

Technische Universität München
TUM School of Engineering and Design

Dynamic Multi-Contact Locomotion for Humanoid Robots

Hard- and Software Design, Contact Planning, and Motion Generation for Autonomous Locomotion

Philipp Seiwald

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Markus Zimmermann

Prüfer der Dissertation:

1. Prof. dr.ir. Daniel J. Rixen
2. Prof. Dr. Timo Oksanen
3. Prof. Abderrahmane Kheddar

Die Dissertation wurde am 26.09.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 06.02.2024 angenommen.

Abstract

Despite the rapid progress of research in the field of humanoid robotics, even modern high-end systems can not compete with human performance when it comes to biped locomotion. In particular the capability of humans to adapt their gait to the environment and to compensate external disturbances remains unrivaled. The goal of this thesis is to bring humanoid robots closer to the locomotion performance of humans. In particular, additional hand support during fast biped walking alias *dynamic multi-contact locomotion* is investigated. The additional contacts are primarily meant to increase the robustness against unforeseen disturbances.

Although the majority of the presented concepts, methods, and solutions can be transferred to any humanoid robot, this thesis is closely related to the particular research prototype LOLA developed at the *Chair of Applied Mechanics* of the *Technical University of Munich*. Since this robot was originally designed for biped locomotion only, extensive modifications had to be made to the hard- and software in order to make it capable of multi-contact locomotion. The changes to the hardware involve the redesign of the entire upper body to make it withstand the increased loads caused by the additional hand support. Moreover, the kinematic topology of the arms is optimized to match the requirements of our target scenarios. Concerning the software revision, this thesis focuses on the planning layer of the locomotion framework which receives an entire new contact planning and motion generation system. Since real-time performance is a primary objective, contact planning is based on a hierarchical search on different levels of detail while the motion generator uses simplified models of the robot to efficiently plan a kinematically and dynamically feasible gait.

The new multi-contact locomotion capabilities of the robot are validated within numerous simulations and real-world experiments. Compared to similar systems, the proposed solution stands out by the (relatively) high walking speed and robustness. Moreover, the entire planning and control software runs onboard and in real-time which represents a distinctive feature when compared to the international competition.

Acknowledgment

The contents of this thesis originate from my work as research assistant and member of the LOLA group at the Chair of Applied Mechanics, TUM. In the following, I want to express my gratitude to the numerous people who supported me in any way during this time.

To begin with, I want to thank my supervisor Prof. dr.ir. Daniel J. Rixen for giving me the opportunity to work on such an inspiring project within this excellent environment and pleasant working atmosphere. I am grateful for his trust and granting me great freedom in my research which he supported through his genuine interest and valuable advice. At this point, I would also like to thank the *German Research Foundation (DFG)* for funding the project *Adaptives Laufen durch Multi-Kontakt Stabilisierung und Nutzung von Teilkontakten für humanoide Roboter* (DFG project number 407378162).

Second, I want to express my gratitude to the previous and present researchers working on the LOLA project. It is obvious that a high-end humanoid such as LOLA cannot be realized or operated by a single person. Instead, a group of highly motivated experts with extraordinary commitment to the project is required. Among many others, this includes Robert Wittmann (who also supported me as mentor of my doctoral project), Arne-Christoph Hildebrandt, Daniel Wahrmann Lockhart, Felix Sygulla, Nora-Sophie Staufenberg, Moritz Sattler, and Tomáš Slimák. Note that working with a mechatronic system of this complexity requires a significant amount of time for maintenance which has to be carried out by the entire project group and is typically not directly visible to outsiders.

My greatest thanks go to Felix Sygulla, with whom I worked most closely during my time at the chair. The success of the multi-contact project would not have been possible without his exceptional expertise and commitment. I will always be grateful for the great time and fun we had during our work on LOLA. I would also like to thank my project partner Shun-Cheng Wu and his supervisor PD Dr. Ing. Habil. Federico Tombari, who contributed the visual perception system and hereby enabled vision-guided experiments.

Concerning the electromechanical realization of the new upper body of LOLA, special thanks are due to the staff of the mechanical and electrical workshop of the chair, namely Simon Gerer, Georg König, Georg Mayr, and Andreas Köstler. In general, my thanks goes to all persons who contributed, directly or indirectly, to the LOLA project. This certainly also includes the numerous students assistants. I also want to thank all my former colleagues at the chair. I learned a lot from each of you and really enjoyed our time together.

Last but not least I want to thank my family for their constant support and Moritz, Tomáš, and Marisa for proofreading this thesis.

Contents

Glossary	vii
1 Introduction	1
1.1 Problem Statement	3
1.2 Author’s Contributions: Overview	3
1.3 Outline	4
2 Fundamentals and State of the Art	5
2.1 System Overview and Hardware Design	5
2.2 Software Design	14
2.3 Contact Planning	19
2.4 Motion Generation	27
2.5 Computer Vision (CV)	40
2.6 Stabilization	42
2.7 The Humanoid Robots JOHNNIE and LOLA	43
2.8 Summary	46
3 Hardware – A New Upper Body for LOLA	47
3.1 Preliminaries	47
3.2 Starting Point	51
3.3 Kinematic Optimization of Arm Topology	54
3.4 Actuation and Sensing	62
3.5 Mechanical Design	68
3.6 Electrical Design	72
3.7 Realization: Manufacturing, Assembly, and Initial Operation	73
3.8 Results and Discussion	74
4 Software – Part A: Locomotion Framework	79
4.1 Overview	79
4.2 Coordinate Systems (CoSys)	82
4.3 Task-Space Definition	84
4.4 Excursus: Computer Vision (CV)	86
4.5 Walking Pattern Generation (WPG)	90
4.5.1 Environment Model	91
4.5.2 Reduced Kinematic and Dynamic Model	94
4.5.3 State Estimation	98
4.5.4 Solution Strategy	99
4.5.5 Planning Pipeline	102
4.6 Excursus: Stabilization and Inverse Kinematics (SIK)	104
4.7 Excursus: Hardware Layer (HWL)	107
4.8 Results and Discussion	108

5	Software – Part B: Contact Planning	111
5.1	Preliminaries	111
5.2	Motion Plan: Higher Level Structure	115
5.3	Quasi-Planar Walking Transition (QPWT)	118
5.4	Transition Planner	120
5.5	Autonomous Locomotion	121
5.5.1	Discretization	122
5.5.2	Pre-Processing	123
5.5.3	A* Algorithm	128
5.5.4	Hierarchical Graph Search	131
5.5.5	Post-Processing	138
5.6	Results and Discussion	143
6	Software – Part C: Motion Generation	146
6.1	Preliminaries	146
6.2	Motion Plan: Lower Level Structure	147
6.3	Phase Planner	150
6.4	Support Area (SA) Planner	152
6.5	Zero-Moment Point (ZMP) Planner	152
6.6	Upper Body Orientation Planner	155
6.7	Foot Motion Planner	156
6.8	Toe Motion Planner	161
6.9	Hand Motion Planner	162
6.10	Head Orientation Planner	165
6.11	Task-Space Selection Factor Planner	165
6.12	Load Factor Planner	166
6.13	External Wrench Planner	168
6.14	Reduced Model Torso (RMT) Planner	168
6.14.1	Vertical RMT Planner	168
6.14.2	Horizontal RMT Planner	172
6.15	Center of Mass (CoM) Planner	176
6.16	Evaluation and Stream Processor	177
6.17	Results and Discussion	177
7	Software – Part D: Ecosystem	180
7.1	Overview	180
7.2	The Open-Source Library Broccoli	182
7.3	The Open-Source Vision Interface	184
7.4	Simulation	185
7.5	Visualization	190
7.6	Control Panel	192
7.7	Conclusions and Suggestions	193
8	Validation – Testing LOLA’s New Capabilities	194
8.1	Simulation	194
8.2	Real-World Experiments	197
9	Closure	201
9.1	Summary	201
9.2	Author’s Contributions and Innovation	202
9.3	Conclusions	204
9.4	Outlook	205

A	Notation	207
B	Quaternion Calculus and Interpolation of Rotations using Quaternions	209
B.1	Fundamentals	209
B.2	Spatial Rotation	212
B.3	Interpolation	215
B.3.1	Linear Interpolation (LERP)	217
B.3.2	Normalized Linear Interpolation (NLERP)	217
B.3.3	Spherical Linear Interpolation (SLERP)	218
B.3.4	Quaternion BÉZIER (QBézier) Curve	220
B.3.5	Spherical Quadrangle (SQUAD) Curve	221
B.3.6	Quaternion B-Spline (QBSpline) Curve	222
B.3.7	Comparison	226
B.3.8	Advanced Speed Control	227
C	Swept Sphere Volumes (SSVs)	228
D	Step Parameters	230
E	Simplified Leg Kinematics	233
F	Dynamics of the Five-Mass Model	234
G	Cubic and Quintic Spline Interpolation and Collocation	237
G.1	Introduction	237
G.2	Materials and Methods	240
G.2.1	Problem Statement	240
G.2.2	Spline Parametrization	241
G.2.3	Spline Interpolation: Preliminaries	243
G.2.4	Cubic Spline Interpolation: Derivation	244
G.2.5	Quintic Spline Interpolation: Derivation	246
G.2.6	Algorithm for Cubic/Quintic Spline Interpolation	249
G.2.7	Spline Collocation: Derivation	250
G.2.8	Satisfying First Order Boundary Conditions for Cubic Splines	253
G.2.9	Algorithm for Cubic/Quintic Spline Collocation	253
G.3	Implementation	253
G.4	Results	255
G.5	Discussion	261
G.6	Attachment: Spline Gradients	262
H	Hardware Details	266
H.1	Mechanical Specifications	266
H.2	Electrical Specifications	268
H.3	Calibration	270
I	Co-authored Publications	271
I.1	Scientific Publications	271
I.2	Published Software	272
I.3	Published Videos	272
I.4	Press Reports (Indirect Publications)	273
J	Supervised Student Theses	274
K	Bibliography	276

Glossary

Acronyms

ABD	Almost Block Diagonal
ADA*	Anytime Dynamic A* (<i>variation of A* search algorithm, see HORNUNG et al. [210]</i>)
AIST	National Institute of Advanced Industrial Science and Technology (<i>Japan</i>)
ANA*	Anytime Nonparametric A* (<i>variation of A* search algorithm, see VAN DEN BERG et al. [423]</i>)
ARA*	Anytime Repairing A* (<i>variation of A* search algorithm, see LIKHACHEV et al. [284]</i>)
ASC	Automatic Supervisory Control (<i>velocity-level IK method, see LIÉGEOIS [283]</i>)
ASIC	Application-Specific Integrated Circuit
BC	Boundary Condition
Bi-RRT	Bidirectional RRT (<i>variation of RRT search algorithm, see LAVALLE and KUFFNER [270]</i>)
Broccoli	Beautiful Robot C++ Code Library (<i>see Section 7.2 and SEIWALD and SYGULLA [15]</i>)
B-Spline	Basis Spline (<i>see Appendix B.3.6 and DE BOOR [129, p. 87ff]</i>)
BVH	Bounding Volume Hierarchy
BVP	Boundary Value Problem
CAN	Controller Area Network (<i>communication bus, see [107]</i>)
CAD	Computer Aided Design
CFRP	Carbon Fiber-Reinforced Polymer
CI	Condition Index (<i>see Equation 3.2, Section 3.3, and MA and ANGELES [296]</i>)
CMOS	Complementary Metal-Oxide-Semiconductor
CMP	Centroidal Moment Pivot (<i>see Section 2.4 and POPOVIC et al. [346]</i>)
CNC	Computer Numerical Control
CNRS	French National Centre for Scientific Research (<i>France</i>)
CoM	Center of Mass
CoP	Center of Pressure (<i>see Footnote 18, Section 2.4, and GOSWAMI [174]</i>)
CoSy	Coordinate System
CoT	Cost of Transport (<i>see Footnote 1 and TUCKER [415]</i>)
CP	Capture Point (<i>see Section 2.4 and PRATT et al. [349]</i>)
CPU	Central Processing Unit
CV	Computer Vision (<i>see Section 4.4</i>)
CWC	Contact Wrench Cone (<i>see Section 2.4 and CARON et al. [110]</i>)
DAE	Differential Algebraic Equation
DARPA	Defense Advanced Research Projects Agency (<i>United States of America</i>)
DC	Direct Current
DCM	Divergent Component of Motion (<i>see Section 2.4 and TAKENAKA et al. [404]</i>)

DDM	Double Description Method (<i>see Footnote 21 and FUKUDA and PRODON [164]</i>)
DDP	Differential Dynamic Programming (<i>see Footnote 24 and MAYNE [303]</i>)
DFG	German Research Foundation (<i>Germany</i>)
DH	DENAVID-HARTENBERG (<i>see HARTENBERG and DENAVIT [191, p. 347ff] and CRAIG [122, p. 65ff]</i>)
DIP	Double Inverted Pendulum (<i>see Figure 2.17 and STEPHENS [392]</i>)
DIY	Do It Yourself
DLR	German Aerospace Center (<i>Germany</i>)
DoF	Degree of Freedom
DRC	DARPA Robotics Challenge (<i>see Footnote 5 and [124]</i>)
DS	Double Support (<i>phase of biped gait: both feet in contact</i>)
DSCB	Distributed Sensor Control Board (<i>see FAVOT [157, p. 18ff]</i>)
EE	End Effector (<i>see Footnote 7</i>)
EMA	Experimental Modal Analysis
EoM	Equation of Motion
ETH	Swiss Federal Institute of Technology in Zürich (<i>Switzerland</i>)
EtherCAT	Ethernet for Control Automation Technology (<i>communication bus, see [77]</i>)
FDM	Fused Deposition Modeling (<i>3D printing technology</i>)
FEA	Finite Element Analysis
FFT	Fast Fourier Transformation
FK	Forward Kinematics
FoR	Frame of Reference (<i>see Appendix A</i>)
FPGA	Field Programmable Gate Array
FRI	Foot Rotation Indicator (<i>see Footnote 19, Section 2.4, and GOSWAMI [174]</i>)
FSM	Finite State Machine
FTS	Force-Torque Sensor
FZMP	Fictitious ZMP (<i>see Footnote 19, Section 2.4, and VUKOBRATOVIĆ and BOROVIAC [428]</i>)
GCoM	Ground projection of the CoM
GFRP	Glass Fiber-Reinforced Polymer
GIWC	Gravito-Inertial Wrench Cone (<i>see Section 2.4 and CARON et al. [108]</i>)
GPIO	General-Purpose IO
GPOS	General-Purpose OS
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRF	Ground Reaction Force
GRW	Ground Reaction Wrench
HD	Harmonic Drive (<i>gear type, see [189]</i>)
HID	Human Interface Device
HWL	Hardware Layer (<i>see Section 4.7 and SYGULLA [401]</i>)
HZD	Hybrid Zero Dynamics (<i>see Footnote 13</i>)
ICP	Iterative Closest Point (<i>see Section 2.5 and BESL and MCKAY [76]</i>)
IHMC	Florida Institute for Human & Machine Cognition (<i>United States of America</i>)
IIT	Italian Institute of Technology (<i>Italy</i>)
IMU	Inertial Measurement Unit (<i>see Section 4.2, Figure 4.2, and Table H.2 for the corresponding CoSy</i>)

IK	Inverse Kinematics
IO	Input/Output
IP	Internet Protocol
IR	Infrared
IVP	Initial Value Problem
JRA	Joint Range Availability (<i>see Equation 3.5 and Section 3.3</i>)
JRL	Joint Robotics Laboratory (<i>cooperation between CNRS and AIST</i>)
KAIST	Korean Advanced Institute of Science and Technology (<i>Korea</i>)
KIT	Karlsruhe Institute of Technology (<i>Germany</i>)
LERP	Linear Interpolation (<i>quaternion interpolation method, see Appendix B.3.1</i>)
LiBM	Linear Biped Model (<i>see Section 2.4 and STEPHENS and ATKESON [393]</i>)
LiDAR	Light Detection and Ranging
LIPM	Linear Inverted Pendulum Mode (<i>see Figure 2.17, Footnote 8, and KAJITA and TANI [230]</i>)
LSE	Linear System of Equations
MEMS	Micro-Electro-Mechanical System
MIT	Massachusetts Institute of Technology (<i>United States of America</i>)
MM	Manipulability Measure (<i>see Equation 3.4, Section 3.3, and YOSHIKAWA [452]</i>)
MPC	Model Predictive Control
MRAC	Model-Reference Adaptive Control (<i>see NGUYEN [322]</i>)
NASA	National Aeronautics and Space Administration (<i>United States of America</i>)
NLERP	Normalized Linear Interpolation (<i>quaternion interpolation method, see Appendix B.3.2</i>)
ODE	Ordinary Differential Equation
OS	Operating System
PC	Personal Computer
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PCL	Point Cloud Library (<i>see RUSU and COUSINS [363]</i>)
PDE	Partial Differential Equation
PDO	Process Data Object (<i>communication object for synchronous data, see CANopen [107]</i>)
PLA	Poly lactide (<i>thermoplastic polymer</i>)
PLY	Polygon File Format (<i>3D triangle mesh format, see TURK [416]</i>)
PMP	PONTRYAGIN's Maximum Principle (<i>see GAMKRELIDZE [166] and NAKAMURA [319, p. 81ff]</i>)
PNM	Portable Anymap Format (<i>family of 2D image file formats, see POSKANZER [347]</i>)
POM	Polyoxymethylene (<i>thermoplastic polymer</i>)
PP	Piecewise Polynomial (<i>description form for polynomial splines, see DE BOOR [129, p. 69ff]</i>)
PSO	Particle Swarm Optimization (<i>see KENNEDY and EBERHART [246]</i>)
QBézier	Quaternion BÉZIER (<i>quaternion interpolation method, see Appendix B.3.4 and SHOEMAKE [378]</i>)
QBSpline	Quaternion B-Spline (<i>quaternion interpolation method, see Appendix B.3.6 and KIM et al. [248]</i>)
QP	Quadratic Program
QPWT	Quasi-Planar Walking Transition (<i>see Section 5.3</i>)
RAM	Random-Access Memory
RCA	Reactive 3D Collision Avoidance (<i>see Section 2.7 and HILDEBRANDT et al. [198]</i>)

RDMA	Remote Direct Memory Access
RGB-D	Red-Green-Blue Depth (<i>image format with three color channels and one depth channel</i>)
RI	Reachability Index (<i>see Equation 3.1 and Section 3.3</i>)
RMC	Resolved Motion Rate Control (<i>velocity-level IK method, see WHITNEY [434]</i>)
RMP	Reaction Mass Pendulum (<i>see Figure 2.17 and LEE and GOSWAMI [274]</i>)
RMS	Root Mean Square
RMT	Reduced Model Torso (<i>see Section 4.5.2 and Figure 4.7</i>)
ROS	Robot Operating System (<i>middleware, see [330]</i>)
RRT	Rapidly-Exploring Random Tree (<i>search algorithm, see LAVALLE [269]</i>)
RTLinux	Real-Time Linux (<i>Linux-based RTOS, see BARABANOV [74]</i>)
RTM	Robot Technology Middleware (<i>middleware, see ANDO et al. [60]</i>)
RTOS	Real-Time OS
SA	Support Area (<i>see Section 2.3</i>)
SDF	Signed Distance Field
SDL	Simple DirectMedia Layer (<i>see LANTINGA et al. [265]</i>)
SDO	Service Data Object (<i>communication object for asynchronous data, see CANopen [107]</i>)
SEA	Series Elastic Actuator (<i>see Footnote 2 and PRATT and WILLIAMSON [348]</i>)
Sercos	Serial Realtime Communication System (<i>communication bus, see [377]</i>)
SIK	Stabilization and Inverse Kinematics (<i>see Section 4.6 and SYGULLA [401]</i>)
SLAM	Simultaneous Localization and Mapping (<i>see Footnote 10</i>)
SLERP	Spherical Linear Interpolation (<i>quat. interp. method, see Appendix B.3.3 and SHOEMAKE [378]</i>)
SLIP	Spring-Loaded Inverted Pendulum (<i>see Figure 2.17, Footnote 4, and BLICKHAN [86]</i>)
SMT	Simultaneous Multithreading
SP	Support Polygon (<i>see Section 2.3</i>)
SPI	Serial Peripheral Interface (<i>communication bus, see [203]</i>)
SQUAD	Spherical Quadrangle (<i>quaternion interpolation method, see Appendix B.3.5 and DAM et al. [123]</i>)
SQP	Sequential Quadratic Program
SS	Single Support (<i>phase of biped gait: one foot in contact</i>)
SSD	Solid State Drive
SSV	Swept Sphere Volume (<i>see Appendix C</i>)
SVD	Singular Value Decomposition (<i>see QUARTERONI et al. [352, p. 17f]</i>)
TCP	Tool Center Point (<i>see Footnote 33</i>)
TCP/IP	Transmission Control Protocol (<i>here: used in combination with the Internet Protocol (IP)</i>)
TOML	Tom's Obvious Minimal Language (<i>configuration file format, see PRESTON-WERNER et al. [351]</i>)
TUM	Technical University of Munich (<i>Germany</i>)
UDP/IP	User Datagram Protocol (<i>here: used in combination with the Internet Protocol (IP)</i>)
UI	User Interface
USB	Universal Serial Bus (<i>communication bus</i>)
VPP	Virtual Pivot Point (<i>see Section 2.4 and MAUS et al. [301]</i>)
VRP	Virtual Repellent Point (<i>see Footnote 22 and ENGLSBERGER et al. [142]</i>)
WPG	Walking Pattern Generation (<i>Section 4.5</i>)
ZMP	Zero-Moment Point (<i>see Section 2.4 and VUKOBRATOVIĆ and BOROVAC [428]</i>)
ZRAM	Zero Rate of change of Angular Momentum (<i>see Section 2.4, GOSWAMI and KALLEM [175]</i>)

Symbols

a	axis component $a \in \mathbb{R}$
\mathbf{a}	axis $\mathbf{a} = [a_x, a_y, a_z]^T \in \mathbb{R}^3$ (typically with $\ \mathbf{a}\ = 1$)
A	area $A \in \mathbb{R}$ rotation matrix component $A_{ij} \in \mathbb{R}$
\mathbf{A}	rotation matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $n \in \{2, 3\}$ (see Appendix A)
B	B-spline $B \in \mathbb{R}$ (see Appendix B.3.6)
\tilde{B}	cumulative B-spline $\tilde{B} \in \mathbb{R}$ (see Appendix B.3.6)
c	confidence $c \in \mathbb{R}$ (typically with $c \in [0, 1]$) cost $c \in \mathbb{R}$
\mathcal{C}	convex cone (e. g. friction cone see Figure 2.19)
d	diameter $d \in \mathbb{R}$
e	error $e \in \mathbb{R}$
$\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$	unit vector $\mathbf{e} \in \mathbb{R}^3$ in x -, y -, or z -direction (FoR depends on context)
E	elastic modulus $E \in \mathbb{R}$
F	force $F \in \mathbb{R}$ exact solution $F(t) \in \mathbb{R}$ of second-order linear ODE (see Equation G.1)
\mathbf{F}	force vector $\mathbf{F} = [F_x, F_y, F_z]^T \in \mathbb{R}^3$
g	gravitational acceleration $g \in \mathbb{R}$ (here: $g = 9.81 \text{ m/s}^2$) reciprocal $g = 1/h \in \mathbb{R}$ of segment proportion or duration $h \in \mathbb{R}$
\mathbf{g}	gravitational acceleration vector ${}_{\text{w}}\mathbf{g} = [0, 0, -g]^T \in \mathbb{R}^3$
h	height $h \in \mathbb{R}$ segment proportion or duration $h \in \mathbb{R}$
\mathbf{H}	homogeneous transform $\mathbf{H} \in \mathbb{R}^{4 \times 4}$ (see Appendix A and CRAIG [122, p. 28])
\mathbb{H}	set of quaternions (see Equation B.1)
\mathbb{H}^1	set of unit quaternions (see Equation B.10)
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	basic quaternions with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ (see Equation B.1)
I	second moment of area $I \in \mathbb{R}$
\mathbf{J}	Jacobian matrix $\mathbf{J} \in \mathbb{R}^{p \times n}$ (see Appendix A)
k	order $k \in \mathbb{N}_1$ of a B-spline curve (see Appendix B.3.6)
l	length $l \in \mathbb{R}$
L^P	angular momentum $L^P \in \mathbb{R}$ at reference point P
\mathbf{L}^P	angular momentum vector $\mathbf{L}^P = [L_x^P, L_y^P, L_z^P]^T \in \mathbb{R}^3$ at reference point P
m	mass $m \in \mathbb{R}$ count of B-splines $m \in \mathbb{N}_1$ of a B-spline curve (see Appendix B.3.6)
M	mass proportion $M \in \mathbb{R}$ (e. g. proportion $M \in [0, 1]$ of the total mass of the robot)
n	dimension of joint-space $n \in \mathbb{N}_1$ (see Figure 4.3) count of knots $n \in \mathbb{N}_2$ of a B-spline curve (see Appendix B.3.6) normal component $n \in \mathbb{R}$
\mathbf{n}	normal $\mathbf{n} = [n_x, n_y, n_z]^T \in \mathbb{R}^3$ (typically with $\ \mathbf{n}\ = 1$)
\mathcal{N}	node \mathcal{N} of \mathbf{A}^* algorithm (see Section 5.5.3)

p	dimension of task-space $p \in \mathbb{N}_1$ (see Figure 4.3) degree $p \in \mathbb{N}_0$ of a B-spline curve (see Appendix B.3.6) linear momentum $p \in \mathbb{R}$
\mathbf{p}	linear momentum vector $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathbb{R}^3$
\mathcal{P}	plane (e. g. see Figure 5.13)
q	joint-space component $q \in \mathbb{R}$
\mathbf{q}	joint-space vector $\mathbf{q} = [q_1, \dots, q_n]^T \in \mathbb{R}^n$ (see Figure 4.3)
r	position $r \in \mathbb{R}$ radius $r \in \mathbb{R}$
\mathbf{r}	position vector $\mathbf{r} = [r_x, r_y, r_z]^T \in \mathbb{R}^3$ (see Appendix A)
R	residual $R \in \mathbb{R}$
s	quaternion component $s \in \mathbb{R}$ segment $s(\eta) \in \mathbb{R}$ of a spline (see Equation G.6)
\mathbf{s}	quaternion $\mathbf{s} = s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k} \in \mathbb{H}$ (see Equation B.1)
$\bar{\mathbf{s}}$	conjugate quaternion $\bar{\mathbf{s}} = s_w - s_x \mathbf{i} - s_y \mathbf{j} - s_z \mathbf{k} \in \mathbb{H}$ (see Equation B.14)
\mathcal{S}	state \mathcal{S} of A^* algorithm (see Section 5.5.3)
$\bar{\mathcal{S}}$	“conjugate” state $\bar{\mathcal{S}}$ of A^* state \mathcal{S} (equivalent to \mathcal{S} but with opposite stance foot)
t	time $t \in \mathbb{R}$
T	torque $T \in \mathbb{R}$
\mathbf{T}	torque vector $\mathbf{T} = [T_x, T_y, T_z]^T \in \mathbb{R}^3$
v	task-space velocity component $v \in \mathbb{R}$
\mathbf{v}	task-space velocity vector $\mathbf{v} = [v_1, \dots, v_p]^T \in \mathbb{R}^p$ (see Table 4.1)
V	volume $V \in \mathbb{R}$
\mathcal{W}	workspace (e. g. see Figure 3.4)
\mathbf{W}^P	wrench $\mathbf{W}^P = [F_x, F_y, F_z, T_x, T_y, T_z]^T \in \mathbb{R}^6$ acting at point P
x	task-space component $x \in \mathbb{R}$
\mathbf{x}	task-space vector $\mathbf{x} = [x_1, \dots, x_p]^T \in \mathbb{R}^p$ (see Figure 4.3)
y	spline approximation $y(t) \in \mathbb{R}$ of second-order linear ODE solution (see Equation G.4)
α	first left hand side coefficient $\alpha \in \mathbb{R}$ of second-order linear ODE (see Equation G.1)
β	second left hand side coefficient $\beta \in \mathbb{R}$ of second-order linear ODE (see Equation G.1)
γ	load factor $\gamma \in \mathbb{R}$ ($\gamma_e \in [0, 1]$ and $\gamma_{RF} + \gamma_{LF} := 1$, see Footnote 57) third left hand side coefficient $\gamma \in \mathbb{R}$ of second-order linear ODE (see Equation G.1)
$\boldsymbol{\gamma}$	load vector $\boldsymbol{\gamma} = [\gamma_{RF}, \gamma_{LF}, \gamma_{RH}, \gamma_{LH}]^T \in \mathbb{R}^4$
ε	machine precision $\varepsilon \in \mathbb{R}$ with $0 < \varepsilon \ll 1$ (or equivalently a number very close to zero)
η	interpolation parameter $\eta \in \mathbb{R}$ (typically $\eta \in [0, 1]$)
ϑ	rotation vector component $\vartheta \in \mathbb{R}$
$\boldsymbol{\vartheta}$	rotation vector $\boldsymbol{\vartheta} = [\vartheta_x, \vartheta_y, \vartheta_z]^T \in \mathbb{R}^3$ (see Appendix B.2)
$\boldsymbol{\Theta}^P$	mass moment of inertia tensor $\boldsymbol{\Theta}^P \in \mathbb{R}^{3 \times 3}$ with respect to point P
κ	curvature $\kappa \in \mathbb{R}$ (see Section 5.5.2)
$\boldsymbol{\kappa}$	mean curvature normal operator $\boldsymbol{\kappa} \in \mathbb{R}^3$ (see Section 5.5.2 and MEYER et al. [306, p. 44])
μ	friction coefficient $\mu \in \mathbb{R}$
ξ	task-space selection factor $\xi \in \mathbb{R}$ ($\xi \in [0, 1]$ with $\xi = 0$: “in null-space” and $\xi = 1$: “in task-space”)
$\boldsymbol{\xi}$	task-space selection vector $\boldsymbol{\xi} = [\xi_{RH}, \xi_{LH}]^T \in \mathbb{R}^2$

ρ	density $\rho \in \mathbb{R}$
σ	singular value $\sigma \in \mathbb{R}$ of a matrix (see QUARTERONI et al. [352, p. 17f]) standard deviation
τ	timing factor $\tau \in \mathbb{R}$ (typically $\tau \in [0, 1]$) knot $\tau \in \mathbb{R}$ of a B-spline curve (see Appendix B.3.6) right hand side $\tau \in \mathbb{R}$ of second-order linear ODE (see Equation G.1)
$\boldsymbol{\tau}$	knot sequence $\boldsymbol{\tau} = \{\tau_1, \dots, \tau_n\}$ of a B-spline curve (see Appendix B.3.6)
φ	angle $\varphi \in \mathbb{R}$
ω	angular velocity $\omega \in \mathbb{R}$
$\boldsymbol{\omega}$	angular velocity vector $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^3$
$\mathbf{0}$	vector or matrix with all components being zero
$\mathbf{1}$	identity matrix with all components being one on the diagonal and zero otherwise
$\mathbf{0}_{\mathbb{H}}$	additive identity quaternion (see Equation B.12)
$\mathbf{1}_{\mathbb{H}}$	multiplicative identity quaternion (see Equation B.11)
$E(n)$	Euclidean group in dimension $n \in \{2, 3\}$ (all transformations of \mathbb{R}^n that can be expressed as a composition of a translation, a rotation, and (optional) a reflection [185, p. 10f])
$SE(n)$	special Euclidean group in dimension $n \in \{2, 3\}$ (all transformations of \mathbb{R}^n that can be expressed as a composition of a translation and a rotation)
$O(n)$	orthogonal group in dimension $n \in \{2, 3\}$ (all transformations of \mathbb{R}^n that can be expressed as a rotation or a composition of a rotation and a reflection [185, p. 6ff])
$SO(n)$	special orthogonal group in dimension $n \in \{2, 3\}$ (all transformations of \mathbb{R}^n that can be expressed as a rotation [185, p. 6ff])

Indices

$(\cdot)_{u v}$	component in u - or v -direction (barycentric coordinates)
$(\cdot)_w$	component in w -direction (barycentric coordinates) real (scalar) component of a quaternion (see Appendix B.1)
$(\cdot)_{x y z}$	component in x -, y -, or z -direction (FoR depends on context)
$(\cdot)_W$	world frame (inertial, equivalent to PW, see Section 4.2 and Figure 4.2)
$(\cdot)_{PW}$	planning world frame (inertial, equivalent to W, see Section 4.2 and Figure 4.2)
$(\cdot)_{VW}$	vision world frame (inertial, see Section 4.2 and Figure 4.2)
$(\cdot)_t$	as body part: torso or RMT as CoSy: torso segment frame (attached to torso, see Figure 3.5)
$(\cdot)_{UB}$	upper body frame (origin: undef., orientation: same as IMU, see Section 4.2, Figure 4.2, and Table H.2)
$(\cdot)_{RF}$	as body part: right foot as CoSy: TCP frame of right foot (attached to zfr , see Section 4.2, Figure 4.2, and Table H.2)
$(\cdot)_{LF}$	as body part: left foot as CoSy: TCP frame of left foot (attached to zfl , see Section 4.2, Figure 4.2, and Table H.2)
$(\cdot)_{SF}$	as body part: current stance foot as CoSy: TCP frame of current stance foot (inertial, see Section 4.2 and Figure 4.2)
$(\cdot)_{\overline{SF}}$	as body part: current swing foot (opposite of stance foot, i. e., $\overline{SF} := \{RF, LF\} \setminus SF$) as CoSy: TCP frame of current swing foot
$(\cdot)_{MF}$	mean foot TCP frame (see Section 5.1)
$(\cdot)_f$	foot $f \in \{RF, LF\}$

$(\cdot)_{RH}$	as body part: right hand as CoSy: TCP frame of right hand (<i>attached to efr, see Section 4.2, Figure 4.2, and Table H.2</i>)
$(\cdot)_{LH}$	as body part: left hand as CoSy: TCP frame of left hand (<i>attached to efl, see Section 4.2, Figure 4.2, and Table H.2</i>)
$(\cdot)_h$	hand $h \in \{RH, LH\}$
$(\cdot)_e$	EE $e \in \{RF, LF, RH, LH\}$ (<i>without head</i>)
$(\cdot)_{VTCP}$	vision TCP frame (<i>attached to vt, see Section 4.2, Figure 4.2, and Table H.2</i>)
$(\cdot)_{EO}$	environmental object frame (<i>see Section 4.5.1 and Figure 4.6</i>)
$(\cdot)_{acq}$	acquisition
$(\cdot)_{act}$	actual
$(\cdot)_{beg}$	begin
$(\cdot)_{clo}$	closest close
$(\cdot)_{coll}$	collocation
$(\cdot)_{cont}$	contact
$(\cdot)_{cur}$	current
$(\cdot)_{def}$	default
$(\cdot)_{des}$	desired
$(\cdot)_{dist}$	distance
$(\cdot)_{dur}$	duration
$(\cdot)_{end}$	end
$(\cdot)_{ext}$	external
$(\cdot)_{free}$	free
$(\cdot)_{gau}$	GAUSS
$(\cdot)_{goal}$	goal
$(\cdot)_{hom}$	homogeneous
$(\cdot)_{human}$	human
$(\cdot)_{idle}$	idle
$(\cdot)_{lag}$	lag
$(\cdot)_{lead}$	lead
$(\cdot)_{lift}$	lift
$(\cdot)_{lower}$	lower
$(\cdot)_{max}$	maximum
$(\cdot)_{mean}$	mean
$(\cdot)_{min}$	minimum
$(\cdot)_{mult}$	multi-contact
$(\cdot)_{new}$	new
$(\cdot)_{next}$	next
$(\cdot)_{open}$	open
$(\cdot)_{opt}$	optimum
$(\cdot)_{orig}$	original
$(\cdot)_{par}$	parent
$(\cdot)_{peak}$	peak

$(\cdot)_{\text{perm}}$	permanent
$(\cdot)_{\text{pha}}$	phase
$(\cdot)_{\text{pre}}$	previous
$(\cdot)_{\text{proj}}$	projection
$(\cdot)_{\text{ref}}$	reference
$(\cdot)_{\text{seg}}$	segment
$(\cdot)_{\text{start}}$	start
$(\cdot)_{\text{step}}$	step
$(\cdot)_{\text{str}}$	stride
$(\cdot)_{\text{succ}}$	successor
$(\cdot)_{\text{surf}}$	surface
$(\cdot)_{\text{test}}$	test
$(\cdot)_{\text{thres}}$	threshold
$(\cdot)_{\text{tra}}$	transition
$(\cdot)_{\text{virt}}$	virtual
$(\cdot)_{\text{vis}}$	vision
$c(\cdot)$	quantity related to cubic spline interpolation or collocation (e. g. see Equation G.8)
$q(\cdot)$	quantity related to quintic spline interpolation or collocation (e. g. see Equation G.11)

Operators

$(\cdot)!$	factorial $\alpha! \in \mathbb{N}_1$ of a non-negative integer $\alpha \in \mathbb{N}_0$
$(\cdot)^T$	transpose
$(\cdot)^{-1}$	for scalars: reciprocal for matrices: inverse for quaternions: multiplicative inverse (see Equation B.15)
$(\cdot)^{\#}$	pseudoinverse of a matrix (here: MOORE-PENROSE pseudoinverse [319, p. 41ff])
$ \cdot $	for scalars: absolute value for sets: cardinality for quaternions: modulus (absolute value) (see Equation B.9)
$\ \cdot\ $	for vectors: norm of the vector (Euclidean norm $\ \cdot\ _2$ if not specified otherwise) for matrices: norm of the matrix (spectral norm $\ \cdot\ _2$ if not specified otherwise)
$(\cdot) \otimes (\cdot)$	HAMILTON product of two quaternions (see Equation B.7)
$(\cdot) \odot (\cdot)$	dot product of two quaternions (see Equation B.8)
$\text{atan2}(y, x)$	2-argument arctangent $\text{atan2}(y, x) \in]-\pi, \pi]$
$\text{axAng}(\cdot)$	constructor of axis-angle representation from unit quaternion (see Equation B.26)
$e^{(\cdot)}, \exp(\cdot)$	for scalars: natural exponential function $e^\alpha = \exp(\alpha) \in \mathbb{R}$ for quaternions: quaternion natural exponential function (see Equation B.16)
$\det(\cdot)$	determinant of a square matrix
$\text{diag}(\cdot)$	diagonal matrix $\text{diag}(\alpha_1, \dots, \alpha_n) \in \mathbb{R}^{n \times n}$ composed of the given scalars or matrices
$\text{dim}(\cdot)$	dimension of a vector or matrix
$\text{heur}(\cdot)$	heuristic (estimated cost of remaining path) for the given A^* state (see Section 5.5.3)

$\ln(\cdot)$	for scalars: natural logarithm function for quaternions: quaternion natural logarithm function (<i>see Equation B.17</i>)
$\max(\cdot)$	maximum element of the arguments
$\min(\cdot)$	minimum element of the arguments
$\text{node}(\cdot)$	constructor of A* node from state, predecessor node, and costs (<i>see Section 5.5.3</i>)
$\text{norm}(\cdot)$	for vectors: normalization for quaternions: normalization (quaternion sign) (<i>see Equation B.13</i>)
$\text{pred}(\cdot)$	predecessor of an A* node (<i>see Section 5.5.3</i>)
^{real/vec} $\text{quat}(\cdot)$	constructor of quaternion from real and vector part (<i>see Equation B.4</i>)
^{ax/ang} $\text{quat}(\cdot)$	constructor of unit quaternion from axis-angle representation (<i>see Equation B.25</i>)
^{rot-vec} $\text{quat}(\cdot)$	constructor of unit quaternion from rotation vector (<i>see Equation B.27</i>)
$\text{rank}(\cdot)$	rank of a matrix
$\text{real}(\cdot)$	real (scalar) part of a quaternion (<i>see Equation B.3</i>)
$\text{rotMat}(\cdot)$	constructor of rotation matrix from unit quaternion (<i>see Equation B.24</i>)
$\text{rotVec}(\cdot)$	constructor of rotation vector from unit quaternion (<i>see Equation B.28</i>)
$\text{stateCost}(\cdot)$	state cost for the given A* state (<i>see Section 5.5.3</i>)
$\text{succList}(\cdot)$	list of possible successors for the given A* node (<i>see Section 5.5.3</i>)
$\text{tranCost}(\cdot)$	transition cost from a given A* node to a given A* state (<i>see Section 5.5.3</i>)
$\text{vec}(\cdot)$	vector part of a quaternion (<i>see Equation B.3</i>)

Chapter 1

Introduction

Over the past 200 years, the global human population has increased significantly. This led to a growing demand for goods and energy, which in turn caused severe industrial pollution and depletion of resources. Although the growth rate of the population declined in the last decades [361], the worldwide production capacities and thus the accompanying negative impacts on our environment are expected to further increase. The industrial revolution, however, also had positive effects on ecology: through extensive automation and production in large scales, a sudden rise of efficiency allowed to reduce the amount of resources required to satisfy the needs of such a large population. Since efficiency is also important from an economic point of view, companies continuously try to raise the level of automation. Today, almost all goods of our everyday life are produced through an automated process which is partly or completely taken over by industrial machinery and robots.

Nevertheless, there remain countless physical tasks a machine is not yet able to perform. This includes problems, which require a very flexible reaction to uncertainties and changing environmental conditions. As an example, robotic manipulators in industry are optimized for high throughput and reliability while performing a repetitive task in a well-controlled environment. While this fits well into the context of an industrial production line, this type of robots cannot be used in uncertain environments or in the close vicinity of humans. However, recent developments show that there is high interest of industry and society in making robots more versatile and safe to explore new fields of application such as autonomous driving or the last mile in parcel delivery. Apart from technological challenges, there are also societal implications, especially with regard to (social) interaction between human and machine.

This thesis investigates a certain type of such versatile machines, namely *Humanoids*, i. e., robots with a visual appearance similar to humans. The focus of this work lies on core physical skills, in particular robust biped locomotion, which is a base requirement for most future applications. Currently, the majority of mobile robotic systems, e. g. in logistics, is wheel- or chain-driven, which seems reasonable when considering stability, robustness, and cost. The focus on legged robots with a topology similar to humans can be motivated through their potential future applications as they are expected to perform better in environments which were designed for humans such as the interior of buildings. In particular, climbing stairs or stepping over obstacles is typically easier to achieve for legged machines. Moreover, acceptance by the society strongly depends on the visual appearance of the robot which is relevant, e. g. for nursing, elderly care, or the entertainment industry. Although the focus of this thesis lies on biped robots, many findings can be transferred to other types of autonomous, mobile systems.

Robust locomotion of biped robots is a broad area of research. This work focuses on so-called *dynamic multi-contact locomotion*, where the term *multi-contact* stands for additional support with the robot's hands to increase stability and robustness. In comparison to regular biped walking, the additional contacts introduce higher requirements on the hardware and also make controlling the robot much more complex. This work specifically addresses the challenges in hard- and software design, contact planning, and motion generation and demonstrates the performance of the proposed hard- and software through experiments under real-world conditions. Apart from this, multi-contact locomotion also sets high requirements on low-level stabilization

and control. These topics have been investigated for the same research platform by SYGULLA and are described in his dissertation [401], which represents a complementary work to this thesis. Finally, by the term *dynamic*, a moderate walking speed is assumed, such that the main characteristics of biped gait are preserved. Up to now, most humanoid robots have been limited to rather low walking speeds when compared to humans. This holds true especially for multi-contact situations where most humanoids slow down to quasi-static motion. In contrast, the achievements of the work presented in this thesis include walking speeds of up to 1.8 km/h under multi-contact conditions. Examples of target scenarios matching the label *dynamic multi-contact locomotion* are depicted in Figure 1.1. Note that this thesis does not investigate *non-gaited* multi-contact actions like crawling, climbing ladders, or manipulation. In addition, only unilateral contacts, i. e., pushing forces, are considered.

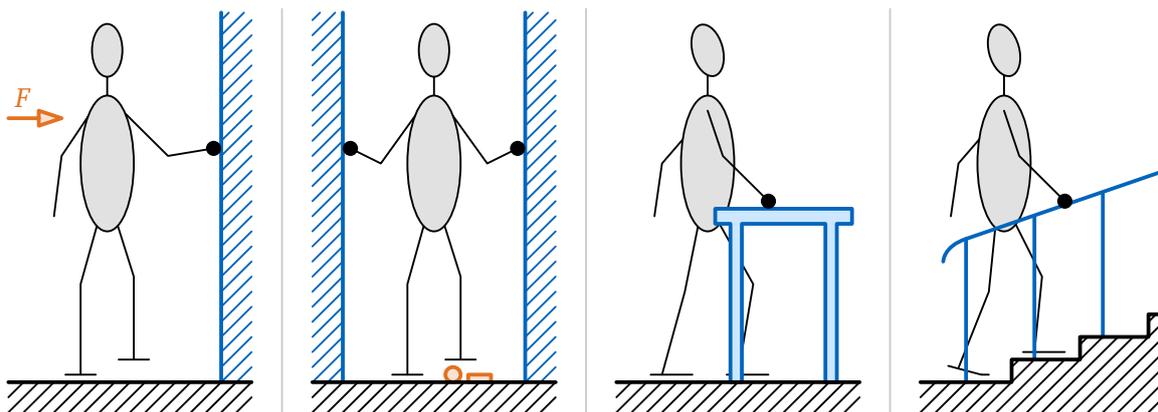


Figure 1.1: Target scenarios for dynamic multi-contact locomotion. From left to right: additional support in the presence of walls, corridors, tables, and handrails (blue). The augmentation by explicit hand-contacts is meant to improve robustness against unforeseen external disturbances and uncertainties in the environment (orange).

In the following section, the actual problem which is to be investigated by this thesis is formulated. Subsequently, in Section 1.2 a rough overview of the contributions of the author is given. Finally, Section 1.3 draws an outline of the structure of this thesis.

Notation and Quaternions An overview of the notation of mathematical formulations used throughout this document is given in Appendix A. Moreover, for describing orientations in the three-dimensional space, in most cases quaternions are used. Since some readers might not be familiar with quaternion calculus, a brief introduction is given in Appendix B.

Hyperlinks The electronic version of this document is equipped with clickable hyperlinks, which allow the reader to quickly jump to the corresponding reference. This also applies to the various referred videos, for which a special citation style is used: e. g. [20 🎥]. Here, the leading number is a classical (in-document) reference to the corresponding bibliography entry and the subsequent “play button” redirects to the online video. For long videos, an additional time stamp might be attached, e. g. [20 🎥@t=10s]. The time stamp is integrated into the underlying link. Moreover, not only citations highlighted in blue are clickable, but also

- references to chapters, sections, paragraphs, footnotes, figures, etc., e. g. Figure 1.1,
- acronyms, e. g. ZMP (redirects to the corresponding glossary entry),
- (most) symbols, indices, and operators, e. g. $\mathbf{F} = [F_x, F_y, F_z]^T$ (redirects to glossary),
- author names, e. g. SEIWALD et al. (redirects to the corresponding bibliography entry),
- (most) robot names, e. g. LOLA (redirects to a photo if included in this document), and
- segment names of the robot LOLA, e. g. arr|| (redirects to its location in the topology).

1.1 Problem Statement

The goal of this work is to improve the autonomy, versatility, and robustness of biped locomotion. This seems to be essential in order to push humanoid robots further into the direction of potential future applications. Impressive progress has been demonstrated in the past years, however, even modern high-end prototypes in academia and industry do not satisfy the high requirements on safety, reliability, and versatility when it comes to real-world use cases in the vicinity of humans. Obviously, a life-sized, fully-actuated humanoid robot is a highly complex system with a large number of subsystems, each of which representing an area of research on its own. The main challenges lie in the design of potent hardware, the identification of dominating dynamic effects, the creation of appropriate models, and the development of efficient planning and control algorithms, which run in real-time under the restrictions of limited onboard computing power. In contrast to wheeled systems, locomotion of legged robots is characterized through repeatedly opening and closing contacts, which is an additional challenge when considering the multi-body dynamics for motion planning and stabilization purposes.

This thesis focuses on a special walking skill: autonomous, dynamic multi-contact locomotion. In particular, the hard- and software of the research platform LOLA (introduced in Section 2.7) was enhanced, so that it is capable of performing the target scenarios depicted in Figure 1.1. Here, the additional hand support during locomotion is meant to improve robustness and further enhance the performance in biped walking as a core skill of humanoid robots. Compared to regular biped walking, multi-contact locomotion sets higher requirements on the physical capabilities of the robot and further raises the complexity in planning and control. For the humanoid LOLA, this requires substantial modifications to the hard- and software design, the contact planning, and the motion generation system.

1.2 Author's Contributions: Overview

The research on legged robots by the *Chair of Applied Mechanics* at the *Technical University of Munich (TUM)* and in particular the research on the humanoids JOHNNIE and LOLA was mainly funded by the *German Research Foundation (DFG)*. Accordingly, the work described in this thesis was funded by the DFG project *Adaptives Laufen durch Multi-Kontakt Stabilisierung und Nutzung von Teilkontakten für humanoide Roboter* (DFG project number 407378162). The project was a cooperation with the *Chair for Computer Aided Medical Procedures & Augmented Reality*, TUM, responsible for work packages related to visual perception. The primary focus of the project was set on autonomous multi-contact locomotion, where the main workload was split into

- a new visual perception system (contributed by WU et al., see [446, 448] and the brief summary of this module in Section 4.4),
- a new upper body hardware design and a new contact planning and motion generation pipeline (contributed by the author of this thesis, see Chapters 3 to 7), and
- an extension of the stabilization system (contributed by SYGULLA, see [401, p. 130ff] and the brief summary of this module in Section 4.6).

Certainly, the overall results presented in this thesis have to be understood as common achievement. Moreover, the presented work substantially benefits from the outstanding contributions of the former researchers of the JOHNNIE and LOLA projects (see Section 2.7). A more detailed summary of the author's individual contributions is given in Section 9.2.

1.3 Outline

This thesis starts in Chapter 2 with an introduction to the fundamentals and the state of the art with regard to legged robots and a special focus on biped and multi-contact locomotion. This is followed by the core contents, which separately discuss the (upper body) hardware in Chapter 3, the locomotion framework in Chapter 4, and the software ecosystem in Chapter 7. An overview of the organization of chapters and sections, visualized as simplified flowchart representing the complete LOLA platform, is given in Figure 1.2. For a clear presentation, the sections within Chapter 4 are sorted according to the main data flow. Since the modules for contact planning and motion generation represent core contributions of the author to LOLA’s locomotion framework, they are presented in full detail in the Chapters 5 and 6, respectively. In Chapter 8, the new capabilities of the robot are validated on the basis of various locomotion scenarios which are evaluated within simulations and real-world experiments. Finally, Chapter 9 contains a summary, a detailed list of the author’s individual contributions, and an outlook including suggestions for future investigations.

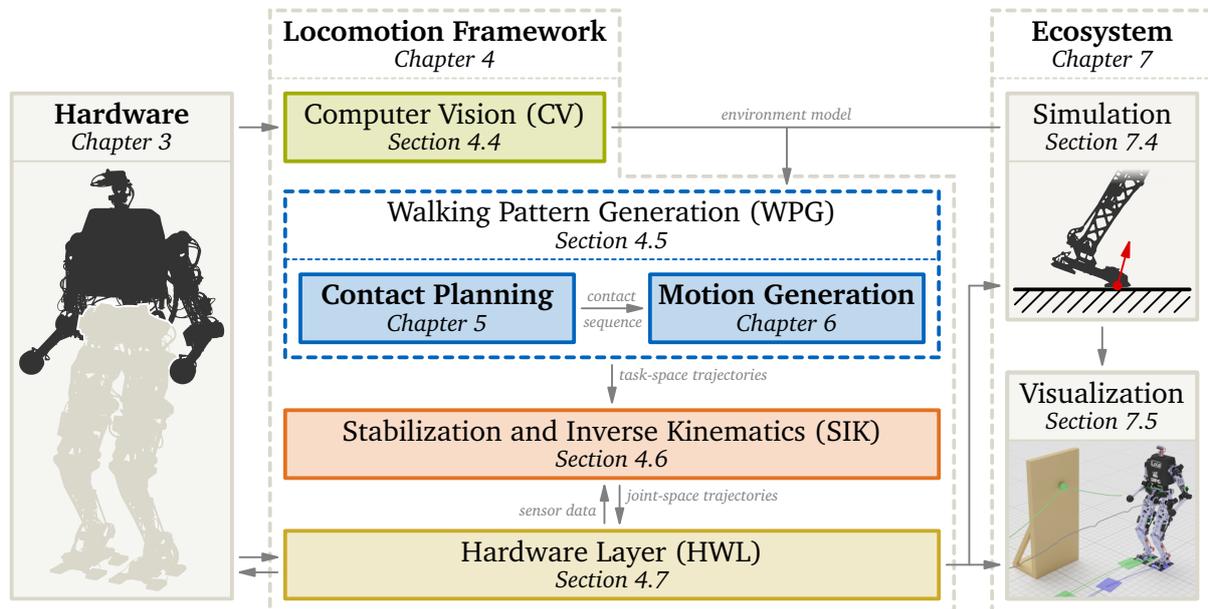


Figure 1.2: Overview of the organization of the core contents of this thesis on the basis of a simplified flowchart of the LOLA platform. The description of the main contents (Chapters 3 to 7) is preceded by an introduction to the fundamentals and state of the art (Chapter 2, not shown) and followed by a validation of the new locomotion capabilities (Chapter 8, not shown) and closure (Chapter 9, not shown).

This document includes descriptions of software which has been contributed by others – in particular the modules *Computer Vision (CV)* see Section 4.4, *Stabilization and Inverse Kinematics (SIK)* see Section 4.6, and *Hardware Layer (HWL)* see Section 4.7. These sections are kept rather short and are meant to give the reader a clue on the role of the corresponding module in the governing framework. Moreover, Section 7.4 describes various extensions of the author to the preexisting simulation framework of LOLA, which was originally developed by BUSCHMANN [100] and SCHWIENBACHER [372].

Chapter 2

Fundamentals and State of the Art

Within the scope of this thesis, the challenges of multi-contact locomotion are investigated with regard to the rather diverse disciplines hard- and software design, contact planning, and motion generation. In the following sections, the state of the art for legged, biped, and multi-contact locomotion of robots regarding the aforementioned disciplines is summarized. This is meant to give a general overview and to introduce the reader to the fundamentals of this special field of robotics. Moreover, it locates the results of this thesis in the broad context of robotic research.

2.1 System Overview and Hardware Design

Whenever a new legged robotic system is to be developed, special focus has to be put on a carefully thought-out hardware design. The most impressive performances have been achieved by high-end prototypes such as ASIMO [367] by *Honda* or ATLAS [321] by *Boston Dynamics*. Although some design flaws might be compensated through smart control strategies, the overall performance still strongly depends on a well engineered system and hardware design.

Nowadays, there exist various types of legged robots in the consumer market. These mainly serve entertainment purposes, such as the internationally very successful toy robot dog AIBO [162] (Figure 2.1, left) by *Sony* or the huge variety of *Do It Yourself (DIY)* kits for two-, four-, and six-legged robots. All commercially sold systems have in common that they are of rather small size. This also holds true for the prominent humanoids SDR-4X alias QRIO [163] (discontinued – Figure 2.1, center) by *Sony* and NAO [176] (mainly intended as research platform – Figure 2.1, right) by *SoftBank / Aldebaran Robotics*, which are both less than 60 cm high.



Figure 2.1: Small sized, commercial quadruped and humanoid robots. From left to right: toy robots AIBO and QRIO by *Sony* and research platform NAO by *SoftBank / Aldebaran Robotics*. Images modified from [383, 386, 387].

Apart from cost reasons, the small form factor has two essential advantages: on the one hand, relatively weak actuators can be used, which is important for product safety considerations. On

the other hand, the low mass in relation to the material strength makes these robots quite robust against damage due to falling or unintended collisions. In contrast, with full-sized humanoids, the potential damage to the machine or nearby humans is significantly higher. Thus, this taller class of robots is – regardless of their not (yet) sufficient usability for real-world applications – still restricted to prototypes and demonstrators in academic and industrial research institutes.

Some kind of intermediate step is made with large scale quadrupeds, for which significant progress in terms of robustness has been made with BIGDOG [356] (Figure 2.2, left) and its successors, e. g. the LS3 alias ALPHADOG [39] (Figure 2.2, center) by *Boston Dynamics*. While BIGDOG was a first demonstrator for potential applications in military, ALPHADOG represented a further developed heavy-duty version of it. Apart from a high-strength mechanical design driven by hydraulic actuators, these robots also featured a very robust control which has evolved from the heuristics-based methods developed for hopping machines at the *Massachusetts Institute of Technology (MIT) Leg Laboratory* led by RAIBERT [355].

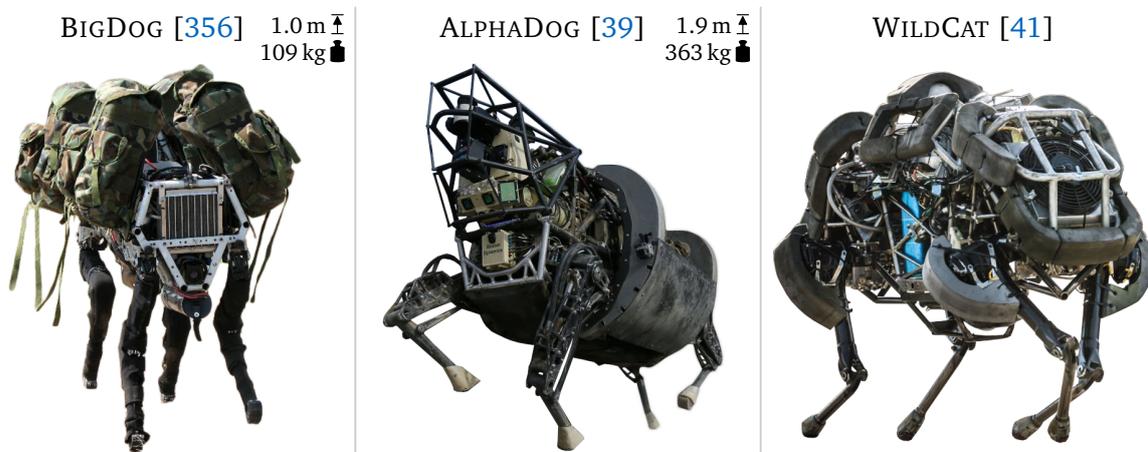


Figure 2.2: Robust, hydraulically actuated quadrupeds developed by *Boston Dynamics* as prototypes for potential military applications. From left to right: BIGDOG, ALPHADOG, and WILDCAT. Images modified from [90].

Further improvements in regard of locomotion speed were made with CHEETAH [40] (tethered, 2D, up to 45 km/h) and its successor WILDCAT [41] (untethered, 3D, up to 25 km/h – Figure 2.2, right) both by *Boston Dynamics*. Especially the latter seems to be very robust, however, energy efficiency at high running speeds is assumed to be poor. Concerning locomotion efficiency of untethered quadrupeds, the MIT CHEETAH series achieved impressive results with minimum *Costs of Transport (CoT)*¹ of 0.5@21.6 km/h for the first [337], 0.47@14.4 km/h for the second [338], and 0.45@4.9 km/h for the third [84] generation of the robot (Figure 2.3, left). In contrast to the aforementioned examples, the MIT CHEETAH series uses electromechanical, high-torque actuators featuring planetary gears with low transmission ratio (“almost” direct-drives), resulting in low friction losses and good impact mitigation properties which is important for locomotion at high speeds and jumping motions [337, 338].

Even greater robustness against impacts can be achieved with *Series Elastic Actuators (SEAs)*², which have been used for example by HUTTER et al. for STARLETH [215] (Figure 2.3, center) and its successor ANYMAL [216] at the *Swiss Federal Institute of Technology in Zürich (ETH)*. Especially the latter was designed for applications in harsh environments, such as search and

¹The *Cost of Transport (CoT)* as defined by TUCKER [415] is a measure for the efficiency of locomotion and is calculated by the power consumption divided by the product of mass, the gravitational acceleration, and the velocity. For comparison: a human of 70 kg weight has a minimum CoT of 0.376@6.3 km/h and 0.467@12.6 km/h [415].

²A classical *Series Elastic Actuator (SEA)* as described by PRATT and WILLIAMSON [348] consists of a motor, a gear, and a passive stiffness (e. g. a spring) which are chained in series to obtain inherent compliance. This makes the actuator very shock resistant and allows direct torque control since it turns the torque control problem into a position control problem. Unfortunately, since the stiffness acts as a low-pass filter, it also limits control bandwidth.[348]

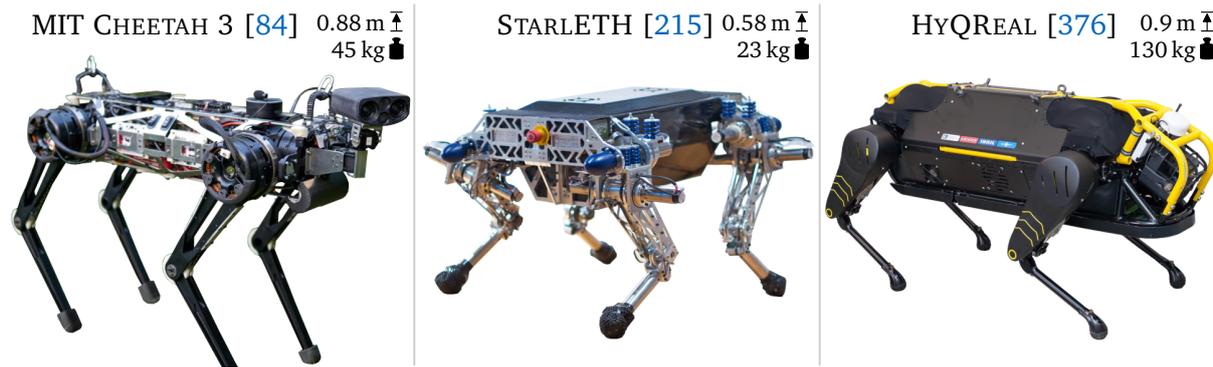


Figure 2.3: Quadruped robots with different principles of actuation. From left to right: MIT CHEETAH 3 (electric, “almost” direct-drives), STARLETH (electric, SEAs), and HYQREAL (hydraulic). Images modified from [84, 215, 376].

rescue as well as oil and gas site inspections. Through numerous demonstrations, ANYMAL showed to be very robust and versatile and thus, represents a large step towards real-world applications. However, with a CoT of 1.2@2.9 km/h [216], it cannot achieve the same power efficiency as the MIT CHEETAH series.

A combination of hydraulic and electric actuators was realized with the quadruped HYQ [374] by SEMINI et al., where brushless *Direct Current (DC)* motors for the less heavily loaded hip abduction/adduction joints allowed to use a smaller hydraulic pump and thus, reduce the overall weight. For its successor, HYQ2MAX [375], all active *Degrees of Freedom (DoF)* are hydraulically driven. In contrast to its predecessor, it also does not feature a passive (spring-based) DoF in the “ankle” for shock absorbance. Note that the same can be observed for the evolution from BIGDOG to ALPHADOG and WILDCAT. Compared to other quadrupeds of similar size, HYQ and HYQ2MAX are rather heavy with a total mass of more than 80 kg while still being tethered. This holds also true for the untethered, high-capacity variant HYQREAL (130 kg – Figure 2.3, right) which demonstrated its strength by pulling a small airplane [376]. HYQREAL represents the most recent generation of this series and the way it was presented suggests that we may see a commercial variant in the near future.

As the given examples show, the development of quadruped robots has made tremendous progress in the last years. Indeed, we observe a transition from pure research prototypes to real, commercially sold products. Prominent examples are SPOT [184] (Figure 2.4, left) as the most recent electrically actuated quadruped by *Boston Dynamics* and the enhanced, commercial variant of ANYMAL [44] (Figure 2.4, center) by the ETH spin-off *ANYbotics*. Additional to these high-end systems, which are mainly targeting industrial applications, also much cheaper (but also smaller) robots, e. g. GO1 [47] (Figure 2.4, right) from *Unitree Robotics*, emerged, which bring robust and versatile quadrupeds also into the consumer market.

Although quadruped and biped robots share similar hard- and software concepts in most aspects, especially humanoids are – at the time of writing – not yet ready for real-world deployment. The inherent lower stability of bipeds when compared to tetrapods, the higher count of active joints, and the greater total size are only examples for additional challenges which put life-sized humanoid robots to another level of complexity. Apart from historic mechanisms and automatons mimicking certain types of human motion, first dedicated investigations on full-sized humanoid robots started with WABOT-1 [245] in 1973 at the *Waseda University*, Japan. Ever since, impressive progress has been made with prototypes developed by various research groups distributed all over the world. So far, a huge variety of biped robots emerged, which differ in their size, count of DoF, actuation principles, control strategies, and hence capabilities and potential future applications.

One way to structure this diversity with regard to hardware aspects, is to cluster biped robots by their level of versatility, where a higher versatility in general also comes with a higher com-

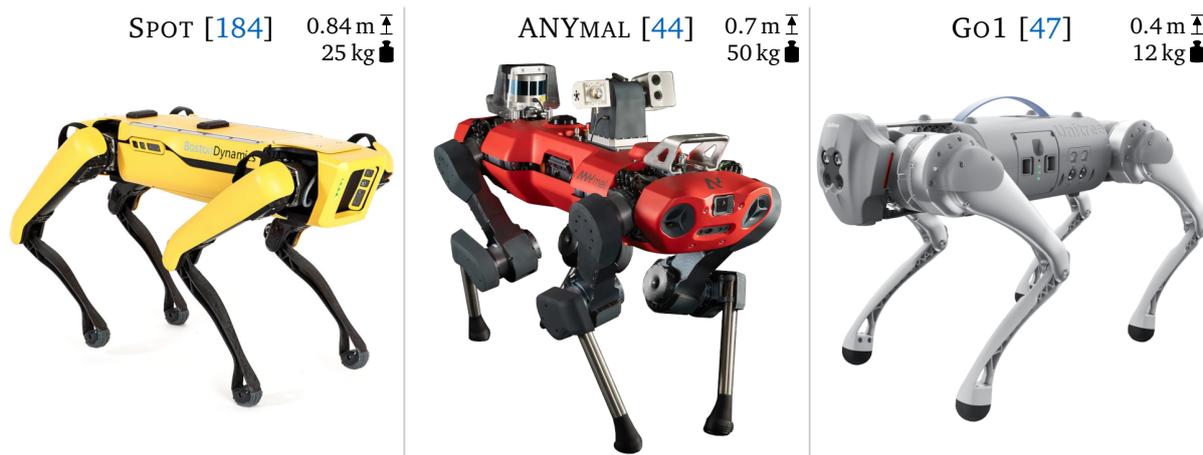


Figure 2.4: Commercial quadruped robots. From left to right: SPOT (formerly SPOTMINI) by *Boston Dynamics*, ANYMAL (generation “C”) by *ANYbotics*, and GO1 by *Unitree Robotics*. Images modified from [44, 92, 418].

plexity and power consumption. The first group, so-called *passive-dynamic*³ *walkers*, does not rely on any integrated actuation. Instead, they are typically driven by the potential energy released from descending a shallow slope. Early analysis of such mechanisms was done in the late 1980s by MCGEER [304], which triggered the investigation of more advanced systems, such as the first 3D passive-dynamic walker by COLLINS et al. [119]. One might extend the definition of this first group to also include *quasi passive-dynamic walkers*, which introduce a minor internal actuation while the passive dynamics still dominate the motion. These are able to walk also on level ground, nevertheless, they are still restricted to a single locomotion pattern based on a limit cycle, thus representing the lowest level of versatility. The main motivation for such systems is their simplicity and typical low CoT, e. g. 0.2@1.44 km/h for the CORNELL BIPED [120].

If we further increase the level of actuation such that active and passive elements have about equal contribution to the total system dynamics, we enter the field of *semi-passive walkers*. A prominent example is given by ATRIAS [213] (Figure 2.5, left), developed at the *Oregon State University*, USA, which set a milestone with regard to versatility for this class of robots, since it was one of the first to achieve untethered 3D motion. It features two series-elastic legs based on fiberglass plate springs coupled with a four-bar mechanism in the sagittal plane for mechanical compliance and to cyclically store and release energy. The robot embodies a realization of the well-known *Spring-Loaded Inverted Pendulum (SLIP)*⁴ model and achieves a CoT of 1.13@3.06 km/h while being robust enough to handle difficult terrains without prior knowledge of the environment [213].

The experience gained with ATRIAS was used to develop CASSIE [42] (Figure 2.5, center), which was commercialized by the spin-off *Agility Robotics*. The step from ATRIAS to CASSIE was huge, leading to a very robust biped system relying on proprioceptive feedback only. Apart from the many optimizations required to move from a research prototype to a commercial product, a prominent change is the different leg kinematics (“ostrich” alike), which allows CASSIE to be even more efficient while using smaller motors [42]. Since CASSIE – like ATRIAS – basically consists of two legs connected by a pelvis assembly, the real-world use cases are rather limited.

³In this context, the term *dynamic* indicates that the corresponding mechanism does not maintain static equilibrium throughout its motion [304]. Indeed, some passive-dynamic walkers are also not stable at standstill.

⁴The *Spring-Loaded Inverted Pendulum (SLIP)* locomotion model consists of a single point mass representing the *Center of Mass (CoM)* which is connected to a massless spring representing the current stance leg during contact, see Figure 2.17. It was derived (among others) by BLICKHAN [86] from an analysis of human and animal hopping and running and became very common in the field of biomechanics. It approximates the real dynamics best for frequencies above 2 Hz while being less representative for slower locomotion [86]. The pogo-stick-like hopping machines by RAIBERT [355] can be seen as an early adoption of a similar concept in the field of robotics. See [168] for an overview of the origins of the SLIP model and its usage in robotics.

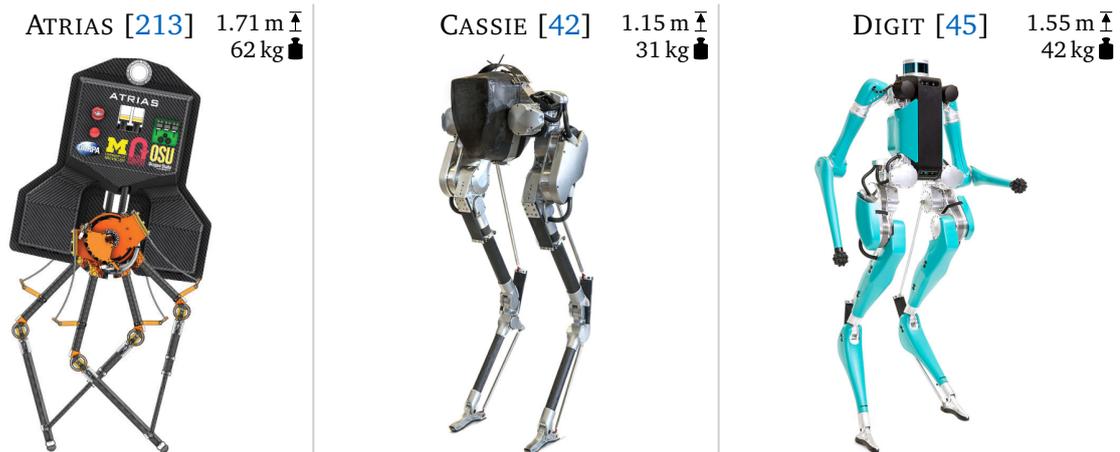


Figure 2.5: Evolution of semi-passive biped walkers from research prototypes to commercial products. From left to right: ATRIAS by the *Oregon State University* and its successors CASSIE and DIGIT by the spin-off *Agility Robotics*. Images modified from [51, 52, 213].

Thus, *Agility Robotics* developed the humanoid DIGIT [45] (Figure 2.5, right), which represents an extension of CASSIE by an upper body integrating multiple depth cameras, a *Light Detection and Ranging (LiDAR)* sensor and (rather rudimentary) arms for simple manipulation tasks. The target markets for DIGIT are promoted as warehouse logistics and the last mile in parcel delivery. However, real-world usability in these fields remains to be shown.

Finally, the last group represents *fully-actuated walkers*, which sacrifice simplicity and energy efficiency in exchange for highest versatility in locomotion and manipulation. Prime examples are the humanoids developed by high-tech companies such as ASIMO [367] (Figure 2.6, left) by *Honda*, PARTNER [402] by *Toyota*, and ATLAS [321] by *Boston Dynamics*. The first two examples are electrically driven and demonstrated unmatched smoothness in biped walking and running on flat ground. At their time, they gave a vision of how humanoid service robots might look in a perfect world, i. e., without large disturbances and uncertainties. In contrast, ATLAS has hydraulic actuators and currently sets the bar in terms of robustness while performing highly dynamic and even acrobatic motions. A serious problem with (untethered) hydraulic robots is their excessive weight: the variant *Atlas-Unplugged* which was designed for the *DARPA Robotics Challenge (DRC)*⁵ finals in 2015 had a weight of 182 kg [321]. Through elaborate lightweight design, the mass could be reduced to only 80 kg for the most recent generation [91] (see Figure 2.8 right). As a negative side effect, extensive customization in general also significantly increases the production costs. Additionally, ATLAS' great strength in combination with the general dangers of hydraulic systems (especially the high oil pressure) still raises serious safety concerns when thinking of operation in the close vicinity of humans.

Unfortunately, the development of ASIMO and PARTNER has been discontinued. In contrast, ATLAS is still subject to ongoing research. However, it is handled as internal prototype and hence – at least in its most recent form – not accessible to others. Note that *Honda* and *Toyota* still show interest in the field of humanoid robotics through their more recent developments E2-DR [451] (Figure 2.6, center) and T-HR3 [43] (Figure 2.6, right), respectively. While E2-DR is specifically designed for disaster response on industrial sites, the focus of T-HR3 as successor of the classical PARTNER robots seems to lie on teleoperation in domestic environments through a sophisticated exoskeleton, thus, offloading the motion planning problem to a human operator.

In order to provide potent hardware to the research community, the company *PAL Robotics* offers the humanoid platforms REEM-C [332] and its successor TALOS [390] (Figure 2.7, right).

⁵The *DARPA Robotics Challenge (DRC)* was an international competition, where semi-autonomous robots had to perform “complex tasks in dangerous, degraded, human-engineered environments” [124]. Throughout the event it became clear that there is still plenty of work left when it comes to reliability of hard- and software [70].

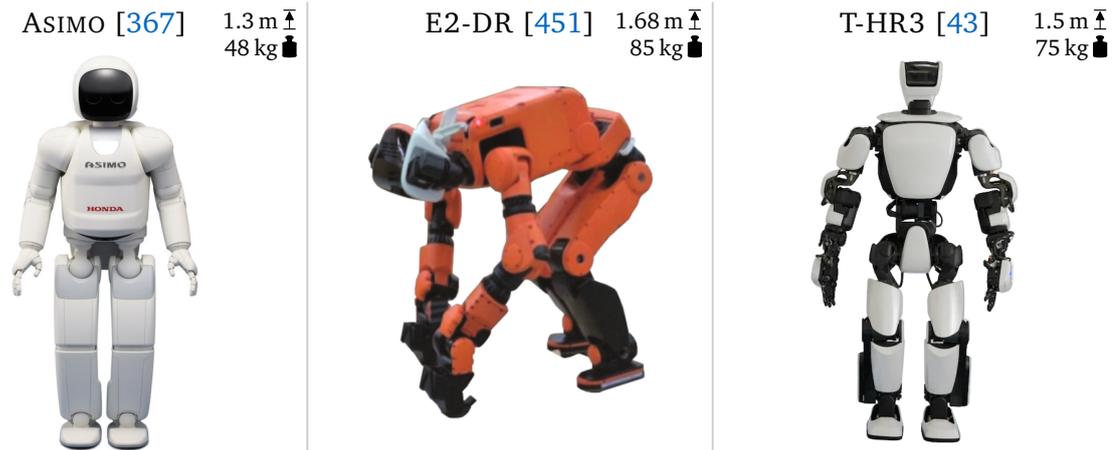


Figure 2.6: High-end research prototypes of fully-actuated humanoids by automotive companies. From left to right: ASIMO (domestic service) and E2-DR (disaster response – shown in transition from biped to quadruped mode) by *Honda* and T-HR3 (without human teleoperator) by *Toyota*. Images modified from [43, 207, 451].

TALOS is electrically actuated, has torque sensors (except for the DoF in the head and wrists), and handles payloads of up to 6 kg with out-stretched arms [390]. Up to now, only rather slow walking motions have been demonstrated with it. Another prominent humanoid platform is the HRP series developed by the *National Institute of Advanced Industrial Science and Technology (AIST)*, Japan and *Kawada Industries*, which started with the famous HRP-2 [236] in 2004, see Figure 2.7 left. Over the years, various new variants have been presented, such as HRP-3 [237] with focus on dust and spray protection for outdoor applications, the compact and slender HRP-4C “Cybernetic Human” [238] designed for the entertainment industry (especially exhibition and fashion shows), the lightweight and low powered HRP-4 [239] (based on HRP-4C) targeting low cost and safety in the vicinity of humans, and HRP-2KAI [240] as a complete overhaul of HRP-2 to make it ready for disaster response tasks as evaluated in the DRC. Finally, their newest prototype HRP-5P (Figure 2.8, left) represents a heavy-duty variant meant for assembly tasks at construction sites, aircraft facilities, and shipyards [241]. With 101 kg it is also the heaviest of the series which puts it between JAXON (Figure 2.8, center) from the *University of Tokyo*, Japan (127 kg, [255]) and TALOS (95 kg, [390]). Although JAXON has a lower mean joint power to weight ratio when compared to HRP-5P [241], it also fits well into the category of “heavy duty humanoids” due to its protective armor and special shock-absorbing structures, see [255] for details.

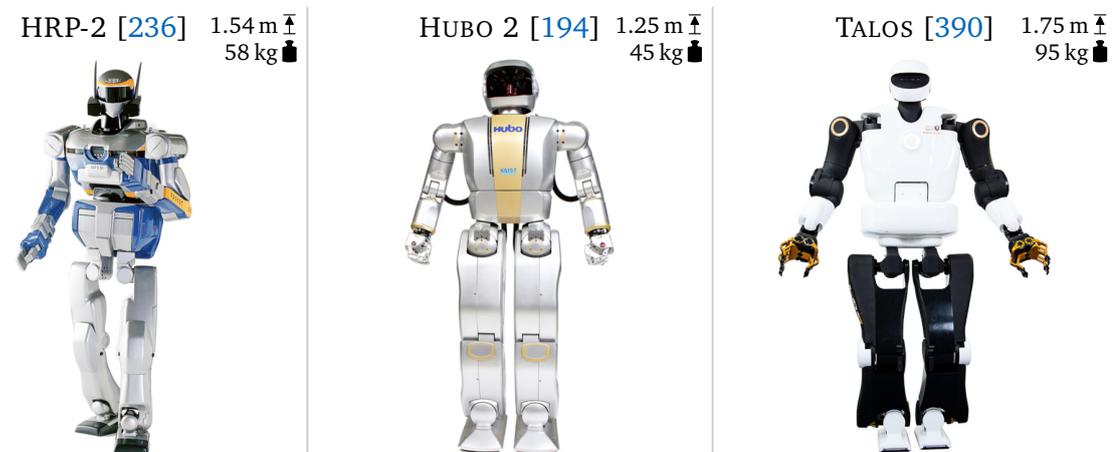


Figure 2.7: Fully-actuated humanoid robots following conventional design approaches. From left to right: HRP-2 by *AIST* and *Kawada Industries*, HUBO 2 by *KAIST*, and TALOS by *PAL Robotics*. Images modified from [55, 228, 333].

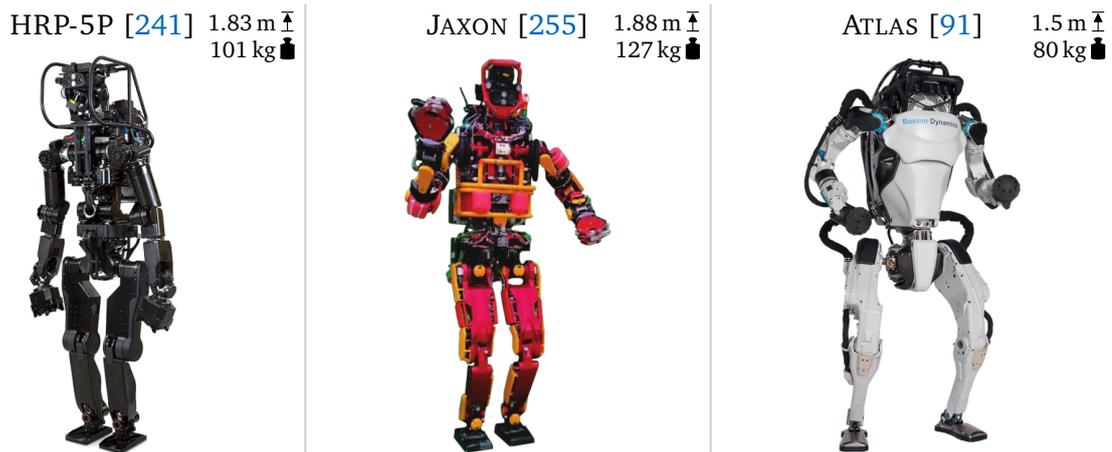


Figure 2.8: High-strength humanoids for heavy-duty manipulation and acrobatic locomotion. From left to right: HRP-5P (electric) by AIST and *Kawada Industries*, JAXON (electric) by the *University of Tokyo*, and the most recent version of ATLAS (hydraulic) by *Boston Dynamics*. Images modified from [56, 91, 255].

Apart from (more or less) shared humanoid platforms, there are numerous unique, in-house developments. Examples are HUBO 2 (Figure 2.7, center) by the *Korean Advanced Institute of Science and Technology (KAIST)* as the most recent version of their KHR/HUBO series [194], TORO [143] (Figure 2.9, left) by the *German Aerospace Center (DLR)*, WALK-MAN [414] as full-sized successor of COMAN [413] by the *Italian Institute of Technology (IIT)*, or VALKYRIE [354] (Figure 2.9, center) by the *National Aeronautics and Space Administration (NASA)*, USA as biped successor of their ROBONAUT 2 [134] space robot. While the HUBO series relies on a conventional hardware design and control strategy, TORO has limbs which are based on DLR's lightweight LWR III [205] robotic arm featuring strain-gauge-based torque sensors in its joints. TORO is used mainly to investigate locomotion and manipulation methods based on torque-control rather than the classical position-control [193]. In contrast to the stiff mechanical design of HUBO and TORO, WALK-MAN and VALKYRIE implement SEAs where the main intention is passive compliance. With 132 kg for WALK-MAN [414] and 129 kg for VALKYRIE [354] they are rather heavy when compared to systems of similar size and power. Nevertheless, their integration of SEAs is an interesting approach, especially for safety-critical applications such as human-machine interaction. Up to now, only rather slow, quasi-static locomotion has been demonstrated with the SEA-based humanoids WALK-MAN and VALKYRIE.

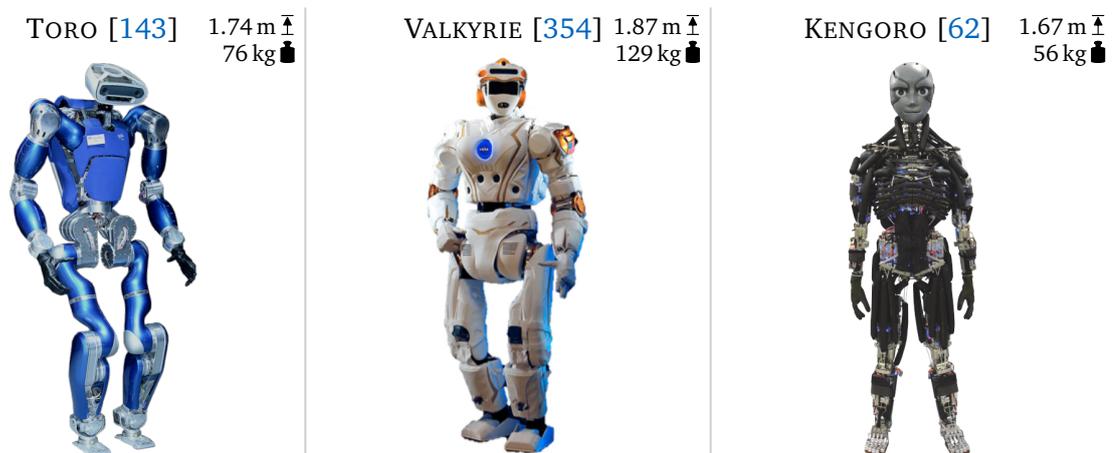


Figure 2.9: Full-sized humanoid robots following non-conventional design approaches. From left to right: TORO (based on LWR III robotic arms) by DLR, VALKYRIE (incorporates SEAs) by NASA, and KENGO (musculoskeletal humanoid) by the *University of Tokyo*. Images modified from [62, 137, 354].

All robots mentioned so far share an (in most parts) conventional joint design. A different approach is made with so-called *musculoskeletal* humanoid robots, such as KOJIRO [308] and its most recent successor KENGO [62] (Figure 2.9, right) by the *University of Tokyo*. Instead of a single actuator – or at the most a pair of actuators – to drive a single rotational or prismatic DoF, a set of artificial muscles allows to realize highly complex joints, such as the human-like shoulder mechanism of BLADE [382] (a pre-study to KOJIRO). Typically, the high count of muscles realizes not only antagonism but also redundancy, thus, a malfunction of a single element may be compensated by its neighbors. The electromechanic, tendon-based muscle design has been published by ASANO et al. in [61] and is used for KENGO to realize the excessive count of 114 DoF (without hands) while keeping the total weight of the 1.67 m tall humanoid below 56 kg. The low mass is achieved by a lightweight design of the skeletal structures using *Carbon Fiber-Reinforced Polymers (CFRPs)* and 3D printed aluminum components. The latter integrate a pipe system for water cooling which in turn also allows artificial perspiration, i. e., “sweating”, through special porous regions [62]. To date, the motion of musculoskeletal humanoids has always looked somewhat “wobbly and clumsy” in demonstrations. However, due to the similarity to human anatomy, they promise very flexible and dexterous motions once this technology becomes more mature. Finally, it is worth noting that there are also hydraulically and pneumatically driven artificial muscles. Unfortunately, they typically suffer from high response times and modeling difficulties, especially for braided artificial muscles [411].

What has not been mentioned so far is the group of non-legged humanoids, i. e., robots with a full-scale anthropomorphic upper body, but a lower body which is realized through a static socket or a wheeled platform. Examples are the research prototypes JUSTIN [331] (with wheeled platform called ROLLIN’ JUSTIN [161] - Figure 2.10, left) and DAVID [177] by the DLR, but also the ARMAR series⁶, including the prominent ARMAR-III [66] and its newest successor ARMAR-6 [68] (Figure 2.10, center) by the *Karlsruhe Institute of Technology (KIT)*, Germany. The restriction to the upper body circumvents the complexity of legged locomotion and instead allows to put full focus on manipulation capabilities. While the natural motivation of the ARMAR series is to develop domestic service robots helping in the household, there is also high interest in non-legged humanoids as smart guides in public places. Here the arms and hands are typically used for social interaction rather than manipulation, such as for the well-known PEPPER [334] robot (Figure 2.10, right) by *SoftBank / Aldebaran Robotics*.

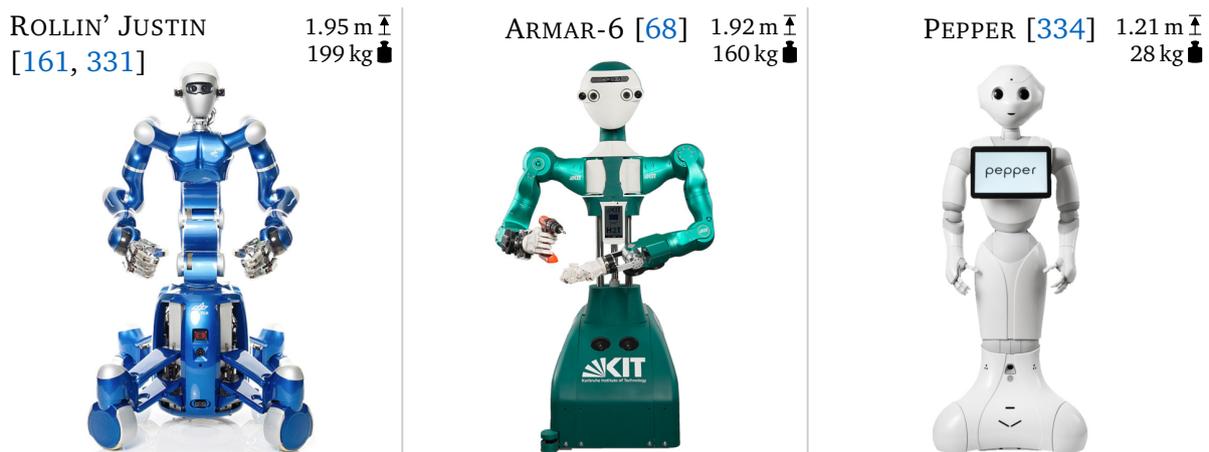


Figure 2.10: Non-legged humanoid robots designed for manipulation and human-machine interaction. From left to right: ROLLIN’ JUSTIN (dexterous manipulation) by DLR, ARMAR-6 (collaborative manipulation) by KIT, and PEPPER (social interaction) by *SoftBank / Aldebaran Robotics*. Images modified from [136, 253, 384].

⁶With ARMAR-4 [67], this series also has a legged variant (making it a “full” humanoid). Although walking controllers have been proposed and evaluated through simulations, cf. [172], to the author’s best knowledge, no real-world walking has been demonstrated yet.

Recently, rather exotic hardware designs also emerged which allow the transition between different locomotion modes. An example is the hominid robot CHARLIE [257] (Figure 2.11, left) by the *German Research Center for Artificial Intelligence*, which is based on the morphology of chimpanzees and can switch between quadruped and biped gait. Another approach is to drop the strict analogy to human or animal topology and instead allow the robot to transform itself into a different locomotion mode. This has been successfully realized with DRC-HUBO+ [223] (Figure 2.11, center) by the KAIST, which allowed the humanoid to switch to a wheeled mode in a kneeling pose. The effectiveness of this approach has been demonstrated at the DRC finals, where the team KAIST took the first place [223]. Instead of auxiliary wheels attached to the knees such as for DRC-HUBO+, a more extreme approach is to replace the feet of a legged robot by (active) wheels. For bipeds this has been realized with HANDLE [183] by *Boston Dynamics* and for quadrupeds with a modification of ANYMAL, cf. [81] (Figure 2.11, right). By combining the advantages of legged and wheeled systems, both demonstrated excellent locomotion skills in various real-world scenarios. While the control of HANDLE seems to rely on classical strategies for stabilizing an inverted pendulum, the modification of ANYMAL, as explained by BJELONIC et al. in [81], includes a novel holistic *Model Predictive Control (MPC)* approach for wheel-legged robots which allows complex hybrid maneuvers. Exploiting the natural efficiency of wheeled locomotion, they achieve a CoT of 0.1@7.2 km/h for pure driving and 0.3@3.6 km/h for a hybrid trotting gait [81]. BJELONIC et al. promote the last mile in parcel delivery as potential future application for wheeled quadrupeds, which makes this modification of ANYMAL a direct competitor of DIGIT.



Figure 2.11: Alternative hardware designs of legged robots allowing different locomotion patterns. From left to right: CHARLIE (transition between quadruped and biped), DRC-HUBO+ (transition between biped and wheeled), and *wheeled* ANYMAL (hybrid wheeled/legged locomotion). Images modified from [133, 229] and [82 @t=34s].

It remains open to classify the mentioned systems, in particular the subclass of legged humanoids, according to their multi-contact capabilities. Obviously, as soon as actuated arms are available, they may be used to establish supporting contacts during locomotion – at least for less demanding, slow motions. However, if we consider moderate or even high walking speeds, multi-contact locomotion sets much higher requirements on the mechanical design of the upper body. To the author’s best knowledge, the torso and the arms of most presented humanoids have been designed primarily with manipulation in mind. Unfortunately, versatility and dexterity in manipulation goes hand in hand with complexity, (unintended) structural flexibility, and weight. This seems to be one reason why real-world experiments of multi-contact locomotion are often restricted to rather slow, quasi-static movements. An exception is the newest generation of ATLAS, which has proven that its hardware is well-suited for highly dynamic motions. Except for some early experiments with its predecessor ATLASPROTO, cf. [321], multi-contact locomotion does not seem to lie in the focus of *Boston Dynamics* (yet).

Hardware Design A natural design goal for mobile robots is to keep the total weight low. Especially for legged systems, an additional directive is to shift the mass from the limbs to the torso. On the one hand, a high torso mass simplifies stabilization due to positive effects of inertia. On the other hand, low mass in the limbs allows faster *End Effector (EE)*⁷ motion, which in turn enables faster overall locomotion. To realize these directives, the presented robots use lightweight structures made of high-strength steel, aluminum, and titanium, but also composite materials and 3D printed elements. Moreover, in some cases limb joints are driven by motors located in or close to the torso. The force is then transmitted through special mechanisms or tendons. Finally, almost all examples implement custom actuators, which further allow to optimize power density, cooling, and sensor integration. In recent prototypes, we see different actuation principles including electric (rigid or as SEA), hydraulic, and pneumatic servos, each of which having their advantages and disadvantages. Although there is no obvious “winner”, it has to be mentioned that commercially available systems are electrically driven only.

Relations to this Thesis In Chapter 3, an upper body design especially targeting multi-contact locomotion is proposed. Although most considerations are applicable to humanoids in general, the presented implementation is tightly linked to the research prototype LOLA [291], which is briefly introduced in Section 2.7. If we apply the same classification as above, LOLA can be categorized as a life-sized, fully-actuated humanoid. Moreover, it is electrically driven, where no SEAs are involved, thus, it can be labeled as conventional, “stiff” robot. With LOLA, autonomous, robust, and fast biped locomotion has been investigated [100, 104, 201, 430, 443]. So far, there have not been concrete plans for manipulation with LOLA, which is why it does not have active hands. In comparison to other systems of similar size and strength, LOLA has a rather lightweight design, mainly due to its aluminum structures with complex geometric shapes and the custom actuator design [291]. However, it has no onboard battery and thus, requires an external power supply, which in turn also prevents an untethered operation. Additional to the power supply, the robot is also linked to a safety harness since it does not feature a protective armor or explicit shock-absorbing structures to prevent damage from falls. Same as for other fully-actuated humanoids, autonomy and versatility are the primary goals. Therefore, the actual CoT of LOLA has not been evaluated yet. In general, energy efficiency considerations are not in the scope of this thesis.

2.2 Software Design

Despite the rapid progress of digitalization, which made complex software omnipresent in our modern society, the term *software design* is still ambiguous and not really tangible. Indeed, the concept of *design* itself has countless meanings, which raises the urge for a universal definition, see for example the one formulated by RALPH and WAND [357]. In the following, a definition similar to the one proposed by FREEMAN and HART in [160] is used, which belongs to the “broad-scoped” [357] type: instead of restricting software design to the high-level phases of architecture conception and interface specification like it is typically defined as part of a formal software engineering process, we also include lower level activities such as implementation and optimization. Moreover, for the purpose of reviewing the state of the art related to legged robots as embedded real-time systems, the definition is further extended to cover also the platform needed to run the software, i. e., the hardware for computation and communication.

⁷In general, the term *End Effector (EE)* denotes the part of a robot which is in physical interaction with its environment to fulfill the robot’s main purpose. Often this represents the *end* of a kinematic chain. Accordingly, in the context of humanoid robotics, this term describes the robot’s feet and hands. It may also include the head since it is the termination of the neck chain and typically carries sensors for visual perception of the environment.

Same as with the hardware of legged robots, there are various different approaches related to the software design. While the core concepts seem to be quite similar, the state of the art still represents a mixture of differences and commonalities, which makes a strict separation into groups difficult. Hence, the following summary tries to extract core properties and give prominent examples for each direction.

Planning vs. Control Within this thesis, a software design for contact planning and motion generation is presented, which in the following is referred to as the task of *Walking Pattern Generation (WPG)* or shorter but less precise: *planning*. Certainly, the WPG represents only a part of the software required to operate a humanoid robot. Another task of at least the same importance is the *control*. Here, the term *control* has to be narrowed down since it is highly ambiguous and can in general denote any piece of software related to the operation of a robot: within the scope of this thesis, the term *control* describes the part of a legged robot's software responsible of executing a given motion while at the same time stabilizing the robot and compensating unforeseen disturbances and modeling inaccuracies.

In contrast to the framework of LOLA where planning and control are clearly separated, the boundaries between these two tasks are often blurred in related work. In some frameworks, especially the ones based on simple heuristics or machine learning relying on proprioceptive sensor information only, a dedicated planning module may not even exist at all. Therefore, the following summarizes the state of the art related to the overall software design of legged robots and is not restricted to the planning task. Additionally, there will be no explicit discussion on multi-contact locomotion. Although it typically comes with higher computational cost and thus, sets higher requirements on an efficient software design, incorporating multi-contact dynamics in general does not require special design patterns.

Architecture: Centralized vs. Distributed Early work on legged robots which used rather simple heuristics for stabilization, like the hopping machines by RAIBERT [355], implemented all routines on a single interface computer. While the processing power might have been less of a concern for these purposes, it becomes more important as soon as the complexity of the hardware (e. g. the count of DoF, sensors, peripherals, etc.) increases or the used algorithms get more elaborate. A paramount example of a highly complex system still operated by a centralized compute unit is the fully-actuated humanoid robot H7 [324] by the *University of Tokyo*. For H7, a central *Personal Computer (PC)*, equipped with dedicated *Input/Output (IO)* boards, directly interfaces sensors and actuators, thus, running the low-level servo-loop together with higher level tasks such as stabilization, motion planning, and even processing of visual input on the same processor [324]. Additional to custom, optimized interface drivers, time-critical modules had to be run in the kernel-space of a *Real-Time Linux (RTLinux)* [74] *Operating System (OS)* in order to maintain real-time performance. Other humanoids of the mid 2000s, such as the HRP-2, started to offload computational expensive visual and audio processing to a separate non-real-time computer [236].

While raw processing power of integrated circuits made incredible progress over the past two decades, also the complexity of recent high-end robots increased significantly. To date, the trend from a centralized to a distributed architecture seems to continue. In fact, most recent systems feature decentralized servo controllers for each joint which allows to increase the cycle time for individual motor control and at the same time reduce the load of the control computer and communication bus. Moreover, low-level control and high-level planning are often split to run on separated boards, see for example the three PCs of the quadruped ANYMAL [216]. The main advantages of such an architecture are the ability to encapsulate functionality and thus, counteract the steady increase of complexity, but also the possibility to cluster modules according to their real-time priorities and in particular their target cycle frequencies. The communication between onboard PCs is typically realized through standard *Ethernet*, while distributed components such

as servo controllers are interfaced mostly through industrial bus systems like *Ethernet for Control Automation Technology (EtherCAT)* [77], *Controller Area Network (CAN)* [107], or *Serial Realtime Communication System (Sercos)* [377] – in rare cases such as for KENGO [62] with *Universal Serial Bus (USB)*. An example for a modern, distributed architecture is shown in Figure 2.12.

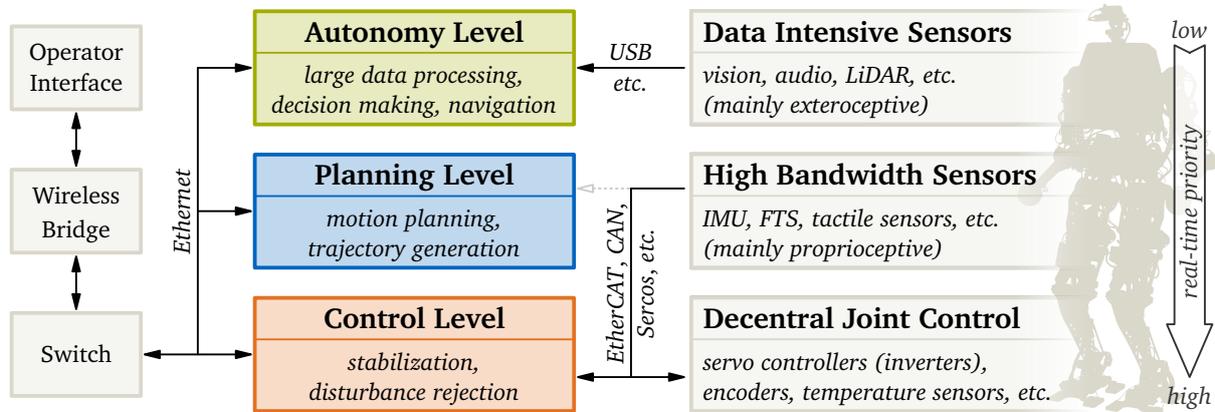


Figure 2.12: Simplified example for a typical distributed software architecture of a modern legged robot. The core functionality of the software is clustered according to the respective real-time priority into an *autonomy* (green), *planning* (blue), and *control* (orange) level, which typically run at different cycle times and on separated PCs. Data intensive sensors are processed by the autonomy level and interfaced through vendor specific protocols. The distributed joint controllers and high bandwidth sensors (most prominently the *Inertial Measurement Unit (IMU)* and *Force-Torque Sensors (FTS)* in humanoid robots) are linked by an industrial (real-time capable) bus.

Computing Hardware In the past decades, the continuous shrinking of structures in semiconductor lithography together with major advances in the architectural design of integrated circuits led to a significant increase of computing performance and power efficiency. Unfortunately, these improvements did not result in a proportional universal speedup of code execution. This is because the trend of raising clock frequencies slowed down due to technological and physical limitations. Instead, chip manufacturers now focus on parallelism, e. g. by increasing the count of *Central Processing Unit (CPU)* cores, in order to achieve the desired performance gains. This makes an adaption of software necessary – in particular the transition from sequential to parallel workflows – which, depending on the actual algorithm, can be difficult or even impossible. Moreover, certain resources such as main memory and cache are shared among CPU cores (or at least groups of cores), which can easily become the bottleneck especially in combination with hard real-time requirements on embedded systems. This represents another motivation for the aforementioned trend to an architecture featuring multiple, distributed PCs, which typically consume more space and power when compared to a single board solution. Additionally, in some cases time-critical and computational expensive workloads get offloaded to distributed *Field Programmable Gate Arrays (FPGAs)*. Examples are the motor driver for the artificial muscles of KENGO [61], distributed control boards interfacing multiple joints and sensors such as in VALKYRIE [354], and the FPGAs used in CHARLIE to realize control loops for complex multi-dimensional joints with local calculation of *Inverse Kinematics (IK)* [257]. Due to the low quantities, custom *Application-Specific Integrated Circuits (ASICs)* are typically not found in legged robots. However, ASICs are often implemented indirectly, e. g. through the use of fully-integrated vision sensors which internally take care of heavy-duty image processing such as tracking, e. g. [219], and depth mapping, e. g. [218]. Additionally, some recent robots such as ARMAR-6 integrate dedicated *Graphics Processing Units (GPUs)* as part of the autonomy level (see Figure 2.12) to accelerate visual processing and neural networks [68].

Real-Time Capabilities Related work proposing novel planning and control techniques for legged locomotion rarely gives details on the actual implementation of algorithms. Although it is reasonable to put emphasis on the proposed method, one has to keep in mind that whenever real-time performance is discussed, supposed details in the implementation might have – together with the available computing hardware – a significant impact. Thus, only a rather shallow comparison of the state of the art with regard to the real-time capabilities is possible. Moreover, the discussion is restricted to the task of (ahead of time) planning since the task of control, as defined earlier, always has to satisfy real-time requirements in order to allow real-world experiments, hence, it is rarely benchmarked. Depending on the application, the planning task might be allowed to take an unlimited amount of time. In such a case, the robot simply starts moving once planning is finished. However, in most real-world scenarios either

- the planning horizon, i. e., the time span for which motion is planned (typically a couple of strides), is too short to reach a certain user defined goal position, or
- a cyclic pattern (e. g. continuous walking) is desired, or
- the user input (e. g. goal position) or the environment (e. g. moving obstacles) changes,

which makes so-called *dynamic replanning* necessary. Thus, there is high interest in making planning as fast as possible. In general, the planning time heavily depends on the complexity of the underlying kinematic and dynamic models: while rather coarse approximations such as the classical *Linear Inverted Pendulum Mode (LIPM)*⁸ [230] introduced by KAJITA and TANI allow fast online methods for WPG [233], incorporating the full multi-body dynamics during planning often requires offline computations such as in [317]. In order to make computations faster, there are efforts to derive analytical formulations of complex kinematic and dynamic models by automated symbolic tools such as the one proposed by HUTTER and GEHRING [214], which has been used for STARLETH and later ANYMAL. However, the resulting equations are used for MPC rather than explicit ahead of time planning. Moreover, considering full multi-body dynamics still seems to be infeasible, which is why the MPC approach proposed for (wheeled) ANYMAL in [81] uses the full robot model for incorporating *kinematics* only. In contrast, *dynamics* are still restricted to the torso body. The same applies to the methods behind the impressive parkour and dance performances of ATLAS, cf. [46]: an *offline* large-scale, non-linear trajectory optimization creates template behaviors which are then executed and adapted by an *online* MPC considering centroidal dynamics only [260]. Finally, if a discrete contact sequence for foot (and hand) placement is not manually defined, but instead has to be computed autonomously based on a model of the robot’s environment, the execution time increases significantly. Especially for complex multi-contact actions, planning time may reach minutes or even hours [151].

Operating System (OS) In general, the choice of the actual OS does not have much impact on the usability or performance of high-level applications such as a *User Interface (UI)*. Contrary, it is crucial when considering hard real-time requirements on embedded systems, e. g. for implementing a low-level interface to the hardware of a robot. For such purposes, a *Real-Time OS (RTOS)* is required, which minimizes kernel overhead and interrupt latency, but also grants extensive access to the scheduling for task prioritization. The majority of the aforementioned robots makes use of a *Linux* based RTOS. While *RTLinux* (e. g. in H7 [324]) seems to be rarely used nowadays, many recent systems, such as TALOS [390], employ the well known *Linux* kernel patch *PREEMPT_RT* [359]. Same as for most other *Linux* based RTOSs, *RTLinux* is based on a cokernel running the *Linux* kernel on top as a fully preemptible process [359]. In contrast, the *PREEMPT_RT* patch directly modifies the *Linux* kernel, which simplifies development

⁸The *Linear Inverted Pendulum Mode (LIPM)* approximates the dynamics of a legged robot by a point mass attached to a massless leg which together form an inverted pendulum, see Figure 2.17. In order to derive linear dynamics, the point mass is constrained to move along a straight line (typically chosen to be horizontal). Note that, in contrast to the “classical” linearized inverted pendulum known from control theory, the LIPM is not restricted to small inclinations (i. e. small strides), since it assumes a mechanism which can actively change the length of the leg [230].

of real-time applications in user-space, but typically also has higher latencies and is less predictable [359]. For other robots, commercial RTOSs such as *QNX Neutrino* [83] by *BlackBerry*, e. g. in *WABIAN-2(R)* [327] by the *Waseda University* (most recent successor of *WABOT-1*), or *VxWorks* [437] by *Wind River Systems*, e. g. in the *Honda* robots *ASIMO* [207] and *E2-DR* [451], are used. In rare cases, a *Windows* derivative is combined with other OSs, such as in *E2-DR* to resolve driver compatibility restrictions of some peripherals mainly related to recognition [451]. Unfortunately, *YOSHIKE* et al. do not specify in [451] the *Windows* variant, i. e., if it is the *General-Purpose OS (GPOS)* for the consumer market or a variant of the independent RTOS sub-family *Windows Embedded Compact*. *FAYYAD-KAZAN* et al. compare the real-time performance of *PREEMPT_RT* (v3.6.6-rt17) with *QNX Neutrino* 6.5 and *Windows Embedded Compact* 7, which shows that the *Linux* patch slowly approaches the performance of its commercial competitors [158]. Since the publication of this study in 2013, all competitors received substantial updates, thus, the results should be treated with caution. A recent study by *PARK* et al. [339] showed, that the scheduling latency of their setup using *PREEMPT_RT* (v4.18.16-rt9) did not exceed 11 μ s in idle and 26 μ s under heavy CPU load.

Middleware On top of an RTOS, a middleware for robotics applications, such as the well-known *Robot Operating System (ROS)* [330], may be used, which provides a software ecosystem for robotics applications and typically ships with a collection of state-of-the-art methods and abstraction layers for common peripherals. Despite its name, *ROS* is not an OS but rather an assembly of libraries which – thanks to recent efforts in the context of *ROS 2* – can significantly improve its real-time capabilities when used on top of an RTOS [339]. Certainly, also a combination of different middlewares is possible, such as for *E2-DR*, which uses a custom extension of the *Robot Technology Middleware (RTM)* [60] for motion control and recognition on the two low-level boards under hard real-time, and additionally *ROS* on the high-level console PC under soft real-time [451].

Remarks and Conclusions Apart from the actual methods for planning and control, the software design of the enclosing locomotion framework has a significant impact on the overall performance and robustness but also the reliability and versatility of a legged robot. Especially the latter two properties are rarely discussed in scientific publications, since they are typically more important for commercial products rather than research prototypes in academia. A general issue within the community seems to be that most research groups have their own individual frameworks, which have grown over the years and have reached an extend that can barely be maintained. Unfortunately, this also makes the exchange of methods and algorithms between frameworks, e. g. for direct comparisons, time-consuming and tedious.

A successful framework requires a harmonic overall concept and components which are matching to each other. Moreover, to achieve impressive results such as those showcased by high-tech companies like *Honda*, *Toyota*, or *Boston Dynamics*, extensive optimization and tuning of the software is necessary. This requires notable manpower, which typically cannot be carried out in academia. Finally, with the continuous progress in performance and skills of legged robots, we also see a steady increase in software complexity. Thus, a good software design becomes more and more important in modern systems.

Relations to this Thesis The software design proposed by this thesis is described in Chapter 4 (main concept) and Chapter 7 (realization). A clear focus is put on the *WPG* module (Chapters 5 and 6), which represents the main contribution of the author with regard to the locomotion system of *LOLA*. In contrast, the surrounding software ecosystem dates back to the various previous researchers working on this project, such as (among others) *LÖFFLER* [289] for the robot *JOHNNIE* and *BUSCHMANN* [100] for the robot *LOLA*.

Referring to the properties discussed above, the software design of LOLA has in large parts a modern, distributed architecture: it features decentral joint controllers and clearly separates modules according to their real-time priority. However, except for the *Computer Vision (CV)* module, all planning and control algorithms run on the same board which may be changed in future to further improve real-time performance. As part of the redesign of the upper body of LOLA (see Chapter 3), the hardware for computation also received a substantial upgrade. For the main workload, two industrial PCs are available. Additionally, a single dedicated GPU attached to one of the PCs allows to accelerate CV algorithms. No dedicated FPGAs are used, although some microcontrollers are present in the lower body – mainly for interfacing sensors.

Previous projects on JOHNNIE and LOLA always had a strong focus on real-time performance. This directive is continued with the methods presented in this thesis, which are based on efficient models of the robot's kinematics and dynamics. For the new WPG module, this results in total execution times for fully-autonomous locomotion of less than 10 ms for motion generation and around 1 s for contact planning in various complex scenarios including multi-contact situations. Compared to the state of the art, this represents a significant improvement. The developed algorithms are implemented on top of a *QNX Neutrino* RTOS. Additionally, a GPOS is deployed on the PC dedicated to the CV module (not in the scope of this thesis).

Due to the strong focus on real-time performance, no robotics middleware is utilized in order to avoid any kind of overhead, which also significantly increases the effort in implementing core functionality. Though, common general purpose libraries, such as *Eigen* [182] for linear algebra and vectorization, are used. All in all, the locomotion framework of LOLA is very generic and versatile: all maneuvers discussed in this thesis can be executed without any modifications to the source code or change of parameters. Thus, different experiments can be conducted subsequently without restart of the system. Moreover, the reliability and robustness of the software of LOLA has proven itself through various public demonstrations with live-experiments. Similar as it is done by other research groups, large parts of the source code used in LOLA are published as free, open-source library.

2.3 Contact Planning

In the context of legged locomotion, the task of *planning* can be split up into *contact planning* (alias *navigation*) and *motion generation* (alias *trajectory generation*). The first computes a discrete contact sequence to reach a certain goal given a model of the robot's environment, see Figure 2.13. The second connects these discrete states by smooth trajectories which can be executed by the subsequent *control* module and finally the hardware. Within the scope of this thesis, contact planning and motion generation together form the WPG module⁹ (see Figure 1.2). Depending on the considered planning approach, there might not be a clear separation between these two parts. Nevertheless, this section tries to summarize the main directions of the state of the art related to contact planning, while the same is done in Section 2.4 for motion generation.

In order to find a discrete contact sequence leading the robot to a user-defined goal while maintaining multiple constraints (such as step length, collisions, etc.), various well-known path planning and search algorithms have been adopted. In general, one might distinct between so-called *continuous* and *discrete* planners. The former typically first generate a continuous path bypassing obstacles (often based on common mobile planners for “car-alike” systems) and afterwards place footholds along this path. Naturally, continuous planners are very efficient and easily satisfy the real-time requirements of legged robots. In contrast, the great majority uses discrete planners, which directly search for an optimal contact sequence and typically exploit

⁹Note that in the literature some authors use the term *Walking Pattern Generation (WPG)* for the motion generation part only, while others (like in this thesis) also include the contact planning part. Since the term *walking pattern* is quite imprecise anyway, the choice between these two definitions seems to be a matter of taste.

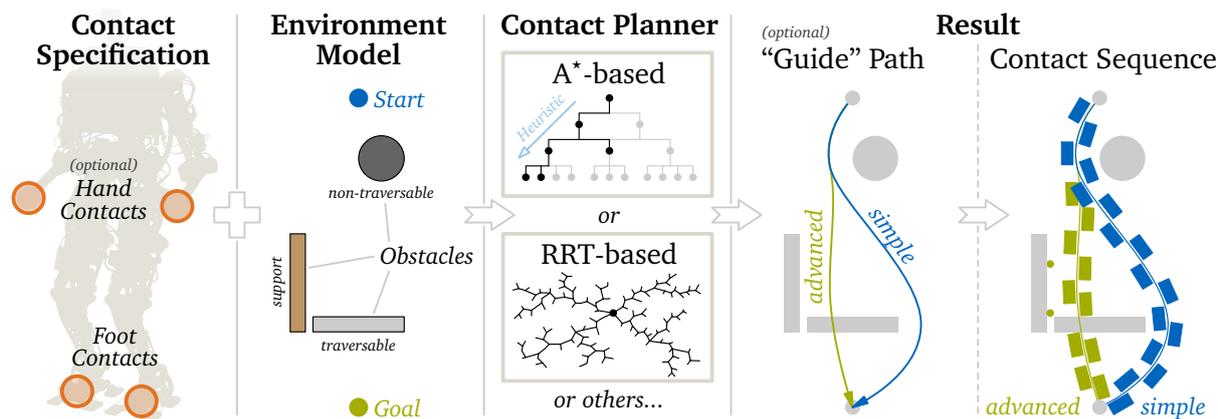


Figure 2.13: Simplified illustration of a typical *contact planning* workflow. From left to right: specification of possible contact areas on the robot; environment model containing traversable and non-traversable obstacles as well as potential support areas for hand contacts; contact planner based on well-known search algorithms or optimization techniques (or combinations of those); resulting contact sequence with (optional) preceding “guide” path for acceleration (simple (blue): planar (“2D”) only; advanced (green): with stepping over, hand contacts, etc.).

the kinematic capabilities of legged robots better than continuous planners, especially when cluttered terrain and stepping over obstacles are considered. As a general drawback, the increase of solution quality also comes with higher computational cost. Therefore, they are often accelerated by a preceding (continuous or discrete) planner creating a “guide” path.

Finally, it has to be mentioned that various frameworks for legged locomotion do not integrate a contact planner at all. These are typically systems where only proprioceptive sensors are used (“blind” walking), cf. ATRIAS and CASSIE. They do not maintain an explicit model of the environment which, however, is a fundamental requirement for a navigation system. Although robust locomotion over rough terrain has been demonstrated, there exist numerous scenarios where locomotion planning benefits from visual perception.

Continuous Planners An example for a *continuous* planner is given by BUSCHMANN et al. in [99] introducing the very first navigation system of the humanoid LOLA. They propose a reactive approach, which does not require an explicit model of the environment and thus, avoids the (at that time) expensive task of *Simultaneous Localization and Mapping (SLAM)*¹⁰. Instead, multiple circular paths (alias “tentacles”) starting from the current position of the robot are used to test the passability of the local surrounding as it is observed by the vision system. Similar to teleoperated walking using a joystick, the robot then simply follows the best “tentacle” by setting the walking direction and speed accordingly. In comparison to other approaches, this method is very simple and fast, however, it is limited to flat floors and scenes of low complexity. Another continuous footstep planner is proposed by DEITS and TEDRAKE in [130]. Their key idea is to look at the problem the other way around: instead of searching for a path through a scene containing non-convex obstacles, they formulate the environment as a set of admissible obstacle-free convex regions which allows them to apply a mixed-integer convex optimization. Drawbacks of this approach are that the automatic extraction of convex obstacle-free regions from a “usual” environment description might be costly and that the restriction to these regions artificially limits the solution space. As last example for continuous planners, KARKOWSKI and BENNEWITZ suggest in [243] to segment the environment into connected planar regions. At first, a global path is planned using a grid-based search, which is used to continuously update the sub-

¹⁰*Simultaneous Localization and Mapping (SLAM)* describes a class of methods in the field of CV, which processes data from vision sensors (e. g. cameras, LiDAR, etc.) and generates a contiguous model of the environment (alias *mapping*). During locomotion, each new viewpoint is used to gradually extend and update the scene. Additionally, the current position and orientation of the robot within its environment is estimated (alias *localization*).

goal of a subsequent local search. The local search computes a path consisting of connected line segments which are iteratively splitted and shifted such that obstacles are bypassed. Footsteps are then placed along this path. Same as for most continuous planners, climbing stairs or stepping over obstacles (which is a key motivation for legged robots) is not covered.

Discrete Planners Representatives of the second group – *discrete* planners – typically are based on well-known search algorithms, most prominently the famous A^* algorithm originally introduced by HART et al. in [190]. A^* is an “informed” graph-based search algorithm which extends DIJKSTRA’s algorithm [135] by a heuristic guiding the node expansion towards the goal. In its original form, A^* is *admissible*, i. e., it is guaranteed to find the optimal path, and *optimal*, i. e., it expands the fewest possible nodes to obtain the optimal path [190]. Indeed, A^* is also used by the contact planner proposed in this thesis – a detailed explanation of this algorithm is given in Section 5.5.3. Seminal work with regard to navigation planning for legged robots especially using A^* is given by CHESTNUTT, e. g. in his dissertation which also includes a good overview of related work until the year 2007 [114, p. 9ff]. He successfully deployed his methods on ASIMO, LITTLEDOG [315] by *Boston Dynamics*, HRP-2, and H7. Other early work based on A^* is the navigation system of QRIO, which uses potential fields imposed on a 2D occupancy grid, where each grid cell holds a probability value for being covered by an obstacle and the time of its last update [364].

Due to its popularity, there exist various extensions and modifications of A^* which are also used in the field of legged locomotion. Examples are *Anytime Repairing A^* (ARA *)* [284] and R^* [285] as used by HORNUNG et al. in [210], *Anytime Dynamic A^* (ADA *)* as used by the same author in [209], and *Anytime Nonparametric A^* (ANA *)* [423] as used by LIN and BERENSON in [286]. The main motivation for these variations is often acceleration (by sacrificing optimality), efficient anytime replanning (providing a non-optimal solution upon interruption e. g. due to a user-specified time limit), or efficient re-use of previous search results. An interesting approach is proposed by LIN and BERENSON, who accelerate the search by a so-called *experience retrieval module*, which maintains a motion plan library by clustering previous motions based on their contact pose similarity [286]. A new motion plan is generated by either planning from scratch (using ANA *) or by retrieving and adapting a similar previous motion from the database, whatever execution branch finishes first. Unfortunately, the performance gain vanishes if the library exceeds a certain size. Finally, there are acceleration techniques which are tailored to the specific application. As an example, KARKOWSKI et al. propose an adaptive node expansion using a reachability map (pre-computed through an IK) for each new foothold based on the previous step [244]. Many other works apply similar approaches to speed up contact planning.

After A^* , the second most common search algorithm used for discrete contact planning are *Rapidly-Exploring Random Trees (RRTs)* introduced by LAVALLE in [269]. In contrast to A^* , RRTs are typically faster but neither guarantee optimality of the solution nor provide a measure for the error bound – at least in their original form. There also exist multiple variations and extensions, such as RRT-Connect [259], where two RRTs are grown simultaneously, one from the start and one from the goal. Later, this concept was referred to as *Bidirectional RRT (Bi-RRT)* [270]. In RRT-Connect, the tree expansion is combined with a greedy heuristic which guides the two trees to each other but also makes the algorithm less prone to local minima [259]. As one of the first adoptions in the field of legged locomotion, OKADA et al. deployed in [328] an extension of RRT-Connect on the humanoid HOAP-1 [251] by *Fujitsu Automation*.

Environment and Collision Model Independent of the chosen search algorithm, there is the need for an appropriate environment model describing the robot’s surroundings. Certainly, a rich and detailed model enables high-quality plans but also causes severe computational cost during pre-processing and the actual search. In many cases, the terrain is described by an occupancy or height map, e. g. obtained from a *Red-Green-Blue Depth (RGB-D)* camera and/or

LiDAR, cf. [326], or a stereo camera system, see Figure 2.14. Apart from 2D or 2.5D grids, also 3D representations such as voxel-based occupancy grids (e. g. in the form of an *OctoMap* as proposed by HORNING et al. in [211]) or volumetric *Signed Distance Fields (SDFs)* (e. g. in [154]) are possible. However, especially the latter is very expensive to compute. Note that visual perception always suffers from occlusion (“shadows”) leading to a sparse, incomplete reconstruction. Thus, methods have been developed to “guess” occluded areas by reasoning about the semantics of the corresponding object (e. g. chair, table, etc.), cf. WU et al. [446].

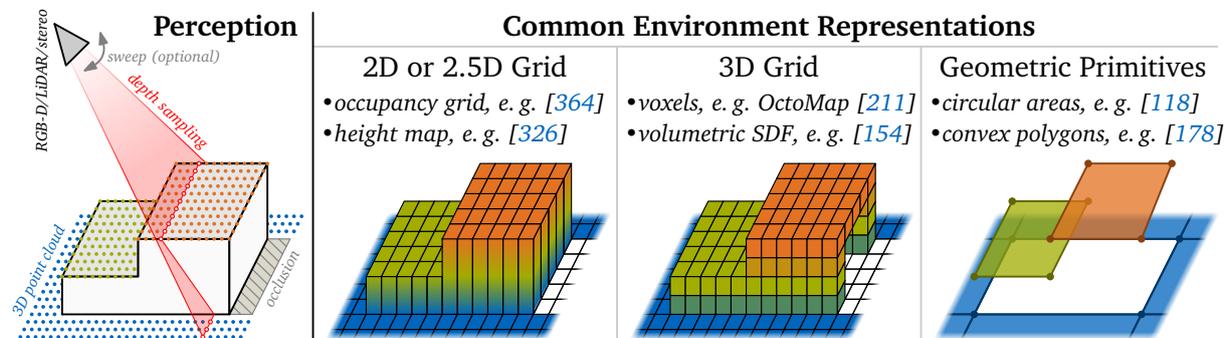


Figure 2.14: Common ways of representing the robot’s environment (in particular the terrain) for the example of a simple staircase. Left: visual perception through an RGB-D, LiDAR (with optional sweep), or stereo camera system to obtain a 3D point cloud. Right: representation as 2D, 2.5D, 3D grid or in the form of geometric primitives.

If a grid representation is too expensive, a common approach is to abstract the terrain by geometric primitives. A recent footstep planner following this idea is presented by GRIFFIN et al. in [178] for DRC ATLAS¹¹ and VALKYRIE. Herein, rough terrain is approximated by planar surfaces which in turn are split into convex regions. Together with the drop of optimality by using a weighted¹² A* implementation, this leads to a comparatively low overall runtime. Another approach proposed by HORNING and BENNEWITZ in [209] is to introduce different levels of detail. The environment is segmented into regions which are classified according to their complexity, i. e., obstacle density. In open spaces, a coarse but fast grid-based planning is used, while more expensive ADA*-based footstep planning is performed in the vicinity of obstacles. The classification of the environment is based on a distance map, which is expensive to compute. Therefore, this method seems to be restricted to static scenes.

A general problem of common 3D sensors is the rather low accuracy of depth measurements which can lead to displacement errors in the range of multiple centimeters [326]. For laser range finders, an additional issue is the typically low image resolution or a restriction to line scans. The latter can be compensated by an explicit sweep motion [326], cf. Figure 2.14. Unfortunately, this also slows down the data acquisition and thus, degrades the model quality for moving objects due to motion blur.

Finally, for avoiding collisions, not only the obstacle representation is relevant, but also its counterpart: the model of the robot. A rather simple way is to approximate the robot by a bounding box which may change its orientation and/or size to obtain a best fit for the actual silhouette as proposed by KUMAGAI et al. in [261]. This way, collision checks become very efficient, especially when lots of obstacles are involved. Certainly, there exist contact planners which make use of a more detailed collision model of the robot, e. g. the one presented in [328], which incorporates a cylinder and convex hull representation of the HOAP-1 robot.

¹¹The most recent version of ATLAS is handled as an “internal” prototype by *Boston Dynamics*. However, previous models – in particular the variant which was provided to various competitors in the DRC (alias DRC ATLAS) – are still available to some research institutes such as the *Florida Institute for Human & Machine Cognition (IHMC)*.

¹²With *weighted A**, variations of the exact A* algorithm [190] are meant, which release some of the original restrictions (typically by multiplying the heuristic by a factor greater than one) and thus, sacrifice optimality in order to accelerate the search while still providing an upper bound on suboptimality – see [139] for details.

Quadrupeds Although motion generation and control for quadrupeds often differs in various aspects when compared to bipeds, contact planning can be quite similar, cf. [114, p. 65ff]. A major difference is that individual foot contacts are less important for quadrupeds when considering stability, however, the constraints (reachability, feasibility and coordination of foot positions) typically are slightly more complex. An example for a recent contact (and motion) planner for quadrupeds, in particular ANYMAL, is given by FANKHAUSER et al. in [154]. Nominal (“default”) footholds are checked for kinematic feasibility and against the terrain which is classified in advance using a binary foothold score based on a height map evaluation. Subsequently, the swing leg motion is planned, where collisions are checked using a volumetric SDF of the environment. Finally, a pose optimization for the robot base (“torso”) is performed by formulating a *Sequential Quadratic Program (SQP)* with analytic gradients and Hessians. The cost function of the SQP pulls the *Ground projection of the CoM (GCoM)* towards the centroid of the *Support Area (SA)* and additionally ensures that joint limits are respected.

Excursus: Support Area (SA), Support Polygon (SP) and Static Equilibrium In the field of legged locomotion, the *Support Area (SA)* alias *support region* or *area of support* is a common entity used to analyze static and dynamic balance. For coplanar contacts, e.g. standing or walking on level ground, it represents a polygon alias *Support Polygon (SP)*, which is defined as the convex hull of all active ground contacts, see Figure 2.15 and [18 @t=4m7s] for an animated explanation. A straightforward method to evaluate the balance of a static pose, e.g. standing, is checking if the *GCoM* lies within the *SP*. In case it lies outside this area, the robot will start to tip over. Moreover, by replacing the *GCoM* with another entity, the *SP* can also be used to check balance during motion (will be explained in Section 2.4). Unfortunately, these criteria for static and dynamic balance are only valid if all active contacts lie within a horizontal plane. For non-coplanar contacts, e.g. for stair climbing or multi-contact locomotion, the *SP* can be generalized to a curved *support region* as the projection of a nonlinear convex set, see Figure 2.15 left and BRETEL and LALL [95] for details. In combination with the *GCoM*, this restores the possibility of quite simple checks for static equilibrium. In contrast to the coplanar case, this generalized form considers individual friction cones at the contact point locations. In [95], the authors additionally present an iterative algorithm, which gradually refines an inscribed (convex) polygon approximating this curved region.

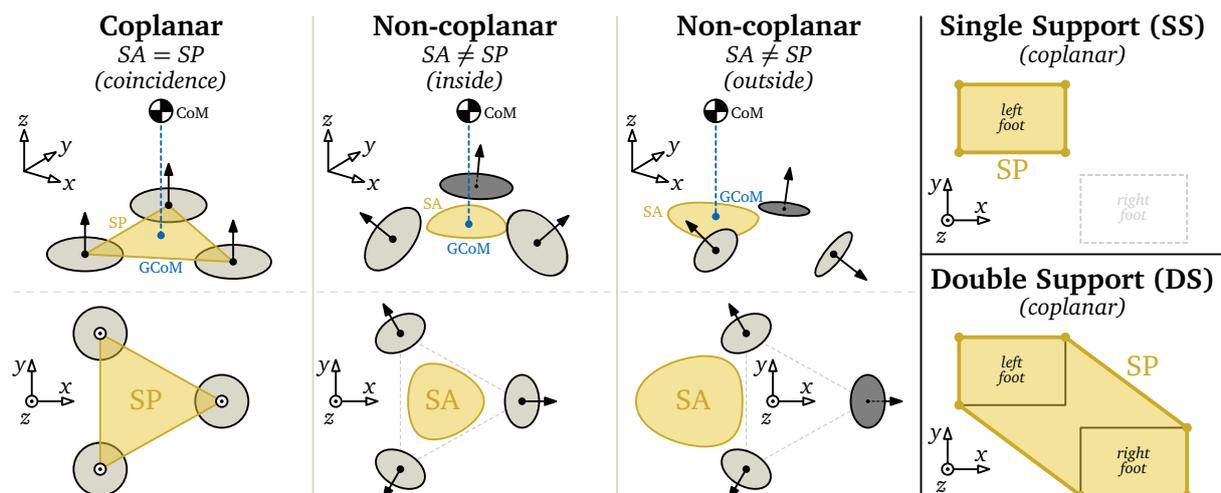


Figure 2.15: Static equilibrium test based on the *GCoM* and the *SA/SP*. Left: shape of the *SA* for coplanar (polygon) and non-coplanar (curved) contacts (modified from [95]). The gray discs represent the individual friction cones at the contact points (centroid) and the arrows the corresponding contact normal. Right: construction of the *SP* during regular biped walking (level ground, rectangular feet) in the *Single Support (SS)* (top) and *Double Support (DS)* (bottom) phase by computing the convex hull of all active ground contacts.

Multi-Contact Most of the contact planners mentioned so far are so-called *footstep planners* for bipeds, i. e., they are restricted to foot contacts. In general, the extension to a “full” multi-contact planner is straightforward but may require some extensions especially if multi-contact stability is to be considered during contact planning. At this point, it has to be mentioned that in some related work the term *multi-contact* has alternative meanings. In [455] for example, ZHAO and SENTIS use it to describe the DS phase of a biped gait where both feet are in contact with the ground. Since their method is based on an inverted pendulum model for which typically an instantaneous stance leg switch is assumed (infinitesimal short DS phase between two SS phases), their wording highlights the explicit investigation of the DS phase. Another example is [264], where two or more contacts with the feet are meant, which requires special attention when investigating *Hybrid Zero Dynamics (HZD)*¹³. Moreover, multi-contact is also used with regard to locomotion of non-biped, legged robots such as the running tripod by THOMAS and SENTIS in [409]. Within this thesis, the term is defined such that it complies with the general understanding in the humanoid robotics community: *multi-contact* indicates explicit *non-feet* contacts during legged locomotion. This includes but is not restricted to additional hand support during biped locomotion.

Pioneering work for multi-contact planning following this definition has been made at the *University of Montpellier*, France and the *Joint Robotics Laboratory (JRL)* of the *French National Centre for Scientific Research (CNRS)* and AIST, which reaches from early work by ESCANDE et al. [147] to recent investigations by MUROOKA et al. [313]. Their work focuses mainly on complex, non-gaited, acyclic scenarios such as sitting down on and standing up from a chair [149], simultaneously holding a glass of water as additional task [150], climbing ladders and crawling through tunnels [151], and alternating or simultaneous object manipulation and locomotion alias *loco-manipulation* [313]. Their framework follows the so-called *contact-before-motion*¹⁴ planning approach, which is in accordance with the separation of contact planning and subsequent motion generation as proposed in this thesis. An overview of the CNRS-AIST JRL framework is given in [94], which also presents common approaches for considering multi-contact stability. The contact planner is based on a best-first graph search in contact space, which is guided by a potential function pulling towards the goal and pushing away from obstacles [151]. Their method is very general and capable of generating highly complex motions with the disadvantage of very high planning times in the range of minutes to several hours [151]. Additionally, actions like crawling require the robot to make contacts not only at the feet and hands, but also at other parts of its hull, which makes the contact planning problem even more complex and costly. In recent work by MUROOKA et al. [313], an ADA*-based graph search is used in a loco-manipulation scenario for footstep and regrasp planning. They propose a special transition model including a feasibility check by a pre-computed (offline) reachability map. This leads to comparatively low online planning times of only 1 s for loco-manipulating a rolling object, despite the highly complex reachability considerations in this scenario [313].

In order to incorporate stability considerations during multi-contact planning, a common approach is to test for static equilibrium only, e. g. as proposed by TONNEAU et al. in [412]. Their contact planner is segmented into two stages. In the first stage, a guide path for the root of the robot is generated using an RRT-based search and a feasibility check by testing contact reachability. In the second stage, a discrete sequence of whole-body configurations is created by

¹³The *Hybrid Zero Dynamics (HZD)* of a biped walker are defined by WESTERVELT et al. as “the largest internal dynamics compatible with the output being identically zero” [433]. Here the term *zero dynamics* is derived from equivalent concepts in control theory, cf. [294, p. 193ff] or [49, p. 319ff]. By *hybrid*, the combination of continuous multi-body dynamics described as *Ordinary Differential Equations (ODEs)* and discrete impacts is meant. The primary goals of the HZD community are the development of controllers for exponentially stable gait cycles and a formal stability analysis. This typically requires the restriction to planar (“2D”) biped walkers where additionally hypotheses are made for the robot (e. g. no closed kinematic chains), the gait (e. g. DS phase is instantaneous allowing discontinuities in velocities) and the impact model (e. g. rigid, no rebound) [433].

¹⁴Originally, HAUSER et al. adopted the idea of *contact-before-motion* planning for humanoids from earlier work by BRETL et al. which was related to free-climbing robots [192].

expanding the root poses from the guide path and enforcing several constraints such as static equilibrium and opening or closing only one contact during each transition. They focus on acyclic motions and achieve computation times of only a few seconds depending on the actual scenario. Note that generating a guide path or some kind of potential to accelerate the actual contact planner is a common approach and can be found in most frameworks of the community.

A justifiable critique of contact-before-motion planning is made by CHUNG and KHATIB in [118] by pointing out that it “might generate unnatural postures” [118] and that most algorithms following this approach “require to manually assign or to uniformly sample the available contact-points in the environment” [118]. In order to relax the strict separation of contact and motion planning while still preserving the advantages of a fast global planner in contact space, they propose a framework based on so-called *contact-consistent elastic strips*. In particular, they extract circular contact regions from a segmented point cloud of the environment and let a global planner compute a feasible sequence of contact regions. The constraints of the subsequent elastic strips algorithm are then formulated in a way such, that contacts can be repositioned on the corresponding circular contact region. This gives the motion planner the opportunity to modify previously planned contact positions – at least to a certain extend.

Finding Contact Candidates When comparing multi-contact to biped locomotion planning, we observe a significant raise of computational cost due to the larger branching factor (more possible contact states) and the more complex stability considerations. Another difficulty is finding feasible support regions in the environment for potential (non-feet) contacts. While a conventional height map is suitable for placing footholds, other approaches are necessary for planning contact with the hands. KUMAGAI et al. propose in [262] to compute the intersection of the reachability volumes of the hands with environmental contact areas represented as convex polygons. As governing framework, they use a similar approach as TONNEAU et al. in [412] by combining an RRT-based global planner (guide path) with a subsequent A*-based footstep planner. Hand contacts are then assumed to be feasible, if they are “geometrically sustainable during one foot step” [262]. In [432], WERNER et al. present a multi-contact framework for TORO, which circumvents this problem by planning footholds only. In contrast, hand contacts have to be specified manually by the user. A rather unconventional approach is presented by KAISER et al. in [227]. Inspired by natural language processing, they create an n-gram model describing transition probabilities between whole-body poses. Herein, poses are equivalent to words while motions are equivalent to sentences [227]. The model is trained by human motion, which has been recorded using an optical tracking system. Additionally, each surface in the environment is assigned a *whole-body affordance*¹⁵, which is used by the planner to derive an optimal contact sequence. Additional to explicitly planned contacts, LIN et al. propose in [287] to consider alternative foot/hand contacts, which can be used to recover from unforeseen disturbances. Due to the great number of potential alternatives, the computational cost increases significantly. In order to accelerate contact planning, dynamic feasibility of zero- and one-step capturability for contact candidates is predicted by a neural network. The network is trained offline with data from a kino-dynamic optimization.

Conclusions The vast majority of contact planners for legged locomotion builds upon well-known search algorithms such as A* and RRTs. Although these core elements differ in their strengths and weaknesses, it seems like the actual choice does not have too much impact on the overall effectiveness of a planner. Instead, it seems to be much more important *how* these algorithms are adapted for the particular use-case, combined to exploit the advantages of each technique, and finally embedded in a surrounding locomotion framework linking perception,

¹⁵*Affordances* describe action possibilities of the robot for locomotion and/or manipulation which are assigned to environmental surfaces and objects [224]. In [225], KAISER et al. differentiate between higher level affordances (e. g. bimanual object handling) which are composed out of lower level affordances (e. g. grasping).

planning, and control. A common trend is to first plan a coarse guiding path, which then is used to accelerate the actual heavy-duty search in contact space. A fair comparison of the proposed contact planners with respect to their runtime performance is considered to be impossible, since the investigated scenarios strongly differ with regard to

- the scene complexity (from flat ground and perfect environment models to previously unknown, cluttered terrain with moving obstacles),
- the robot's skills (from simple biped walking to climbing stairs, stepping over obstacles, and finally complex, acyclic multi-contact loco-manipulations), and
- the considered model of the robot (from a mobile "car-alike" behavior to incorporating whole-body kinematics and dynamics).

Note that some kind of collision avoidance is always involved since this is the main idea of navigation. The same holds true for kinematics, which are always considered, at least in the form of a reachability concept for EE placement. In contrast, dynamics are rarely included by the contact planner and are instead introduced by the subsequent motion generator. For a cheap feasibility check, however, it is common to perform a static equilibrium test for key poses. Since the early works from the beginning of the 2000s, contact planners for legged locomotion have received many improvements. For multi-contact planning, we observe a tremendous progress over the past ten years, especially with regard to the real-time capabilities. Although the steady increase of computational power plays a certain role here, the main reasons for this evolution seem to be methodological improvements such as focusing on dominant effects by introducing simplifications at the right places.

Relations to this Thesis The contact planner proposed in this thesis is described in Chapter 5. Although there has been previous work on navigation planning for LOLA by BUSCHMANN et al. [99] and later HILDEBRANDT et al. [13, 199, 200, 202], the contact planner has been completely redesigned from scratch (based on the existing experience) in order to make it capable of multi-contact planning. We restrict ourselves to *gaited* multi-contact locomotion, which means that we aim at preserving the main characteristics of biped gait. This way, multi-contact situations can be incorporated in an efficient way without combinatorial explosion in our A^* -based graph search. By separating contact planning and motion generation, we followed the contact-before-motion paradigm. Moreover, we adopt the idea of a preceding coarse guide path by formulating a hierarchy of two consecutive A^* searches with different discretization for both, the state space and the environment model. The terrain is represented as a height map, while objects are approximated by triangle meshes representing their surface (for finding potential hand contacts) and *Swept Sphere Volumes (SSVs)*, see Appendix C, representing their volume (for fast distance queries). Compared to related work, our method is very versatile since it is capable of planning contact sequences which involve stepping over traversable and bypassing non-traversable obstacles, stepping up/down platforms, climbing stairs and ramps, and making explicit hand contacts for extra robustness simultaneous to all those actions mentioned before. All of this is done fully autonomously given the mentioned height map and obstacle representation but without any user interaction. Despite this high level of autonomy and versatility, we achieve runtimes of only around 1 s for various complex scenarios which outperforms most other planners. Although the source codes of various other contact planners has been published, the implementation is rarely discussed in the associated papers. Since this thesis has a strong focus on real-time execution with limited onboard computation power, some implementation aspects with great impact on overall planning time are discussed in Chapter 5.

2.4 Motion Generation

Subsequent to the contact planner, the second component of a typical WPG (following the contact-before-motion paradigm) is the *motion generator*. It takes the computed contact sequence as input and connects the discrete states by kinematically and dynamically feasible transitions. In case no contact planner is involved at all, e. g. for “blind” walking, the next couple of contacts may be chosen such that a certain walking speed and direction on flat ground is maintained. As another option, future footholds may be obtained from a dynamics model predicting the motion for a certain time horizon. So far, numerous approaches for motion generation on legged robots have been proposed. They mainly vary in

- the comprehensiveness of the underlying robot and contact model,
- the solution strategy, i. e., the way how the model is used to compute smooth trajectories and simultaneously ensure kinematic and dynamic feasibility,
- the robustness of the planned motion against modeling uncertainties and unforeseen disturbances during real-world execution, and
- the computational complexity, which has direct impact on the real-time performance.

In many cases, the choice of a certain model does not prescribe a specific solution strategy and vice versa. This leads to a large variety of possible combinations and makes a classification of methods difficult. However, some combinations are very common in literature, while others are used not so often. The following tries to summarize the state of the art by clustering the available frameworks into the two most common variants, see Figure 2.16.

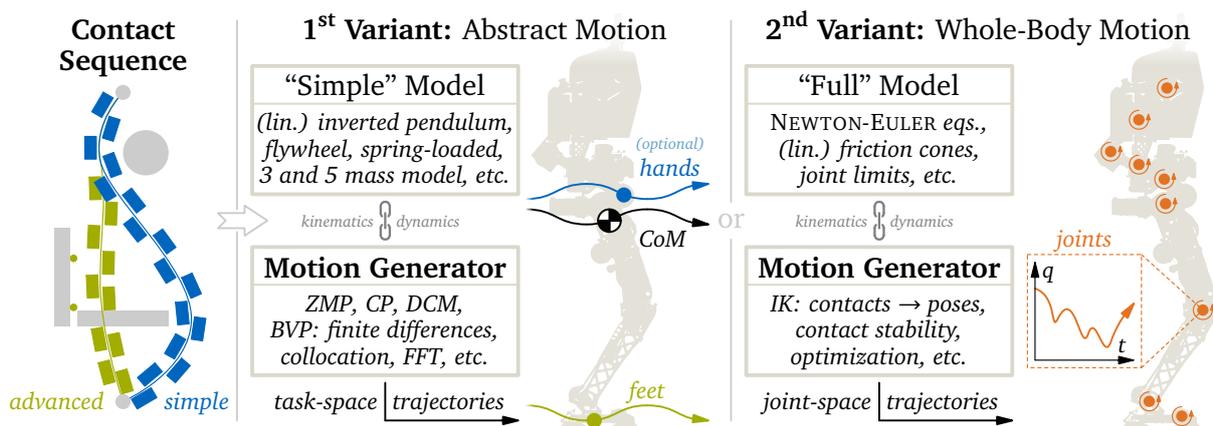


Figure 2.16: Simplified illustration of common workflows for *motion generation*. From left to right: discrete contact sequence as obtained from a contact planner (see Figure 2.13 right) used as input; 1st variant of motion generators linked to a “simple” model of the robot providing kinematics and dynamics for generation of abstract motion in the form of task-space trajectories; 2nd variant of motion generators linked to a “full” model of the robot providing kinematics and dynamics for generation of whole-body motion in the form of joint-space trajectories.

Generation of Abstract Motion – Overview The first and probably most common variant generates an abstract motion represented by *task-space* trajectories, which typically encompass the motion of the EEs and the CoM or alternatively the root body (alias torso). The main focus lies on the CoM motion, which is typically derived from a rather simple model of the robot in combination with an appropriate feasibility concept. In many cases, this is done by formulating a *Boundary Value Problem (BVP)* with the dynamics of the model describing the underlying ODE and the first and last state of the given contact sequence defining the boundaries. Alternatively, the boundaries might be chosen such that a cyclic gait is obtained. The motion of the EEs is typically computed in advance based on heuristics. These ensure suitable contact transitions, e. g. heel-strike and toe-lift-off, and may avoid collisions with the environment, e. g. for stepping

over an obstacle. In contrast, multi-body dynamics are rarely used to optimize the EE motion. The corresponding joint motion is then obtained by a subsequent IK (which may not be part of the motion generator), automatically resolving the typical redundancy of limb kinematics. The advantages of this first class of motion generators are the low computational cost due to the simplicity of the underlying model, but also the robustness of the resulting motion under real-world conditions. As a general drawback, the coarse approximation of the kinematics and dynamics may lead to solutions which are not feasible on the real system, especially for difficult maneuvers such as climbing stairs. As a countermeasure, certain safety margins might be introduced, which, unfortunately, also artificially limit the robot's full potential. In related work, this first class of motion generators is mainly used to plan *gaited* motion.

Generation of Whole-Body Motion – Overview The second variant of motion generators aims at computing an optimal whole-body motion in *joint-space*, i. e., individual trajectories for each joint. It is mainly promoted by the CNRS-AIST JRL group, see for example [280]. Typically, these approaches make use of a comprehensive multi-body model of the robot and derive the *Equations of Motion (EoM)* by setting up the NEWTON-EULER equations for the full topology, i. e., all bodies. Since the full configuration of the robot is available at any time, individual joint limits and per-segment collisions can be considered (in contrast to a task-space formulation where only EE collisions can be checked). A typical first step is to compute a discrete sequence of whole-body configurations from the given contact sequence using an IK method combined with a check for static equilibrium (see Section 2.3). To obtain feasible transitions between these configurations, a comprehensive optimization problem is formulated, which considers kinematics, joint limits (position, torque, etc.), contact stability, and collision avoidance. The advantages of this approach are that rather complex maneuvers can be handled and that the planned motion is very likely to be executable on the real system since many kinematic and dynamic effects and hardware constraints are considered in advance. Unfortunately, this comes with an increased computational cost, which often makes online planning infeasible. Moreover, a very detailed model may also decrease overall robustness in real-world experiments due to inevitable inaccuracies and unforeseen disturbances which can represent significant deviations from the model. In related work, this second variant of motion generators is mainly used to plan complex, *non-gaited* motion.

Stability vs. Feasibility Before having a closer look on related work, the herein commonly used terms *stability* and *feasibility* have to be clarified. In the context of legged locomotion, *stability* typically means that the robot does not tip over and fall. Note that the robot might deviate from the original planned motion but still remain stable as long as a lower level control module *stabilizes* the system by compensating errors and disturbances. Theoretically, even for complex, high-DoF robots, a formal stability analysis for a certain motion is possible by considering a sufficiently comprehensive multi-body model. However, even the best model only represents an approximation of the real world and cannot consider all effects. Especially for legged locomotion, the repeatedly opening and closing contacts introduce highly complex dynamics which are difficult to model. Indeed, there is no way to guarantee that the planned motion leads to a stable behavior in a real-world experiment. Therefore, it is common practice to use the term *feasibility* instead, which describes that a planned motion is theoretically doable under certain assumptions. Depending on the underlying model, these assumptions may include for example sufficient contact friction, sufficient joint torque, and – most important and independent of the model – that there are no (unknown) external disturbances. Using this definition, the goal of a motion generator is to plan a *feasible* motion. In combination with a lower level control module supervising the actual motion, this hopefully leads to a *stable* real-world execution.

Static Feasibility In related work, the feasibility of a static robot pose is almost exclusively evaluated by checking static equilibrium (already introduced in Section 2.3 together with the concept of the SA). Although static equilibrium is a necessary condition for static feasibility, it is not sufficient: in addition, the joints of the robot have to be strong enough to keep the robot in the specified pose. This becomes relevant for extreme configurations such as extra long steps or standing on tiptoes. Obviously, this type of constraints can only be checked reliably by whole-body approaches which have the necessary joint-level information. Static feasibility checks are mainly used during contact planning, but are sometimes also used within motion generation, especially for multi-contact frameworks where an evaluation of dynamic balance would be much more complex. Moreover, for very slow, quasi-static motions, testing only for static feasibility seems to be a valid simplification.

Robot Model Especially for motion generators using a task-space formulation, various simplified models for approximating dominant dynamic effects during legged locomotion have been proposed, see Figure 2.17. Naturally, these models have a strong focus on centroidal dynamics in the sagittal plane. In general, they are not restricted to motion generation but are also often used within lower-level stabilization.

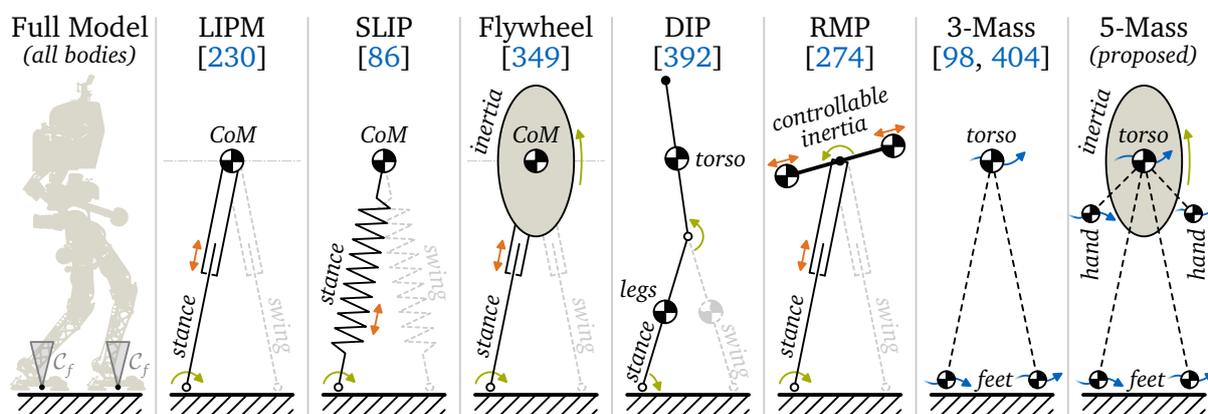


Figure 2.17: Selection of models for approximating the robot's dynamics. From left to right: full model for comparison (all bodies, with friction cones C_f in contacts); *Linear Inverted Pendulum Mode (LIPM)* [230]; *Spring-Loaded Inverted Pendulum (SLIP)* [86]; LIPM with flywheel; *Double Inverted Pendulum (DIP)* [392]; *Reaction Mass Pendulum (RMP)* [274]; three-mass model; five-mass model. The colored arrows highlight rotational (green), 1D translational (orange), and 2D translational (blue) DoF. Apart from the depicted planar case, also a 3D variant exists for most models.

The probably most prominent example is the LIPM [230] by KAJITA and TANI (already introduced in Footnote 8), which has been shown in [231] to be a suitable approximation of biped gait dynamics and thus, has been used for planning and stabilization [234]. Originally formulated for the planar case, it also has been generalized to the three-dimensional case in [232]. Methods using the LIPM are typically restricted to a constant CoM height. Moreover, only the current stance leg is considered, thus, focusing on the SS phase and making the DS phase infinitesimal short (instantaneous leg switch). By considering the current stance leg as linear spring, we obtain the SLIP model [86] (already introduced in Footnote 4), which is very common especially in the field of biorobotics and mainly used to model hopping and running. Another extension of the LIPM model is made (among others) by PRATT et al. in [349] through replacing the point mass by a flywheel to model the angular momentum around the CoM. The mass moment of inertia is set to be a constant model parameter.

Other examples for simplified robot models are the *Double Inverted Pendulum (DIP)* model [392] by STEPHENS, the *Reaction Mass Pendulum (RMP)* model [274] by LEE and GOSWAMI, and the *Linear Biped Model (LiBM)* [393] by STEPHENS and ATKESON. The DIP model consists of two linked segments abstracting the legs and the torso. This allows to subdivide the control

into an ankle strategy (counteracts small disturbances) and a hip strategy (counteracts large disturbances) known from human balance control [392]. The RMP model splits the single mass of the LIPM into a pair of equal point masses. The position of the point masses is symmetric around the upper pivot center (CoM) and can be changed, such that the mechanism allows to control the inertia (so-called “inertia shaping”) while maintaining the same CoM [274]. In the three-dimensional case, this generalizes to three mass pairs forming an ellipsoid. Each pair is aligned along a different axis and can be controlled independent of the others. In [274], the RMP is presented as a tool to analyze the centroidal momentum as abstraction of the aggregate limb dynamics. Finally, the LiBM resembles two superimposed LIPM (one for each leg) and allows to explicitly consider the DS phase [393]. Within the SS phase, it remains identical to the LIPM. In [393], a method to obtain periodic motion for the LiBM based on the concept of orbital energy is presented. An unconventional but interesting approach is made by NAGARAJAN and YAMANE. In [316], they automatically derive a robot model by performing a model order reduction based on balanced truncation. In [316], they focus on balancing, fast arm swing, and hip rock and roll only and do not consider actual locomotion such as walking.

In order to incorporate swing foot dynamics, the LIPM has been extended by TAKENAKA et al. in [404] with two additional point masses at the feet forming the so-called three-mass model¹⁶. The model has been adopted by BUSCHMANN et al. in [98], where, in contrast to the LIPM, the upper body (torso) mass is typically not constrained to move along a straight line, but instead can have an arbitrary (known) vertical motion. In [100, p.79], it has been shown that including the effect of foot masses gives a more accurate approximation, especially for high walking speeds. The three-mass model forms the basis of the five-mass model, which is used within this thesis, see Section 4.5.2 for details. The five-mass model adds explicit hand masses to introduce arm dynamics similar to the swing foot dynamics. It also introduces a mass moment of inertia for the torso body which models the dynamic effects of (planned) upper body rotation.

Finally, it has to be mentioned that the foot-ground contact can be modeled in different ways. On the one hand, some motion generators assume the feet to be point-alike, hence the contact acts as a simple pivot point. On the other hand, if real feet are considered, an additional torque representing an active ankle joint may be applied. Moreover, for most simplified robot models used within task-space approaches, the contact is assumed to provide “sufficient” friction such that no slipping occurs. In contrast, “full” models, typically used in whole-body approaches, explicitly consider contact stability by incorporating (often linearized) friction cones (assuming COULOMB friction) at each EE in contact, see [94] for details.

Dynamic Feasibility – ZMP, CoP, and FRI/FZMP For evaluating dynamic feasibility, a classic approach is to extend the well-known criterion for static equilibrium (see Section 2.3) to a criterion for so-called *dynamic balance*. Here *dynamic balance* means, that the contact between the robot and the ground remains closed (no relative motion) and thus, the robot does not tip over. A prominent example for such a criterion has been introduced by VUKOBRATOVIĆ and STEPANENKO [427]: a gait is dynamically balanced, if the *Zero-Moment Point (ZMP)* remains within the SP at all times. Herein, the ZMP denotes the point in the contact plane, at which the *horizontal*¹⁷ torques of the net *Ground Reaction Wrench (GRW)* become zero [428]. If the robot is in rest, the ZMP coincides with the GCoM, which shows that the ZMP criterion is indeed a generalization of the static equilibrium test. Moreover, the definition of the ZMP makes it a suitable choice for the (torque-free) pivot point in pendulum based simplified robot models such

¹⁶Extending the LIPM by a mass accounting for the swing leg has for example already been proposed by PARK and KIM in [336]. However, in [336] it is assumed, that gravity represents the dominating effect and thus, the inertia of the swing foot mass is neglected. In contrast, TAKENAKA et al. also consider the effects of inertia [404].

¹⁷Note that there might be a non-zero vertical torque component at the ZMP, which is why the wording *zero-moment* may be perceived as misleading, cf. [174]. Indeed, under assumption of sufficient friction, a non-zero vertical torque component has no impact on dynamic balance. However, to avoid slipping around the vertical axis due to limited real-world friction, one typically tries to minimize the vertical torque component too.

as the LIPM, see Figure 2.17. A more detailed explanation of the ZMP and how it is used in the motion generator presented in this thesis is postponed to Section 4.5.4. In literature, multiple other formulations with close relation to the ZMP exists. Prominent examples are the *Center of Pressure (CoP)*¹⁸ or the *Foot Rotation Indicator (FRI)*¹⁹ point [174], which both are (almost) equivalent to the ZMP. According to VUKOBRATOVIĆ and BOROVIAC [428], the difference between ZMP, CoP, and FRI point can be summarized as follows (see also Figure 2.18):

- If the gait is *dynamically balanced*, then ZMP, CoP, and FRI point coincide and lie within the SP. The contact of the robot with the ground is stable (no tipping motion).
- If the gait is *dynamically unbalanced*, then the FRI point lies outside the SP, the CoP lies on the border of the SP, and the ZMP does not exist. The robot tips over the edge of the SP. The distance of the FRI point to the SP is a measure for the “unbalance” [174].

The motion generator presented in this thesis shall produce a dynamically balanced gait. Since all three entities coincide in this case, only the ZMP will be discussed in the remaining chapters.

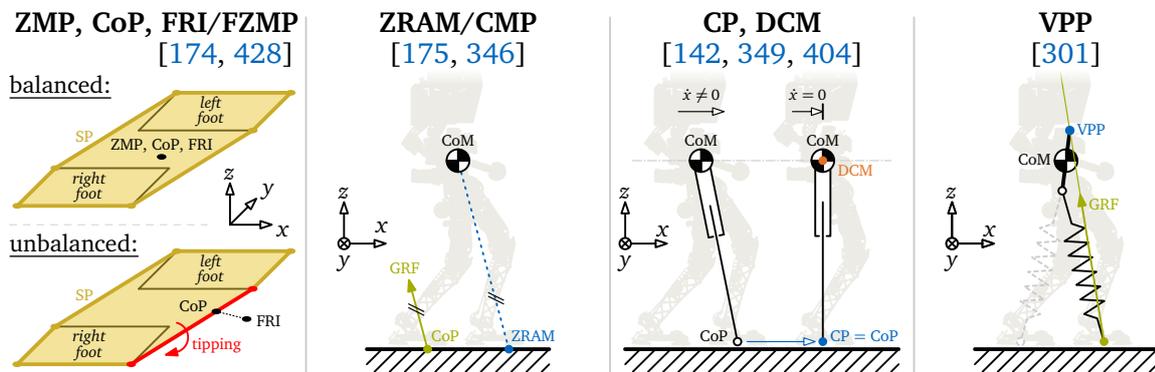


Figure 2.18: Illustration of common concepts related to the analysis of dynamic feasibility. From left to right: *Zero-Moment Point (ZMP)*, *Center of Pressure (CoP)*, and *Foot Rotation Indicator (FRI)* alias *Fictitious ZMP (FZMP)* for the balanced and unbalanced case; *Zero Rate of change of Angular Momentum (ZRAM)* alias *Centroidal Moment Pivot (CMP)*; *Capture Point (CP)* and *Divergent Component of Motion (DCM)*; *Virtual Pivot Point (VPP)*.

A severe limitation of the ZMP criterion is that all contacts have to lie in the same horizontal plane. In practice, many motion generators (such as the one proposed in this thesis) use this criterion also for scenarios with non-coplanar contacts, such as climbing stairs. Especially if the step height is small when compared to the step length, the introduced error seems to be negligible. Another important restriction of the ZMP criterion is the assumption of sufficient friction, i. e., that the robot does not slip. Although the ZMP is introduced in [427] by modeling the robot’s foot with a finite dimension, the ZMP criterion can also be used for robots with point-alike feet, e. g. quadrupeds (three feet in contact are equivalent to a single triangle-shaped foot). However, for bipeds with point-alike feet, the SP would degrade to a line in the DS phase and a point in the SS phase. Aside from the interface of the robot with the ground, the ZMP criterion does not set any requirements on the mechanism “above” the contact plane, i. e., the choice of the robot model or (known) external forces and torques. This can be exploited for multi-contact planning which will be explained later.

¹⁸ GOSWAMI describes the *Center of Pressure (CoP)* as the “point on the foot/ground surface where the net ground reaction force actually acts” [174]. The CoP is very common in the field of biomechanics, where gait analysis is mainly performed in the sagittal plane. In contrast to the *Ground Reaction Wrench (GRW)*, which can be described at an arbitrary reference point (the ZMP is only a special choice), the *Ground Reaction Force (GRF)* is defined to act at a point such that there is no (planar) torque component (thus the name *ground reaction “force”*). This highlights the equivalence between the CoP and the ZMP.

¹⁹ GOSWAMI defines the *Foot Rotation Indicator (FRI)* point as a “point on the foot/ground contact surface, within or outside the convex hull of the foot support area, at which the resultant moment of the force/torque impressed on the foot is normal to the surface” [174]. In [428], VUKOBRATOVIĆ and BOROVIAC call this point *Fictitious ZMP (FZMP)* to indicate, that – in contrast to the ZMP – it also exists outside of the SP.

Dynamic Feasibility – ZRAM/CMP Besides the ZMP as the probably most widespread balance concept in robotics, various other entities describing core characteristics of legged locomotion have been discovered. In [175], GOSWAMI and KALLEM introduce the *Zero Rate of change of Angular Momentum (ZRAM)* point as the location on the ground, at which the net GRF would have to act in order to obtain a zero rate of change of the robot’s centroidal angular momentum. Later, they also call this point *Centroidal Moment Pivot (CMP)*²⁰ [346]. They propose to enforce a constant angular momentum, which is motivated by the fact, that the total angular momentum changes when the robot starts to fall. However, there exist stable gaits which do not satisfy this condition, see for example the motion generator proposed by TAJIMA et al. in [402], which explicitly plans a non-constant angular momentum depending on the horizontal CoM dynamics in order to avoid “unusual” motion. BUSCHMANN also reported from experience with JOHNNIE and LOLA, that gaits satisfying this condition “appear very unnatural and lead to large compensating motions of arms and/or the upper body in order to cancel the change of angular momentum produced by the legs” [100, p. 51].

Dynamic Feasibility – Running It has to be highlighted, that although dynamic balance is an often used criterion, it is not strictly required for stability or dynamic feasibility of a gait pattern. An obvious example is locomotion with ballistic phases such as running where no SP exists during the flight phase. Motion generators for running have been proposed for example for Honda’s ASIMO by TAKENAKA et al. in [405] (10 km/h, 0.1 s flight phase) or for Toyota’s PARTNER by TAJIMA et al. in [402] (7 km/h, 0.1 s flight phase). The key idea for ASIMO’s running gait was to split it into a vertical, horizontal, and rotational component for which different robot models are applied (vertical: single point mass, horizontal: three-mass model, rotational: flywheel) [405]. Motion generation for PARTNER was realized by first planning running in place and then superposing translational velocities to obtain a certain direction and speed [402]. Note that the ZMP concept can still be used for the phases where one or both feet are in contact. During the flight phases, the CoM is simply planned according to ballistics. Special care has to be taken for the transition between contact and flight phases, where the horizontal components of the GRF have to zero out to prevent slipping.

Dynamic Feasibility – CP and DCM A strategy similar to the ones for running gait generation has been proposed by PRATT et al. in [349] with the *Capture Point (CP)* concept. Assuming a biped modeled as a LIPM, the CP denotes the point on the ground where the next step has to be placed in order to come to a full stop (zero kinetic energy) [349], see Figure 2.18. In case a flywheel is added to the model (cf. Figure 2.17), this generalizes (together with certain actuation limits) to a so-called *capture region* as set of all possible CPs. Among the first systems using the CP concept was the SEA driven lower-body biped M2V2 [350] by the IHMC. Although this approach was originally meant for disturbance rejection, it was later adopted for motion generation, e. g. in [141]. Another analysis of the LIPM was done by TAKENAKA et al. in [404], who separated the dynamics into a convergent and *Divergent Component of Motion (DCM)*. The convergent component represents the stable part of the LIPM dynamics and is typically ignored since it converges without the need of an external control. The DCM represents the instable part, for which various control techniques have been proposed. Originally, TAKENAKA et al. formulated the DCM in the plane, nevertheless, it has been generalized to the three-dimensional case by ENGLSBERGER et al. in [142]. In the two-dimensional case, the CP and the ground projection of the DCM are equivalent (see Figure 2.18), which does not hold true for the three-dimensional case, cf. [144].

²⁰In [346], the authors give a formal definition of the *Centroidal Moment Pivot (CMP)* as “the point where a line parallel to the ground reaction force, passing through the CM [=CoM – author’s note], intersects with the external contact surface” [346], see Figure 2.18. In this publication they also show, that the mean separation distance between ZRAM/CMP and ZMP measured in human gait is rather small.

Dynamic Feasibility – VPP Another balance indicator, which became quite prominent in the field of biomechanics, has been found by MAUS et al. In [301], they observed that for healthy humans and some animals like chicken and dogs, the GRF (as seen from a reference frame attached to the CoM) passes through a certain point above the CoM. The authors named this point *Virtual Pivot Point (VPP)*, since it can be seen as a virtual “hook” above the CoM introducing regular (non-inverted) pendulum dynamics which automatically compensate disturbances. Their observations are restricted to the SS phase during straight walking and running and also only consider the sagittal plane. For their analysis, they use a rather simple biped model, where the CoM and VPP are fixed to the torso body having a constant distance to each other, see Figure 2.18. Recently, the VPP also received attention in the community around humanoid robotics. An investigation by STAUFENBERG et al. [391] showed, that the motion generator proposed in this thesis (and also its predecessor by BUSCHMANN et al., see [98]) does not establish a VPP, yet it generates a quite smooth gait. Indeed, the VPP may not represent a stability or feasibility criterion, but instead gives an insight on how nature solves the biped gait problem.

Dynamic Feasibility – Multi-Contact, Friction, and CWC In case of multiple, non-coplanar contacts (e. g. climbing stairs, multi-contact, etc.), the aforementioned concepts are not directly applicable anymore. To evaluate the feasibility of a certain motion also in these scenarios, a common approach is to check if the individual contact forces at the EEs in contact remain within their respective friction cones (which implies the assumption of COULOMB friction and point contacts). This condition is also known as *weak contact stability* [94]. Pioneering work in this direction was made by SAIDA et al. in [366], which was later generalized by HIRUKAWA et al. in [204]. Instead of considering individual contact forces, they require the net (total) contact wrench acting on the robot to lie within a certain set. By linearizing the individual friction cones, this set becomes a polyhedral convex cone (see Figure 2.19), which is called *Gravito-Inertial Wrench Cone (GIWC)* [108] or equivalently *Contact Wrench Cone (CWC)* [110]. The net contact wrench and gravito-inertial wrench are obtained by splitting the NEWTON-EULER equations into terms related to the robot-environment interface (contacts) and gravitational and inertial multi-body effects, respectively. Since both parts together form an equilibrium, they can be used synonymously (only the sign may differ). HIRUKAWA et al. consider the cases of *weak contact stability* (with satisfied COULOMB friction) and *strong contact stability* (with assumption of sufficient friction) as originally classified by PANG and TRINKLE in [335]. In [204], they show that for a biped robot walking on a horizontal plane with sufficient friction, their formulation for strong contact stability is equivalent to the ZMP criterion.

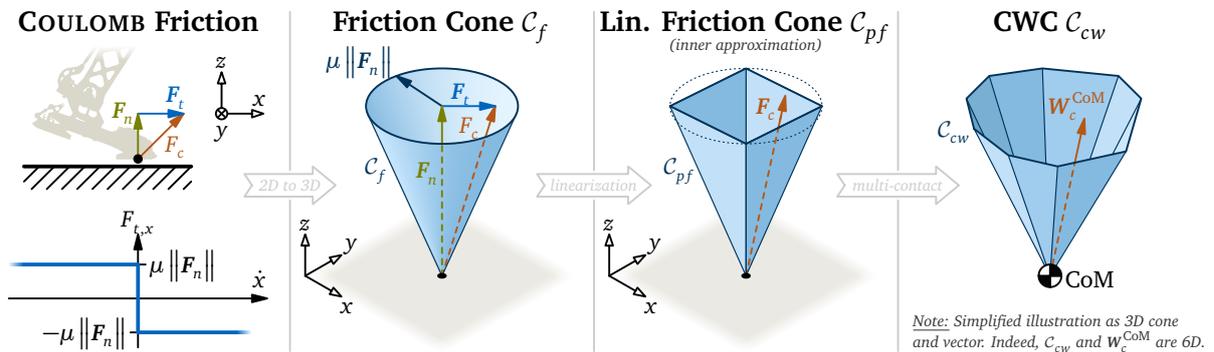


Figure 2.19: Contact Wrench Cone (CWC) as polyhedral convex cone. From left to right: COULOMB friction model with friction coefficient μ and contact force F_c as vector sum of normal force F_n and tangential force F_t ; friction cone C_f ; linearized (polyhedral) friction cone C_{pf} ; CWC C_{cw} and net (total) contact wrench W_c^{CoM} . See [108] for details.

The seminal work by SAIDA et al. and HIRUKAWA et al. motivated numerous extensions and generalizations, such that this field now represents a new area of research on its own. A general difficulty is to efficiently compute the CWC as a polyhedral convex cone for which typically an

algorithm based on the *Double Description Method (DDM)*²¹ is used. A common simplification for the use in a motion generator is to assume zero CoM acceleration, which results in a pure static feasibility criterion [151]. Recently, new methods have emerged which also take the CoM acceleration into account. An example is given by CARON and KHEDDAR in [109], who limit the CoM position to compute an envelope of feasible CoM accelerations. In a dual manner, AUDREN and KHEDDAR propose in [71] to limit the CoM acceleration for computing feasible CoM positions. A typical drawback of methods based on polyhedral representations is that for a slight increase in the number of contacts, the computation time often increases drastically [71].

Another approach for testing dynamic feasibility has been proposed by CARON et al. in [110] where they generalize the classical SA used within ZMP-based methods by explicitly considering friction and multiple non-coplanar contacts. They first formulate a “full” SA (incorporates friction) and subsequently a “pendular” SA as a subset. The latter additionally introduces the constraints of a linear, regular (non-inverted) pendulum between the CoM and the ZMP. In [110], the method is understood as a generalization of the ZMP criterion to check dynamic feasibility also for non-coplanar contacts. However, the derivation is rather complex and requires the incorporation of pendulum dynamics while the classical ZMP criterion for coplanar contacts does not set any requirements on the dynamics model of the robot.

The explicit consideration of friction within motion generation can be seen ambivalently. On the one hand, assuming infinite friction is surely unrealistic. On the other hand, if a friction model is involved, it also has to be parameterized. In reality, the friction coefficients depend on the actual material pairing (foot/ground) and are not known in advance. Moreover, the aforementioned methods assume COULOMB friction with linearized friction cones, which is still a very rough approximation when compared to the high complexity of real-world contacts.

Kinematic Feasibility Besides multi-body dynamics, also kinematic constraints of the robot have to be considered in order to maintain feasibility of the generated motion. Since most humanoid robots do not feature the same flexibility and range of motion when compared to the human anatomy, certain limitations arise even for allegedly simple maneuvers such as climbing stairs or stepping over obstacles. Within a whole-body approach, incorporating individual joint limits is straight forward. In contrast, considering kinematic feasibility within a task-space formulation always implies some kind of approximations since explicit joint-level information is not available. Typical approaches are to use pre-computed reachability areas/volumes or a preliminary check of joint limits by a simplified kinematic model approximating the real limb topology. Herein, the most important quantity is the vertical CoM/torso position since it has the greatest influence on leg kinematics in a biped. Methods for planning a suitable vertical trajectory have been proposed (among others) by STASSE et al. in [389] and NISHIWAKI and KAGAMI in [325]. For stepping over large obstacles, STASSE et al. propose to generate a vertical waist trajectory built from third-order polynomials, which are connected by a key configuration. The key configuration is obtained from a so-called *feasibility unit*, which considers the obstacle’s geometry (height and length). NISHIWAKI and KAGAMI focus on stepping up and down a staircase for which they use a simplified model of the leg kinematics in the sagittal plane. The upper bound for the torso height is determined based on the existence of an IK solution (stretched knee) and the maximum joint velocity in the knee DoF. Apart from the vertical CoM/torso position, also the horizontal components affect kinematic feasibility. However, they are typically computed by the method for accomplishing dynamic feasibility instead. Indeed, some frameworks use a classical LIPM-based approach, although – strictly speaking – this conflicts with the requirement of a constant CoM height introduced by the LIPM.

²¹ A polyhedral convex cone can be described in a “face” form (set of inequalities) or in a “span” form (positive combination of base vectors), hence the name “double description”, [108]. The *Double Description Method (DDM)* is an algorithm to solve the so-called *extreme ray enumeration problem*, i. e., to compute one form from the other. It was originally introduced by MOTZKIN et al. in [311] and later revisited by FUKUDA and PRODON in [164].

Solution Strategies The chosen robot model together with the concepts for kinematic and dynamic feasibility can be seen as input for the solution strategy, which defines how this information is used to generate motion in the form of smooth trajectories. Among the frameworks using a task-space formulation, the probably most prominent method has been proposed by KAJITA et al. in [233]. Herein, the ZMP concept has been first coupled with the three-dimensional LIPM (the ZMP acts as pivot point of the pendulum), which is then used to formulate a preview controller tracking a given ZMP reference trajectory under consideration of a certain time horizon. In general, most ZMP-based approaches are formulated as a so-called *inverse problem* [233]: instead of computing the ZMP from a given robot motion, the required robot motion (in particular the CoM) is computed from a given (reference) ZMP trajectory. As a consequence, one has to distinct between the desired (planned) ZMP and the actual (executed) ZMP.

A motion generator based on the DCM was first introduced by TAKENAKA et al. in [404], where a desired ZMP trajectory is modified in a way such that the DCM matches a cyclic gait at the boundaries. They use a three-mass model, where the torso mass follows the dynamics of a LIPM and the CoM is finally computed as a combination of the torso and foot mass motion. With this approach, they achieved impressive walking speeds of up to 4 km/h for ASIMO [404]. Besides the extension of the DCM concept to the three-dimensional case by ENGLSBERGER et al. in [142], the authors also propose an accompanying motion generation workflow. First, they prescribe the motion of a so-called *Virtual Repellent Point (VRP)*²² by placing it at a user-defined height above the CoP. The CoP is given by the foot placement and the assumption of point-alike feet. The VRP then defines the dynamics of the DCM which in turn determines the dynamics of the CoM. Unfortunately, the assumption of instantaneous stance leg switch in [142] led to discontinuous joint torques. This assumption was dropped in [144] by smoothing the DCM reference trajectory in the DS phase using third-order polynomial interpolation. However, other assumptions such as point-alike feet are still necessary to allow an analytic solution. In [145], the smoothness was further improved by interpolating also the VRP reference trajectory through polynomial splines. Earlier work at the DLR focused on the closely related CP, cf. [141]. Another prominent advocate for CP-based motion generation and control is *Boston Dynamics* since they used it for ATLAS, see [321].

The three-mass model proposed by TAKENAKA et al. was adapted by BUSCHMANN et al. in [98], where the torso mass is not restricted to LIPM dynamics anymore but instead can have arbitrary vertical motion. BUSCHMANN et al. formulate a second-order, linear BVP for the horizontal components of the torso mass, which is solved by a spline collocation method. The motion generator proposed in this thesis integrates an extension of this approach, thus a detailed explanation is postponed. A main advantage of this strategy is that there are no restrictions for the reference ZMP trajectory or the vertical torso motion. In particular the latter is beneficial since it makes incorporating kinematic limits simple, e. g. by the methods proposed in [8, 325, 389].

The solution strategies mentioned so far use a rather simple model of the robot. In contrast, there are numerous methods which incorporate a “full” multi-body model. Due to the significant increase of complexity, these typically involve offline computations (at least in parts). An example is given by DENK and SCHMIDT in [131], where a database of dynamically feasible gait primitives is synthesized by optimal control techniques solved through direct collocation. The precomputed walking patterns assume symmetric gait and differ in parameters like step length (walking speed), direction (curve walking), or clearance (stepping over obstacles). The synthesis is time consuming, thus performed offline, while the concatenation of primitives to an executable sequence is done online. In [97], BUSCHMANN et al. propose a nonlinear parameter optimization considering the full multi-body dynamics of JOHNNIE. The computation is accelerated by using recursively derived analytic gradients, however, the runtime still remains in the range of several minutes which makes it an offline-only method. Another approach is suggested

²²The *Virtual Repellent Point (VRP)* can be understood as the endpoint of a (virtual) linear spring attached to the CoM, such that the spring force equals the total force acting on the CoM due to gravity and ground reaction [142].

by BESSONNET et al. in [79], where the EoM of a full, but two-dimensional (sagittal plane) multi-body model of the robot are transformed into a state space representation. An optimum gait minimizing driving torques and reaction forces is found by applying PONTYAGIN's *Maximum Principle (PMP)* [166] to the state space EoM leading to a corresponding BVP. Constraints are defined for transitions (SS, DS), states (joint limits, collision avoidance), and forces (unilateral ground contact and maximum joint torques). The BVP is solved through a combination of shooting (for initial guess) and finite differences (final solution). In [403], TAKANISHI et al. propose to linearize the EoM of the torso (alias "trunk") as obtained from a full multi-body model where the remaining motion (limbs) is assumed to be known. The resulting decoupled (x/y), linear, second-order ODEs for the horizontal torso components are solved iteratively using a *Fast Fourier Transformation (FFT)* together with periodic *Boundary Conditions (BCs)*. Another example for a motion generator tracking a certain reference ZMP trajectory while using a full robot model is given by NAGASAKA et al. in [317]. They optimize the horizontal torso motion of the humanoid H7 using a steepest gradient approach, where initial trajectories are obtained from a static²³ walking pattern. Since the full model has to be evaluated within each iteration of the optimization, the method is also restricted to offline computation. In later work on H7, real-time performance for planning up to three steps ahead has been achieved by NISHIWAKI et al. in [323]. They modify the horizontal motion of all bodies by the same distance except for the feet (fixed to the ground) in order to follow a given ZMP trajectory. Their BVP is solved by an iterative finite difference method, which requires a feasible initial solution.

For frameworks using a full model of the robot, we observed a transition from classical ZMP-based approaches to more generalized ones using contact stability and the CWC as feasibility criterion. Here, contact stability (together with various other constraints) is embedded within a comprehensive optimization to compute whole-body motion in the form of joint-space trajectories. A rough overview of the typical workflow of this class of motion generators (2nd variant in Figure 2.16) has already been given above. Seminal contributions by the CNRS-AIST JRL group are for example [149, 279, 280]. Unfortunately, the high degree of generality together with the focus on complex multi-contact actions comes hand in hand with high computational cost, which makes real-time execution infeasible in most cases. Another issue of the formulation as a large-scale optimization problem is that the solver is prone to get stuck in local minima and that manual corrections still might be necessary in some cases, cf. [149].

Model Predictive Control (MPC) A major step towards real-time performance has been made (among others) by CARPENTIER et al. in [111]. Their key idea is to introduce various simplifications to the optimization problem, most importantly using only centroidal dynamics (CoM and angular momentum) for dynamic feasibility instead of considering all bodies. The optimal control problem is solved by a MPC approach based on multiple shooting. Subsequently, the desired whole-body motion is computed using a second-order hierarchical IK. For a multi-contact scenario where HRP-2 climbs a stair while using a nearby handrail, the total planning time for a single stair is less than 6 s, which represents a significant improvement for this class of motion generators. Although their framework has been demonstrated on a biped, it can also be used for quadrupeds. Note that especially for quadrupeds, numerous MPC-based approaches for motion generation exist, see for example the already mentioned publications around STARLETH and ANYMAL and the impressive results recently achieved with wheeled ANYMAL [81]. For humanoids, the clearly most impressive demonstration of motion generation using MPC have been given by *Boston Dynamics* with the highly dynamic dance and parkour moves of ATLAS. Apart from the general structure of their framework, which involves partial offline computation

²³The terms *static* and *walking* might seem to be mutually exclusive at first sight. However, in this context, a *static walking pattern* indicates that only static feasibility has been considered during motion generation. Although this does not necessarily mean that the executed motion has to be slow, typically only rather slow, quasi-static motion is planned to avoid falling due to unconsidered dynamic effects.

[260], not too much details have been published yet. Finally, there also exist various MPC based approaches which achieve real-time performance by considering a rather simple model of the robot, such as the LIPM/ZMP-based method proposed by MAXIMO et al. in [302].

Embedding Multi-Contact Dynamics Per definition, whole-body motion generators using contact stability as feasibility criterion (2nd variant in Figure 2.16) automatically support multi-contact situations such as additional hand support during walking. For performance reasons and to reduce complexity, multi-contact actions are often tested for static feasibility only (e. g. by a static equilibrium check at key configurations). Nevertheless, approaches which are based on solving a comprehensive optimization problem still suffer from high computation times.

On the other side, methods for generating abstract motion (1st variant in Figure 2.16) are typically much faster. However, since they often rely on a feasibility criterion assuming coplanar contacts, custom extensions for supporting multi-contact situations are necessary. An example for such an extension is given by MASON et al. in [300]. They propose to generate the CoM trajectory by MPC assuming LIPM dynamics. In case the ZMP is about to leave the classical SP, a mixed-integer *Quadratic Program (QP)* is triggered to find an optimal hand contact position. Once the hand is in contact, another QP is solved to compute the required external force at the hand, which centers the so-called *shifted SP* to the current ZMP. By considering the additional hand support as a planned, external force shifting the SP (or equivalently the ZMP) rather than as a non-coplanar contact, dynamic feasibility under multi-contact conditions is established without violating the assumptions of the ZMP criterion (see also Figure 2.20). Similarly, MUROOKA et al. present in [314] a multi-contact pattern generator tracking a reference ZMP trajectory by extending the classical preview control of KAJITA et al. from [233]. They use the LIPM to describe the core dynamics of the biped and consider hand contacts as external forces shifting the ZMP (in [314] called *ext-ZMP*). The subsequent DCM-based stabilizer is also extended to account for the additional hand contact forces.

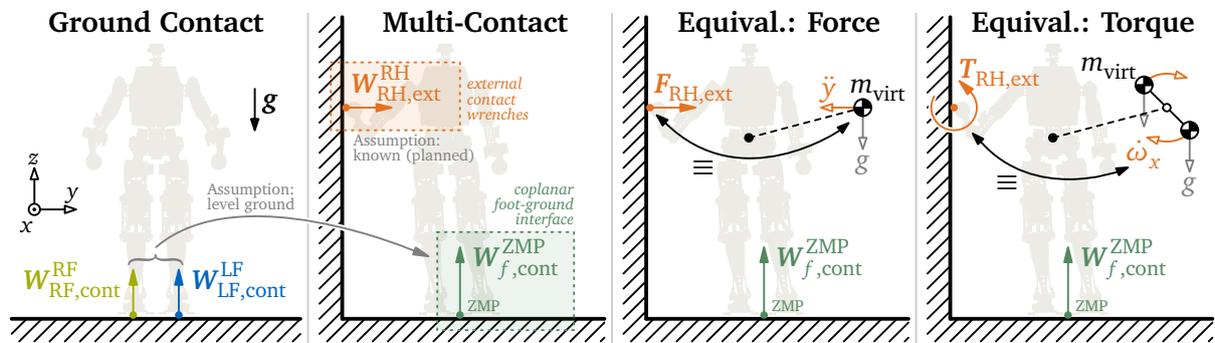


Figure 2.20: Restoring the validity of the ZMP criterion for multi-contact scenarios by making certain assumptions. The interaction of the robot with its environment can be split up into coplanar contacts representing the foot-ground interface (assuming level ground) and external contact wrenches at the hands (assumed to be known). Each external wrench is equivalent to the gravito-inertial effects of additional (virtual) masses of the robot making a corresponding contribution to the EoM. Since the ZMP criterion only sets requirements on the foot-ground interface but not on the model of the robot, it remains valid as long as the external wrenches are considered in the EoM of the model.

Certainly, there are also numerous hybrids between the 1st and 2nd variant of motion generators such as the one proposed by KUMAGAI et al. in [262], where abstract motion is planned using the concept of contact stability for evaluating multi-contact feasibility. Herein, the optimum CoM motion and contact forces are computed by solving a QP. Since feasibility is checked by testing for static equilibrium (using linearized friction cones), their method is restricted to quasi-static motion. Although it is faster than classical whole-body approaches solving a comprehensive optimization problem, total planning times still remain in the range of several seconds due to the subsequent whole-body IK based on a prioritized QP.

Machine Learning The methods presented so far can be understood as “conventional” approaches. In the past decade, the great popularity of machine learning in the field of robotics and in particular CV also triggered the investigation of corresponding motion generators for legged locomotion. Here, a common approach is to implement joint-level controllers (often of PD-type), which get parametrized through a learning process fed by simulated or real experiment data. So far, most of these methods rely on proprioceptive sensor data only, thus learning based methods are currently more associated with the task of *control* rather than *planning*. This might change once CV is coupled with motion generation to maintain a certain planning horizon.

A direct comparison of a conventional LIPM/ZMP-based (preview control) method and a novel, reflex based neuro-controller was conducted on ARMAR-4 by GOLDBECK et al. in [172]. With their neuro-controller they could slightly improve the CoT, maximum speed, and robustness (walking on a slope and with external disturbances). However, their study is entirely based on simulation data, which limits its meaningfulness for a real-world application. A successful transition of a learning based controller to real hardware was achieved by SIEKMANN et al. in [381] for the lower body biped CASSIE. They demonstrate very robust biped stair climbing (only proprioceptive sensors – no CV) based on reinforcement learning, where the training is performed exclusively in simulation and the resulting controller is deployed on the real hardware without further modification (following the so-called *sim-to-real* paradigm). The key to their success seems to be the randomization of dynamics parameters (friction, damping, mass, etc.) in their simulation environment used for training, which makes the controller robust against real-world inaccuracies. A similar approach based on reinforcement learning was presented earlier by LEE et al. in [275] for the quadruped ANYMAL. They propose a two-stage learning process. First, a “teacher” policy is trained having access to privileged information such as ground truth knowledge on terrain and contacts. In the second stage, a proprioceptive “student” policy is trained under guidance by the teacher. Their system is robust enough to handle challenging natural terrain including mud, snow, rubble, dense vegetation, and flowing water.

A remarkable feature of such methods is their *zero-shot generalization*, i. e., that they can handle situations which did not occur during training. A general drawback is that it is difficult to understand why certain learned policies work, which makes it hard to further improve methods. Moreover, it makes the system unpredictable and thus, potentially more dangerous than conventional approaches. Finally, one has to keep in mind that a large amount of data is not enough, but instead “every learner must embody some knowledge or assumptions beyond the data it is given in order to generalize beyond it” [138]. Otherwise, the resulting policy will not be better than random guessing, see the famous “no free lunch” theorem by WOLPERT [444].

Hybrid Zero Dynamics (HZD) Most frameworks based on HZD (already introduced in Footnote 13) focus on a rather simple, planar model of the robot and a pure simulative analysis, see for example [264]. However, there are also recent investigations, which deploy HZD-based motion generation on real, full-sized humanoids. In [195] for example, HEREID et al. propose to use the full multi-body dynamics together with HZD-typical virtual constraints (i. e. constraints enforced through feedback control) to formulate a nonlinear optimization problem, which is solved through direct collocation. They achieve stable, three-dimensional walking with a mean CoT of only 1.33 for the spring-legged humanoid DURUS. Due to the involved large-scale optimization, the computation time lies in the range of several minutes.

Conclusions So far, countless approaches for motion generation of legged and in particular biped robots have been published. The mentioned works have to be understood as examples for the main directions in this field. Moreover, the separation into a 1st and 2nd variant made in Figure 2.16 is just one way to look at the state of the art. The huge amount of possible combinations of the core ingredients, such as the robot model, the concepts for kinematic and dynamic feasibility, and the solution strategy, makes a further classification difficult.

The common primary goal of all motion generators is to plan a feasible motion such that the robot does not fall. Similar to the field of hardware design, there are different secondary objectives depending on the research direction. Frameworks originating from the field of biorobotics and the HZD community typically put strong focus on locomotion efficiency. The generated motion is often restricted to straight, cyclic gait patterns, which are optimized for low CoT or to behave similar to human or animal gait. In contrast, the community around fully-actuated humanoids is more focused on versatility, traversing obstacles, or complex, acyclic multi-contact maneuvers. Naturally, the real-time capabilities of a motion generator are directly related to the complexity of the underlying model and feasibility concepts. Online processing has been possible ever since – it’s just a matter of which kinematic/dynamic effects are considered.

Despite the great progress in research, there is still much room for improvement. Especially for multi-contact scenarios, only rather slow, quasi-static motion has been achieved in real-world experiments, see for example the videos attached to [150] or [314]. Here, further improvements in the hardware design but also the incorporation of multi-contact dynamics during motion generation are necessary. Both issues are tackled by this thesis, see Chapters 3 and 6.

Apart from conventional techniques, also locomotion frameworks based on machine learning have recently been successfully deployed on hardware. Although this class of motion generators is in its early days, impressive results have already been demonstrated, which promise even better results once this area of research becomes more mature. In the author’s personal opinion, a smart combination of both, conventional and learning based methods, has the potential to achieve human like locomotion performance in the mid-term future.

Relations to this Thesis The motion generator presented in Chapter 6 is based on the previous work by BUSCHMANN [100] (EE motion, robot model, dynamic feasibility, and solution strategy) and HILDEBRANDT [201] (kinematic feasibility). In order to support multi-contact locomotion, it has been rewritten from scratch, which allowed to integrate the necessary extensions. The presented technique belongs to the first class of motion generators creating an abstract motion (EE and CoM), which is translated to individual joint-space trajectories by the subsequent SIK module (see Section 4.6 and SYGULLA [401]). The focus lies on real-time generation of *gaited multi-contact* locomotion (less than 10 ms planning time in most scenarios). In this work, multi-contact is meant to implicitly (preemptively) improve robustness. In contrast, some related work focuses on explicit disturbance rejection similar to footstep modification, see for example [431].

For testing dynamic feasibility, a five-mass model as extension of the three-mass model proposed by BUSCHMANN et al. [98] is used, see Figure 2.17. Moreover, the ZMP criterion is applied, where hand contacts in multi-contact scenarios are considered as planned external forces (similar to [300, 314]). To obtain a dynamically feasible CoM motion from a given ZMP reference trajectory, a BVP is formulated, which is solved through quintic spline collocation as extension of the cubic spline collocation method proposed by BUSCHMANN et al. in [98]. The use of quintic polynomials leads to a smoother CoM trajectory and an easier incorporation of boundary conditions without the need of tuning parameters. The proposed BVP describes locomotion, which is gaited but *not* cyclic. Thus, individually placed contacts (as obtained from the contact planner) allow arbitrary walking patterns such as straight-, backwards-, curve-, and lateral-walking, climbing stairs, additional hand-support, and all imaginable combinations of those. Note that multi-contact dynamics are embedded without any approximation. In contrast, the proposed motion generator violates the requirements of the ZMP criterion for climbing stairs or walking on uneven ground (non-coplanar contacts). As successful experiments demonstrate, the introduced error seems to be small. Although, the motion generator assumes that the contacts provide sufficient friction, (linearized) friction cones at the EEs are considered in the subsequent SIK module for computing an optimal contact wrench distribution, see [401, p. 80ff] for details.

For evaluating kinematic feasibility, a planar, simplified leg model based on the one proposed by HILDEBRANDT et al. in [8] is used. In contrast to the torso height optimization suggested in

[8], the method presented in this thesis determines kinematic limits through a pure geometric approach, which avoids manually tuned parameters and also significantly reduces the computational cost. In contrast to the aforementioned LIPM-based method by STASSE et al. [389], a varying torso height does not violate the assumptions of the used five-mass model. Compared to the method of NISHIWAKI and KAGAMI [325], kinematic feasibility is considered in this thesis only on position level (no velocity limits). However, a more complex leg model is used and not only an upper, but also a lower bound for the torso height is provided.

Finally, the motion generator proposed in this thesis is capable of producing dynamic multi-contact locomotion with (comparatively) high walking speeds. The effectiveness has been demonstrated by various experiments on the real hardware, where the robot copes with difficult multi-contact scenarios at up to 1.8 km/h, cf. [20] @ $t=10s$. This is (at the time of writing) much faster than comparable systems.

2.5 Computer Vision (CV)

The WPG presented in this thesis is meant to be coupled with a CV system providing information about the surroundings of the robot. Although visual perception is not in the scope of this thesis, the following introduces fundamentals and collects some remarks on the state of the art. This is meant to give a rough insight into the current activities in this field and also highlight joint research areas, where the boundaries between CV and WPG blur.

Simultaneous Localization and Mapping (SLAM) To achieve full autonomy, a CV system maintaining a model of the environment and localizing the robot within it (known as SLAM, already introduced in Footnote 10) is required. This information is then used by the contact planner to find an optimal contact sequence from the current position of the robot to a user-defined goal. Indeed, SLAM has a very large research community, where mobile robots are only one out of many applications. A core component of many SLAM pipelines is an efficient implementation of the *Iterative Closest Point (ICP)* algorithm [76], which aligns two point clouds as obtained from an RGB-D, LiDAR, or stereo sensor by minimizing a mean-square distance metric. This can be used to update a 3D surface reconstruction or to localize the sensor relative to the scene. At the time where SLAM was too expensive to be run onboard, some path finding algorithms have been developed which do not need an explicit environment model but rather directly use the (2D) image stream from the camera(s), see for example the already mentioned “tentacle”-based method by ROHE et al. and BUSCHMANN et al. [99, 360]. Meanwhile, real-time SLAM has not only become feasible, but also (partially) integrated into ASIC-accelerated low-cost sensors (e. g. [218, 219]), such that current research focuses more on various extensions building on top of SLAM. Examples are methods for completion of unseen/occluded areas from semantics [446] and automatic generation of semantic scene graphs for high-level scene understanding [448]. Both examples, [446] and [448], have been developed by WU et al. in parallel to the WPG presented in this thesis as part of the new multi-contact locomotion system of LOLA. Similar to most modern perception systems, the heavy-duty work is carried out by neural networks, which are trained with synthetic or real datasets on potent workstations and allow efficient real-time execution on the onboard hardware.

Accuracy In order to evaluate the accuracy of a SLAM system, a common strategy is to compare the results against the ground truth obtained from an external motion capture system such as the ones provided by *Vicon Motion Systems* [426]. On the one hand, the localization error can be directly obtained by comparing the localized pose with the motion capture pose as suggested by LASGUIGNES et al. in [268] (around 2 cm / 2° error for ICP-based localization using a combination of a low-cost tracking sensor [219] and a LiDAR). On the other hand, the accuracy of the

scene reconstruction (surface accuracy) can be evaluated by computing the local per-point difference between the RGB-D SLAM result and the measurements of a potentially more accurate LiDAR, cf. SCONA et al. [373] (around 5 cm error for a stereo camera system). Unfortunately, the depth accuracy of common low-cost sensors is still in the range of several centimeters for typical sensing distances of 2 m to 5 m, which can lead to severe disturbances in vision guided walking due to wrong contact expectations. Finally, although external motion capture is a convenient tool to obtain ground truth data, one has to keep in mind that these systems also have limited accuracy. In the author's personal experience, the official vendor specifications only match the relative accuracy over small distances and in the center of the tracked volume, while the absolute error over long distances or at the boundaries of the tracked volume is often much higher even for a well calibrated setup.

Segmentation and Pre-Processing Providing a point or triangle cloud representing the surroundings of the robot is in general not enough. Instead, the scene has to be segmented into meaningful primitives (floor, walls, obstacles, etc.) in order to be usable by a subsequent contact planner. In [96], BROSSETTE et al. give a good overview of their pipeline, which represents a very common approach for extraction methods based on point clouds. They obtain an input point cloud from the well-known *Asus Xtion Pro Live* RGB-D sensor [69], which gets filtered by voxelization in order to reduce the amount of data. Then they cluster points belonging to the same planar surface using a region growing algorithm, after which a convex hull algorithm is applied to obtain an environment model consisting of convex polygons. Similar to most other extraction pipelines based on point clouds, BROSSETTE et al. make extensive use of the prominent *Point Cloud Library (PCL)* [363], which implements most of the algorithms used by this pipeline. In order to choose from a set of potential contact surfaces, KAISER et al. propose in [225] to assign *affordances* (already introduced in Footnote 15). This metric can be used either to assist a human operator during high-level control in supervised autonomy (cf. [226]), or to formulate contact costs in a fully-autonomous contact planning process. For fast distance evaluations, e. g. in the context of collision avoidance, a coarse approximation of an obstacle's geometry by volumetric primitives can be useful. As an example, WAHRMANN et al. present in [429] a method to compute an SSV representation of an object from its corresponding point cluster. Apart from the description of an object's surface and volume, a classification (e. g. as floor, table, chair, etc.) helps the WPG to choose between possible path and contact options. The task of splitting a scene into disjoint parts and adding semantic labels to them is called *semantic segmentation*. The classification often relies on trained networks, cf. WU et al. [446].

Visual Servoing Aside of providing input data for ahead of time planning, CV in humanoid robots may also be used for visual servoing. In [406], TANGUY et al. propose a so-called *closed-loop RGB-D SLAM* framework for multi-contact scenarios. They align the perceived environment model with a 3D *Computer Aided Design (CAD)* model of the scene used during planning, to continuously update the current pose of the robot and the target contact points. The updated poses are fed into a QP-based whole-body controller compensating the discrepancies between the planned and executed motion. Another example is given by GIRAUD-ESCLASSE et al. in [171], where a whole-body motion generator based on a *Differential Dynamic Programming (DDP)*²⁴ algorithm is augmented by so-called *visual-tasks*. These tasks describe the error between the desired (expected) and actual (observed) position of visual features (e. g. points of interest identified by edges, corners, etc.) expressed in the camera's image plane. The error dynamics

²⁴*Differential Dynamic Programming (DDP)* as introduced by MAYNE in [303] is an optimal control algorithm for trajectory optimization. The main idea is to perform a preceding backward pass and subsequent forward pass in alternation until convergence is achieved. During the backward pass, a new sequence of optimal control inputs with regard to a user-defined cost function is computed (while moving backwards in time). The forward pass then evaluates the resulting trajectory, which serves as new reference for the following iteration.

of the visual features are derived from the image plane motion, which in turn is defined by the (known) camera motion. The gradient of the error dynamics is then fed into the DDP to minimize the costs introduced by the visual tasks.

Active Camera Control For robots with active DoF controlling the pose of the camera(s) (alias “head”), it is possible to track an object or certain point in space. Although this is mainly used in manipulation scenarios, cf. [181], it can be useful during locomotion for fixation of close-by obstacles, cf. [100, p. 84ff]. A general difficulty here is to stabilize the cameras since the active DoF in the head cause additional vibrations and motion blur.

2.6 Stabilization

The WPG proposed in this thesis assumes a preceding CV system delivering appropriate input data. Similarly, the WPG also expects its own output, i. e., the planned motion, to be processed by a low-level stabilization system before being sent to the hardware. The stabilization module, which is responsible for compensating any kind of real-world inaccuracies and disturbances, is not in the focus of this thesis. For an overview of the fundamentals and the state of the art in this research field, the literature review given by SYGULLA in his dissertation [401] is recommended. In the following, only few notes in direct relation with the topics of this thesis are added.

Hardware Design – Sensors Since online stabilization relies on proprioceptive data, the hardware design of a legged robot has to integrate the corresponding sensors. The most important components are the *Inertial Measurement Unit (IMU)*, often located in the torso/trunk, and a six-axis *Force-Torque Sensor (FTS)* at each EE meant to get in contact. The IMU provides (among other data) inclination and angular velocity as the most important indicators for balance in legged locomotion. Depending on the available budget and space, commercial IMUs range from *Micro-Electro-Mechanical System (MEMS)* based low-cost chips as used in handhelds to high-end, low-drift systems with fiber-optic gyroscopes as used in aircrafts and missiles, cf. [217]. A FTS delivers the current contact wrench at the corresponding EE, which can be used to control the load distribution between the EEs and also to synchronize the desired (planned) with the actual contact state. Additionally, one might include dedicated contact switches to determine if an EE contact is open or closed. Due to gravity and inertia effects, the same information might be difficult to derive from FTS data only (depends on the EE mass “after” the FTS). A further step would be to integrate a tactile skin, e. g. at the sole of the foot, such as the low-cost, sewed sensor presented by SYGULLA et al. in [399] or the one based on hexagonal printed circuit boards alias *HEX-O-SKIN* proposed by MITTENDORFER and CHENG in [307]. Apart from the complex mechanical integration of tactile sensors, designs based on a matrix of so-called *taxels*²⁵ also introduce high amounts of data, which have to be handled by the low-level communication bus. Finally, various humanoids such as TORO and TALOS integrate torque sensors in their joint modules to allow direct torque feedback. Indeed, there is a debate in the research community whether position- or torque-control should be used for legged robots. Here, the distinction between position- or torque-control indicates, whether the *inner* control loop of the joints uses position or torque as control variable, respectively. It does not provide information about the presence or type of an *outer* control loop or a higher level strategy such as hybrid force/motion control. In general, position-control is well-suited for fast and precise EE motion as required

²⁵In the tactile sensor community, the term *taxel* is often used to describe an elementary tactile transducer measuring pressure, friction, etc. at a certain point. Similar to *pixels* in an image, a matrix of *taxels* delivers a two-dimensional data field. Considering a tactile skin on the sole of a foot, this allows to measure the pressure distribution and, in case of uneven terrain, the contact area, i. e., the shape of the terrain underneath the foot.

in locomotion, while torque-control introduces virtual compliance in favor of impact mitigation and safety in human-machine interaction. A recent (purely simulative) comparison of position- and torque-based whole-body locomotion controllers is given by RAMUZAT et al. in [358] for the humanoid platform TALOS (provides both, a position- and a torque-control mode).

Software Design – Real-Time Capabilities For a quick reaction to disturbances, the stabilization module has to run with a high update rate. Typical cycle frequencies lie in the range of 200 Hz (e. g. DRC-HUBO+, cf. [223]) to 1 kHz (e. g. LOLA, cf. [400]) and are limited mainly by the IO capabilities of peripherals and the processing time of the stabilization algorithm. Note that in some systems the real-time communication bus connecting sensors and actuators with the control level (see Figure 2.12) runs at even higher frequencies of up to 4 kHz (e. g. LOLA, cf. [10]). This allows input-data filtering and output-data extrapolation (based on target velocities) for increased smoothness of the motion [10]. The high cycle frequency of the stabilization module implies that a higher level WPG has to be capable of providing trajectory data at this rate. In particular, it must not be slowed down by other planning tasks running in parallel. Thus, heavy duty loads such as the contact planning algorithm are typically offloaded to the autonomy level (see Figure 2.12) running on a different thread, process, or even PC.

Motion Generation – Inverse Kinematics (IK) As mentioned earlier, the tasks of *planning* and *control*, or – more precisely – motion generation and stabilization, are not always clearly separated. This holds true in particular for the IK. For whole-body approaches using a joint-space formulation, an IK is typically already integrated in the motion generation process in some way, e. g. to compute a valid robot pose from a set of contacts or indirectly in the context of a large-scale optimization. In contrast, motion generators providing abstract motion in the form of task-space trajectories require an explicit IK, which may be part of the subsequent stabilization system. In both cases, the IK typically allows to incorporate secondary goals due to the kinematic redundancy of most legged robots. Common velocity level IK methods with redundancy resolution are *Resolved Motion Rate Control (RMC)* introduced by WHITNEY in [434], which (locally) minimizes joint velocities, and its extension *Automatic Supervisory Control (ASC)* introduced by LIÉGEOIS in [283]. The latter is more popular, since it allows to additionally (locally) minimize a custom cost function by taking its corresponding gradient into account. This can be used to formulate secondary objectives such as preferring a certain “comfort” pose or avoiding joint limits and collisions. Furthermore, it is possible to prioritize objectives, either by simple weighting of the corresponding cost terms, or by formulating a strict hierarchy alias *stack of tasks*, see NAKAMURA et al. [318] and SICILIANO and SLOTINE [379] for corresponding extensions of ASC. In general, tasks incorporating unilateral constraints such as joint limits are often realized by regularization, e. g. through formulating the cost as a repulsive potential, cf. [371]. In [299], MANSARD et al. propose a method for strict enforcement of unilateral constraints in a task hierarchy.

2.7 The Humanoid Robots JOHNNIE and LOLA

The contents of this thesis are tightly coupled to the research platform LOLA developed at the *Chair of Applied Mechanics*, TUM. This institute has an almost 30 year long history with regard to the design, realization, and control of legged robots. Starting in 1993 with the six-legged robot MAX [342] based on the stick insect *Carausius Morosus*, further investigations led to the eight-legged, pipe-crawling maintenance robot MORITZ [344], the biped JOHNNIE [169], and finally its successor LOLA [417].

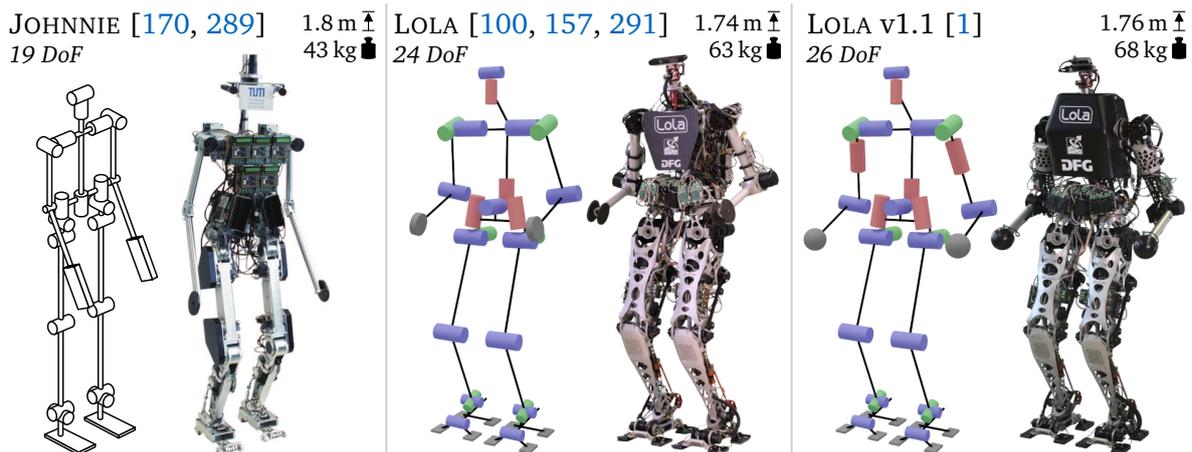


Figure 2.21: Humanoid robots (with topology) developed at the *Chair of Applied Mechanics*, TUM. From left to right: JOHNNIE (1997-2003), LOLA (2004-2019), and LOLA v1.1 (since 2020). Photo of JOHNNIE modified from [170]. The topology of JOHNNIE was taken from [343] and extended by the two DoF of the vision system (see “head”).

Previous Work – JOHNNIE The walking machine JOHNNIE (see Figure 2.21 left) was capable of autonomously climbing stairs and bypassing or stepping over obstacles, which made it one of the best performing humanoids at its time. With only 19 DoF²⁶ and 43 kg at a height of approximately 1.8 m, it is quite slender and rudimentary when compared to modern high-end humanoids. Since the focus was set on biped walking, the arms are rather simple omitting an explicit hand design or an active elbow joint. For motion generation, a LIPM/ZMP-based approach is used where the horizontal CoM motion is derived analytically from the decoupled LIPM-ODEs assuming a known, polynomial ZMP reference trajectory [288]. All algorithms are executed online on a single onboard PC, which receives only high-level signals from an external operator via *Ethernet* [290]. The CV system of JOHNNIE was developed by the *Chair of Automatic Control Engineering*, TUM. The data stream of two stereo cameras is used to build a two-dimensional global and a three-dimensional local (close vicinity) map of the environment. Appropriate step length, direction, and gait type (e. g. for stepping over obstacles) are transmitted from the CV system to the motion generator [292]. Unlike most modern CV solutions, the navigation system of JOHNNIE is restricted to structured environments, i. e., previous knowledge (to a certain extent) is required. A comprehensive description of the hard- and software of JOHNNIE is given in the dissertations of its main contributors, namely GIENGER [170] (design and realization), LÖFFLER [289] (simulation and control), and LORCH [292] (visual perception and step sequence planning).

Previous Work – LOLA The experience gained with JOHNNIE was used during the development of its successor LOLA (see Figure 2.21 center), which should enable autonomous, fast, and human-like walking together with an overall performance enhancement [417]. Prominent changes in the hardware design are the additional DoF in the pelvis, elbows, and toes. Note that active toe joints in humanoid robots are rare but not unique, cf. H7 [324] or PARTNER [402]. LOLA was originally 1.74 m tall, weighted 63 kg, and had 24 electrically actuated DoF. The very first version of the robot had 25 DoF since the “head” featured an additional *vergence* joint allowing to change the focus of the stereo camera system [291, p. 101ff]. With the transition to a new CV system, the stereo cameras were replaced by a single RGB-D sensor such that only the two DoF for *pan* and *tilt* remained [430, p. 22ff]. In contrast to JOHNNIE’s centralized control architecture, LOLA features a decentralized joint control driven by commercial servo controllers and

²⁶Most previous descriptions of JOHNNIE’s hardware mention only 17 DoF which do not include the two additional DoF in the “head” (vision pan and tilt).

custom *Distributed Sensor Control Boards (DSCBs)* communicating over a *Sercos-III* bus [156]. In a later hardware revision, the *Sercos* network was replaced by an *EtherCAT* bus, see [10] for details. A more detailed overview of the previous hardware configuration of LOLA (representing the starting point of this thesis) is given in Section 3.2.

The first CV system of LOLA was developed by the *Institute of Autonomous Systems Technology* at the *Universität der Bundeswehr München*, Germany, see ROHE et al. [360]. It is based on a stereo camera system providing input data for the “tentacle”-based navigation approach [99] already mentioned in Section 2.3. The second generation of LOLA’s CV system computes an explicit environment model from point cloud data obtained from an RGB-D sensor [6, 429]. The environment model is used by the second generation of LOLA’s navigation system performing an A*-based implicit graph search accelerated by a guiding 2D path to obtain an optimal foothold sequence [13, 199]. For avoiding collisions, obstacles are approximated by SSVs obtained from point clusters [429], which are used during ahead of time planning on the one hand [13, 199] and online trajectory adaption alias *Reactive 3D Collision Avoidance (RCA)* on the other hand [198]. An SSV-representation of the links is also used in the IK to avoid self-collisions [371].

In contrast to the LIPM-based motion generation approach in JOHNNIE, for LOLA a three-mass model in combination with a spline collocation method to solve the decoupled BVP for the horizontal CoM motion is used [98]. This allows a varying torso height, which can be optimized to respect kinematic limits in challenging maneuvers [8]. Moreover, various approaches for global kinematic optimization during motion generation have been investigated [9, 200]. As an alternative to the default IK-based method to obtain joint-space trajectories from an abstract reference motion [371], another approach incorporates the full multi-body model considering kinematic and kinetic quantities alias *inverse kineto-dynamics* [103]. This allows to exactly track the desired contact forces, however, at higher computational costs limiting the control loop frequency [103]. For disturbance rejection, a state estimator for the CoM and acting external force based on a KÁLMÁN filter [235] with LIPM dynamics is used [440]. Moreover, a three-mass model²⁷ allows to predict the robot’s motion in the near future [439], which is used to optimize footstep modifications for compensation of large disturbances [441, 442]. An overview of the combination of predictive collision avoidance and online disturbance rejection in LOLA is given in [202]. For compensating smaller disturbances occurring in early- or late-contact situations (e. g. stepping on an unseen small obstacle or in a hole), event-based control strategies triggering gait phase transitions and modifying phase timings have been investigated [11, 102].

Details on the hard- and software of LOLA (up to the beginning of this thesis) are collected in the dissertations of its main contributors, namely LOHMEIER [291] (mechanical design and realization), FAVOT [157] (electrical design and realization), BUSCHMANN [100] (simulation and control), SCHWIENBACHER [372] (simulation, collision avoidance, IK), EWALD [153] (event based reactive trajectory adaption), WITTMANN [443] (disturbance rejection), HILDEBRANDT [201] (navigation, collision avoidance, kinematic optimization), and WAHRMANN [430] (point cloud based CV). An overview of the previous state and skills of LOLA is given in [25 ◀].

Author’s Contribution – LOLA v1.1 Since the initiation of LOLA’s development in 2004 and the first public walking demonstration at the *Hannover Fair* in 2010, the robot continuously received various hard- and software upgrades. The probably most invasive modification (concerning hard- and software) was made in the context of the upgrade for multi-contact locomotion as presented in this thesis. This includes a complete redesign of the upper body to make it withstand the increased loads in multi-contact scenarios, which finally led to a more “beefy” overall appearance, see Figure 2.21 right. In order to distinct this new, multi-contact capable version from its previous configuration, it is referred to as “LOLA v1.1”.

²⁷Similar to the three-mass model used during motion generation (see Figure 2.17), but extended by explicit foot-ground contacts modeled by a spring/damper pair and a mass moment of inertia for the torso body.

2.8 Summary

To date, there exists a large variety of legged robots reaching from small, commercial toy companions manufactured in large quantities to highly complex, life-sized humanoid prototypes in research facilities. Apart from various DIY kits for hobby use, the community around legged robots mainly focuses on quadrupeds and bipeds. In the last years, tremendous progress has been made, which has also led to an increased interest of the western society in such machines. For quadruped robots, we recently observed a transition from research prototypes to commercial products, which can be operated not only by a trained expert in an industrial environment, but also by amateur users in the private sector. In contrast, large bipeds and full-sized humanoids are still subject to ongoing research. Nevertheless, they have become mature enough resulting in large companies starting to eagerly work on commercial adaptations which are closer to a mass-produced consumer product than the currently available low-quantity humanoids sold to research institutes as test platforms.

Concerning the hardware, there are numerous successful concepts which differ in kinematics, actuation, materials and design of structures. Since each approach has its own advantages, there is currently no clear “winner” design. Key parameters are power density, locomotion efficiency, robustness, and (structural) damping. Here, modern actuator concepts and materials (e.g. composites) promise the highest potential for significant improvement of overall performance. Full-sized humanoids may arrive in the consumer market in the mid-term future, thus new hardware designs should keep potential mass production in mind.

Although there is still much to do on the hardware side, the current bottleneck of high-end prototypes seems to be the software, in particular the way how the numerous joints have to be controlled in order to obtain a versatile, fast, robust, and efficient gait. Same as with the hardware, countless concepts emerged which gradually get refined and extended by the research groups promoting their corresponding strategy. A comparatively new trend are approaches based on machine learning which, however, focus more on the task of control rather than planning. In most classical frameworks, motion generation, i. e., connecting a discrete contact sequence by feasible trajectories, became fast enough to be run online. Current investigations focus on the proper and efficient evaluation of kinematic and dynamic feasibility in complex scenarios, especially multi-contact. In contrast, the contact planner (responsible for generating a discrete contact sequence from a given environment model) often violates the hard real-time requirements. Here, the common objective is to develop more efficient methods and to find a good compromise between generality, versatility, and optimality on the one hand and computational cost on the other hand. For a fully-autonomous system, all involved algorithms have to run under the restrictions of rather limited onboard computational power. An efficient software design is essential, however, related publications often lack of corresponding details.

In the field of visual perception, novel methods based on machine learning led to great advances with regard to the segmentation and understanding of complex scenes. Moreover, core techniques for localization and reconstruction have made their way into consumer products, which accelerates further research building on top of it. A general issue is the performance of mobile 3D sensors, which cannot compete with the versatility, adaptivity, and accuracy of human vision. However, the recent trend of integrating hardware acceleration for neural networks into modern microchips promises to push the limits for depth perception based on stereo vision.

Although the problems of biped and quadruped locomotion can be seen as “solved” on a very fundamental level, today even the best high-end systems do not even come close to the versatility and robustness of human and animal gait. In particular, additional hand support during biped locomotion – a simple and natural skill for humans – is still in its early days. This thesis tries to push the state of the art in this specific area.

Chapter 3

Hardware – A New Upper Body for LOLA

Parts of this chapter have already been published in [1, 3, 5, 28, 75].

This chapter describes the redesign of LOLA’s hardware in order to make it capable of multi-contact locomotion. Since the design of the robot’s hip, pelvis, and legs (see [291] for details) has proven to perform very well in versatile and dynamic walking, cf. [101, 202], it is kept unchanged while the focus is set on modifications to the upper body. The chapter starts in Section 3.1 with general design considerations followed by an overview of the previous state of the robot’s hardware in Section 3.2. In Section 3.3, the kinematic topology of the new arm design is optimized with regard to various multi-contact target scenarios. Subsequently, Sections 3.4 to 3.6 describe the mechanical and electrical design of the new upper body in detail. The realization and initial operation are presented in Section 3.7 after which the results are summarized and discussed in Section 3.8.

A condensed overview of the changes made to the hardware of LOLA has already been published in [1], see also the accompanying video presentation [19 ◉]. Moreover, the evaluation of the structural dynamics of the old and new hardware through an *Experimental Modal Analysis (EMA)* has been performed by BERNINGER et al. and is published in more detail in [75] and [5], respectively. The kinematic optimization of the arm topology presented in Section 3.3 is based in large parts on the master’s thesis of NEUBURGER, which has been published in [28]. An overview of the hardware upgrade is visualized in the video [22 ◉]. First real-world experiments with the new system demonstrating initial operation tests are presented in the video [21 ◉].

3.1 Preliminaries

The review of the state of the art provided in Section 2.1 gives an insight into the complexity of the electromechanical design of legged robots and in particular life-sized, fully-actuated humanoids. Developing a high-performance system requires multiple years of development and numerous design iterations (see for example the long history of ASIMO and ATLAS). Apart from knowledge gained from experience with previous prototypes, there are various general rules and thoughts which should be considered throughout the development. The following paragraphs try to summarize such common design considerations from a rather global perspective.

Legged Robots When compared to wheeled systems, legged robots feature a much higher degree of flexibility and versatility. This makes it difficult to estimate real-world loads (e.g. mechanical stress, joint torques, power consumption, etc.) within the design phase. Hence, it is hard to predict the overall performance of the final system (e.g. maximum speed, interaction forces, payload, etc.). As a consequence, the typical goal is to make the hardware as robust as possible while respecting a certain budget for mass, volume, and cost. The probably greatest impact on robustness is given by the choice of materials. A lightweight design using structures made of steel, aluminum, or titanium alloys is common practice. In order to further reduce weight, these “classical” materials may be replaced by high-strength composites such as CFRPs, which however are typically more difficult to process and introduce stricter constraints on the

geometric shape of a part when compared to 5-axis milling of solid metal. Due to the comparatively low strength, structures made out of conventional synthetic polymers (plastics) are almost exclusively used in rather small robots or for non-essential parts such as covers.

Another way to achieve a high strength-to-weight ratio for structures is to perform a topology optimization, which typically leads to rather complex geometries, see for example the sophisticated leg design of LOLA [291, p. 121ff]. As a drawback, the manufacturing of complex geometries by milling, casting, or 3D printing (see Figure 3.1) is expensive, especially for low quantities. Moreover, for casting and 3D printing, often a subsequent machining step (typically milling) is required to achieve sufficient surface quality and precision for functional surfaces such as junctions. In the author’s personal experience, this additional step accounts for about half of the total part costs and should not be underestimated.

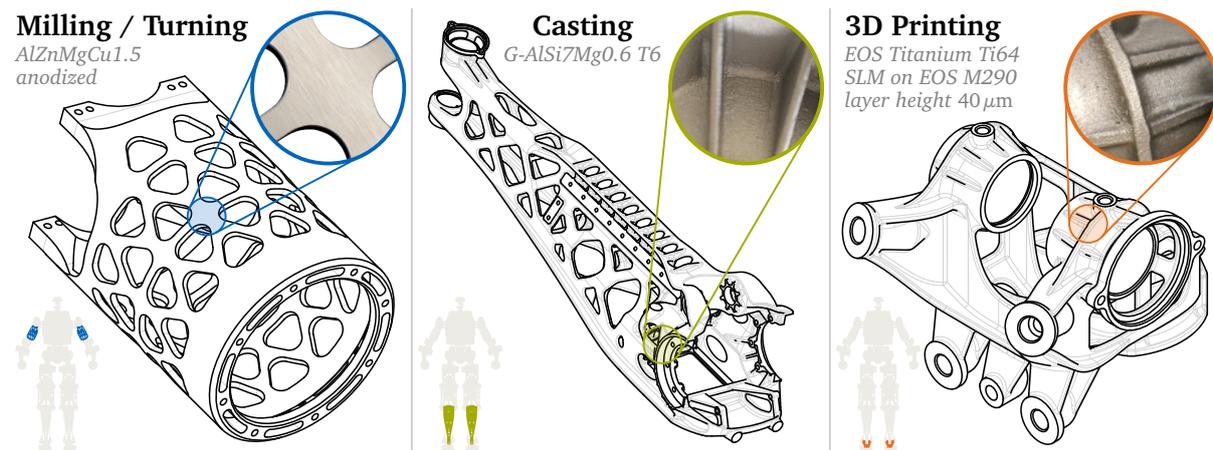


Figure 3.1: Lightweight structures with complex geometry (selection from the humanoid LOLA): geometric shape (CAD contour) and exemplary surface quality (photo) for parts manufactured by milling and turning (left), investment casting (center), and 3D metal printing (right). The upper arm structure (left) has been designed by the author of this thesis in the context of the upper body revision. The lower leg structure (center) and the ankle (right) are original designs by LOHMEIER [291]. Originally, the ankle was cast in aluminum (same alloy as for other leg structures), however, it was replaced in 2018 by a 3D printed version out of a titanium alloy, which is heavier but stronger.

For legged locomotion, not only the total mass but also its distribution is important. A common rule is to avoid heavy components (e. g. joint drives) in parts subject to strong accelerations (e. g. the EEs), but instead locate them close to the torso/trunk body. Moreover, there should be only as many DoF as necessary since each joint causes additional weight, undesired flexibility (gears), and additional effort within planning and control. Note that not every DoF has to be actuated, but may instead be realized as passive joint which is typically simpler, lighter, and may be able to store and release energy having a positive effect on locomotion efficiency. However, the integration of a passive joint typically forces the designer to optimize it for a certain cyclic gait, which drastically limits the versatility of the robot. An extreme example are passive-dynamic walkers as introduced in Section 2.1.

Concerning actuation, it is reasonable to use the same principle (electric, hydraulic, etc.) for all joints. This avoids additional weight due to multiple onboard power supplies. From today’s perspective, electric actuation should be preferred whenever the robot is meant to be deployed in domestic environments (no leakage, low noise, high safety). For inherent compliance and best impact mitigation properties, one might implement SEAs (already introduced in Footnote 2). However, in the author’s personal opinion, the rather poor performance in position tracking makes SEAs less suitable for long kinematic chains such as the limbs of a fully-actuated humanoid (positioning error accumulates). This does not relate to quadrupeds, which typically have a much lower count of DoF per leg and are therefore not too much affected by this issue. Note that also for a robot without SEAs, impacts can be mitigated (to a certain extend) by

placing passive compliance (e. g. rubber coatings / pads) at contact surfaces.

Unfortunately, some flaws in the hardware design of a legged robot first become visible during initial operation tests. Prominent examples are the transmission of vibrations and structural damping properties, which are very difficult to forecast by simulation due to the large amount of components and complex interconnections. This makes a reliable prediction of overall system dynamics infeasible and enforces multiple design iterations to fix related issues. Finally, for a commercial adaption, certain other aspects such as unit costs in the context of mass production, reliability, and durability have to be considered. Due to the high count of bodies undergoing complex motion, physical crashes and (self-)collisions of legged robots are more likely to happen than for wheeled systems. Hence, the hardware must not only be robust, but also simple to maintain, repair, and replace.

Humanoids Giving robots a visual appearance similar to the one of humans can have different motivations. On the one hand, some applications (e. g. elderly care) might require a certain look so that the robot is accepted by the person who interacts with it. On the other hand, a human-alike topology promises good locomotion performance in domestic environments. Similarly, a humanoid robot may be used to physically simulate a human which can be useful for testing prostheses or other medical devices. An example is given by OGURA et al. [327] with the humanoid WABIAN-2(R), which has been designed to simulate human kinematics and motion while supporting against a walking aid machine as used in rehabilitation.

One approach to mimic the appearance and motion capabilities of the human is trying to replicate its anatomy, i. e., its structures (skeleton), transmissions (tendons), and actuators (muscles). This requires to find electromechanical equivalents to highly complex components of the human body, be it complex multi-dimensional joints (e. g. the shoulder, see the approach made with BLADE [382] and KOJIRO [308]) or highly flexible and versatile actuators (e. g. artificial muscles, see the approach made with KENGO [62]). Obviously, a detailed replica of the human anatomy results in an extremely complex musculoskeletal humanoid which is difficult to control and maintain. Thus, a more common approach is to replicate only the “outer” kinematic behavior, e. g. by replacing spherical joints driven by a tendon- or parallel-mechanism with conventionally actuated revolute joints chained in series. In [277], LENARČIČ and UMEK proposed to experimentally identify the outer kinematics by visual tracking of human subjects. They use the resulting kinematic model to compute the reachable workspace of the human arm.

For a natural appearance, the geometric proportions of the segments (in particular the link lengths) should roughly match the human anthropometry, cf. [438, p. 82ff]. While this seems to be fulfilled for most humanoid robots, the mass distribution often significantly deviates from the human reference. An example is the humanoid LOLA (v1.1), where the lower body accounts for 62 % and the upper body for 38 % of the total mass²⁸. For comparison: a rough estimate for the average male human mass proportions is 42 % for the lower and 58 % for the upper body²⁹. Figure 3.2 presents a more detailed comparison. For the multi-contact capable LOLA v1.1, the main focus still lies on biped locomotion where the arms should only be used for additional support and not for heavy-duty manipulation. Thus, the relatively heavy lower body is a result of the high strength of the legs when compared to the arms. However, especially for biped walking machines, it is beneficial to make the upper body heavier than the lower body. Since the core dynamics of a biped walker can be modeled as an inverted pendulum (see Section 2.4),

²⁸The upper body of LOLA v1.1 is defined by the segments torso, afr|l, aar|l, arr|l, efr|l, vp, and vt. The lower body is defined by the remaining segments br, ba, hrr|l, har|l, hfr|l, kfr|l, sar|l, sfr|l, and zfr|l. See Figure 3.2 for details and Figure 3.3 for an explanation of the segment codenames.

²⁹This estimate is extracted from a study conducted by HERRON et al. [196], who measured the mass distribution of the human body using biostereometrics. The presented proportions for the lower and upper body are averaged over the data of the six male cadavers analysed in their study while assuming a homogeneous density (therein labeled as “Density 1”). The upper body is defined by the segments 1, 2, and 4 to 9, while the lower body is defined by the remaining segments 3 and 10 to 15. See Figure 3.2 and [196, p. 33] for details on the segmentation.

a high CoM (i. e. long pendulum length) slows down its internal dynamics effectively making stabilization easier. Since the actual mass distribution has a great impact on the dynamics of a biped, it has to be taken into account whenever gaits are compared.

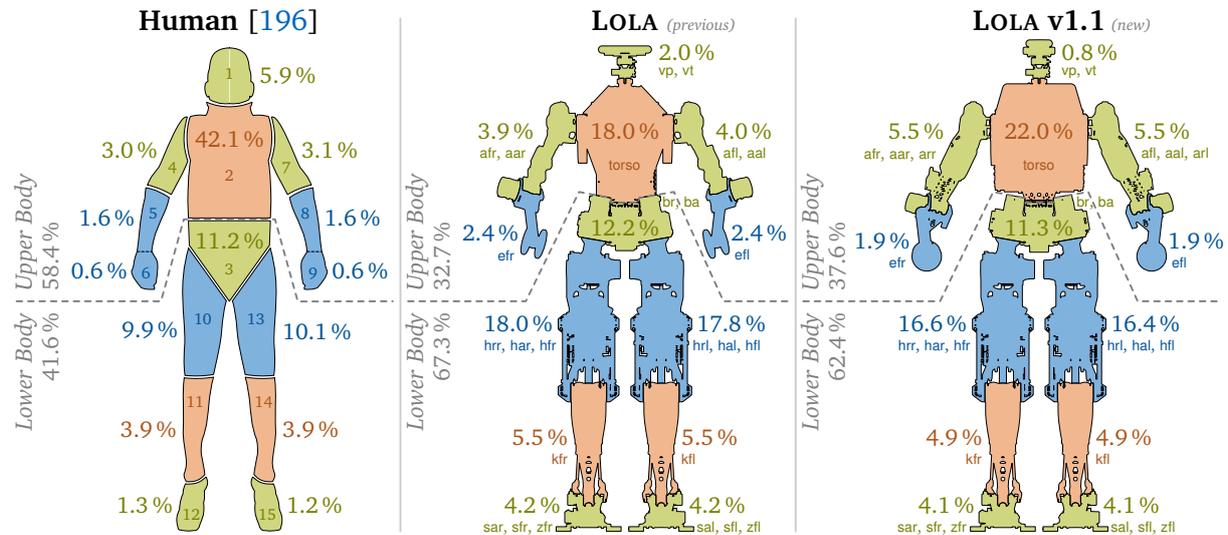


Figure 3.2: Mass distribution of the human body (left) in comparison to the humanoid robot LOLA (center) and LOLA v1.1 (right). The data (and contour) for the human body is extracted from HERRON et al. [196] (homogeneous density, average over six male cadavers). The data for the robot is obtained from CAD (verified by real measurements). For best comparability, the segmentation of the robot is chosen to be as similar as possible to the one of the human body.

Upper Body Design and Multi-Contact A core requirement for a humanoid to be capable of multi-contact maneuvers is that the workspace of the hands is large enough to reach potential contact areas in the vicinity of the robot. Moreover, it is not sufficient to reach a certain contact point, but instead it must also still be possible to perform small EE motions to compensate positioning or perception errors and to establish a certain desired contact force (see also the *manipulability measure* introduced by YOSHIKAWA [452]). Indeed, there are numerous studies on the design of humanoid arms, see for example BAGHERI et al. [73] who investigate the optimum joint arrangement for realizing shoulders. A common objective of these works is to replicate the versatility, dexterity, and strength of the human arm. In many cases, the resulting kinematic configuration is similar to the model identified by LENARČIČ and UMEK [277], which features a 2 DoF inner shoulder³⁰ representing the *sternoclavicular* joint, a 3 DoF outer shoulder representing the *glenohumeral* and *acromioclavicular* joint, and a 1 DoF elbow joint. A wrist is not considered in this model, however, in related works it is typically realized as a 3 DoF joint with intersecting axes similar to the last three joints in an industrial manipulator. In contrast to complex biomechanical models, LENARČIČ and UMEK do not draw explicit connections to the human anatomy but instead derive their model solely from experimental observation. This makes it a great reference for the design of robotic arms, which are meant to have similar (outer) kinematics but typically feature other principles of actuation when compared to the human reference. In practice, many designs omit the inner shoulder joint for simplicity, see for example ARMAR-III [58]. Examples for upper body designs featuring a (1 DoF) inner shoulder joint are ARMAR-4 [67] and the subsequently developed KIT dual arm system [353] as used for ARMAR-6 [68]. Here, the inner shoulder joint was added to maximize the dexterity in dual arm manipulation. Depending on the type of the contact (uni- or bilateral), an explicit “hand”, ranging from a single DoF gripper to a fully articulated hand, might be integrated. Independent

³⁰The influence of the inner shoulder joint on the workspace is extensively discussed by LENARČIČ and KLOPČAR in [278]. Moreover, KLOPČAR and LENARČIČ proposed in [254] to extend the model by an additional translational DoF between the inner and outer shoulder joint.

of this choice, it is highly recommended to place a six-axis FTS close to the part of the arm getting in contact such that the low-level stabilization module can match the desired and actual contact wrench. Finally, if contacts should not only be possible at the hands, but also at other parts of the arm (e. g. the elbow), the hull has to be designed accordingly.

Allowing additional hand contacts also sets higher requirements on the strength of the upper body, which otherwise would only have to sustain the gravito-inertial forces of its own components. Moreover, it is not sufficient to make the torso and arms strong enough to avoid stress induced damage. Instead, also the structural dynamics, i. e., the transmission and damping of impacts and vibrations, have to be considered since these can have a significant influence on the performance of the low-level control, cf. [75]. For simplicity, most upper body designs feature a stiff torso without any active or passive DoF. However, there are also torso designs with an active spine driven by a parallel mechanism such as CAUTO [106] (cable driven) and CHARLIE [258] (STEWART platform [397]). Even more sophisticated, high-DoF spines close to the human anatomy have been realized with the musculoskeletal humanoids KOJIRO [308] and KENGO [62]. For a new upper body design of a humanoid robot, one has to weight the advantages (increase of suppleness, flexibility, versatility, etc.) against the disadvantages (increase of weight, complexity, cost, etc.) of a torso with active DoF.

3.2 Starting Point

Since an in-depth description of the original electromechanical design of LOLA is contained in the dissertations of LOHMEIER [291] and FAVOT [157], only a brief overview is given in the following. It summarizes the status of the hardware *before* the revision of the upper body and focuses on the previous limitations in the context of multi-contact locomotion. Note that a brief overview of the total system (including software) has already been given in Section 2.7.

The previous hardware of LOLA was 1.74 m tall and weighted 63 kg. It featured 7 DoF legs (hip: 3, knee: 1, ankle: 2, toe: 1), 3 DoF arms (shoulder: 2, elbow: 1), a 2 DoF pelvis, and a 2 DoF head, which sums up to 24 electrically actuated DoF in total. Figure 3.3 presents the topology together with a list of codenames which are used to uniquely identify segments.

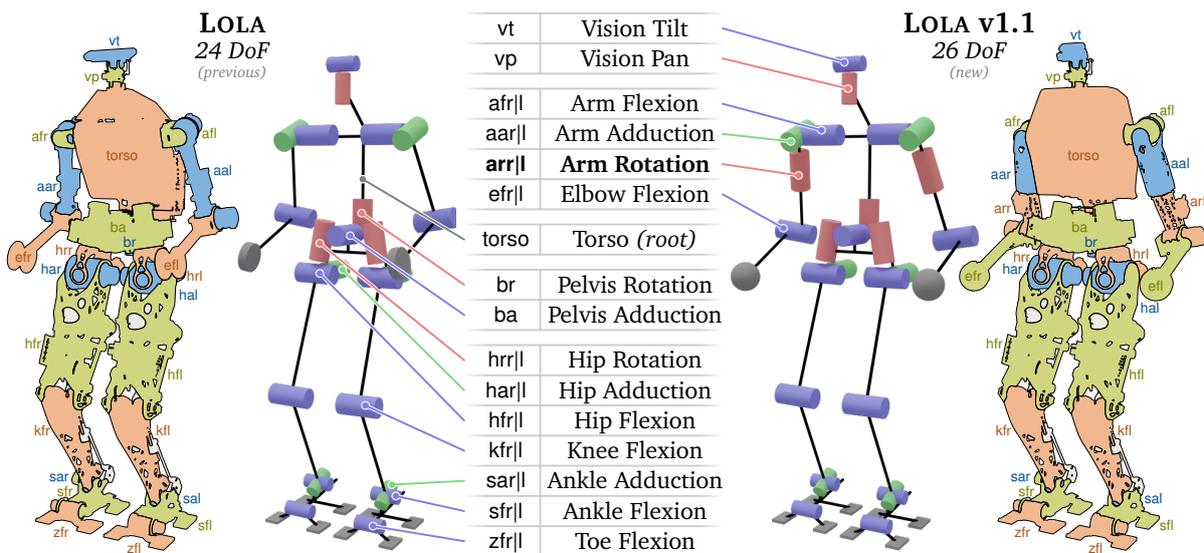


Figure 3.3: Kinematic topology of the previous hardware of LOLA (left, 24 DoF) and its newest revision LOLA v1.1 (right, 26 DoF). Each segment of the robot is assigned a unique codename (center) which is derived from its German name. Here, “r||” denotes *both* sides, right and left. The two topologies differ only in the upper body, where – additional to the change of link lengths – the new design features an additional “Arm Rotation” DoF (**bold**) on each side.

The primary material for custom lightweight structures of the robot are high-strength aluminum alloys as used in aerospace applications. Only critical parts with small cross section are made out of stainless steel or titanium. The main focus of the original design of LOLA was clearly set on the lower body, where highly complex structures of the pelvis and legs were manufactured by an external specialist for investment casting, see Figure 3.1 center. This applies only to certain fundamental structures, while the majority of parts was made by the in-house workshop through milling and/or turning from the solid. Since multi-contact locomotion was not considered at that time and the total mass of the robot had to be minimized, the upper body was kept rather rudimentary. This led to riveted tubular structures for the torso and arms, which were simple but sufficient for its purpose.

The drive system of LOLA is organized into modules, see [291, p. 96ff]. The modules have a custom, lightweight design integrating a brushless DC motor (stator and rotor as separate components), a *Harmonic Drive (HD)* [189] gear (circular spline, flex spline, and wave generator as separate components), a motor-side incremental encoder, a link-side absolute encoder, and a limit switch packed into a custom housing. The custom design allows to optimize the drives for minimum weight, i. e., maximum power density, with respect to their individual requirements. An exception are the low powered drives in the head (vp and vt), for which a commercial solution was chosen for simplicity. Moreover, not all of the custom drive modules are equipped with a high-ratio HD gear. Instead, the motors for the knee and ankle joints (kfr||, sar||, and sfr||) are linked to planetary roller screws and slider crank mechanisms, respectively. This allows to maximize the joint torque and to shift the mass upwards effectively reducing leg inertia. Accordingly, for these DoF the link-side absolute encoders are located directly at the driven joint, i. e., “after” the parallel kinematics, which eliminates errors introduced by these special mechanisms. Except for the ankles (sar|| and sfr||), all joints of the robot are back-drivable and show only minimal backlash.

Each drive is connected to a commercial servo controller by *Elmo Motion Control* [140]. The servo controllers are chained via an *EtherCAT* bus, which also interfaces additional sensors such as the main IMU³¹ located in the torso segment and the custom six-axis FTS and binary contact switches³² located in the feet (sfr|| and zfr||). Further details on the *EtherCAT*-based real-time communication system of LOLA have been published by SYGULLA et al. in [10].

The previous torso of the robot contained two industrial mainboards, each with an *Intel Core i7-4770S@3.1 GHz* quad-core CPU and 8 GiB *Random-Access Memory (RAM)*. The first board directly interfaced the *EtherCAT* bus and executed all real-time planning and control algorithms on a *QNX Neutrino 6.6 RTOS*. The second board was dedicated to the vision system (soft real-time) running on an *Ubuntu 16.04 GPOS* (without real-time patch), which obtained its input data from an *Asus Xtion Pro Live* RGB-D sensor located in the head (vt) and connected via *USB*. The two PCs communicated through an *Ethernet* network, which was also linked to a remote “operator” PC used for monitoring and sending high-level signals.

Since LOLA does not feature an onboard power supply, it is restricted to tethered operation. Note that a potential future upgrade with an onboard battery pack, e. g. located in the torso, would have a significant impact on the overall mass distribution (cf. Figure 3.2), which could make it more similar to human proportions. The previous power network of LOLA was fed by 48 V (joint servo controllers) and 80 V (motors except vp and vt) from an external rack. These input lines were additionally converted onboard to 24 V (motors for vp and vt) and 12 V (computing and various electronics). Further conversion to 5 V, 3.3 V, etc. for sensors and integrated circuits was performed locally on the corresponding *Printed Circuit Boards (PCBs)*.

³¹The main IMU of LOLA, an *iVRU-FC-C167* by *iMAR Navigation* [217], provides a *CAN* interface which is linked to the main communication bus by a commercial *CAN-EtherCAT* gateway, see Figure H.4.

³²The electrical components of the custom FTS in the feet of LOLA provide a *Serial Peripheral Interface (SPI)* [203]. The master of the *SPI* bus is represented by an auxiliary microcontroller located in the upper leg (hfr||), which pre-processes the data from the FTS and the binary contact switches (read from *General-Purpose IO (GPIO)* pins) and transmits it over the *CAN* and finally the *EtherCAT* bus through the *CAN-EtherCAT* gateway, see Figure H.4.

Hardware Revisions Note that the given description of the hardware refers to the state of LOLA *just before* the modifications presented in this thesis began. Indeed, starting from its original design by LOHMEIER and FAVOT in 2010, the hardware of the robot received numerous modifications, such as

- the redesign of the head replacing the stereo camera system by a single RGB-D sensor and the transition from 25 to 24 DoF (removing *vergence* joint),
- the redesign of the contact pads at the feet now featuring binary contact switches (≈ 30 N threshold per pad [401, p. 10]) and a new two-layer foot-sole material,
- the upgrade of the onboard PCs and corresponding OSs, and
- the upgrade of the low-level communication bus replacing the original *Sercos-III* bus by an *EtherCAT* bus and removing the custom DSCBs.

Details on these revisions and the transition of the hardware from its original state to the “starting point” of this thesis are omitted for brevity. However, the interested reader is referred to the more detailed summary on important hardware revisions given by SYGULLA in [401, p. 9ff].

Previous Limitations In consideration of the target multi-contact scenarios shown in Figure 1.1, the previous hardware of LOLA had several limitations which made an extensive revision of the upper body inevitable. First of all, the previous arm design featured only three joints (Figure 3.3, left). This was sufficient to compensate leg dynamics during fast biped walking by performing corresponding counteracting arm motions. However, the resulting reachable workspace of the hands is very limited, especially in the robot’s lateral proximity (Figure 3.6, left). Moreover, if we imagine a fixed torso segment, a 3 DoF arm also provides only a very limited set of joint configurations to reach a certain contact point (lack of redundancy). Additionally, the hands were realized by simple dummy weights (Figure 3.5, top) introducing artificial inertia. There was no dedicated contact surface or FTS to measure the contact wrench.

As a second major issue, the stiffness of the T-shaped, riveted tubular scaffolding representing the backbone and shoulders (Figure 3.11, left) was too low. This manifested itself by undesired oscillations of the upper body limiting the performance of the low-level controller even for regular biped walking without hand contacts [401, p. 72ff]. An EMA evaluating the overall structural dynamics of the robot showed that the first critical eigenmode visible to the low-level controller was characterized by bending of the torso frame (see [19 \bullet @ $t=2m18s$]) and occurred at 9.7 Hz. See Figure 3.18 (top) for a brief overview of the results of the EMA and BERNINGER et al. [5, 75] for details. Unfortunately, biped walking excites modes in such low frequency domains. For multi-contact locomotion where the stress on the torso is substantial, it was expected that the situation gets even worse.

Finally, the previous rudimentary upper body was only meant to carry core components (e. g. the IMU, onboard PCs, etc.) and provide a minimalist design for humanoid arms. Clearly, it would not have been able to withstand the increased loads occurring in a multi-contact scenario.

New Upper Body The aforementioned limitations of the previous hardware configuration required a complete redesign of the upper body. This allowed to change the kinematic topology such that it fits the desired multi-contact scenarios. As a first step, one has to decide whether the torso should contain active or passive DoF (e. g. an artificial spine). For LOLA, the moderate increase in suppleness and flexibility did not justify the increase in mass and complexity. Thus, it was decided to stay with a rigid torso. Instead, the focus was set on optimal arm kinematics. Since the two DoF in the pelvis (br and ba) have shown to significantly increase the kinematic capabilities during challenging maneuvers such as climbing stairs, they stayed mainly the same. However, the mechanical realization of the br segment (structures) was changed in order to make it strong enough for the increased multi-contact loads. Thus, the revision of the upper body basically included all segments “above” the br segment.

3.3 Kinematic Optimization of Arm Topology

The design of new arms raises the question for their optimal kinematic topology, i. e., the count and type (revolute/prismatic) of DoF, the joint arrangement, and link lengths. For LOLA, the new arms should enable supporting capabilities similar to the ones of humans. Although the design is inspired by the human arm, it does not try to replicate it. Since the focus lies on fast gaited locomotion while grasping and manipulation are not considered, we prefer simplicity, stiffness, and dynamic performance over dexterity and versatility. As a consequence, neither an inner shoulder joint, nor an articulated hand are considered.

Workspace Analysis In order to evaluate the kinematic capabilities of a certain arm design, it is reasonable to analyze the workspace of the hand. In the context of multi-contact locomotion, it represents a direct measure for the supporting capabilities and is in particular useful during contact planning for efficient reachability queries. Without further clarification, the term *workspace* is somewhat imprecise and ambiguous. Indeed, most related works build on the original definition by KUMAR and WALDRON [263] and distinguish between the

- *reachable workspace* as set of points the *Tool Center Point (TCP)*³³ can reach in at least one of the considered orientations and the
- *dexterous workspace* as set of points the TCP can reach in all considered orientations.

Following this definition, the dexterous workspace is a small (and often empty) subset of the reachable workspace. Since analyzing the dexterous workspace is complex, only the reachable workspace is considered in the following. However, being able to reach a certain point in multiple orientations is still regarded as beneficial and can be easily mathematically formulated as a corresponding metric indicating the dexterity of the mechanism at this point.

For evaluating the workspace of a humanoid arm, the torso body is typically defined as root of the kinematic chain and considered to be fixed in space. Certainly, if we consider the complete robot, the real reachable workspace of the hands is extended by moving the pelvis or torso accordingly (the feet may still be fixed to the ground). On the one hand, performing a whole-body motion to reach a certain point might conflict with other simultaneous tasks (in our case biped walking). On the other hand, considering the full kinematic topology of the robot significantly increases the computational effort. Thus, within this thesis, the analysis is restricted to the kinematics between the torso and the hand.

With an increasing count of DoF, the analytic computation of the reachable workspace becomes more and more tedious. Moreover, incorporating joint limits drastically increases the complexity, which makes an analytic evaluation infeasible in most cases. A straightforward approach for numerical computation of the reachable workspace is to perform a “brute-force” *Forward Kinematics (FK)* evaluation such as proposed by LENARČIČ et al. in [276]. Here, each DoF gets sampled between its corresponding joint limits leading to a huge set of possible joint-space configurations. For each configuration, the TCP pose is evaluated and registered in a three-dimensional voxel grid as discretized representation of the workspace. An interesting finding of the study by LENARČIČ et al. is that the workspace of the human arm kinematics shows the maximum volume and maximum compactness³⁴ at the same ratio of lower- to upper-

³³The *Tool Center Point (TCP)* is typically a point fixed to the EE which is of special importance for the specific task of the robot. For an industrial manipulator performing a pick-and-place task, the TCP is typically located between the fingers of the gripper. Despite its name, the term TCP is often used to describe a *Coordinate System (CoSy)*, i. e., a position and orientation, rather than just a single point. For LOLA, a TCP frame is defined for the end of each limb ($\{zfr\}$ and $\{efr\}$) and the head ($\{vt\}$). See Table H.2 for the definition of special CoSys of LOLA.

³⁴LENARČIČ et al. define the *compactness* of a workspace as the “ratio between the dispersion of the workspace elements [=reachable voxels – author’s note] and the dispersion of a sphere of the same volume” [276]. Here, the dispersion is computed as the average squared distance of all reachable voxels to the arithmetic center of the workspace. Following this definition, a spherical workspace has the maximum compactness of 1.

arm length (0.8 for an average human) [276]. This can be seen as an indicator that nature may consider similar metrics to optimize limbs within the process of evolution.

The main advantage of the brute-force FK approach is that – assuming a sufficiently fine-grained sampling of the DoF – the entire workspace is guaranteed to be found. As a drawback, the amount of evaluated configurations drastically increases (exponential growth) with each additional DoF making this method infeasible for long kinematic chains. A pragmatic solution to this problem was proposed for example by TONNEAU et al. in [412] where they use a limited set of randomly chosen joint-space configurations to compute TCP positions of the arm of HRP-2. They compute the convex hull of the resulting point cloud, which is subsequently simplified using the *Decimate* mesh modifier of *Blender* [85]. This leads to a rather coarse, convex approximation of the reachable volume, while the real workspace is actually non-convex. Within contact planning, they tackle this issue by intersecting the convex workspace approximation with a (simplified) collision volume of the robot effectively removing “false positives”, i. e., non-reachable areas which were incorrectly labeled as reachable [412].

A more elaborate method was proposed by ZACHARIAS et al. in [453]. In a first stage, they also perform a FK computation of TCP poses with randomly sampled joint-space configurations. In a second stage, they compute multiple IK solutions (with different initial configurations) for each reached grid cell to also explore the null-space. They discretize the workspace in the full SE(3), i. e., including orientations, which results in what they call the *capability map*. For the discretization of SO(3), they use the spiral point algorithm proposed by SAFF and KUIJLAARS in [365] to uniformly distribute points on a sphere inscribed in the currently investigated voxel. At each point on the sphere, they place a CoSy, which is further sampled by rotating it around its local z -axis (pointing towards the center of the voxel). The sampled CoSys of all points on the sphere represent the set of discretized orientations for the voxel. Thus, their workspace is represented by a five-dimensional grid (three parameters for position and two for orientation).

Local Metrics For evaluating the dexterity of a kinematic chain in a certain joint-space configuration (in the following denoted by $\mathbf{q} \in \mathbb{R}^n$) or for a certain workspace cell (in the following localized by the task-space coordinates $\mathbf{x} \in \mathbb{R}^p$), numerous measures have been proposed in literature – see PATEL and SOBH [341] for a compact overview. This paragraph briefly introduces four common local metrics which have been used during the kinematic optimization and the subsequent more detailed (high-resolution) analysis of the finally realized arm topology of LOLA. In the following, we assume that $n \geq p$ holds.

The *Reachability Index (RI)* is probably the simplest metric since it indicates if a certain discrete workspace cell is reachable (Boolean value) or how many joint-space samples hit this cell (integer value). In contrast to the other three metrics, it is linked to a certain workspace cell rather than a certain joint-space configuration. Possible definitions are

$$\underbrace{\text{RI}(\mathbf{x}) := \begin{cases} 1 & \text{if } \exists \mathbf{q} : \text{FK}(\mathbf{q}) = \mathbf{x} , \\ 0 & \text{else} \end{cases}}_{\text{Boolean variant}} \quad \text{or} \quad \underbrace{\text{RI}(\mathbf{x}) := |\{\mathbf{q} \mid \text{FK}(\mathbf{q}) = \mathbf{x}\}| \in \mathbb{N}_0}_{\text{integer variant}} . \quad (3.1)$$

Indeed, there does not seem to be a common, unique definition. Instead, measures based on the reachability are often defined with regard to the specific application. For example, ZACHARIAS et al. compute the RI for a certain voxel as the percentage of reachable orientations out of all possible orientations [453]. Within the scope of this thesis, the definitions from Equation 3.1 are used. Here the Boolean variant is sufficient to compute a surface representation of the reachable workspace, while the integer variant contains more information and can be used as indicator for the dexterity at a certain workspace cell.

The second local metric is given by the *Condition Index (CI)* which was proposed (among others) by MA and ANGELES in [296]. It is derived from the condition number (assuming the

spectral matrix norm) of the homogeneous task-space Jacobian J_{hom} and is defined by

$$\text{CI}(\mathbf{q}) := \frac{\sigma_{\min}(\mathbf{J}_{\text{hom}}(\mathbf{q}))}{\sigma_{\max}(\mathbf{J}_{\text{hom}}(\mathbf{q}))} \in [0, 1] \quad (3.2)$$

with σ_{\min} and σ_{\max} as minimum and maximum singular values of J_{hom} , respectively. The homogeneous task-space Jacobian J_{hom} is given by³⁵

$$\mathbf{J}_{\text{hom}}(\mathbf{q}) := \text{diag}(\alpha_1, \dots, \alpha_p) \mathbf{J}(\mathbf{q}) \in \mathbb{R}^{p \times n} \quad \text{with} \quad \mathbf{J}(\mathbf{q}) := \begin{pmatrix} \partial \dot{\mathbf{x}}(\mathbf{q}) \\ \partial \dot{\mathbf{q}} \end{pmatrix} \in \mathbb{R}^{p \times n} \quad (3.3)$$

and weights the individual task-space directions x_i of the task-space Jacobian \mathbf{J} by the scalars α_i . This allows to compensate dimensional inhomogeneity between translation (e. g. TCP position) and rotation (e. g. TCP orientation) within \mathbf{x} . The weights α_i may be chosen to 1 for rotational and $1/l$ for translational components. Here, l denotes a characteristic length of the mechanism, e. g. the maximum lever for an outstretched arm. Similarly, one can post-multiply \mathbf{J} with a diagonal matrix in case the mechanism has a mixture of revolute and prismatic joints.

Note that the CI is formulated as the reciprocal of the condition number of J_{hom} such that it is bounded [341]. The CI can be seen as a measure for the kinematic isotropy of the Jacobian, where the special case $\text{CI} = 1$ denotes an isotropic configuration and $\text{CI} = 0$ implies that the mechanism is in a singular configuration, i. e., $\text{rank}(\mathbf{J}) < p$.

The probably most prominent Jacobian-based metric is the *Manipulability Measure (MM)* originally introduced by YOSHIKAWA [452]. By replacing \mathbf{J} with J_{hom} in the original definition, the dimensional homogeneous MM is given by

$$\text{MM}(\mathbf{q}) := \sqrt{\det(\mathbf{J}_{\text{hom}}(\mathbf{q}) \mathbf{J}_{\text{hom}}^T(\mathbf{q}))} = \prod_{i=1}^p \sigma_i(\mathbf{J}_{\text{hom}}(\mathbf{q})) \in \mathbb{R} \quad (3.4)$$

with σ_i as the p singular values of J_{hom} . Note that the explicit computation of σ_i (e. g. through a *Singular Value Decomposition (SVD)* of J_{hom}) also allows to efficiently compute the CI as a cheap by-product. While the CI is a “better measure of the degree of ill-conditioning” [341], the MM considers the motion of the TCP in all workspace directions rather than only the two workspace directions related to σ_{\min} and σ_{\max} . YOSHIKAWA also showed, that the MM is proportional to the volume of the so-called *manipulability ellipsoid* as the n -dimensional hyperellipsoid in \mathbb{R}^p linked to the SVD of J_{hom} . Here, σ_i represent the magnitudes of the principal axes (left-singular vectors). Finally, it has to be mentioned that there exist numerous adaptations and extensions of the MM. As an example, KIM and KHOSLA propose in [247] extensions to counteract order and scale dependency.

While the Jacobian is a direct translator from joint- to task-space motion and thus, seems to be a reasonable basis for evaluating local dexterity, it does not contain information about non-linear constraints³⁶. An example for such constraints are joint limits, for which another specialized metric – often referred to as *Joint Range Availability (JRA)* – may be used. Similar to the RI, there exist numerous definitions for the JRA. Possible formulations are

$$\underbrace{\text{JRA}(\mathbf{q}) := \sum_{i=1}^n \frac{(q_i - q_{\text{mean},i})^2}{(q_{\text{max},i} - q_{\text{min},i})}}_{\text{cf. KAPOOR et al. [242]}} \quad \text{or} \quad \underbrace{\text{JRA}(\mathbf{q}) := \sqrt[n]{\prod_{i=1}^n \frac{\min(q_{\text{max},i} - q_i, q_i - q_{\text{min},i})}{\frac{1}{2}(q_{\text{max},i} - q_{\text{min},i})}}}_{\text{cf. NEUBURGER [28, p. 39]}} \quad (3.5)$$

³⁵In [296], MA and ANGELES investigate the design of platform manipulators (parallel mechanisms), hence, they use a different formulation with their Jacobian being the inverse of the one presented in Equation 3.3. However, the concept remains the same. Moreover, they propose to set the weights α_i to l for rotational and 1 for translational components of \mathbf{x} . This is equivalent to a scalar multiplication of J_{hom} from Equation 3.3 with l , which has neither influence on the condition number, nor the CI as its reciprocal.

³⁶There are, however, efforts to incorporate such constraints into the MM. In [421], VAHRENKAMP and ASFOUR formulate an augmented Jacobian integrating penalization terms for joint limits and obstacle- and self-collision. They use their extended manipulability concept to analyze the manipulation capabilities of ARMAR-III and ARMAR-4.

where $q_{\min,i}$, $q_{\max,i}$, and $q_{\text{mean},i} = \frac{1}{2}(q_{\min,i} + q_{\max,i})$ are the minimum, maximum, and mid-range angle/position of the i -th joint, respectively. Note that the JRA as defined by KAPOOR et al. [242] (Equation 3.5 left) is theoretically unbounded and strongly depends on the count of DoF. In contrast, the formulation by NEUBURGER [28, p. 39] (Equation 3.5 right) is bounded to $\text{JRA} \in [0, 1]$. In fact, the formulation as geometric mean pulls JRA to zero if one of the joints reaches its limit. Conversely, a JRA of one implies that all joints are perfectly centered between their corresponding limits. Due to these properties, the formulation of NEUBURGER was used during the optimization and analysis of LOLA's arm topology.

The metrics CI, MM, and JRA are linked to a certain joint-space configuration. In order to obtain a corresponding value for a certain workspace cell, one might use the minimum, maximum, or average of all redundant joint-space configurations pointing to this cell, cf. VAHRENKAMP et al. [420]. In our case, the per-cell average value was used during optimization. For the analysis of the final result, all variants (minimum, maximum, and average) are computed and stored in a dense octree (discretizing the workspace by position), which allows an efficient propagation of metrics from lower to higher levels of the octree.

Kinematic Optimization of LOLA's Arms The kinematic optimization of the arm topology of LOLA was performed mainly by NEUBURGER within the context of his master's thesis. His thesis has been published in [28] and gives a comprehensive explanation of the proposed framework for task-specific optimization of kinematic chains in general and the arm topology of LOLA as specific application in particular. Thus, this paragraph represents only a brief summary while the interested reader is instead referred to [28]. The main workflow of the proposed optimization framework is given by the following steps:

1. Specify the discretization, i. e., step sizes, of the workspace. Positions are represented by a three-dimensional voxel grid while orientations are discretized using the spiral point algorithm of SAFF and KUIJLAARS [365] similar to the method of ZACHARIAS et al. [453].
2. Specify a set of "tasks" by defining desired workspaces \mathcal{W}_{des} . This is done by choosing target volumes (derived from typical dimensions of tables, handrails, etc.) and corresponding orientations (direction of contact depends on surface normal), see Figure 3.4.
3. Specify a parameterized kinematic model of the robot. Typical optimization parameters are link lengths while the count, type (revolute/prismatic), and limits of joints are fixed.
4. Select a valid initial parameter set.
5. Evaluate the workspace for the current parameter set by computing the FK and local metrics (e. g. RI, CI, MM, and JRA) of randomly sampled joint-space configurations.
6. Compute the integral of the local metrics over all reached cells as proposed by KUCUK and BINGUL [256] (normalized by workspace volume). The individual cells are weighted against each other according to the task-specific desired workspaces \mathcal{W}_{des} (see Step 2). This results in a global, yet task-specific cost value.
7. Select a new set of parameters based on the governing optimization scheme and repeat starting from Step 5 or stop upon convergence. Meta-heuristic solvers based on *Particle Swarm Optimization (PSO)* [246], *Simulated Annealing* [252], and *Covariance Matrix Adaption Evolution Strategy* [188] showed comparable performance in our case.

The restriction to simple geometric parameters in Step 3 makes this scheme actually an "ordinary" parameter optimization. However, the whole optimization process was performed for multiple manually specified topologies, i. e., with different count, type (revolute/prismatic), and limits of joints. The resulting optimum parametrizations were compared against each other, hence, making the whole procedure a topology optimization. Note that the comparison of different topologies by a single global cost value requires all involved metrics to be *normalized*, i. e., independent of the count and type of DoF, which turns out to be difficult.

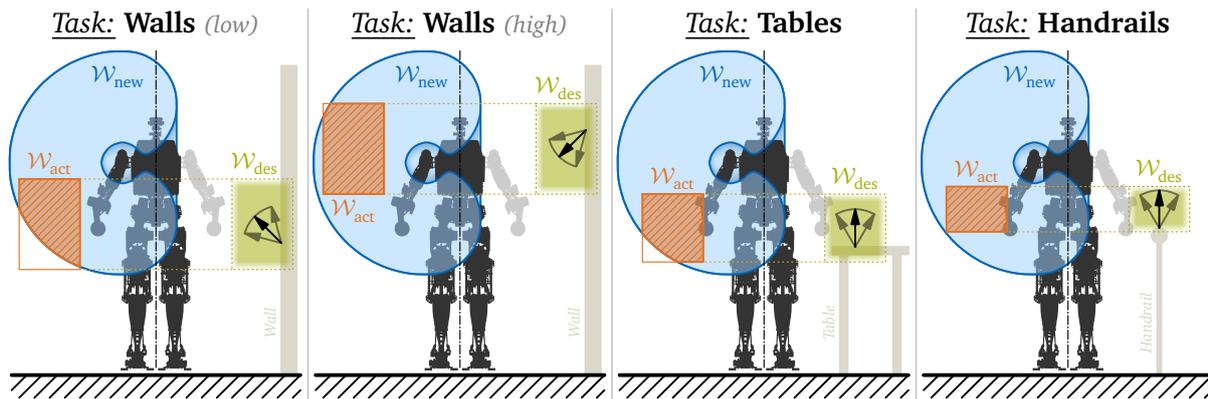


Figure 3.4: Specification of various multi-contact “tasks” by defining desired workspaces \mathcal{W}_{des} . From left to right: tasks for support against walls (low and high contact), tables, and handrails. The right side of each subfigure shows the target volume (green) and corresponding target orientations (arrows) which are defined by local (per-cell) scalar weights fading out at the boundaries. The left side of each subfigure shows the workspace of the new arm design \mathcal{W}_{new} (final result of the kinematic optimization; collisions are not considered) and the actually “matched” task-specific workspace $\mathcal{W}_{act} = \mathcal{W}_{new} \cap \mathcal{W}_{des}$ (right/left arm and tasks are symmetric).

In a first attempt of Step 5, a hybrid FK/IK approach similar to the one of ZACHARIAS et al. [453] was implemented. First, the FK is evaluated for a limited set of joint-space configurations leading to a sparse approximation of the workspace. Second, for all cells in the neighborhood of the already explored cells, a position-level IK is triggered effectively filling the “gaps”. The IK is based on the NEWTON-RAPHSON method [380, p. 133f] using the MOORE-PENROSE pseudoinverse [319, p. 41ff] to deal with redundancy and to be robust against singularities. Here, the pseudoinverse of the task-space Jacobian is computed through an SVD. The iteration is initialized with the already known close-by (neighbor) joint-space configuration. Unfortunately, this method showed a poor runtime performance due to the expensive exploration of neighbor cells at the boundary of the reachable workspace, see [28, p. 55f] for details. However, a pure FK-based approach within Step 5 led to satisfactory results in our case. Thus, the improvement of the developed hybrid FK/IK approach is omitted but may be part of future investigations to allow the optimization of longer kinematic chains where brute-force FK becomes infeasible.

Finally, it has to be mentioned that collisions are not considered during optimization. First, self-collisions of the arm are not possible due to the specific arrangement and limits of the joints. Second, collisions of the arm with the rest of the robot (e. g. the pelvis or hip) are rather unlikely for the chosen target workspaces, see Figure 3.4. Nevertheless, considering collisions would be straightforward by performing corresponding distance tests during sampling, e. g. using an efficient SSV approximation of the link geometry (see Appendix C). For doing so, the lower and upper arm each can be approximated by a line-SSV while a single point-SSV nicely matches the shape of the hand. Hence, the additional computational cost is expected to be small when compared to the cost of the involved SVD for computing the CI and MM.

Final Results – Topology The kinematic optimization by NEUBURGER resulted in a suggestion for an arm topology which is optimal with respect to the selected tasks and metrics. Although other design considerations were also taken into account for the finally realized arm kinematics, the resulting topology complies – except for certain details – with the computed optimum. A comparison of the previous three-DoF and new (finally realized) four-DoF arm design is shown in Figure 3.5. A noteworthy deviation from the optimal topology is the additional offset $l_{new,3}$, which was introduced in order to minimize the risk of collisions between the housing of the elbow drive ($q_{effr|}$) and close-by objects in the lateral proximity of the robot such as walls in multi-contact scenarios. Moreover, the hand is less likely to collide with the hip this way.

With the chosen shoulder topology, all three joint axes ($q_{afr|}$, $q_{aar|}$, and $q_{arr|}$) have a common

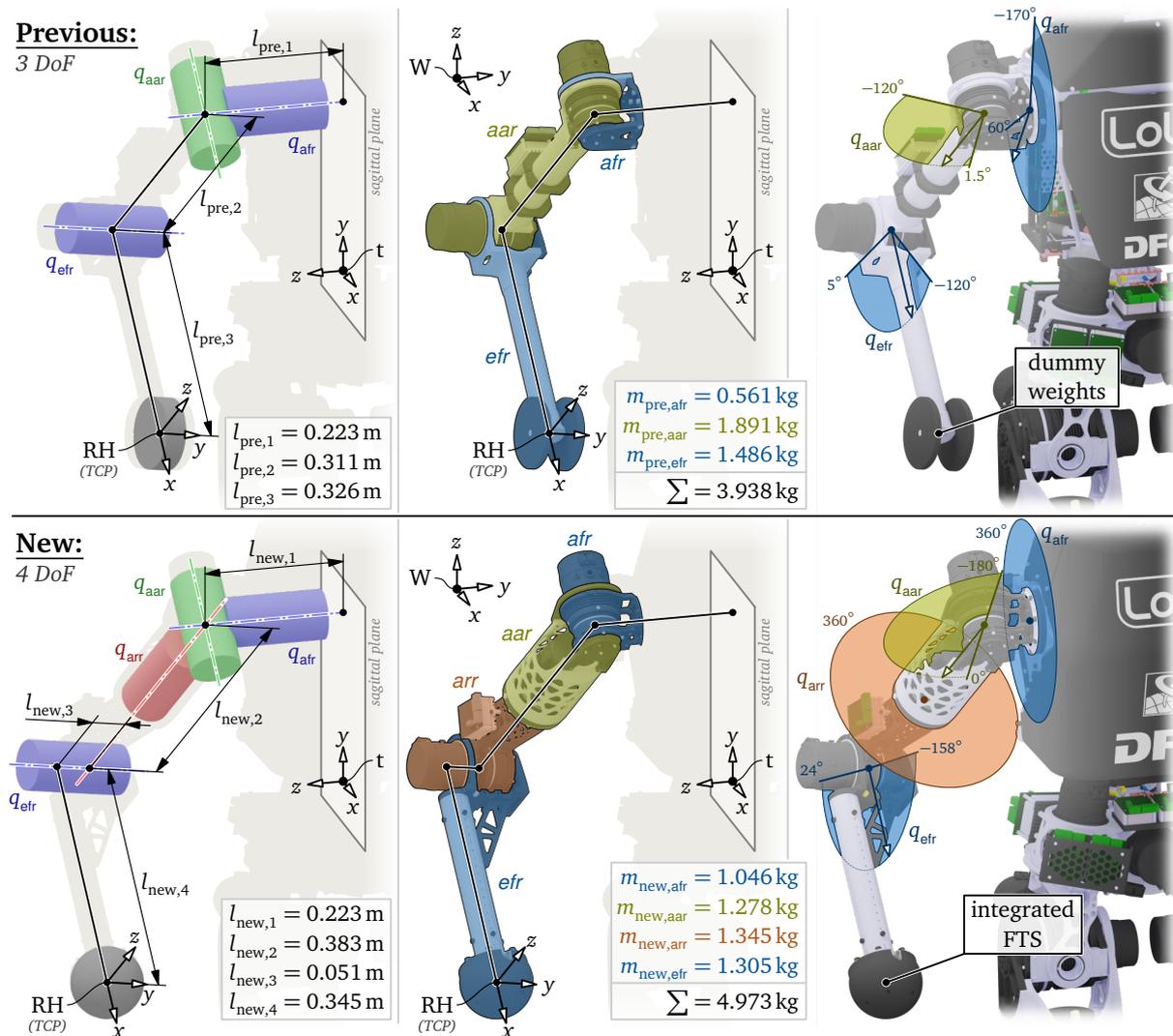


Figure 3.5: Comparison of the previous (top, 3 DoF) and new (bottom, 4 DoF) arm design of LOLA. From left to right: axis arrangement and main dimensions; per-segment and total mass; mechanical joint limits, previous dummy weights (arms are used for compensating leg dynamics), and new integrated six-axis FTS (required for multi-contact; simultaneously replaces dummy weights). The arms are shown in the configuration $q_{afr} = 40^\circ$, $q_{aar} = -15^\circ$, $q_{arr} = 0^\circ$, and $q_{egr} = -90^\circ$. For the workspace analysis, the kinematic chain between the torso frame “t” (in sagittal plane) and the TCP frame of the (right) hand “RH” is evaluated. Note that the torso frame “t” is rotated by 90° relative to the world frame “W” which complies with the CAD definition of the corresponding segment.

point of intersection. From a pure geometric perspective, one might argue that the specific order of these three DoF does not matter, since they behave like a single spherical joint. However, the particular arrangement has a significant impact on the volume (reachability depends on joint limits) and internal constitution (local metrics depend on Jacobian) of the workspace.

The kinematic limits of the DoF are a result of the corresponding mechanical realization. Note that the available joint range of the new arm design is significantly larger than the previous one and even exceeds human capabilities. While the values given in Figure 3.5 represent mechanical limits of the hardware, certain safety margins (5° in most cases) are applied in the software to avoid physical damage in case of an error. Moreover, to avoid “unusual” arm configurations in multi-contact scenarios, the stricter limits $q_{afr} \in [-90^\circ, 90^\circ]$ and $q_{aar} \in [-100^\circ, 0^\circ]$ are set via software and ensure that the elbow stays below the head. Certainly, these artificial restrictions may be disabled to restore the full joint range for other scenarios.

Final Results – Workspace Representative results of a high-resolution workspace analysis for the previous and new (finally realized) arm topology are visualized in Figure 3.6. Here, the task-space vector was chosen to $\mathbf{x}(\mathbf{q}) := {}_t\mathbf{r}_h(\mathbf{q})$ with ${}_t\mathbf{r}_h$ as the position of the TCP frame of the hand described in the (fixed) torso CoSy (see Figure 3.5 for the CoSy definition and Appendix A for the meaning of left and right subscripts). Collisions are not considered in this analysis.

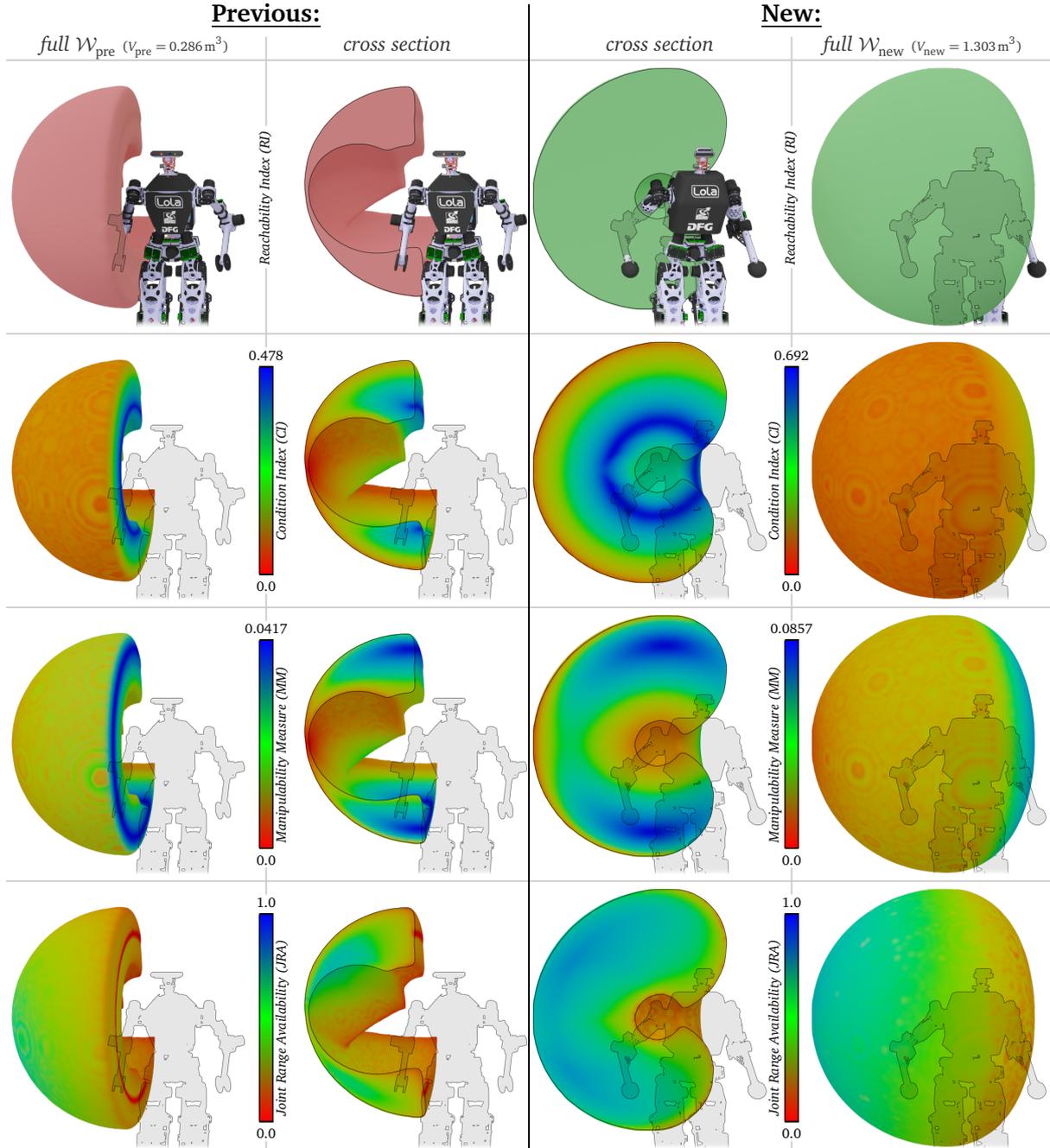


Figure 3.6: High-resolution workspace analysis of the previous (left) and new (right) arm topology. The kinematic chains are set up according to the topologies defined in Figure 3.5. The outer columns show the full workspaces \mathcal{W}_{pre} and \mathcal{W}_{new} , while the inner columns show their cross-sections in the y - z -plane of the torso frame. From top to bottom: reachability as volume, CI from Equation 3.2, MM from Equation 3.4, and JRA from Equation 3.5 (right). For each image in the last three rows, the local color is linearly interpolated between the global minimum and maximum of the corresponding metric (based on the local, per-voxel maximum of the metric) using the provided color gradient. Note that the color pattern appearing on the surface (workspace boundary) is a result of the workspace discretization and the post-processing. See the video [22] @t=2m6s for an animation explaining the post-processing and showing all cross-sections.

The source code for the high-resolution workspace analysis has been published as part³⁷ of the free and open-source *Beautiful Robot C++ Code Library (Broccoli)* by SEIWALD and SYGULLA [15] (see Section 7.2) and is completely independent of the implementation of the kinematic optimization by NEUBURGER [28]. The code is based on an efficient, parallelized brute-force FK and is capable of analyzing serial kinematic chains (without loops) consisting of an arbitrary sequence of revolute and/or prismatic joints. The library also provides an example application demonstrating the analysis of a common 7-DoF industrial manipulator simply by specifying its DENAVIT-HARTENBERG (*DH*) parameters [191, p. 347ff] using the notation of CRAIG [122, p. 65ff]. The example application is meant to encourage and assist others in using the code.

The dataset forming the basis of the visualizations shown in Figure 3.6 was created by the aforementioned *Broccoli* module. For a clear and intuitive three-dimensional presentation, several additional post-processing steps have been performed using *Blender*. These mainly include smoothing of the surface, color interpolation for cross section views, and high-quality rendering. The combined workflow of the high-resolution analysis using *Broccoli* (Step 1 to 7) and the post-processing using *Blender* (Step 8) can be summarized as follows:

1. Setup the kinematic chain from the torso to the TCP of the hand according to Figure 3.5.
2. Auto-compute³⁸ (or manually specify) the count of uniformly distributed joint-space samples for each DoF. The total count of samples is given by the product of all individual joint samples (evaluation of all possible combinations). In our case, this results in $1.1 \cdot 10^8$ and $1.5 \cdot 10^{11}$ total samples for the previous and new arm topology, respectively.
3. *Pre-sampling*: compute the FK (only the TCP position) for a small subset of all discretized joint-space configurations. In particular, evaluate only every second sample for each of the n DoF. This results in processing only $(1/2^n)$ of the total count of samples (with $n_{\text{pre}} = 3$ and $n_{\text{new}} = 4$). From the resulting point cloud, compute a bounding box enclosing the reachable volume (with applied safety margins).
4. Setup an octree representing the discretized workspace. The count of levels (in our case four) and the dimensions of the bottom-level cells (in our case 1 cm for x , y , and z) is manually specified by the user. The top-level grid size is automatically chosen such that the octree encloses the bounding box computed in Step 3. This ensures that the octree is only as large as necessary which minimizes memory consumption.
5. *Main-sampling*: compute the FK (including J_{hom}) for the full set of all discretized joint-space configurations. For each sample, additionally compute the local metrics CI, MM, and JRA and update the corresponding minimum, maximum, and mean value in the corresponding bottom-level cell of the octree. Additionally, each voxel stores the count of joint-space samples reaching it (alias RI).
6. Update the octree by propagating the minimum, maximum, and mean values of the local metrics upwards, i. e., from the bottom- to the top-level. The filled octree is equivalent to similar grid-based workspace representations such as the *capability map* [453].
7. Compute a surface representation (as triangle mesh) of the reachable workspace. For this, an extension³⁹ of the well-known *Marching Cubes* algorithm [293] is used.

³⁷See the class `TaskSpaceEvaluator` in the module `analysis` of *Broccoli*.

³⁸The required resolution for sampling a certain DoF is computed by evaluating the maximum possible TCP movement caused by this joint while considering the maximum lever arm of the subsequent links and a user-defined maximum allowed displacement (e. g. based on the chosen workspace discretization, i. e., voxel dimensions).

³⁹In particular, the method `CGMeshFactory::createVolumeMarchingCubes` from the module `geometry` of *Broccoli* is used, which represents an extension of the original *Marching Cubes* algorithm of LORENSEN and CLINE [293]. The core *Marching Cubes* algorithm is realized based on the implementation of BOURKE [93]. The *Broccoli* implementation additionally interpolates per-vertex colors from the grid (color gradient from local metrics), computes smooth per-vertex normals from the local density gradient (using the RI as density with surface threshold 0.5), and provides an *indexed* mesh (reusing vertices among triangles to save memory by avoiding duplicates).

8. Smooth the surface representation by applying the *Smooth* mesh modifier of *Blender*. For creating colored cross section views, apply the *Boolean* mesh modifier (subtract a cube) and use a *Point Density* based shader to interpolate colors from a corresponding point cloud carrying per-vertex colors (extracted from the octree).

Note that storing the data in the form of an octree gives the user access to different levels of detail. This comes in handy for using the same dataset also for coarse but fast real-time queries. Due to the spatial resolution of only 1 cm, a very high count of joint-space configurations has to be evaluated (see Step 2). Thus, the “pre-sampling” (Step 3) and “main-sampling” (Step 5) as main workloads of the presented process are parallelized, which – thanks to an efficient parallel batch access to the octree – turns out to scale linearly with the count of available CPU cores. In our case, the evaluation of the previous and new arm topology took 69.5 s and 23.8 h ($\approx 1.7 \cdot 10^6$ samples per second) on a 56-core server⁴⁰, respectively. The much higher runtime for the new arm topology complies with the much higher total sample count. The higher sample count in turn is mainly a result of the additional DoF and the larger joint ranges. Moreover, the greater span of the arm ($l_{\text{new},1} + l_{\text{new},2} + l_{\text{new},4} = 0.951$ m vs. $l_{\text{pre},1} + l_{\text{pre},2} + l_{\text{pre},3} = 0.86$ m, see Figure 3.5) requires smaller stepsizes for joint-space sampling.

3.4 Actuation and Sensing

Once the kinematic topology has been determined, the next step towards a new upper body is the placement of core components such as actuators and sensors. While the actuators represent the mechanical realization of an active joint, the sensors transmit valuable data describing the state of the robot and its environment to the locomotion framework. Within this section, the focus lies on “primary” sensors, which excludes the obvious ones integrated in the joint drives such as rotary encoders. For the upper body, this mainly includes the FTSs in the hands, the IMU in the torso, and the vision sensors in the head.

Actuators While the kinematics of the new arms has been optimized with respect to certain target scenarios (see Figure 3.4), it is much more difficult to derive required dynamic capabilities, i. e., maximum joint torques, from such coarse task descriptions. For gaited multi-contact locomotion, the interaction force between the hand and the environment may reach from a gentle touch (just as precaution to be ready for a potential external disturbance) to a “full” support (e. g. for difficult maneuvers such as stepping over large obstacles). Thus, it is not clear which maximum actuator torques are “sufficient” in the general case. Since LOLA shall have supporting capabilities similar to humans, a rather pragmatic solution is chosen instead. In particular, common maximum interaction forces are obtained from the military and NASA standards *MIL-STD-1472G* [132] and *NASA-STD-3000B* [320], respectively. These standards describe the maximum permitted interaction force for human-machine interfaces in the context of military equipment design. Although manipulation is not considered within this thesis, these standards give a good insight in the “minimum” physical capabilities of (adult, male) humans. In particular, [132, p. 80, 330] specifies (maximum) torques T_{human} that can be applied by 95 % of male subjects in the General Forces of the United States of America. It seems reasonable, to design the arms of LOLA such that they surpass these “minimum” capabilities.

As mentioned earlier, the drive system of LOLA (except for the knee and ankles) is organized into modules enumerated from A to G with A as the largest (hip flexion) and G as the smallest (toe flexion) drive [291, p. 96ff]. A brief summary of the components of each drive has already been given in Section 3.2. In order to be consistent with the rest of the hardware, the original

⁴⁰The used server hardware contains two 28-core *Intel Xeon E5-2660v4@2.0 GHz* CPUs on a dual socket board running *Ubuntu 20.04* in a virtualized environment.

joint design by LOHMEIER is also used for the DoF of the new upper body. In particular, the modules D and E are used for the joints $afr||$ and $arr||$, while the module F realizes the joints $aar||$ and $efr||$, see Table 3.1 for the specification and Figure 3.7 for the placement of the actuators. This choice significantly reduces complexity and allows to benefit from the experience gained through the last 10 years with these drive systems. Moreover, complete modules from the previous hardware can be reused such that only two new drives had to be built for the revision of LOLA's upper body. In accordance with the performance of the human arm, the drive with the highest torque rating (module D) is used for the DoF $afr||$. Since module E and F share the same T_{perm} and T_{peak} (limited by the motor), their assignment to the remaining DoF $aar||$, $arr||$, and $efr||$ is not prescribed by their dynamic performance. Instead, their assignment is derived from the best mechanical integration (mechanical interface differs between module E and F). As already explained in Section 3.2, the pelvis segments br and ba basically remain unchanged except for the structural parts of br . Thus, they are still actuated by drive modules of type D which, however, received maintenance during the hardware upgrade.

Table 3.1: Actuator specification of the new 4 DoF arm design. The maximum permanent and peak torques T_{perm} and T_{peak} are determined from the corresponding official ratings of the stator/rotor pair from *Parker Hannifin* [340] and the HD gear [189] (the torque-limiting component is highlighted in red). The equivalent (maximum) human torque T_{human} is computed from the interaction forces specified in [132, p. 80] and the (average) lower/upper arm lengths specified in [132, p. 330]. The mass of the modules has been determined by measuring the real hardware.

Joint	Drive	Motor / Gear	T_{perm}	T_{peak}	T_{human}	Mass
Arm Flexion $afr $	Module D	K064050-8Y1 HFUC-20-100-2A	49 Nm	147 Nm	61 Nm	1.14 kg
Arm Adduction $aar $	Module F	K064025-GD HFUC-17-100-2A	31 Nm	99 Nm	30 Nm	0.97 kg
Arm Rotation $arr $	Module E	K064025-GD HFUC-20-100-2A	31 Nm	99 Nm	34 Nm	1.05 kg
Elbow Flexion $efr $	Module F	K064025-GD HFUC-17-100-2A	31 Nm	99 Nm	49 Nm	0.97 kg

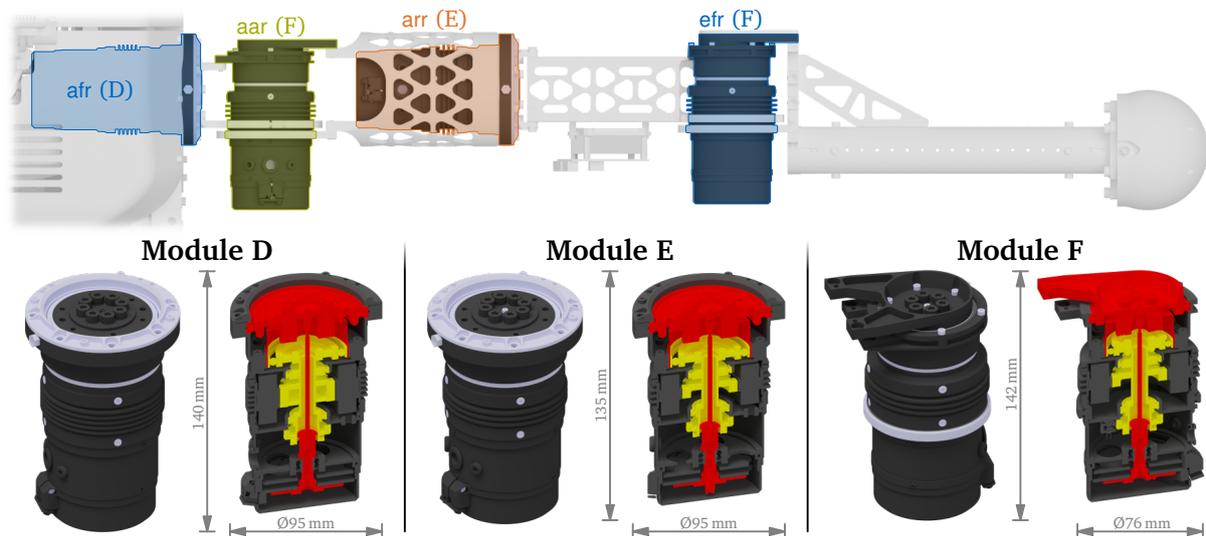


Figure 3.7: Actuator placement in the arm. The DoF are actuated by drive modules of type D ($afr||$), E ($arr||$), and F ($aar||$ and $efr||$). The arm is shown in the configuration $q_{afr} = 0^\circ$, $q_{aar} = -90^\circ$, $q_{arr} = -90^\circ$, and $q_{efr} = 0^\circ$. The section views show the housing with stator (gray), shaft with rotor (yellow), and output (red). See [291, p. 96] for details.

In order to maximize the torque-to-mass ratio, some other humanoids integrate double (or even triple) motor drive systems for certain joints (typically the knee), cf. JAXON [255] or HRP-5P [241]. Although being lighter, this concept increases overall complexity and typically re-

quires more space. For LOLA's arms, the outer geometric shape of the drives is to be minimized in order to avoid collisions in multi-contact scenarios. Therefore, each joint is driven by a single motor. Moreover, the drive modules are placed in a way, such that collisions with the environment are impossible or at least unlikely (e. g. $\text{afr}||$ inside the torso, $\text{arr}||$ inside the upper arm, and $\text{efr}||$ with the offset $l_{\text{new},3}$, see Figure 3.7).

A severe disadvantage of reusing pre-existing drive modules is that no (major) improvements are possible. An example is the lack of a dedicated torque-sensor. Although explicit torque-control is currently not investigated by the research group around LOLA, the author highly recommends to integrate torque-sensors in the joints of a potential successor of LOLA such that alternative control concepts can be tested. For future developments, it also has to be evaluated if commercial, integrated servo drives can be used instead of complex, custom built actuators. To the author's best knowledge, commercial modules still do not reach the high torque-to-mass ratio of LOLA's drives. This might be due to the different margins of safety for dimensioning a versatile and reliable mass-product versus a heavily specialized component for a research prototype. Since legged robots become more and more popular (cf. quadrupeds), it is likely that the market for lightweight, high-torque motors specialized for this particular application will grow.

Force-Torque Sensors (FTSs) In order to measure and control the contact wrench in multi-contact scenarios, a six-axis FTS is placed inside of each hand, see Figure 3.8. For this purpose, the commercial sensor *FTE-Axia80-Dual Si-200-8/Si-500-20* [369] from Schunk has been selected. Its key advantages over competing products are its compactness and its aluminum housing which leads to a relatively low mass of only 276 g (actual measured weight). Moreover, it is fully integrated providing an *EtherCAT* interface such that no external evaluation or translation unit is required. The sensor supports two calibrations affecting the range of measurement which can be selected via software, see Table 3.2.

Table 3.2: Specification of the six-axis FTS *FTE-Axia80-Dual Si-200-8/Si-500-20*. For the experiments conducted within the scope of this thesis, the second calibration option (highlighted in green) has been used. This enables the measurement of higher loads (e. g. due to impacts) by sacrificing absolute accuracy (at 22 °C the error is specified to be < 2% of the upper limit of the measurement range [369]).

Calibration	Range of Measurement			Overload			Sampling Rate
	$F_{x y}$	F_z	$T_{x y z}$	$F_{x y}$	F_z	$T_{x y z}$	
Si-200-8	± 200 N	± 360 N	± 8 Nm	± 2.5 kN	± 4.5 kN	± 100 Nm	up to 4 kHz
Si-500-20	± 500 N	± 900 N	± 20 Nm				

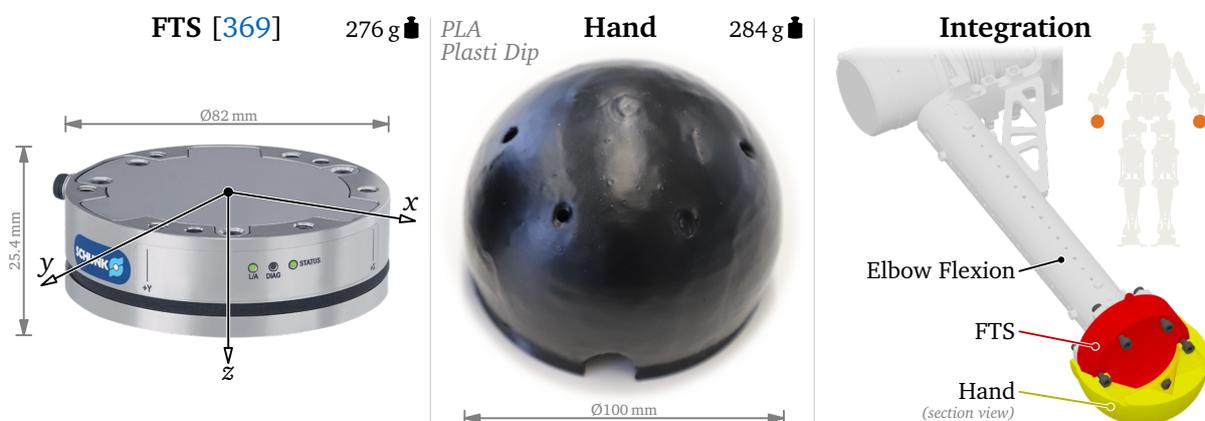


Figure 3.8: Integration of the six-axis FTS *FTE-Axia80-Dual Si-200-8/Si-500-20* measuring the local contact wrench at each hand. From left to right: photo of the sensor (modified from [369]) with definition of the sensor's local CoSy, enclosing 3D printed "hand" with rubber coating, and placement within the $\text{efr}||$ segment.

The sensor is enclosed by a spherical “hand” structure (0.1 m in diameter) acting as contact surface in multi-contact experiments, see Figure 3.8 center. The structure is designed as a single part which is 3D printed through *Fused Deposition Modeling (FDM)* using *Poly lactide (PLA)* as material. This makes the part very light and simple to replace. Here, the low mass minimizes parasitic effects on the FTS measurement due to gravity and inertia. Within the multi-contact scenarios considered in this thesis, the hand does not move during contact, thus inertia effects are (theoretically) eliminated. However, during the transition between contacts, a quickly moving and heavy hand would cause significant inertia forces, which could be interpreted by the robot as an unintended contact with the environment. In contrast to the feet, the hands of LOLA do not feature explicit contact switches, but instead detect the state of a contact (open/closed) by comparing the measured forces against a certain user-specified threshold. Making the hand easy to replace is important since all parts getting in contact with the environment (such as the foot sole) are subject to damage and extraordinary wear. The surface of the hand is coated with liquid rubber commercially known as *Plasti Dip* [345]. The approximately 0.5 mm thick coating significantly increases the friction coefficient compared to the blank PLA surface. Unfortunately, it does not introduce any considerable damping. The multi-contact experiments conducted so far suggest, that the integration of an additional (soft) damping layer similar to the foot sole (see [396]) should be considered in future. Unfortunately, the spherical shape of the hand makes this a non-trivial task. This holds true also for the integration of a contact switch or tactile skin.

For measuring the contact wrench in the foot of LOLA, a custom lightweight FTS (originally designed by SCHWIENBACHER [370]) was used by LOHMEIER [291, p. 135ff]. This was mainly due to the lack of appropriate commercial products at that time. Common off-the-shelf six-axis FTSs – which are typically designed for the EE of an industrial manipulator rather than a mobile robot – often have a relatively low range for sensing torques when compared to forces. However, due to the large footprint of LOLA (currently 0.276 m × 0.22 m), rather high torques may occur (about 95 Nm for single-legged standing on tiptoes). Moreover, the integration in the (strongly accelerated) foot of a humanoid robot sets high requirements on the mass of the sensor such that lightweight, aluminum-based solutions should be preferred. This led to the custom design which is specialized for the loads that occur in biped walking. The sensor additionally features an explicit overload protection mechanism to withstand large impacts [291, p. 70]. Meanwhile, the variety of commercial lightweight and fully-integrated FTSs matching these specific requirements increased. Thus, for future developments it should be considered to use an off-the-shelf product instead of a custom (typically less mature and reliable) solution.

Inertial Measurement Unit (IMU) The main property of a mobile robot is that the root segment is not fixed to the environment. Thus, besides the n joints, a legged robot has six additional passive DoF representing the position and orientation of the root segment relative to the (inertial) world. For walking on level ground, the horizontal position and rotation around the vertical axis represent the core parameters to localize the robot within its environment. Hence, these components of the root segment’s pose are essential for navigation, see Chapter 5. The remaining components are typically used as indicators for balance: the rotation around the horizontal axes (alias inclination) indicates tilting while the vertical position may be used to detect loss of contact with the ground (falling/jumping) although this is more reliably detected by contact switches and/or a FTS located in the foot instead. Accordingly, these components are primary inputs for stabilization, see Section 4.6.

In order to measure these six passive DoF in LOLA, the IMU *iVRU-FC-C167* by *iMAR Navigation* [217] has been selected by LOHMEIER, see [291, p. 153ff] for details and Figure 3.9 left for a photo. It measures translation through MEMS-based accelerometers and rotational velocity through fiber-optic gyroscopes. As a high-end IMU used in aircrafts and missiles, it features high accuracy, low drift, and low mass compared to similar systems [291, p. 154]. However, since position and orientation are computed by integration, there is still considerable drift. Indeed,

most high-end IMUs support drift compensation by using the *Global Positioning System (GPS)* (position) and/or an integrated magnetometer (orientation). Unfortunately, due to the relatively poor accuracy, this does not add much benefit for indoor use. An alternative technique for drift compensation is to fuse the pose data from the IMU (high short-term accuracy) with the pose data from visual odometry (high long-term accuracy) which tries to combine the best of both worlds. This is known as *visual-inertial odometry* (or *visual-inertial SLAM*) and represents a research area on its own, see for example BLOESCH et al. [87] (based on the extended KÁLMÁN filter) or LEUTENEGGER et al. [281] (based on non-linear optimization). Obviously, vision-assisted methods work best in dense, feature-rich environments (e. g. indoors). Finally, in the more specialized field of humanoid robotics, the IMU data may be augmented by a dynamic model of the robot. Various state estimators using simplified models such as the LIPM (e. g. STEPHENS [394]) or a full multi-body model (e. g. XINJILEFU et al. [450]) have been proposed. Within this thesis, a rather pragmatic approach based on merging IMU and planned motion data is used to determine the current state of the robot, see Section 4.5.3 for details.

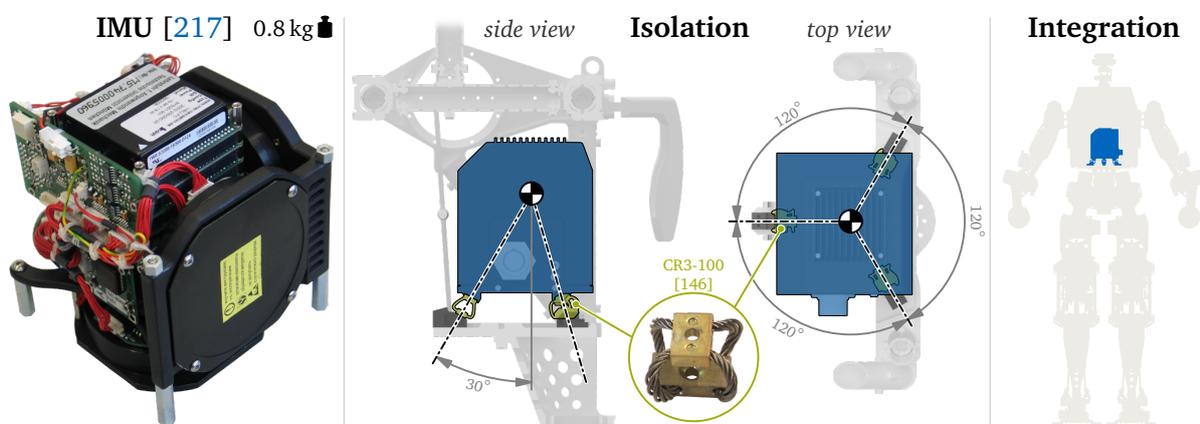


Figure 3.9: Integration of the main IMU *iVRU-FC-C167* measuring the global translation and rotation of the robot relative to the (inertial) world. From left to right: photo of the sensor without housing (modified from [291, p. 155]), mounting through compact wire rope isolators, and placement within the torso segment.

Since the aforementioned IMU fully satisfies the needs for LOLA, it was decided to keep it also for the new upper body. Same as before, the measurement is acquired through the IMU's *CAN* interface (200 Hz sampling rate) where a *CAN-EtherCAT* gateway by *ESD Electronics* [152] is used to integrate the IMU into the main communication bus of LOLA. The IMU is placed within the center of the torso segment, see Figure 3.9 right. Similar to the original design by LOHMEIER, it is mounted to the rigid torso through three compact wire ropes (*CR3-100* by *Enidine* [146]), which isolate the IMU from shocks and (high-frequency) vibrations in all axes.⁴¹ The isolators act as a low-pass filter damping noise due to structural oscillations and impacts. Following the original considerations of LOHMEIER [291, p. 156], we choose a target cutoff frequency of 50 Hz. A rough calculation⁴² of the actual cutoff frequency using the aforementioned components delivers 47.7 Hz. The main axes (primary force direction) of the wire rope isolators intersect in the CoM of the IMU. The isolators are arranged symmetrically (120°) around their common intersection point, see Figure 3.9 center.

A severe drawback of using a high-end IMU is that it is typically much heavier and consumes more space compared to pure MEMS-based systems. Thus, for future developments, it has to be

⁴¹At some point in time, the original wire rope isolators have been replaced by rigid connectors. Unfortunately, to the author's best knowledge, the time of this modification and the reason for it have not been documented. However, comparing the properties of the previous wire rope isolators to the new ones suggests, that the previous setup may have been not stiff enough which resulted in a too low cutoff frequency.

⁴²Here, we use $m_{\text{IMU}} = 0.8 \text{ kg}$ [291, p. 154] and assume that the three wire rope isolators can be approximated by three parallel linear springs each with stiffness 24 kN/m [146]. Note that this is just a rough estimate since the actual alignment is much more complex and the isolators have non-linear, frequency-dependent behavior.

evaluated if modern low-cost IMUs – especially in combination with visual odometry for compensating drift – would satisfy the requirements of a humanoid robot. Moreover, a cluster of distributed mid-range IMUs might outperform a single high-end system while still being lighter and cheaper. Finally, it has to be mentioned that most modern IMUs support sampling rates higher than the mentioned 200 Hz in our case. However, since we are interested in the (comparatively slow) torso dynamics and high frequencies get damped by the wire rope isolators anyway, a higher update rate (such as 1 kHz) is not expected to make a considerable difference.

Vision System The vision system of LOLA v1.1 is a combination of the *Depth Camera D435* [218] and the *Tracking Camera T265* [219] by *Intel RealSense*, see Figure 3.10 left. They have been selected according to the requirements of the new CV system, see Section 4.4, and replace the previously mounted (single) *Asus Xtion Pro Live* camera.

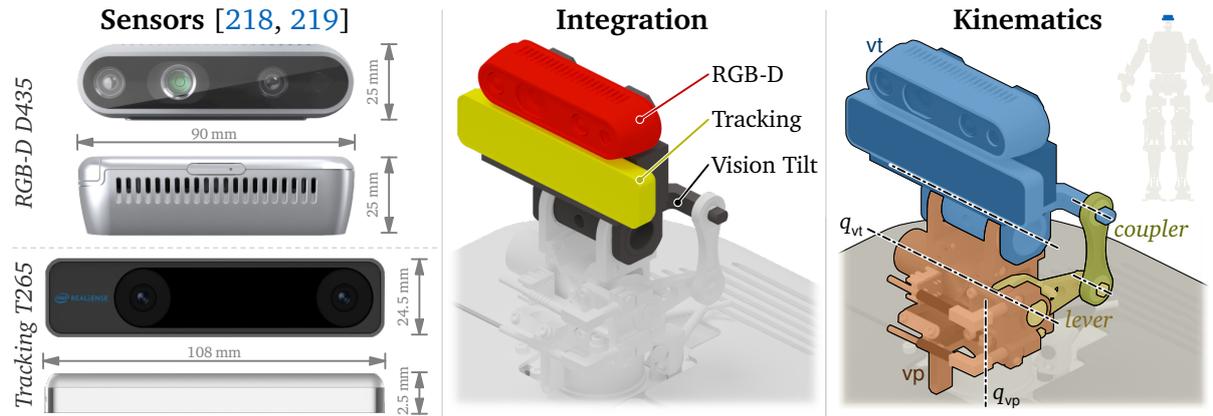


Figure 3.10: Integration of the *Depth Camera D435* and the *Tracking Camera T265* as new vision system of LOLA. From left to right: photo of the sensors (modified from [218, 219]), placement within the vt segment, and kinematics of the head realizing pan and tilt motion (DoF vp and vt – unchanged since [430]).

The *Depth Camera D435* is an RGB-D sensor simultaneously providing a color and depth stream. The color image (up to 1920×1080 and 30 Hz) is obtained from a regular (single) rolling shutter *Complementary Metal-Oxide-Semiconductor (CMOS)* chip. The depth image (up to 1280×720 and 90 Hz) is obtained by projecting a static *Infrared (IR)* pattern on the scene which is observed by two (left/right) IR cameras. The actual depth values are computed by an onboard ASIC. Thus, without external processing the camera provides an RGB-D image stream or, equivalently, a colored 3D point cloud. Due to the projection of an IR pattern, a depth image can be generated for surfaces without texture (e. g. a white wall) or even in complete darkness. As a drawback, scenes with other IR light sources (e. g. sunlight) drastically degrade the quality of depth measurements. Thus, this sensor works best for indoor applications.

The *Tracking Camera T265* consists of two (left/right) monochrome global shutter CMOS image sensors and a MEMS-based IMU. The camera’s IMU is theoretically redundant with the primary IMU in the torso, but it is used to obtain better drift-free pose estimation through visual-inertial odometry. The image sensors are equipped with wide angle (“fisheye”) lenses to maximize the field of view which is important for visual tracking. The device integrates an onboard ASIC, which performs visual-inertial odometry in real-time and provides the host a 6D-pose of the camera (with respect to an inertial world frame) at a rate of 200 Hz.

The realization of the DoF in the head, i. e., the segments vp and vt, have not been discussed so far. This is because their design is kept (almost) unchanged, i. e., the kinematics and actuation is exactly the same as in [430], see Figure 3.10 right. Indeed, only minor modifications to the mechanical interfaces (new torso segment and new cameras) have been made. Within the scope of this thesis, the pan and tilt DoF of the head are used to explore the scene at the beginning of an autonomous walking experiment.

3.5 Mechanical Design

A severe limitation of the previous upper body was the insufficient stiffness of the T-shaped frame representing the backbone and shoulders, see Figure 3.11 left. Moreover, the quite rudimentary scaffold consisting of riveted aluminum tubes was not designed for multi-contact loads. The previous design also had multiple minor issues such as a rather limited extensibility – an important feature for a research platform where experimental components are added and removed frequently. Thus, the upper body was redesigned from scratch. This includes the torso and the arms. In contrast, the head remained mostly unchanged.

Materials As a first step, appropriate materials for structural parts had to be selected. For LOLA, the main goal (apart from increasing the overall strength) was to simultaneously maximize the stiffness and minimize the weight. Prominent materials for lightweight structures in aerospace applications are (sorted by descending density) steel, titanium, aluminum, and magnesium. Note that the specific stiffness, i. e., the ratio between the elastic modulus E and the density ρ , is very similar for these materials and lies in the range of $E/\rho \approx 24 \dots 27 \cdot 10^6 \text{m}^2/\text{s}^2$. With regard to the specific stiffness, these metals and their alloys are easily outperformed by CFRPs. Unfortunately, CFRPs are much more difficult to process and set strong restrictions for the shape, geometric tolerances, and interconnections. Moreover, while metals show plastic deformation before rupture, CFRPs are more brittle and tend to break abruptly which renders the inspection and repair more difficult. In case of a severe crash (not unlikely for a research prototype) a damaged torso frame made out of CFRP would have to be replaced which in turn would mean a complete disassembly of the upper body. Thus, CFRPs are not considered for the main structures of LOLA's new upper body. Nevertheless, they are a serious option for potential mass-produced (and more mature) commercial humanoids in future.

The specific stiffness is a handy parameter for investigating deformation. Indeed, under normal stress, it is inversely proportional to the strain. This can be easily verified by considering a cylindrical beam with cross-section A which is stretched (or compressed) by a certain external force acting along its main axis. We further assume the beam to be slender, the deformations to be small, and the material to be isotropic, linear elastic, and homogeneous. For a fixed load, the strain is proportional to $1/(EA)$ [180, p. 7ff]. If we assume that the beam has a fixed length l and a certain budget for the mass $m = Al\rho = \text{const.}$, then A is proportional to $1/\rho$. As a consequence, the strain is proportional to ρ/E . Thus, given a certain target weight m of the structure, materials with the same specific stiffness E/ρ will show the same deformation under normal stress. However, the weakness of the previous torso of LOLA is mainly related to bending (2nd mode in [75]), for which the second moment of area I has to be taken into account. Given a certain external moment bending our beam, the curvature is proportional to $1/(EI)$ [180, p. 100ff]. With $I = A^2/(4\pi)$ for a cylindrical beam, the curvature is proportional to $1/(EA^2)$ and consequently ρ^2/E . Similar considerations can be made for torsion where the twist is again proportional to ρ^2/E [180, p. 192ff]. Thus, for bending and torsion, the material with the lowest density wins (in our case magnesium). These design rules apply to structural parts where the load, mass, and primary dimensions (here: beam length) represent external constraints, while the material (E and ρ) and the cross-section (A and I) are free design parameters. Obviously, if the cross-section is constrained due to limited space, the material with the highest E should be preferred (in our case steel).

Since aluminum is easier to process compared to magnesium, it was selected as primary material for the upper body structures. Due to its high strength and good machinability, the particular alloy *AlZnMgCu1.5* was used in most cases. For critical parts in narrow spaces, i. e., with limited cross-section, stainless machining steel (mainly *X 10 CrNiS 18 9*) is used. An example is the coupling flange between lower and upper body, see Figure 3.11 right. In contrast, parts subject to low stress are made out of conventional polymers, mostly *Polyoxymethylene*

(POM) due to its high specific strength and dielectric properties. Examples are brackets for mounting electrical components such as the PCs. Finally, the bulky and thin-walled covers of the torso are made out of *Glass Fiber-Reinforced Polymer (GFRP)*. Apart from optical purposes, they prevent that the cables for power supply and high-level communication (attached to an external rack) get caught in one of the numerous components of the torso assembly.

Torso Frame The structural dynamics of a humanoid robot are strongly influenced by the design of the torso since it acts as central segment interfacing the legs and the arms. This especially holds true for multi-contact configurations, where contact forces from the hands propagate through the torso towards the feet and vice versa. In the case of LOLA, an EMA of the previous hardware showed that the first two structural eigenmodes appeared at 6.5 Hz and 9.7 Hz, respectively, see Figure 3.18 and BERNINGER et al. [5, 75] for details. The first mode represents mainly a torsion around the *vertical* axis and seems to be caused by twisting of the legs [5]. Since the legs are connected through the hip (segment ba), the mechanical realization of the torso does not have much influence here. Fortunately, the balance control of LOLA is based on the rotation of the upper body around the *horizontal* axes measured through the primary IMU located in the torso segment, see [401, p. 69] for details. Thus, structural oscillations around the vertical axis do not (directly) affect the balance control. The second mode represents mainly a bending motion around the horizontal axis (walking direction). Indeed, this mode is clearly visible in the experimentally identified open-loop transfer function from vertical foot velocities (main direction of impact in biped walking) to the horizontal inclination rates of the upper body (used by balance controller) [75]. The shape of this mode indicates, that the main deformation occurs in the torso segment. In accordance with these conclusions, the primary design goals for the new torso frame were: (a) increase overall strength for multi-contact capabilities and (b) increase the stiffness with special focus on the aforementioned bending mode. A comparison of the previous and new torso frame is given in Figure 3.11. Note that the second moment of area with respect to the critical horizontal axis has been significantly increased.

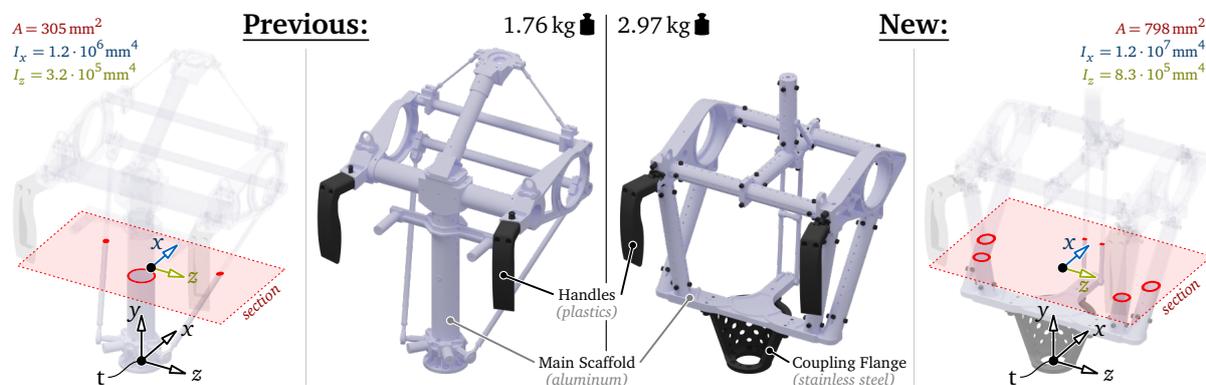


Figure 3.11: Comparison of the previous (left) and new (right) torso frame. An exemplary section view (red) on the same height highlights the increased second moment of area $I_x = \iint z^2 dx dz$ with respect to the x -axis of the torso frame “t”. For this, the x -axis is shifted to the centroid of the cross section, which still lies in the original x - y -plane due to symmetry. The high robustness and stiffness is a result of the cube-alike shape on the one hand, and the increased use of material (+68.8% mass) on the other hand.

Same as before, the new torso frame consists of interconnected aluminum tubes. However, while the original T-shaped scaffold had connections through lightweight rivets, its cube-shaped successor uses a (slightly heavier) screwed push-fit system which is more robust against vibrations and allows precise positioning and alignment, see Figure 3.12. Moreover, each tube is flattened on four sides and provides a pattern of alternating threads and fitting pin holes. This allows an easy attachment of new components with well-defined position and orientation, e. g. through form-fit clamps. Apart from the tubes, the scaffold integrates a lightweight shoul-

der flange for mounting the arm drives, a T-shaped bottom plate, and a high-strength coupling flange as main interface to the lower body. As with the rest of the upper body of LOLA, most parts are made by milling and turning.

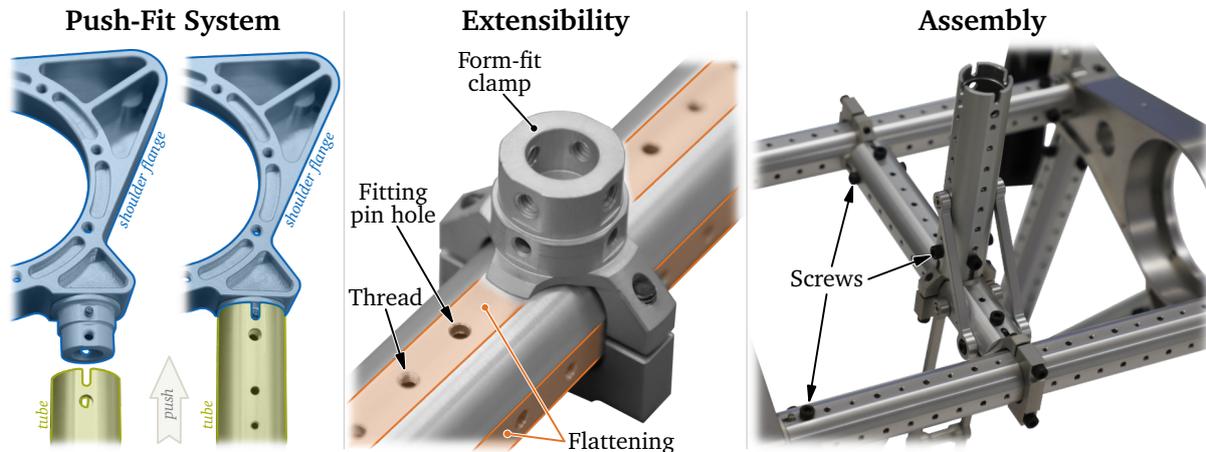


Figure 3.12: Interconnections of the new torso frame scaffold. The ends of each tube are connected via a push-fit system for accurate positioning and alignment. Moreover, each tube features flattened sides and an alternating pattern of fitting pin holes and threads for simple and precise extension by form- and/or force-fit.

Finite Element Analysis (FEA) Most parts of the new upper body have been designed in accordance with conventional engineering practice for low mass and high stiffness. However, for critical⁴³ parts, a *Finite Element Analysis (FEA)* was performed to further adapt the shape according to the expected loads. As mentioned earlier, it is difficult to predict the actual loads on an individual component for a multi-contact scenario. This holds true in particular for structural parts in the torso frame, which typically have numerous mechanical interfaces. However, by considering concrete target scenarios (e. g. horizontal/vertical support against a wall or table), at least a rough estimate for the interface forces can be derived which allows to define load cases required by the FEA. The result of the FEA analysis is then used to design an improved revision of the part – simply by making regions thicker or thinner, placing holes, or adding stiffeners depending on the predicted local stress. Due to the strict time constraints, an automated topology optimization (cf. LOLA’s leg structures by LOHMEIER) was omitted. As a positive side effect, the “manual” optimization of the geometry gave more control on the resulting shape. Thus, the geometry was designed in a way, so that all parts could be manufactured through conventional *Computer Numerical Control (CNC)* milling or turning.

Exemplary results of such an analysis are shown in Figure 3.13 for two different components of the torso frame. Here, it is assumed that both hands of the robot are subject to contact forces of 100 N acting in all three axes (combined to a worst-case scenario). By additionally considering the weight of the upper body, rough estimates for the interface forces can be computed (see F_{A-F} in Figure 3.13). Since LOLA has been designed with the CAD software *Catia* [125], its integrated FEA module has been used. For best results, the parts are meshed with a relatively high resolution using parabolic tetrahedrons (element type TE10). Obviously, the described load case represents a purely static scenario while much more complex loads will appear during locomotion. Due to limited time, further investigations in the design phase have been omitted. Instead, the structures have been tested directly in real-world multi-contact experiments.

⁴³Due to the high structural complexity of a humanoid robot, it is difficult to identify “critical” parts in general. Obviously, this strongly depends on the actual scenario. To obtain an “ideal” hardware, one would have to optimize every single part according to its individual real-world loads. Since development time is limited, a more realistic approach is to design parts such that they are surely strong/stiff enough (by applying a certain safety factor) and optimize only the shape of parts which promise substantial savings in the total mass of the robot.

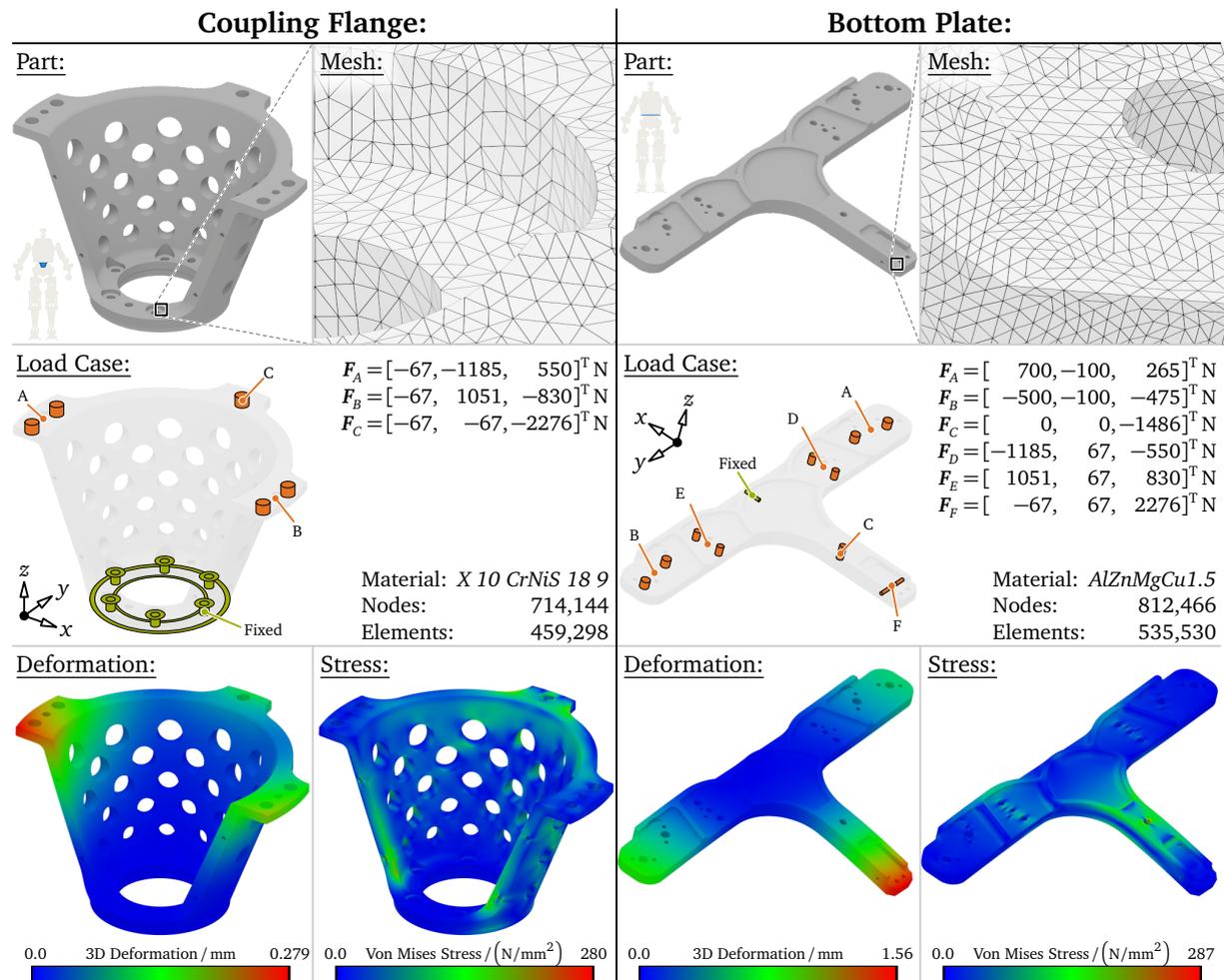


Figure 3.13: Exemplary *Finite Element Analysis (FEA)* of the coupling flange (left) and the bottom plate (right) as critical structural components of the new torso frame. From top to bottom: CAD model and discretized geometry, load case definition, and resulting local deformation and stress. The interface forces F_{A-F} are applied as distributed loads on the highlighted hole or pair of holes (orange). The visualizations in the last row use the deformed geometry (scaled by the factor 10 for better visibility). See the video [22 @t=1m27s] for animated cross section views.

Miscellaneous The majority of structures and components of the new upper body has been designed by the author of this thesis. However, various “non-critical” parts have been designed by the student assistants REINHOLD POSCHER and DANIEL PÖLZLEITNER. This included in particular the three torso frame covers made out of GFRP, the main terminal (interface for the power supply connectors etc., see Figure H.3), and the mountings for: the two PCs, the GPU, the *EtherCAT* junction, the servo controllers (except for efr!), and the *CAN-EtherCAT* gateway. Most of these components are directly attached to the torso frame using the aforementioned modular tube system. The only exception are the mountings for the PCs and the GPU, which are linked to the torso frame through the same compact wire rope isolators as used for the IMU, see Figure 3.14. This is meant to isolate these sensitive components from strong impacts and vibrations.

Overview The video [22 @t=32s] shows all components of the upper body and visualizes their mechanical assembly. A comprehensive list of the main mechanical parameters of the robot (including per-segment mass, inertia, etc.) is given in Table H.1. The main dimensions of LOLA v1.1 are illustrated in Figure H.1.

3.6 Electrical Design

The torso segment of LOLA integrates core electrical components such as the power distribution system, the main computing hardware, and the communication bus junctions. With the mechanical revision of the upper body, these systems have been upgraded, too. Special thanks goes to the staff at the electronics lab of the *Chair of Applied Mechanics*, namely ANDREAS KÖSTLER and GEORG MAYR, who significantly contributed in the design phase of LOLA's new power distribution and communication system.

Power Distribution System Within the context of the upper body revision, the power interface has been changed such that the robot is now provided input voltages of 80 V (unchanged) and 24 V (previously 48 V). Both voltages are delivered by corresponding power supply units located in an external rack. Consequently, the robot is still restricted to tethered operation lacking a dedicated onboard battery pack⁴⁴. Same as with the previous version of LOLA, the 80 V are directly used to feed the majority of motors. An exception are the motors driving the vp and vt joints, which are connected to the 24 V input line. The 24 V line additionally feeds all other onboard components such as the PCs, the IMU, the FTSs, and the joint servo controllers (auxiliary voltage). Indeed, the change from 48 V to 24 V led to higher currents, which required an adaption of the wire cross sections. The main motivation for this change was to get rid of the heavy onboard DC converters (e. g. 48 V to 24 V and 12 V). To a certain extent, the savings in mass compensated the additional material used for the new torso frame (see Section 3.5). The 80 V motor line is secured through emergency stop switches on the human operators desk and the robot itself. While nonhazardous components such as the PCs and the IMU are directly connected to the 24 V supply, the rest (joint servo controllers, etc.) is guarded through an additional onboard switch (triggered through the human operators desk) attached to the coupling flange of the torso frame, see Figure 3.11. This additionally allows a comfortable forced reboot of the joint servo controllers without turning off the onboard PCs. Note that the motors for the head run with 24 V and are therefore not interrupted by the emergency stops. However, due to their low performance, their danger potential is rather limited when compared to the rest of the robot. A graphical overview of the power distribution system of LOLA is given in Figure H.2.

Computing Hardware The new torso integrates two industrial Mini-ITX mainboards (*Advantech AIMB-276* [50]), each with an *Intel Core i7-8700@3.2 GHz* hexa-core CPU, 32 GiB RAM, and a 512 GB *Solid State Drive (SSD)*. One of the PCs is mounted on the back of the torso and is setup with a *QNX Neutrino 7.0* 64bit RTOS. It directly interfaces the *EtherCAT* bus and executes all real-time planning and control algorithms (hard real-time), i. e., the WPG, the SIK, and the HWL module. As required by this RTOS, *Simultaneous Multithreading (SMT)* of the CPU is disabled such that only six threads can be processed simultaneously⁴⁵. The second PC is mounted on the front of the torso and runs *Ubuntu 20.04* 64bit (without real-time patch). It is dedicated to the new CV system of LOLA (soft real-time). Here, SMT of the CPU is enabled such that up to twelve threads can be processed in parallel. For acceleration of heavily parallelized computations (e. g. neural networks), it additionally features an *Nvidia Quadro P2000* GPU. In order to avoid damage due to impacts, the GPU is mounted separately (again featuring wire rope isolators) and is connected to the *Peripheral Component Interconnect (PCI)* bus of the mainboard through a corre-

⁴⁴Although equipping LOLA with an onboard power supply would be possible, it was not part of the present multi-contact project and would have required significant additional effort in the design, realization, and testing phase. This was not compatible with the strict time schedule of the project. A potential successor of LOLA may directly integrate batteries to allow a power-autarkic operation. Note that for untethered operation, the hardware additionally has to be robust enough to withstand falls which is currently not the case.

⁴⁵Note that more than six concurrent threads may be active. Their distribution to the available CPU cores is managed by the scheduler of the OS which, based on the thread priority, assigns appropriate time slices.

sponding riser cable, see Figure 3.14. Note that the vision sensors (see Section 3.4) are directly connected to the front PC using *USB* as data and power link.

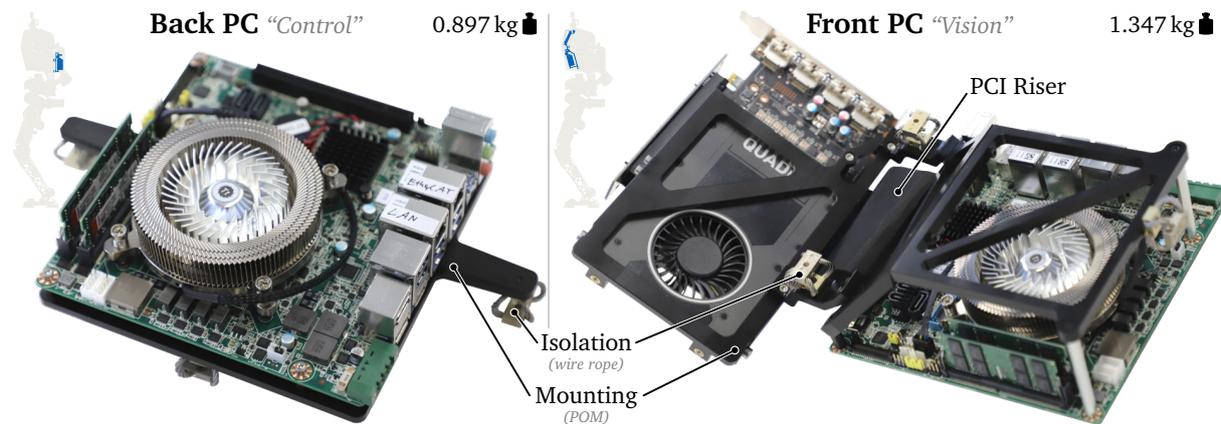


Figure 3.14: Main computing hardware of LOLA v1.1 consisting of two industrial boards. The PC mounted on the back of the torso (left) is running planning and control algorithms on an RTOS. The PC mounted on the front of the torso (right) is running perception algorithms on a GPOS and is equipped with a dedicated GPU linked through a PCI riser cable. The mountings for both PCs feature isolation from impacts and vibrations through compact wire ropes.

Joint Servo Controllers and Communication System Same as before, each drive module of the upper body is connected to a corresponding commercial servo controller by *Elmo Motion Control*. These directly control the motor current while interfacing the absolute (link side; used for initialization) and incremental (motor side; used for commutation and control) rotary encoder to implement an enclosing velocity and position control loop (classical cascaded motor control). Furthermore, they read the binary signal from the limit switch (integrated into the drives) which is used as emergency stop to prevent collisions. The majority of servo controllers is located in the torso segment (in the “neck”). An exception are the controllers for the elbow joints $efr||$, which are instead mounted on the lower arm segments $arr||$. For data exchange, the servo controllers are directly attached to the main *EtherCAT* communication bus of LOLA, which in turn is linked to the real-time PC acting as master (see Figure 3.14 left). Apart from the joint servo controllers, the *EtherCAT* bus also interfaces the already mentioned *CAN-EtherCAT* gateway and the new FTSS in the hands. The *CAN* bus on the other hand connects the IMU and the custom FTSS in the feet. Finally, there is also a *Gigabit-Ethernet* network connecting the two onboard PCs, the operator PC (monitoring and high-level signals), and optionally a PC used to interface the external motion tracking system. A graphical overview of the communication system of LOLA is given in Figure H.4.

3.7 Realization: Manufacturing, Assembly, and Initial Operation

The CAD design of the new upper body started in August 2019 and was finished in January 2020. This was the starting point for the mechanical manufacturing of the 129 custom parts (excludes the two newly built drive modules of type D). More than 94% of all custom parts were manufactured through milling and turning by staff of the in-house mechanical workshop of the *Chair of Applied Mechanics*, namely SIMON GERER and GEORG KÖNIG. The remaining 6% were provided by external contract manufacturers.⁴⁶

⁴⁶The reason for the external production of custom parts was the strict time schedule of the project. Unfortunately, the main manufacturing period fell into the time of the first local lockdown due to the Covid-19 pandemic. This resulted in a limited access to the facilities (including the mechanical and electrical in-house workshops), which made parallel manufacturing necessary.

In May 2020, the majority of custom parts was finished such that assembly could be started, see Figure 3.15. As a preceding step, the weight of every single part of the upper body (reaching from structural parts to single screws and pins) was measured with a high precision scale. Subsequently, the density of the corresponding CAD part was adjusted to match the real measured mass. While this is accurate for parts with homogeneous materials such as aluminum structures, virtual correction masses were added to inhomogeneous components such as the PCs (heavy CPU cooler) to account for the individual mass distribution. The main advantage of this extra effort is that the CAD model becomes very accurate and can be used for exporting the mass, CoM, and mass moment of inertia of each segment to the multi-body simulation framework of LOLA (see Table H.1 for an overview of this data). Since the used multi-body simulation is restricted to rigid bodies, flexible components such as wires cannot be considered. However, their weight is accounted for by adding (rigid) correction masses to the corresponding segments. As a result, the difference in the total mass of the robot between the CAD model (67.3 kg) and reality (68.2 kg) is only 1.3%. While the upper body was mechanically assembled by the author of this thesis, the electrical wiring of components was done by the staff of the electronics lab of the *Chair of Applied Mechanics*, namely ANDREAS KÖSTLER and GEORG MAYR.

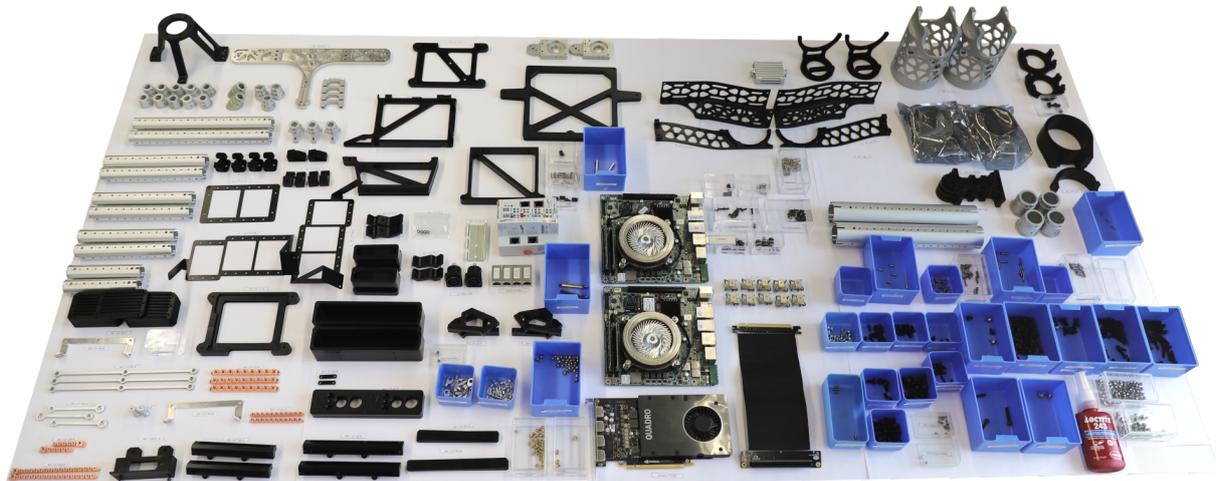


Figure 3.15: Selection of components of the new upper body before their assembly. The weight of every single part has been measured in advance to update the CAD model with accurate mass information.

In October 2020, the new hardware was ready for initial operation tests. This included the mechanical calibration of the robot by attaching it to a special fixture which brings it into a well-defined pose. The joint angles reported by the absolute encoders are saved in this special pose for later use as (constant) offset within the initialization after powering up the robot. The calibration jig was mainly designed by the student assistants DANIEL PÖLZLEITNER and REINHOLD POSCHER, see Appendix H.3 for further details. Finally, the joint servo controllers had to be tuned (i. e., setting the gains of the cascaded feedback control) which was done by FELIX SYGULLA together with the adaption of the SIK and HWL module to support the new hardware. First walking experiments are shown in the video [21 ▶].

3.8 Results and Discussion

The revision of the upper body of LOLA represents a major upgrade which significantly extends the capabilities of the robot. The final configuration of the upper body (without covers) is shown in Figure 3.16 together with a brief summary of the main specifications of the robot. An interactive 360° view is available at the *IEEE Robots Database*, see [113] or alternatively the video [21 ▶@t=3s]. The general conclusion of the revision is positive: the original goals have

been achieved such that the hardware is now capable of multi-contact locomotion. This has been demonstrated through various real-world experiments such as described in Chapter 8.

LOLA v1.1:

Height: 1.763 m

Mass: 68.2 kg

Biped: up to 3.38 km/h

Multi-Cont.: up to 1.8 km/h

Active Joints: 26

Power: 80 V + 24 V

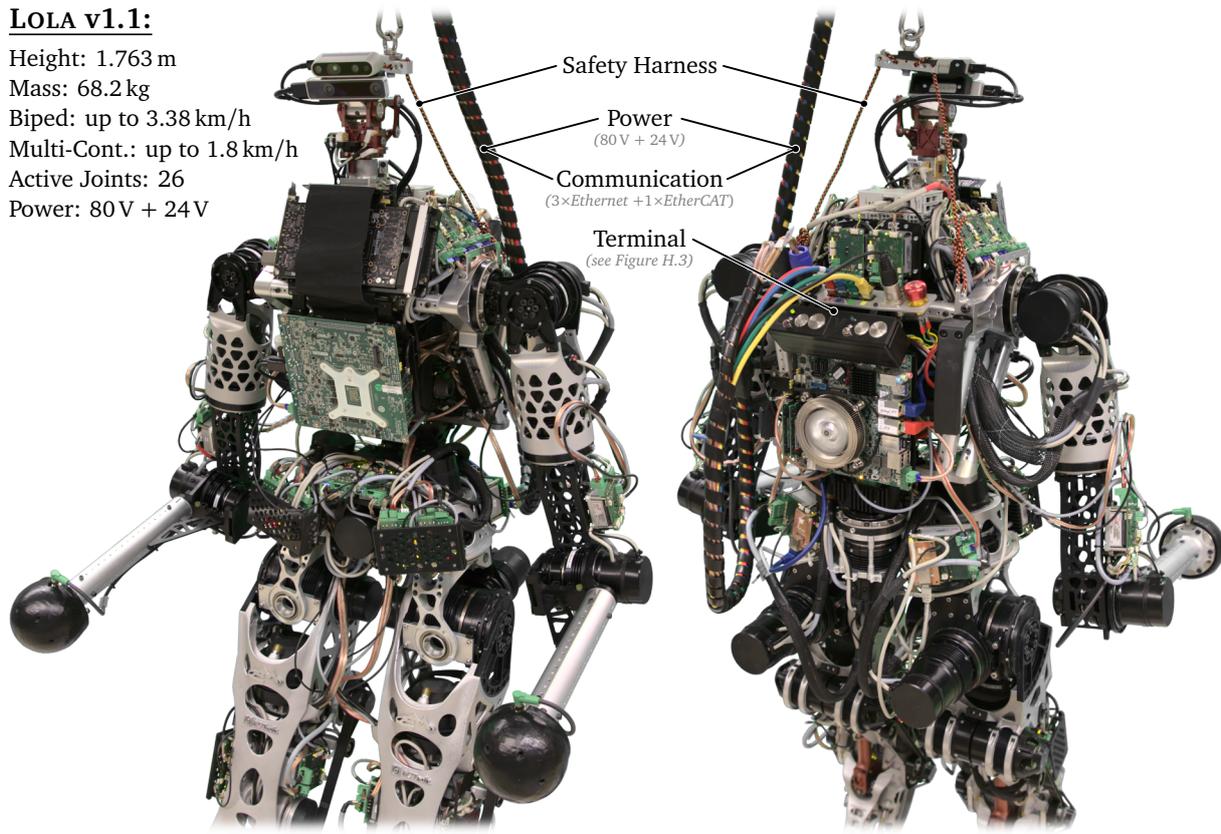


Figure 3.16: Front and back view of the new upper body of LOLA v1.1 (without torso covers). The robot is tethered for safety reasons and to supply it with power (no onboard batteries) and high-level signals (from human operator via *Ethernet*). The additional *EtherCAT* interface in the terminal is meant for testing purposes and is not used in general.

Compared to other fully-actuated humanoids of this size (cf. Section 2.1), LOLA is still very light. Through the recent hardware modification, the total mass of the robot gained only 4.9 kg which equals a plus of 7.74%. This is mainly because the hands are still quite simple, lacking an explicit wrist or gripper/fingers. Moreover, there is still no onboard power supply. However, the robot in its current form matches its purpose for our multi-contact scenarios.

Kinematic Optimization of Arm Topology In Section 3.3, the kinematic topology of the new arms has been optimized for selected target scenarios. The analysis of the arm mechanism shows that the reachable workspace significantly increased. While the total volume gained about 350%, also the local dexterity (measured by the metrics CI, MM, and JRA) has been improved, see Figure 3.6. For the considered multi-contact scenarios, especially the region in the lateral proximity of the robot is important. A comparison with the previous hardware clearly shows the improvement in this area. It is important to note that the hands of LOLA have a spherical shape, thus the actual orientation of the EE upon contact is not prescribed. This makes many things easier and is an important difference compared to an arm designed for manipulation.

Actuation and Sensing Concerning actuation, the new upper body relies on a proven modular design, see Section 3.4. Fortunately, existing modules of the previous arms could be reused such that only two new drives had to be build. The multi-contact experiments conducted so far show that the performance of the joint drives is clearly sufficient. Indeed, using the full torque capabilities would probably damage some of the lightweight arm structures. Thus, if the

performance of the arms has to be increased in future (e. g. for carrying large payloads), one should first design stronger (but also heavier) structures connecting the DoF.

While the main IMU located in the torso segment was basically kept unchanged (except for a slightly different mounting), the new six-axis FTSs in the hands have been added to measure the contact wrench in multi-contact situations. Using commercial sensors instead of a custom solution drastically reduced the complexity and effort during design, realization, and operation. Although the selected sensor model is located in the lower price segment, it provides sufficient accuracy and bandwidth for our use case. Note that the measurements of the FTSs are not used by the WPG described in this thesis, thus, it is not further discussed. Instead, the FTS data represents a primary input for the SIK module, which has been extensively explained in the dissertation of SYGULLA. See in particular [401, p. 130ff] for a description of the used multi-contact force control which also shows the desired and actual (measured) contact forces on the right hand for a certain validation experiment (standing/stamping with right hand supporting against a wall while being pushed by a human, cf. [20 @t=35s]).

To be prepared for fully-autonomous locomotion in unknown environments, also the vision system has been upgraded now featuring two components, an RGB-D sensor and a tracking camera. Unfortunately, the performance of these two components is mixed. On the one hand, the tracking camera represents a major upgrade as it provides a reliable, drift-free 6D-pose through visual-inertial odometry. This represents a very important input for the reconstruction algorithms of the new CV module. An additional advantage is that the used IMU is integrated into the camera which minimizes the error due to relative motion (e. g. when considering the kinematic chain between the torso IMU and the vision system). On the other hand, the selected (low-cost) RGB-D camera shows poor accuracy in depth sensing, see Figure 3.17. Moreover, the already borderline⁴⁷ performance in a best-case setting drastically degrades once sunlight (strong IR component) hits the scene. For the experiments presented in this thesis, it was necessary to keep the lab clear from sunlight. Unfortunately, the circumstances of the governing project did not allow the exchange of the RGB-D sensor. For future investigations, it is highly recommended to use another system – ideally a high-class industrial camera or LiDAR.

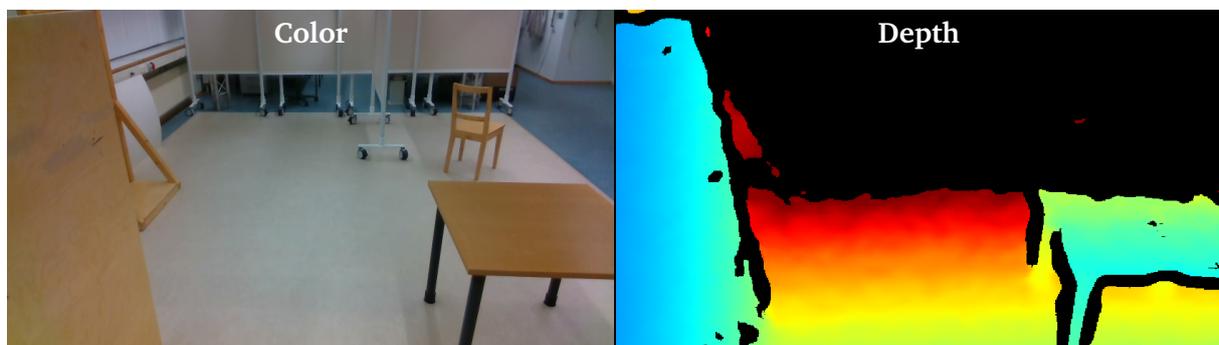


Figure 3.17: Exemplary captured frames from the RGB-D sensor of the new vision system. Left: color image showing the environment. Right: color-coded depth image (averaged over 10 consecutive frames to filter noise) with blue indicating close, red indicating distant, and black representing unknown points. Besides the rather poor absolute accuracy of depth measurements, the data is also subject to severe noise and a “wavy” reconstruction of flat surfaces such as the floor. See [17 @t=22s] for the full video of the corresponding experiment.

Mechanical Design An apparent proof that the new upper body is robust enough to withstand multi-contact loads is given by the numerous videos of successfully conducted experiments, e. g. [20 @]. Besides robustness, an additional goal of the mechanical redesign was to increase the

⁴⁷Certainly, this rating has to be seen in the present context: while a depth accuracy of 4 cm at a distance of 2 m [218] may be sufficient for numerous applications in robotics, it is highly problematic for identifying potential foot- or hand-contact points for a full-sized humanoid robot.

structural stiffness in order to counteract undesired oscillations affecting the bandwidth of the low-level controller. To check if this goal has been achieved, BERNINGER et al. performed a second EMA evaluating the structural dynamics of the new hardware, see [5] for details. The first five mode shapes for the previous and new version of the robot as identified by the EMA are shown in Figure 3.18.

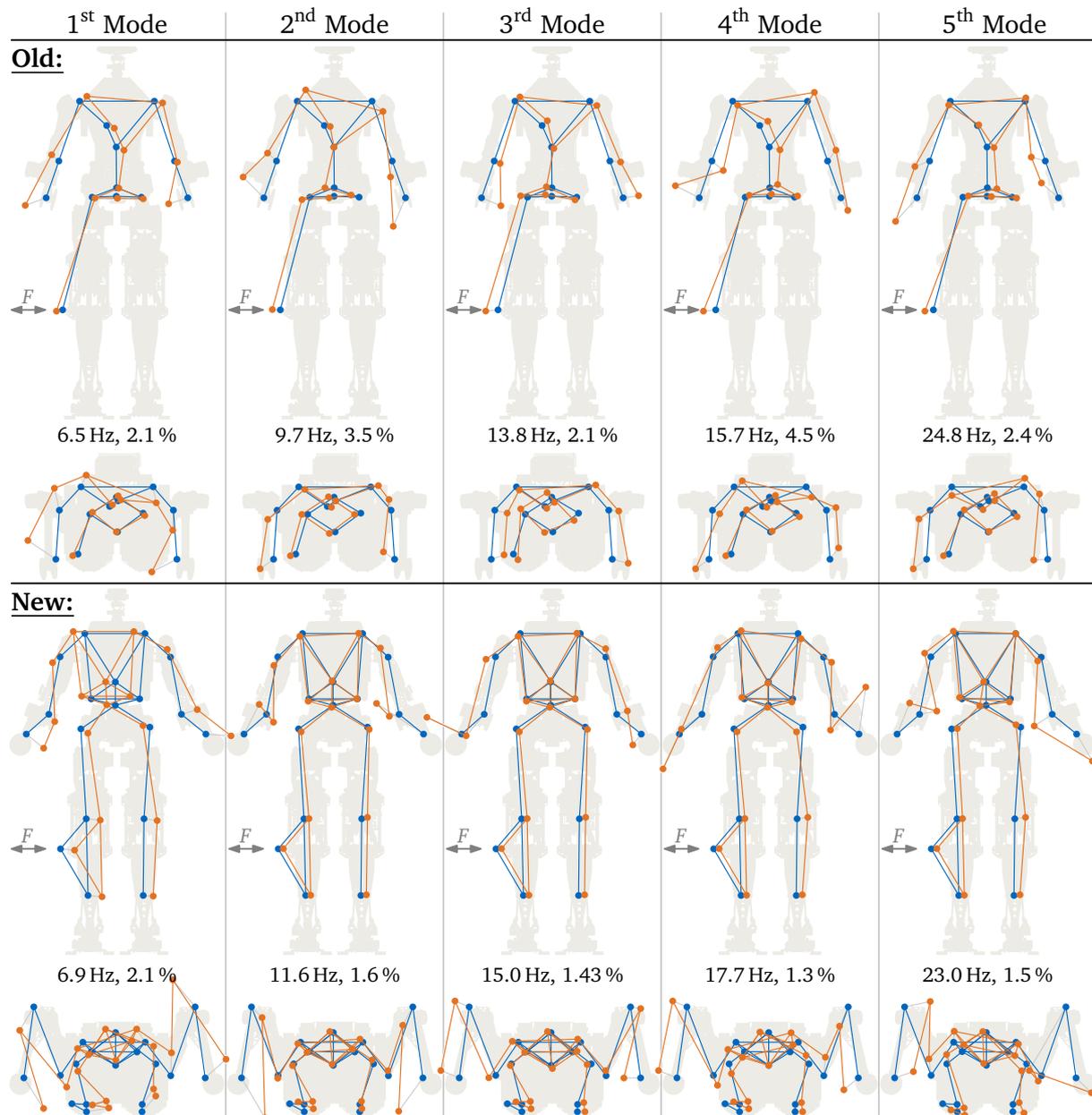


Figure 3.18: First five mode shapes for the previous (row 1+2) and new (row 3+4) version of the robot presented in the front (row 1+3) and top (row 2+4) view. For each mode, the respective eigenfrequency and damping ratio are given. The dots represent the location of the attached accelerometers in their initial (blue) and displaced (orange) state. During the EMA, the robot is commanded to remain in a default standing pose (in contact with the ground) while it is excited in the lateral direction by an external shaker linked to the right knee (indicated by F). Note that modes (same as for eigenfrequencies and damping) are theoretically independent of the location of the excitation as they represent possible free vibrations of the system. Modified from [5].

As already predicted in Section 3.5, the revision of the upper body had only a small impact on the shape and eigenfrequency of the first mode. Since torsion around the vertical axis due to twisting of the legs is not visible to the balance controller, this mode is not considered critical.

Nevertheless, the strengthening of the torso frame seems to remove its contribution to the torsional mode which leads to a slight overall improvement. Starting with the second mode, the results differ when comparing the new to the previous hardware. Indeed, the eigenfrequency of the second mode has been increased by 20 %. Much more importantly, the shape of the second mode shows that we got rid of the undesired bending of the torso frame around the horizontal axis. In fact, for the 2nd to the 5th mode, the torso stays almost rigid while the main deformation appears in the arms instead. Certainly, structural oscillations of the arms are also an undesired behavior. However, a stiff torso (carrying the IMU) is much more important in this case.

For potential future investigations it should be considered to further increase the stiffness of the arms. According to the results of the EMA, it can be expected that this would effectively raise the eigenfrequencies (starting with the second mode), eventually shifting them out of the domain typically excited by legged locomotion. Besides stiffness, also damping plays an important role. Since the new upper body design relies on the same core concept, i. e., scaffolds made out of aluminum tubes, the damping behavior did not change significantly. For a potential successor of LOLA, one might consider to use other materials, e. g. CFRPs, or to integrate passive damping elements, e. g. sandwich composites (cf. intervertebral disc in the human spine).

For the purpose of tuning the low-level control, numerous walking experiments have been conducted where the robot was commanded to move as fast as possible. Indeed, a new speed record of 3.38 km/h (step length: 0.63 m; step duration: 0.67 s) could be set for regular biped walking (no hand support) which is a slight improvement over the previous record of 3.34 km/h set in 2011 [101]. This demonstrates that the revision of the upper body and the associated increase of the total mass did not decrease the biped walking performance of the robot which makes it a true upgrade. Certainly, LOLA received also many other improvements over the past 10 years. However, the revision of the upper body represents the most “invasive” modification.

Electrical Design The revision of the power distribution system represents another remarkable improvement. The transition from 48 V to 24 V for the auxiliary voltage made multiple DC converters obsolete, hence, reducing complexity and mass. As a consequence, the wire cross-sections need to be greater which actually does not represent a notable disadvantage. In fact, 24 V are an industry standard in the field of automation, thus most components support it.

The upper body revision also led to a substantial improvement with regard to computational power. For the PC dedicated to real-time control, this is simply a result of upgrading to modern components. In contrast, the PC running CV algorithms now additionally features a dedicated GPU, which significantly accelerates massive, parallel computations. Moreover, it allows to use other vision sensors which run their algorithms on the host PC and, therefore, require a dedicated GPU. An example is the stereo camera *ZED 2i* [395] by *Stereolabs* which seems to be a reasonable alternative to the current (problematic) RGB-D sensor. Note that the particular model of the GPU has been selected to provide the highest computing performance while still being powered exclusively over the PCI bus (maximum 75 W). This avoids a direct power supply of the GPU with corresponding DC converter.

For low-level communication, the already existing EtherCAT bus was adopted. For LOLA, it has proven to be fast and reliable while still being simple to integrate. Unfortunately, due to some legacy components such as the IMU and the custom FTSS in the feet, additionally CAN and SPI buses are operated in parallel. For a potential successor of LOLA, a single EtherCAT bus connecting all components should be considered to get rid of the complexity, latency, and weight (cf. *CAN-EtherCAT* gateway) of translating between bus systems.

Software – Part A: Locomotion Framework

Parts of this chapter have already been published in [1–4].

In this chapter, the present locomotion framework of LOLA is introduced. The current state of the framework represents the result of more than 20 years of evolution starting in 1997 with JOHNNIE. Thus, it has to be seen as joint achievement of all researchers working on JOHNNIE and LOLA so far (see Section 2.7 for a concrete list of persons). Although this chapter gives an overview of the entire framework, it strongly focuses on the task of *planning*, i. e., the WPG module integrating the contact planner and motion generator. As part of the upgrade for multi-contact locomotion, the entire WPG has been redesigned and rebuilt from scratch. Besides the revision of the upper body hardware described in Chapter 3, this represents the second core contribution of the author to the LOLA project. To begin this chapter, the framework and its main workflow are introduced in Section 4.1. This is followed in the Sections 4.2 and 4.3 by the definition of frequently used CoSys and the “task-space” of LOLA, respectively. The actual WPG module is discussed in Section 4.5, which is preceded by a brief summary of the CV system in Section 4.4 and succeeded by a brief summary of the SIK and HWL module in the Sections 4.6 and 4.7, respectively. Finally, the chapter is concluded in Section 4.8. Note that the focus of this chapter is the governing locomotion framework, while the actual contact planner and motion generator are described in more detail in the Chapters 5 and 6, respectively.

A coarse overview of the multi-contact locomotion framework of LOLA has already been published in [1], see also the accompanying video presentation [19]. With regard to the WPG, its core method for generating a dynamically and kinematically feasible CoM motion (without multi-contact) has been published in [3]. The underlying spline collocation algorithm has been presented separately in much more detail in [2]. Moreover, planning of the CoM motion is based on a simplified five-mass model of the robot, which has been parameterized according to the optimum derived by ANIAN LEYERER within the context of his term paper [29]. Although this term paper has not been published, it is briefly summarized in [4].

4.1 Overview

Developing an enclosing software framework for a real-time locomotion system is part of the discipline *software design*, for which the state of the art has already been summarized in Section 2.2. The main objective of such a framework is to connect the individual concepts for perception, planning, and control and to provide an interface to the surrounding software infrastructure for communication, simulation, logging, and visualization. According to this brief description, a locomotion framework may seem to be just a conglomeration of “glue code” necessary for holding the essential parts together without actually affecting the final outcome, i. e., the motion of the robot. This might hold true for certain purely simulative investigations. However, if the locomotion system is to be executed on real hardware, hard real-time constraints have to be satisfied. While the failure of high-level tasks such as speech recognition are typically not considered critical, a violation of the real-time constraints in the low-level locomotion system of a humanoid robot is likely to cause a fatal crash.

Indeed, the hard real-time constraints do not only set high requirements on the efficiency of algorithms and interfaces, but also the overall architecture of the framework itself. In particular, the possibility for efficient parallel execution (to reduce total cycle time) and advanced synchronization (for low latency and jitter) are important. To achieve these goals, a common approach is to cluster the components of the locomotion system according to their real-time priorities. This typically results in a *control*, a *planning*, and an *autonomy* level (see Figure 2.12) with *control* being the computational lightest and most time-critical and *autonomy* being the computational heaviest and least time-critical level. The locomotion framework of LOLA follows this hierarchical approach by forming the modules *Hardware Layer (HWL)*, *Stabilization and Inverse Kinematics (SIK)*, *Walking Pattern Generation (WPG)*, and *Computer Vision (CV)*. Figure 4.1 gives an overview which is explained in the following paragraphs.

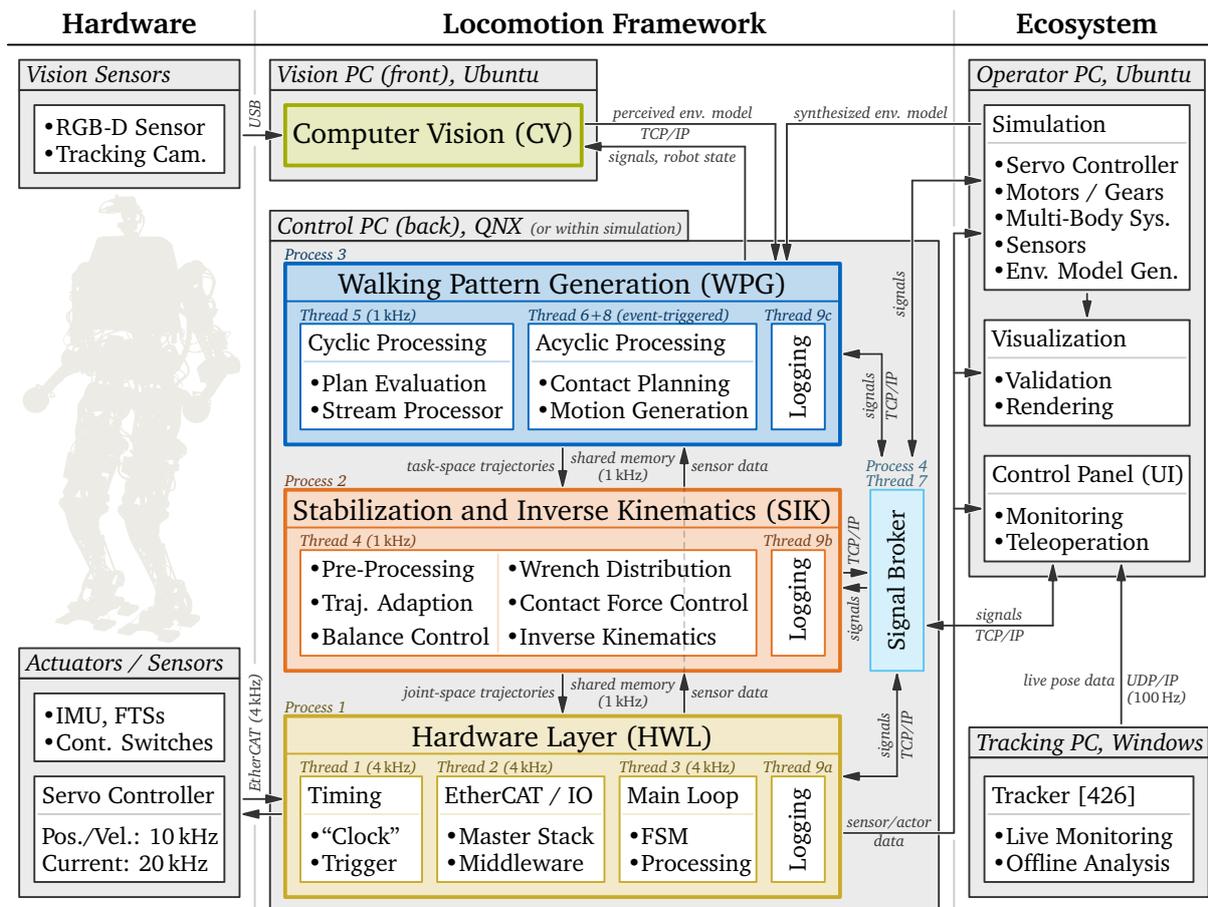


Figure 4.1: Overview of the real-time locomotion framework of LOLA and its interface to the hardware and the software ecosystem. The CV module runs on the front PC with a GPOS under soft real-time, while the WPG, SIK, and HWL modules run on the back PC with an RTOS under hard real-time. The software executed on the control PC is further split into numbered processes and threads. The thread number indicates its respective QNX real-time priority with 1 representing the highest priority. The logging threads 9a, 9b, and 9c share the same (lowest) QNX real-time priority. For details on the communication network, see Figure H.4.

Main Workflow In order to explain the main workflow, let us assume an autonomous locomotion scenario where the robot is commanded to walk from its current position to a certain goal, e. g. the opposite side of the laboratory. As a first step, the scenario has to be triggered by a human operator selecting the corresponding action for autonomous walking (with corresponding goal position) in the UI alias *Control Panel* running on the *Operator PC*, see Figure 4.1 right. The request is encoded as a corresponding high-level signal and transmitted to the *Control PC* where

it is received by a *Signal Broker* (Process 4, Thread 7) forwarding it to the modules HWL (Process 1), SIK (Process 2), and WPG (Process 3). Certain high-level signals are additionally passed (together with the current state of the robot) from the WPG to the CV which gives the human operator also control over the CV module. The particular signal for autonomous locomotion triggers a new planning activity in the WPG, which uses the current model of the environment to create a corresponding motion plan leading from the start (current pose) to the destination (goal position). The environment model is received and continuously updated from the CV module, which is briefly summarized in Section 4.4. The creation of the motion plan is an acyclic task (Thread 6) and consists of a preceding contact planning and subsequent motion generation step. Once the motion plan is created, it is passed to the main WPG loop (Thread 5), which evaluates the plan for the current time step and performs some additional post-processing. A more detailed view on the general workflow of the WPG is given in Section 4.5.5. The output of the WPG is (among other data) a current sample of the task-space trajectories, which is cyclically passed to the SIK module. In the main loop of the SIK module (Thread 4), the task-space sample is modified for the purpose of stabilization and finally converted to a corresponding joint-space sample, see Section 4.6 for a brief summary and [401] for a full description. The joint-space sample is then cyclically transferred to the HWL module which simultaneously provides the current sensor data to the SIK module. The main loop of the HWL (Thread 3) operates a *Finite State Machine* (FSM) and performs some additional processing such as extrapolation of the target joint-space data in order to upsample it from a 1 kHz to a 4 kHz cycle. The sensor and actuator data is exchanged by a dedicated IO module (Thread 2) running a commercial *EtherCAT* master stack by *Acontis Technologies* [48] and a custom middleware [10] for device abstraction. The transmission of *EtherCAT* packets is triggered by a separate timing module (Thread 1, highest priority), which also represents the main “clock” of the real-time locomotion system and is used for precise synchronization between HWL, SIK, and WPG with low latency and jitter. A brief summary of the HWL module is given in Section 4.7, while a more detailed description is found in [401]. The presented workflow of the locomotion framework is (almost) the same for other scenarios, e. g. teleoperated walking, where corresponding high-level signals (walking direction, speed, etc.) are cyclically emitted from the *Control Panel*.

Communication The real-time communication between the HWL, SIK, and WPG module is realized through shared memory interfaces. This allows efficient transfer of large (communication) payloads and is precisely synchronized by the dedicated timing thread of the HWL (Thread 1). Moreover, the communication is synchronized with the *EtherCAT* bus connecting the majority of actuators and sensors (except vision sensors) using the distributed clocks functionality of *EtherCAT* [10]. For high-level signals with small payloads, which typically consist of a unique identifier and an (optional) small data packet such as a single float, LOLA uses a publish/subscribe system originally introduced by BUSCHMANN, see [100, p. 122f] for details. It basically consists of a central hub alias *Signal Broker* (Process 4) and multiple local brokers (not shown in Figure 4.1) as counterparts within the connected modules. For communication between processes, the signals are transmitted either using *POSIX* message queues (on same PC) or *Ethernet* (separated PCs). Examples for such high-level messages are trigger signals or locomotion parameters which can be changed “online”, e. g. the step duration (speed) or step height (ground clearance). For efficient transmission of large payloads in cases where a shared memory interface is not possible (e. g. between separated PCs), a third option has been introduced by the author of this thesis. It abstracts a regular network socket using either the *Transmission Control Protocol* (TCP/IP) or the *User Datagram Protocol* (UDP/IP) and equips it with features such as

- internal serialization of complex types such as classes containing vectors and matrices,
- (optional) internal compression using the *Deflate* algorithm from *zlib* [165],
- thread-safe circular buffers for input/output objects to avoid dynamic memory allocation and (optionally) run the costly serialization and compression in an asynchronous thread.

The implementation of this enhanced socket has been published as part⁴⁸ of *Broccoli*. A prominent example for its usage is the connection between the CV and the WPG module, where a large payload (the environment model) has to be transmitted efficiently without the overhead of the publish/subscribe system.

Real-Time Logging For analyzing experiments in the aftermath, the core modules HWL, SIK, and WPG each integrate a real-time logging module (*Thread 9a, 9b, and 9c*). Here, real-time means that data logging is performed in an asynchronous thread of lowest priority such that the impact on the actual locomotion system is kept minimal. The corresponding manager class uses thread-safe circular buffers for storing the sequence of data objects to be written to the various log files. Similar to the aforementioned network socket, the costly encoding, (optional) compression using *zlib*, and actual file writing is offloaded to an asynchronous thread running in the background. The real-time logging framework of LOLA is part⁴⁹ of *Broccoli*.

Ecosystem In order to operate, test, and analyze a locomotion framework, a comprehensive software ecosystem is required. Within the LOLA project, this is mainly realized through dedicated applications for *Simulation* (testing and debugging, cf. Section 7.4), *Visualization* (validation and analysis, cf. Section 7.5), and the already mentioned *Control Panel* (operation and monitoring, cf. Section 7.6) each of which running on the external *Operator PC*.

The simulation system of LOLA was originally developed by BUSCHMANN and SCHWIENBACHER [100, 372]. It replaces the hardware components of the real robot by corresponding virtual counterparts and simulates their physical interaction with the environment. Moreover, for the simulation system, the real-time locomotion framework is not executed on the *Control PC* anymore, but instead integrated into the simulation application running on the *Operator PC*. The sequential execution within a single process and single thread guarantees a deterministic, repeatable output. The simulation is briefly summarized in Section 7.4, which also introduces an extension contributed by the author of this thesis automatically synthesizing an environment model as it would be generated from the CV system in a real experiment. The results of simulations and experiments, i. e., the logged data, can be visualized through a dedicated graphical tool described in Section 7.5.

Finally, the LOLA laboratory is also equipped with a commercial motion capture system from *Vicon Motion Systems* [426]. The corresponding *Vicon Tracker* software runs on a dedicated *Windows PC* and transmits a live stream of 6D pose data for various tracked CoSys (e. g. the torso) via *Ethernet* using a UDP/IP broadcast. The data stream is received by the *Control Panel* which forwards it to a corresponding logging module saving the current object poses together with the current time stamp of the HWL for synchronized logs. It has to be highlighted that the motion capture data is only used for offline analysis. It is *not* used by the real-time locomotion system of LOLA since this would contradict the idea of autonomy.

4.2 Coordinate Systems (CoSys)

Before the task-space of LOLA is specified, frequently used CoSys have to be introduced. On the one hand, they are used to describe the 6D pose of special parts of the robot. On the other hand, certain quantities such as positions and orientations are typically described with respect to a CoSy making it a so-called *Frame of Reference (FoR)*. The used notation for a FoR as well as the transformation from one frame to the other is explained in Appendix A. An overview of the most important “special” CoSys of the robot LOLA is given in Figure 4.2. From a theoretical point of view, the definition of a CoSy describing the pose of a (rigid) part of the robot is arbitrary. In

⁴⁸See the class `NetworkSocket` in the module `io` of *Broccoli*.

⁴⁹See the class `Logger` in the module `io` of *Broccoli*.

practice, for the sake of a clear and intuitive handling, special frames are defined such that their origin has a certain meaning, e. g. the center of the hand. Moreover, the orientation is chosen such that the z -axis points upwards and the x -axis points forwards (for straight walking).

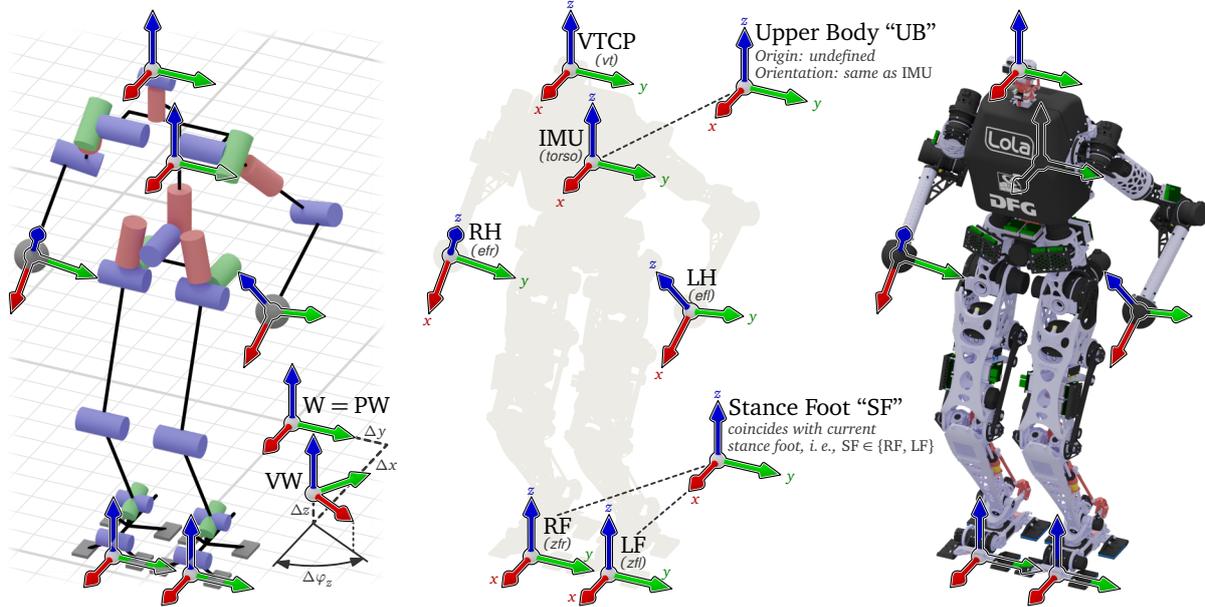


Figure 4.2: Special CoSys of LOLA v1.1. The (planning) world frame “W” (alias “PW”) and the vision world frame “VW” (shown for $\Delta x = 0.5$ m, $\Delta y = 0.3$ m, $\Delta z = 0.1$ m, and $\Delta \varphi_z = 60^\circ$) are inertial. The remaining frames are attached to one of the 27 segments of the robot, see Table H.1. An explicit specification of the presented CoSys through their (constant) pose with respect to the corresponding parent segment is given in Table H.2. The upper body frame “UB” and the stance foot frame “SF” represent exceptions with a special definition.

World Frame First of all, an inertial world frame “W” is defined. Its concrete position and orientation is arbitrary, however, we require that the positive z -axis, i. e., the unit vector ${}_W e_z$, points into the opposite direction of gravitational acceleration such that ${}_W g := [0, 0, -g]^T$ holds. Since the CV system of LOLA is in large parts detached from the rest of the locomotion framework (see Figure 4.1), it uses its own world frame in the following denoted as *vision* world frame “VW”. For a clear distinction, the *planning* world frame is also denoted by “PW” which is just a more explicit alias for “W”. Indeed, VW has the same requirement for the z -axis such that the relative transform between these two world frames can be described by a three-dimensional translation and planar rotation (Δx , Δy , Δz , and $\Delta \varphi_z$ in Figure 4.2). Although both, PW and VW, are *inertial* such that time derivatives described in these frames are *absolute*, they are allowed to “jump”, i. e., instantaneously change their position and orientation (may be used to reset odometry). The parallelism of the z -axes simplifies the use of the environment model (described in VW) within contact planning by avoiding expensive transformations of large data.

Tool Center Point (TCP) Frames Next, a TCP frame is defined for each EE, i. e., for the feet (“RF” attached to zfr and “LF” attached to zfl), the hands (“RH” attached to efr and “LH” attached to efl), and the head alias *vision* (“VTCP” attached to vt). The origins of RF and LF lie in the centroid of the contact surface of the corresponding toe segment. The origins of RH and LH lie in the center of the sphere representing the contact surface of the hand. Finally, the origin of VTCP is set to the mean optical center of both vision sensors (shifted along ${}_{VTCP} e_x$ to lie “outside” the cameras). Note that the term *TCP frame* is used as synonym for the corresponding *EE frame*. However, it is *not* equivalent to the body-CoSy of the respective robot segment which it is attached to. As an example, RH is located in the center of the right hand, while the efr segment (to which RH is attached) has its body-CoSy in the elbow.

Stance Foot Frame While it would be possible to use the world frame W as FoR for all calculations, in some cases it makes more sense to use a body attached frame instead. Especially in biped walking, it is reasonable to describe the motion of the robot with respect to the TCP frame of the current stance foot “SF”, which cyclically switches between RF and LF. Within the scope of this thesis, we assume that once a foot becomes the current stance, it remains fixed to the ground (idealized closed contact). Moreover, we assume the ground underneath the stance foot to be static. Thus, same as for the world frame W , the stance foot frame SF remains inertial (but jumps cyclically) and time derivatives described in SF are absolute, hence, do not require special attention. An extension for scenarios with moving ground, e. g. an escalator, would require only minor modifications to the proposed locomotion framework – at least if the ground motion is known. However, more effort would be required to describe running, i. e., situations where no stance exists. Both, moving grounds and running, are not in the scope of this thesis. The counterpart of the stance foot SF is the swing foot \overline{SF} , whose TCP frame is *not* inertial.

Inertial Measurement Unit (IMU) and Upper Body Frame As a reference for the measurements of the primary IMU located in the torso segment of the robot, we explicitly define an “IMU” frame located in the mean⁵⁰ origin of the integrated accelerometers and gyroscopes. We additionally introduce the upper body frame “UB”, which will be used to specify a certain upper body inclination (e. g. for stooped walking) since it represents a more intuitive alias for this purpose. The UB frame is special in such that we do not specify its origin since it will not be used anyway and would probably lead to confusion otherwise (What is the “origin” of the upper body?). The UB frame copies the orientation of the IMU frame, i. e., it rotates with the torso segment. Note that the IMU frame does not have the same orientation as the torso frame t (which is only used for describing the pose of the corresponding CAD segment), but is instead rotated by 90° around the common x -axis, see Figure 3.5 and Table H.2.

4.3 Task-Space Definition

The locomotion framework of LOLA is designed in a way such that the entire planning (module WPG) and large parts of the control (module SIK) are describing motion in the *task-space*. In general, the task-space should contain quantities relevant to the specific task of a robot, e. g. the TCP pose for an industrial manipulator. It represents a set of constraints which has to be satisfied under all circumstances. For redundant robots, the dimension of the joint space $\dim(\mathbf{q}) = n$ is greater than the dimension of the task-space $\dim(\mathbf{x}) = p$ such that there remains a certain freedom in motion typically called *null-space*⁵¹ of dimension $n - p$. Certain IK methods, e. g. ASC in the case of LOLA, allow to exploit the null-space by specifying secondary objectives during the transition from task- to joint-space such that the whole potential of the hardware is used. Thus, the task-space definition should include the primary objectives of the robot, while a certain freedom ($n - p > 0$) should be kept for fulfilling secondary objectives within the IK.

Since the main objective of LOLA is biped walking, its task-space includes components which are essential in legged locomotion. The probably most important component is the overall 6D pose of the robot with respect to its environment. For this, LOLA uses the position of the CoM and the orientation of the upper body frame UB, which both represent essential characteristics describing centroidal dynamics (cf. the various simplified models presented in Section 2.4). For biped robots, it is also natural to include the position and orientation of both feet, here through the TCP frames RF and LF attached to the corresponding toe segments. This allows exact placement of footholds (e. g. for climbing stairs) and to align the foot sole with the ground

⁵⁰In fact, the x -, y -, and z -axis of the IMU *iVRU-FC-C167* do not intersect exactly in a single point, thus there is no clear “center”. To keep errors low, a reasonable mean position is computed to define the origin of the IMU frame.

⁵¹The name *null-space* originates from the fact that a corresponding motion in the joint-space does not produce any motion in the task-space, i. e., it lies in the kernel (=null-space) of the task-space Jacobian $J = (\partial \dot{\mathbf{x}} / \partial \dot{\mathbf{q}})$.

(e. g. for climbing ramps). A special property of LOLA’s hardware are the active toe joints $zfr||$. The corresponding DoF of the left and right foot is directly mapped into the task-space, which allows explicit control within in the planning stage. The toe joints are mainly used to overcome kinematic limits in fast walking (long strides) and climbing stairs (rapid change of CoM height) for which corresponding heuristic-based trajectories are planned. Besides the toe joints, also the DoF vp and vt are directly mapped from joint- to task-space. This gives explicit control over the head motion, e. g. for visual tracking of environmental objects. If a third head DoF for “roll” motion is added in future, one should consider to include the orientation of the vision TCP frame VTCP instead, since this may give a more natural control over the head, e. g. through the quaternion-based multi-axis interpolation methods presented in Appendix B.3.

For multi-contact locomotion, it is reasonable to add also the position of the hands, in particular the TCP frames RH and LH, to the task-space definition. For LOLA, a special approach is chosen where the hand positions are *optional* members of the task-space. Indeed, their membership can be changed dynamically, i. e., during locomotion. This allows us to specify an explicit target position whenever it is necessary (e. g. for making a contact) or otherwise assign the hand to the null-space for “assistance” in fulfilling primary objectives (e. g. the desired CoM position) and secondary objectives (e. g. the compensation of leg dynamics). Dynamically changing the task-space definition is realized by running four IK instances in parallel (without hands; with right hand; with left hand; with both hands) and blending between these options by smooth bilinear interpolation in joint-space, see Section 4.6 for a brief overview and [401, p.67f] for details (includes a brief overview of other/similar approaches in related work). Note that because of the low DoF count in the arm, the orientation of the hand is not included in the task-space. Due to the spherical contact surface, this is not necessary anyway for our target multi-contact scenarios. The complete list of task-space components is given in Table 4.1.

Table 4.1: Components of the task-space vector \mathbf{x} and the task-space velocity vector \mathbf{v} of LOLA v1.1 (sorted according to the location of the corresponding component within the vector \mathbf{x} respectively \mathbf{v}). The last two rows (gray) are optional. Note that rotational quantities are represented in \mathbf{x} by rotation vectors $\boldsymbol{\vartheta}$ and in \mathbf{v} by angular velocities $\boldsymbol{\omega}$. Thus, in general $\mathbf{v} \neq \dot{\mathbf{x}}$ holds. Modified from [401, p. 49].

\mathbf{x} -Comp.	\mathbf{v} -Comp.	Description
${}^W r_{CoM}$	${}^W \dot{r}_{CoM}$	position of the CoM described in the world FoR
${}^W \boldsymbol{\vartheta}_{UB}$	${}^W \boldsymbol{\omega}_{UB}$	orientation of the upper body frame described in the world FoR
${}^W r_{RF}$	${}^W \dot{r}_{RF}$	position of the right foot TCP frame described in the world FoR
${}^W \boldsymbol{\vartheta}_{RF}$	${}^W \boldsymbol{\omega}_{RF}$	orientation of the right foot TCP frame described in the world FoR
q_{zfr}	\dot{q}_{zfr}	right foot toe angle
${}^W r_{LF}$	${}^W \dot{r}_{LF}$	position of the left foot TCP frame described in the world FoR
${}^W \boldsymbol{\vartheta}_{LF}$	${}^W \boldsymbol{\omega}_{LF}$	orientation of the left foot TCP frame described in the world FoR
q_{zfl}	\dot{q}_{zfl}	left foot toe angle
q_{vp}	\dot{q}_{vp}	vision pan angle
q_{vt}	\dot{q}_{vt}	vision tilt angle
${}^W r_{RH}$	${}^W \dot{r}_{RH}$	position of the right hand TCP frame described in the world FoR
${}^W r_{LH}$	${}^W \dot{r}_{LH}$	position of the left hand TCP frame described in the world FoR

The task-space vector $\mathbf{x} \in \mathbb{R}^p$ and the task-space velocity vector $\mathbf{v} \in \mathbb{R}^p$ with $p \in \{22, 25, 28\}$ presented in Table 4.1 together form the *task-space trajectories* describing the motion of the robot. Indeed, this represents the main output of the WPG module which is transmitted to the subsequent SIK module, see Figure 4.1. Since the SIK module uses an IK approach on velocity level and planned accelerations are not used by the stabilization algorithms modifying the task-space trajectories, it is sufficient for the WPG module to provide \mathbf{x} and \mathbf{v} . However, since the motion generator presented in this thesis uses an analytic trajectory description (see Section 6.2), computing higher order time derivatives would be cheap.

Unfortunately, dealing with rotations in three-dimensional space is always somewhat tricky. There are numerous ways of describing spatial rotation, each of which having their own advantages and disadvantages. Within the WPG module, a representation by (unit) quaternions is used almost exclusively. This is motivated by the absence of singularities, the computational efficiency (especially in chaining rotations), and the capability of creating “natural” motions when interpolating multi-axis rotation, see Appendix B. In contrast, the SIK module mainly uses rotation vectors and angular velocities, which simplifies the modification of trajectories for the purpose of stabilization. Thus, they are also used in the task-space formulation instead of the rather unintuitive quaternions and their time derivatives. As a positive side-effect, the dimension of \mathbf{x} and \mathbf{v} is kept minimal. The conversion from the quaternion formulation to the equivalent rotation vectors and angular velocities is done as a very last post-processing step of the WPG module. See Appendix B.2 for details on the conversion from one representation to the other.

In Figure 4.3, an overview of the task- and joint-space definition of LOLA is given. Since the robot is mobile, its joint-space vector \mathbf{q} contains – additional to the 26 actuated DoF – the 6D pose⁵² of the torso acting as “root” segment. Note that the torso pose can be seen as a virtual (massless) mechanism consisting of three prismatic and three rotational (passive) joints connecting the torso frame t with the world frame W . A detailed specification of the kinematic topology of LOLA (including the specification of all components of \mathbf{q}) is given in Table H.1.

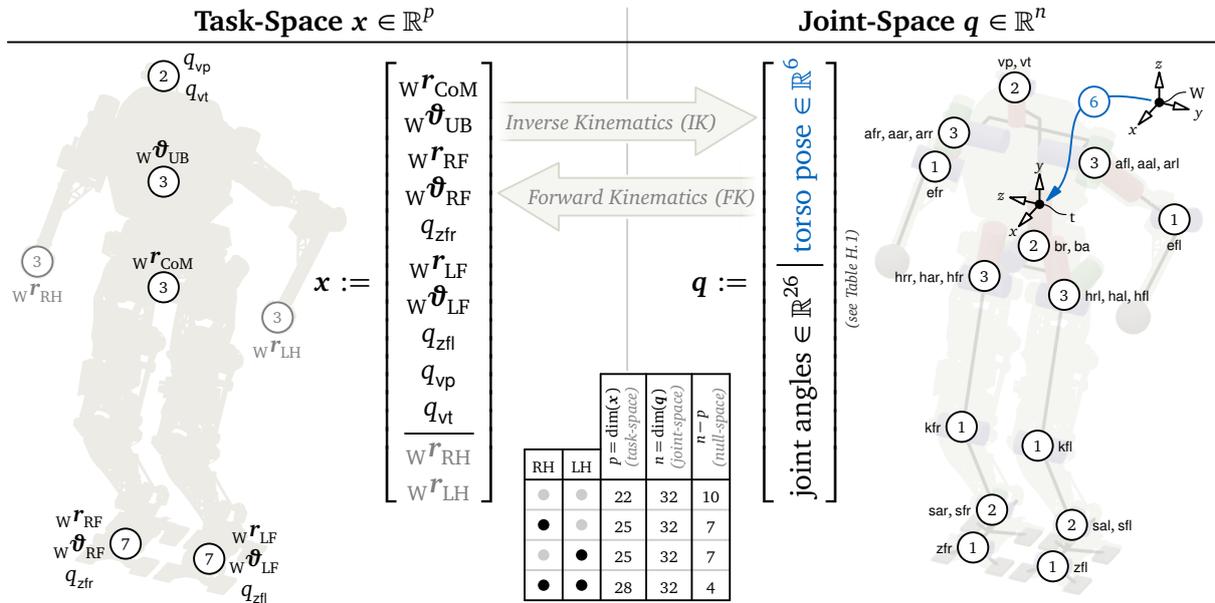


Figure 4.3: Overview of task- (left) and joint-space (right) definition of LOLA v1.1. The numbers in the circles indicate the DoF count at the corresponding location. Note that the dimension of the null-space $n - p$, i.e., the degree of redundancy, decreases when the right and/or left hand are assigned to the task-space.

4.4 Excursus: Computer Vision (CV)

The CV system processes visual input obtained from the cameras to generate a virtual reconstruction of the surroundings of the robot and provide an appropriate environment model to the WPG module, see Figure 4.1. Within the WPG, the environment model is used by the contact planner for generating a feasible contact sequence leading from the current position of the robot

⁵²The 6D pose of the torso frame t with respect to the world frame W defines the first six components of \mathbf{q} . Originally introduced by BUSCHMANN [100], the pose is specified by the orientation described with z - x - z EULER angles ($q_1 = \psi$, $q_2 = \vartheta$, and $q_3 = \varphi$, see [100, p.153]) followed by the position $q_4 = {}_W r_{t,x}$, $q_5 = {}_W r_{t,y}$, and $q_6 = {}_W r_{t,z}$.

to a user-specified goal. Indeed, the CV module is only used for fully-autonomous locomotion in real experiments. Within simulation, the environment model is instead automatically synthesized from a manually created scene description, which is also used for evaluating multi-body interactions, see Section 7.4 for details. In other locomotion modes, an explicit environment model is not required at all. Instead, either level ground is assumed (e. g. for teleoperated walking) or the contact sequence is specified manually (bypassing the contact planner). Since the CV system is not in the scope of this thesis, it is only briefly summarized to show its connection to other components of the locomotion framework. A short introduction into the fundamentals and state of the art has already been given in Section 2.5.

Revision for Multi-Contact Locomotion The previous CV system of LOLA was developed by WAHRMANN et al. and was designed to detect the floor (as plane), platforms (as polygon), and simple obstacles approximated by SSVs, see [6, 429, 430] for details. Within the context of the multi-contact revision, it has been completely replaced by a novel implementation developed by WU et al. [446, 448]. For the new CV system, the focus was set on

- SLAM using visual-inertial odometry to minimize the error between model and reality,
- semantic segmentation, i. e., splitting the scene into meaningful objects, and
- classification, i. e., labeling objects to assist the contact planner in decision making.

In our target scenarios for multi-contact locomotion, the hand-environment contacts are meant to increase the overall robustness and should not introduce additional disturbances due to unexpected impacts caused by a misperception of contact surfaces. For this reason, the estimated 6D pose of the robot should be as accurate as possible. Moreover, the rather coarse approximation of objects by SSVs is not sufficient anymore. Instead, a more detailed surface representation (in our case triangle meshes) is required for precise hand placement. However, SSVs are still used for representing the volume of an object enabling efficient distance calculations for collision avoidance and proximity tests. Finally, objects have to be classified such that the contact planner can distinguish between “pure” obstacles (e. g. a swivel chair) and objects allowing potential hand-support (e. g. walls, tables, etc.). The new CV system by WU et al. was specifically designed with regard to these additional requirements.

Interface Specification Because its development is in large parts decoupled from the rest of the locomotion system of LOLA, the CV module can be easily replaced by another solution “plugged into” the generic interface provided by the WPG module. This interface was developed by the author of this thesis and has been published as a free and open-source C++ library called *am2b-vision-interface* [16] (utilizes *Broccoli* [15] and *cpptoml* [167] as dependencies). It uses the enhanced network socket described in Section 4.1, which allows an efficient, compressed transmission of large data objects under soft real-time⁵³ conditions. An overview of the data which is cyclically exchanged through this interface is given in the Tables 4.2 and 4.3.

The members t_{cur} and t_{acq} are used to synchronize the clocks of both endpoints and to assign timing information to a data packet (e. g. the time of capturing images, i. e., the time of the environment “snapshot”). The WPG further transmits an estimation of the robot’s current state, in particular with regard to the vision TCP frame (\mathbf{r}_{VTCP} , \mathbf{s}_{VTCP} , $\dot{\mathbf{r}}_{\text{VTCP}}$, and $\boldsymbol{\omega}_{\text{VTCP}}$) and the torso IMU frame (\mathbf{r}_{IMU} , \mathbf{s}_{IMU} , $\dot{\mathbf{r}}_{\text{IMU}}$, and $\boldsymbol{\omega}_{\text{IMU}}$), both described in the planning world frame PW. The robot’s current state is computed by combining (planned) odometry, measurements from the torso IMU, and joint encoder data through a rather simple sensor data fusion which is described in more detail in Section 4.5.3. The motion of the VTCP and IMU frame together with the

⁵³While the WPG, SIK, and HWL modules are connected through a shared memory interface within a precisely timed 1 kHz cycle, a CV system typically runs at a much lower update rate of about 30 Hz (often limited by the cameras). For comparison: the delay introduced by the (local) *Ethernet* network and the TCP/IP stacks of both endpoints is expected to be much less than 3 ms, which would represent < 10% of the cycle time at 30 Hz.

Table 4.2: Data transmitted from the WPG to the CV module (see the class `ControlToVisionContainer` in *am2b-vision-interface* [16]). This container is sent with a typical cycle frequency of 100Hz. The external motion capture data (last row, gray) is optional and only used for calibration and (offline) analysis.

Symbol	Description
t_{cur}	current time according to HWL “master” clock (<i>Thread 1</i> in Figure 4.1)
t_{acq}	time of acquiring the data packaged within this container (acc. to master clock)
${}_{\text{PW}}\mathbf{r}_{\text{VTCP}}$	position of vision TCP frame described in planning world FoR
${}_{\text{PW}}\mathbf{s}_{\text{VTCP}}$	orientation of vision TCP frame described in planning world FoR
${}_{\text{PW}}\dot{\mathbf{r}}_{\text{VTCP}}$	abs. transl. velocity of vision TCP frame described in planning world FoR
${}_{\text{PW}}\boldsymbol{\omega}_{\text{VTCP}}$	abs. angular velocity of vision TCP frame described in planning world FoR
${}_{\text{PW}}\mathbf{r}_{\text{IMU}}$	position of (torso) IMU frame described in planning world FoR
${}_{\text{PW}}\mathbf{s}_{\text{IMU}}$	orientation of (torso) IMU frame described in planning world FoR
${}_{\text{PW}}\dot{\mathbf{r}}_{\text{IMU}}$	abs. transl. velocity of (torso) IMU frame described in planning world FoR
${}_{\text{PW}}\boldsymbol{\omega}_{\text{IMU}}$	abs. angular velocity of (torso) IMU frame described in planning world FoR
$\varphi_r, \varphi_p, \varphi_y$	Roll φ_r , pitch φ_p , and yaw φ_y measured by (torso) IMU (raw sensor data)
${}_{\text{IMU}}\boldsymbol{\omega}_{\text{IMU}}$	abs. angular velocity of IMU described in IMU FoR (raw sensor data)
${}_{\text{IMU}}\ddot{\mathbf{r}}_{\text{IMU}}$	abs. transl. acceleration of IMU described in IMU FoR (raw sensor data)
–	list of high-level signals, e. g. “start calibration”, “reset scene”, etc.
–	data stream from external motion capture system (optional)

Table 4.3: Data transmitted from the CV to the WPG module (see the class `VisionToControlContainer` in *am2b-vision-interface* [16]). This container is sent whenever new information about the environment is available (depends on the cycle time of the reconstruction pipeline).

Symbol	Description
t_{acq}	time of acquiring the data packaged within this container (acc. to master clock)
${}_{\text{PW}}\mathbf{r}_{\text{VTCP}}$	position of vision TCP frame described in planning world FoR
${}_{\text{PW}}\mathbf{s}_{\text{VTCP}}$	orientation of vision TCP frame described in planning world FoR
${}_{\text{VW}}\mathbf{r}_{\text{VTCP}}$	position of vision TCP frame described in vision world FoR
${}_{\text{VW}}\mathbf{s}_{\text{VTCP}}$	orientation of vision TCP frame described in vision world FoR
–	list of high-level signals, e. g. “calibration finished”, “reset model”, etc.
–	list of terrain patches (height map) to reset or update (see Section 4.5.1)
–	list of objects to remove or update (see Section 4.5.1)

raw sensor data of the torso IMU ($\varphi_r, \varphi_p, \varphi_y, \boldsymbol{\omega}_{\text{IMU}}$, and $\ddot{\mathbf{r}}_{\text{IMU}}$) is meant to be used by the CV system for improving the localization of the robot within the reconstructed scene. Finally, the WPG module sends a list of high-level signals to control the CV system (forwarded from the UI), which is typically empty most of the time. The WPG-CV interface is also used to transmit a pre-processed data stream of the external motion capture system. It has to be highlighted, that the CV system of LOLA does *not* use this data during experiments. The external motion capture data is only used for the purpose of calibration and (offline) analysis, e. g. as “ground truth” reference for evaluating the performance of the localization method.

In the opposite direction, the CV module transmits recent changes of the environment model formulated as lists of terrain patches and objects to update, see Section 4.5.1. Moreover, it sends the pose of the vision TCP frame described in two different FoR: the vision world frame VW and the planning world frame PW. The first variant (${}_{\text{VW}}\mathbf{r}_{\text{VTCP}}$ and ${}_{\text{VW}}\mathbf{s}_{\text{VTCP}}$) represents the result of visual-inertial odometry, e. g. obtained from the tracking camera integrated in the head of LOLA. The second variant (${}_{\text{PW}}\mathbf{r}_{\text{VTCP}}$ and ${}_{\text{PW}}\mathbf{s}_{\text{VTCP}}$) is a simple copy of the pose previously received from the WPG module with the restriction that it was taken from the time when the last image used

for visual-inertial odometry was captured by the camera. This way, the CV system ensures that both descriptions of the VTCP frame are synchronized, i. e., linked to the same time t_{acq} . Hence, through

$$\mathbf{p}_W \mathbf{s}_{VW} = \mathbf{p}_W \mathbf{s}_{VTCP} \otimes \mathbf{v}_W \bar{\mathbf{s}}_{VTCP} \quad \text{and} \quad \mathbf{p}_W \mathbf{r}_{VW} = \mathbf{p}_W \mathbf{r}_{VTCP} - \underbrace{\text{rotMat}(\mathbf{p}_W \mathbf{s}_{VW})}_{\mathbf{p}_W \mathbf{r}_{VW-VTCP}} \mathbf{v}_W \mathbf{r}_{VTCP} \quad (4.1)$$

it is possible to determine the current position $\mathbf{p}_W \mathbf{r}_{VW}$ and orientation $\mathbf{p}_W \mathbf{s}_{VW}$ of the vision world frame described in the planning world frame, which fully specifies the transform $\mathbf{p}_W \mathbf{H}_{VW}$ between these two frames (cf. Appendix A). Since the environment model is described with respect to the VW frame⁵⁴, this information is essential for the WPG which internally uses PW as FoR. Note that for real-world experiments the transform $\mathbf{p}_W \mathbf{H}_{VW}$ changes over time due to inevitable drift caused by slippage. By updating $\mathbf{p}_W \mathbf{H}_{VW}$ according to Equation 4.1 with every message received from the CV system, a feedback loop is realized effectively eliminating any discrepancy between model and reality caused by drift.

Workflow The CV system developed by WU et al. is built on top of dense mapping algorithms which utilize the full depth input from the RGB-D sensor located in the head of LOLA (cf. Figure 3.17). Indeed, two variants of the reconstruction pipeline have been investigated. The first variant, *Semantic Completion Fusion* alias *SCFusion* [446], focuses on the completion of missing geometry which is typically caused by view occlusion. The missing geometry is extrapolated by a data-driven deep neural network trained to recognize objects by their shape. Real-time performance is achieved through incremental updates of regions and acceleration through parallel execution on the CPU and GPU. The second variant, *Scene Graph Fusion* alias *SGFusion* [448], represents a more lightweight and efficient method for scene reconstruction. It fuses the input depth stream into a map of so-called *surfels*. Although related work defines the term *surfel* in slightly different ways, its core concept of an “oriented disc” is mostly the same. Within the reconstruction pipeline of WU et al., a *surfel* explicitly assigns an orientation (describing the local surface normal), a radius (derived from the local sparsity of the input point cloud), and a *confidence* value (measure for the local shape accuracy) to a 3D point representing the center of the disc. For segmentation and classification of the surfel map, a graph neural network is used.

The source code for both, *SCFusion* and *SGFusion*, has been made publicly available through [447] and [449], respectively. For obtaining the results presented within this thesis, the second, surfel-based variant was used. In order to generate a representation of the environment which can be used by the contact planner of the WPG module, the surfel map is further converted into

- a model of the terrain represented as conventional height-map (by “rasterizing” surfels on a 2D grid aligned with the x - y -plane of the vision world frame),
- a model of the (per-object) surface represented as triangle mesh (by using *SurfelMeshing* as introduced by SCHÖPS et al. in [368]), and
- a model of the (per-object) volume represented as SSV segment (by extracting Euclidean clusters and computing a minimum volume bounding box estimation).

Within this conversion, the confidence value of each surfel is transferred to the corresponding height map and triangle mesh representations, which makes local confidence information available to the contact planner. A more detailed specification of the environment model is postponed to Section 4.5.1. The surfel map reconstruction and the resulting environment model is shown in Figure 4.4 for two exemplary multi-contact scenarios. For a clearer, animated presentation, the video [17] is recommended.

⁵⁴Typically, large parts of the environment model (in particular all regions which do not lie in the current field of view of the vision sensors) remain static with respect to the vision world frame. Since the relative transform $\mathbf{p}_W \mathbf{H}_{VW}$ between VW and PW changes over time (drift), describing the environment with respect to the planning world frame would require frequent and expensive coordinate transformations of the entire model.

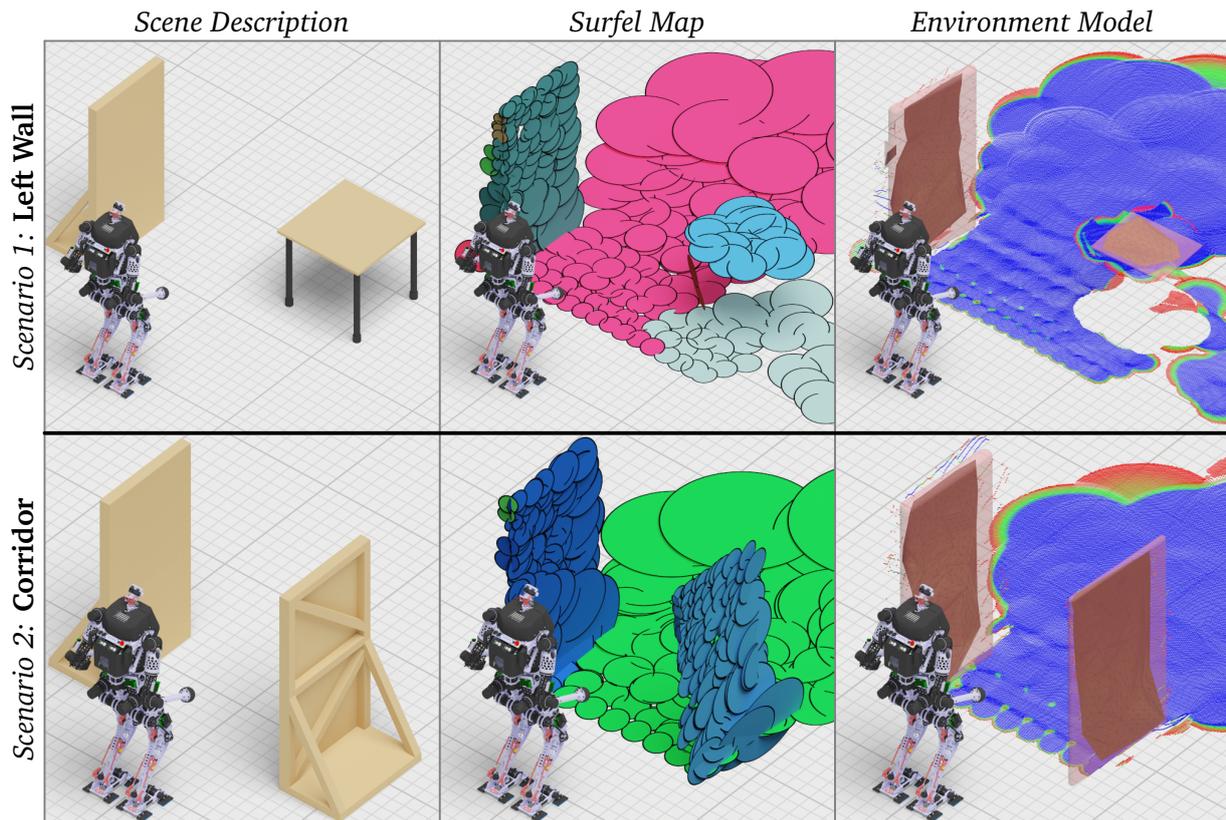


Figure 4.4: Scene reconstruction by the *Computer Vision (CV)* system for two multi-contact scenarios: “Left Wall” (top) and “Corridor” (bottom). From left to right: description of the scene, reconstructed surfel map (colorized according to segmentation), and final environment model consisting of the terrain as height map (colorized according to confidence) and objects represented by triangle meshes (surface) and SSVs (volume). See also the video [17].

4.5 Walking Pattern Generation (WPG)

The WPG system is responsible of planning a feasible motion (in the form of ideal task-space trajectories) to be transmitted to the subsequent SIK module, see Figure 4.1. For this purpose, it maintains an environment model and formulates its own simplified representation of the kinematics and dynamics of the robot, see the Sections 4.5.1 and 4.5.2, respectively. Moreover, it maintains an estimation of the current state of the robot which is obtained from a fusion of planned and measured data, see Section 4.5.3. The environment model represents the primary input of the contact planner which assembles a discrete contact sequence connecting the current with a user-specified target position. The reduced model of the robot together with the solution strategy described in Section 4.5.4 is used by the motion generator to derive a kinematically and dynamically feasible gait. Both, the contact planner and motion generator, are integrated into a sequential planning pipeline which is introduced in Section 4.5.5.

The original WPG system of LOLA was designed and implemented by BUSCHMANN [100]. At that time, LOLA’s core capabilities were teleoperated walking and rather rudimentary autonomous walking realized through a reactive contact planner which has already been described in Section 2.3. Later, HILDEBRANDT et al. [13] extended this WPG system by a discrete, A*-based footstep planner which allowed the robot to autonomously step on platforms (represented by polygons) and bypass or traverse small obstacles (represented by SSVs). In order to enable multi-contact locomotion, the previous WPG system would have required extensive changes in almost all parts of the source code. Additionally, after more than 15 years of continuous modifications and extensions, this module had definitely reached the end of its life, especially with regard to robustness, efficiency, traceability, coding style, and documentation. For this reason,

the author of this thesis recreated the entire WPG module from scratch. This resulted in native support for multi-contact locomotion but also partial footholds, i. e., toe-only contacts of the foot (e. g. for walking on tiptoes and climbing stairs). Furthermore, the new planning pipeline has a modern and modular design with clear interfaces, which allows an easy extension and/or replacement of components in future. Indeed, the new WPG module represents the author’s main contribution to the locomotion system of LOLA. The following sections introduce core planning concepts and give an overview of the planning pipeline, while the actual contact planner and motion generator are described in the Chapters 5 and 6, respectively.

4.5.1 Environment Model

In order to safely navigate through space, a mobile robot needs information about its surroundings. For this purpose, a model of the environment has to be present which translates raw perception data from the physical world (e. g. a 3D point cloud) into a format which is usable for path-planning. Note that the state of the art for environment representations has already been briefly reviewed in Section 2.3 (see also Figure 2.14). For structured environments such as an industrial plant, one might generate the environment model from a corresponding CAD model of the facility such that the robot only needs to localize itself. Within the scope of this thesis, the focus lies on previously unknown and unstructured environments, thus, we assume that the model is originally generated and cyclically updated by a CV system. From the point of view of the WPG module, we handle the CV system as a black box since its actual realization does not matter as long as it supports the interface specified in Section 4.4.

Terrain Although multi-contact locomotion is our primary objective, another core requirement of the new WPG system is that LOLA must not lose existing biped skills, e. g. stepping up and down platforms and bypassing or traversing small obstacles. The previous environment description by HILDEBRANDT et al. and WAHRMANN et al. [6, 202] relied on a rather coarse approximation of the ground by planes (floor) and polygons (platforms). With the new WPG, a more feature-rich representation of the terrain through a height map is used. Apart from being more general, this allowed to extend the biped capabilities of LOLA to climbing stairs with short treads (partial contact) and walking up and down ramps (inclined ground).

In order to allow efficient real-time updates (the field of view of a CV sensor typically covers only a small part of the entire scene), the terrain is partitioned into so-called *patches* which are aligned with the vision world frame VW, see Figure 4.5. This also reduces memory consumption since patches are only allocated if necessary (dynamic map size).

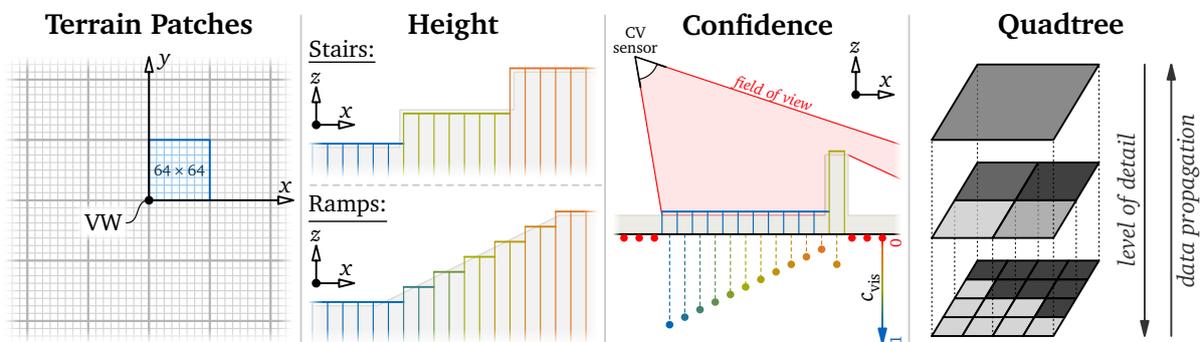


Figure 4.5: Representation of the terrain within the new environment model formulation. From left to right: partitioning of the x - y -plane of the VW frame into patches (each containing 64×64 cells of size $1 \text{ cm} \times 1 \text{ cm}$); discrete per-cell height values approximating the shape of the ground (resolution 1 mm); discrete per-cell confidence values c_{vis} (256 steps, e. g. derived from the distance to the CV sensor and/or the local point cloud sparsity); single terrain patch formulated as a quadtree (7 levels, $2^{7-1} = 64$) storing minimum, maximum, and mean of height and confidence data.

Each terrain patch consists of $64 \times 64 = 4096$ cells, where each cell stores an individual height and confidence value. Since the z -axis of the VW frame is defined⁵⁵ to point into the opposite direction of gravitational acceleration (see Section 4.2) and the terrain patches lie in the x - y -plane, the height value indeed represents the “real” (physical) height of a cell relative to a horizontal ground plane. The confidence value $c_{\text{vis}} \in [0, 1]$ indicates with which certainty the corresponding cell has been observed by the CV system. Here, a value of 1 denotes the highest possible confidence, while 0 indicates that the corresponding cell has not been detected yet. The confidence data will be used by the contact planner to avoid uncertain regions.

With a horizontal resolution of 1 cm and a vertical resolution of 1 mm, the discretization of the terrain is rather fine when compared to related work. In general, using a high-res terrain for contact planning leads to a more accurate placement of footholds which typically leads to better overall results. Unfortunately, this also implies higher computational costs. In order to maintain real-time performance, the contact planner presented in Chapter 5 is designed to work on different levels of detail (coarse to fine). For this reason, we implement each terrain patch as a quadtree (7 levels), where the lowest level cells store the data obtained from the CV system and the higher level cells are filled by propagating data upwards (fine to coarse). Each cell of the quadtree stores minimum, maximum, and mean of the corresponding height and confidence values such that comprehensive terrain information is available on different levels of detail.

Objects Previously, each environmental object was modeled by an enclosing SSV segment only, cf. WAHRMANN et al. [6]. Since bypassing and traversing obstacles can be realized using the newly introduced height map, an explicit object representation is not required to this end. However, in contrast to the previous system, the new WPG does not handle objects exclusively as obstacles but instead may use them for additional hand support. Thus, we need to model objects independently of the terrain. In particular, an accurate representation of the object’s surface is required for precise hand placement during multi-contact locomotion. Moreover, an approximation of the object’s volume is necessary for efficient distance evaluations within collision and proximity tests. We formally specify an environmental object through (see also Figure 4.6)

- a unique identifier for efficient handling within an object database,
- its classification (e. g. “floor”, “wall”, “table”, “chair”, etc.),
- the 6D pose $({}_{\text{VW}}\mathbf{r}_{\text{EO}}, {}_{\text{VW}}\mathbf{s}_{\text{EO}})$ of the object’s local frame “EO” relative to the VW frame,
- a surface model formulated as a triangle mesh described in the EO frame, and
- a volume model formulated as an SSV segment described in the EO frame.

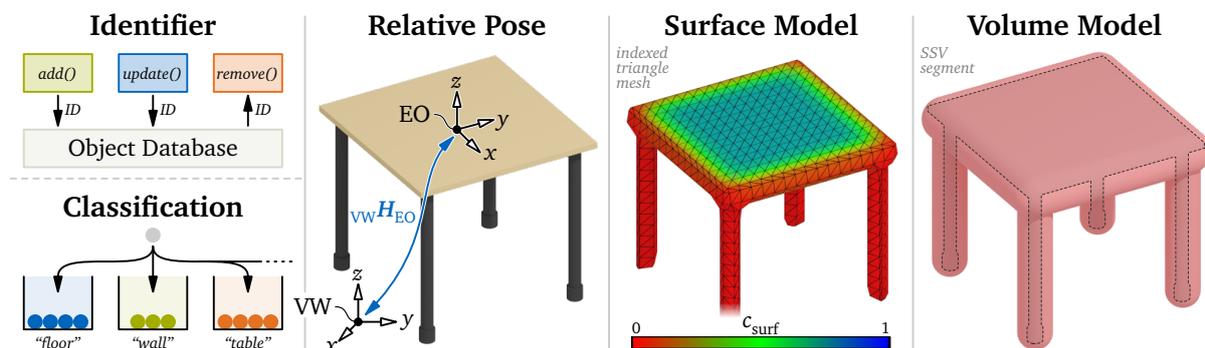


Figure 4.6: Representation of an object within the new environment model formulation. From left to right: unique identifier for efficient handling within an object database; classification (e. g. derived by CV system from semantics); relation between *Environmental Object* frame “EO” and vision world frame VW; surface modeled as indexed triangle mesh (colorized according to local *surface confidence* c_{surf} , cf. Section 5.5.2); volume modeled as SSV segment.

⁵⁵In our case, the CV system guarantees this by initializing the VW frame in a static configuration using measurements of the IMU (the one integrated into the tracking camera) to identify the direction of gravitational acceleration.

Describing the surface and volume model relative to the local EO frame allows efficient updates of an object’s pose without expensive transformation of geometry data (e. g. for moving rigid objects). The surface is modeled as an indexed triangle mesh which contains buffers for

- vertex coordinates described in the EO frame (alias *vertex buffer*),
- per-vertex normals described in the EO frame (alias *normal buffer*),
- per-vertex (real-world) colors used for visualization purposes (alias *color buffer*),
- per-vertex confidence values (alias *confidence buffer*), and
- 3-tuples of vertex indices where each tuple specifies a triangle (alias *index buffer*).

Within the scope of this thesis, we assume that the CV system transmits only non-degenerated triangles (non-zero area). Moreover, we require neighboring triangles to be coupled through their vertices (same vertex indices of shared edge). This gives us the required topology information for building the n -ring neighborhoods for vertices, edges, and triangles, such that we can efficiently “move along” the surface. This is used in Chapter 5 to find an optimal contact point for hand support. Note that the mesh does not necessarily have to be closed, i. e., there might be one or multiple “borders” (edges which are connected to a single triangle).

The normal buffer carries additional information about the local shape of the surface. If not provided by the CV system, angle weighted normals (as introduced by THÜRRNER and WÜTHRICH [410]) are computed as fallback. Besides an individual normal and color (used for visualization only), each vertex is also linked to a certain confidence value. For surface models, the WPG module distinguishes between the *visual perception* confidence c_{vis} describing a measure for the local shape accuracy and the *multi-contact / surface* confidence c_{surf} as a local metric justifying how well the corresponding point is suited for a potential hand contact. The visual perception confidence c_{vis} is transmitted by the CV system similar to the terrain confidence. The surface confidence c_{surf} represents a fusion of the visual perception confidence, the local mesh curvature, and the distance to the surface boundary (for open meshes). See Section 5.5.2 for details on the pre-processing of environmental surfaces.

There are much less constraints with respect to the volume model. Indeed, the only requirements are that the SSV segment represents a meaningful approximation of the object’s geometry and that it covers the entire geometry, i. e., it encapsulates all triangles of the surface mesh. Since the SSV segment is used for collision avoidance, a too small volume (not fitting the entire object) is not allowed. In contrast, a too large volume is still valid but may have negative influence on the performance of the contact planner.

Real-Time Logging For the purpose of analysis, debugging, and visualization, it is necessary to log the environment model. Especially for dense scenes with many objects the amount of data becomes considerable. Hence, in order to maintain real-time performance within experiments, one has to carefully select the format in which the environment model is logged. Here we choose a hybrid approach which combines simplicity and readability for lightweight data and efficiency and compactness for heavyweight components.

For the enclosing container and small data such as object identifiers, classifications, and 6D poses, the human-readable *Tom’s Obvious Minimal Language (TOML)* [351] is used. As a configuration file format, TOML is simple to write and parse and makes manual modifications easy (e. g. for debugging purposes). Since an object’s volume is simply described by a group of point-, line-, and triangle-SSV elements, we directly embed it into the TOML container.

For the triangle mesh describing an object’s surface, we use the *Polygon File Format (PLY)* [416]. PLY is a native format for indexed triangle meshes with per-vertex normals and colors and comes with a computationally efficient binary variant (uncompressed). Moreover, it allows custom attributes which is used in our case to encode per-vertex confidence values. For terrain patches and their corresponding quadtrees, the *Portable Anymap Format (PNM)* [347] as collection of native 2D raster image formats is used. Besides a Boolean variant, the PNM family

includes also a one- and three-channel (each 8bit or 16bit depth) pixel format. Depending on the type of 2D data, we pick the variant with the least overhead. The PNM formats are rather simple and lack (integrated) compression. However, their binary variants allow extremely fast reading and writing (copy of entire memory block) such that they are a suitable choice for real-time systems. Another advantage is the wide support by image manipulation software.

For interfacing the TOML format, the *cpptoml* [167] library is used. In contrast, custom implementations are used for the PLY and PNM formats (see the module *io* in *Broccoli*). This allowed further optimizations and an efficient integration into the WPG module. Depending on the type of experiment, either the entire environment model or its components (e. g. triangle meshes and terrain patches) are compressed using *zlib* [165] to reduce memory consumption. Moreover, PLY and PNM structures are either referenced as separate files, or directly embedded into the TOML container using the well-known *Base64* [222] binary-to-text encoding.

4.5.2 Reduced Kinematic and Dynamic Model

The WPG proposed within this thesis plans abstract motion based on a simplified representation of the robot. In the following, reduced models of LOLA’s kinematics and dynamics are presented which together provide a reasonable approximation of the full system while still allowing an efficient online evaluation. In Chapter 6, these models will be used to plan a kinematically (joint limits) and dynamically (balanced gait) feasible motion in task-space. Note that the state of the art for simplified robot models has already been briefly reviewed in Section 2.4.

Idle Pose As preparation, we first introduce the so-called *idle* pose as the configuration of the robot which represents “default standing” shown in Figure 2.21 right. In particular, we define the task-space vector \mathbf{x}_{idle} such that

- the GCoM coincides with the centroid of the SP (static equilibrium with maximum safety margins) while the CoM has a height of 0.9 m above the ground,
- the upper body is upright (${}_W\mathbf{s}_{\text{UB}} = \mathbf{1}_{\mathbb{H}}$),
- the feet are parallel to each other and have full contact with the ground (${}_W\mathbf{s}_f = \mathbf{1}_{\mathbb{H}}$ and $q_{z\text{frl}} = 0$) while their TCPs have a lateral separation of 0.275 m,
- the hand TCPs have a lateral separation of 0.68 m, a height of 0.97 m above the ground, and a shift in positive walking direction of 0.22 m relative to the foot TCPs, and
- the head is inclined to look at the ground in front of the robot ($q_{\text{vp}} = 0$ and $q_{\text{vt}} = -25^\circ$).

The given default height of the CoM has proven to be a suitable choice since slightly bent knees provide a certain kinematic “reserve” (e. g. for walking on uneven ground). Note that due to the two joints in the pelvis of LOLA, an outstretched knee does *not* represent a singularity in general. Within idle, the only EEs in contact are the feet such that we choose the hands to be in null-space. However, we still define an idle position for the hands since it will be used as starting point for blending the hands from null- to task-space, see Section 6.9 for details.

After defining the idle pose in task-space \mathbf{x}_{idle} , we can compute the corresponding joint-space configuration \mathbf{q}_{idle} . For this purpose, we simply trigger the same velocity-level IK as used by the SIK module and stop the iteration upon convergence ($\|\Delta\mathbf{q}_{\text{idle}}\| \leq \varepsilon$). Since we define the idle pose to be static, we use $\mathbf{v}_{\text{idle}} = \mathbf{0}$ which consequently leads to $\dot{\mathbf{q}}_{\text{idle}} = \mathbf{0}$. Indeed, each action of the robot (e. g. teleoperated walking, autonomous walking, balancing, etc.) is planned to start and end in the static idle pose. This guarantees that actions can be chained arbitrarily.

The Models The reduced models for LOLA’s kinematics and dynamics are depicted in Figure 4.7 while their constant parameters are specified in Table 4.4. The kinematic model has a strong focus on the leg mechanism in the sagittal plane which is in particular relevant to avoid

joint limits when climbing stairs. In contrast, the dynamic model considers the system as a whole and approximates the robot by five point masses and an additional mass moment of inertia. The models are coupled through the positions of the foot and hand TCPs and the position of the “virtual” torso alias *Reduced Model Torso (RMT)*, in the following abbreviated with “t”. **Attention:** The RMT is not to be confused with the “physical” torso segment of the robot, whose position is not part of the task-space vector \mathbf{x} and therefore not known to the WPG. Instead, the RMT describes the position of the central mass element of the five-mass model.

Table 4.4: Parametrization of the reduced kinematic and dynamic model of the robot (see also Figure 4.7). The lengths l_i and the diameter d_h are derived from the CAD model of the robot. An exception is l_7 , which is computed from the idle pose ($\mathbf{x}_{\text{idle}}, \mathbf{q}_{\text{idle}}$) by resolving the kinematic chain $l_1 \dots l_7$ for the default CoM height of 0.9 m. The mass proportions M_f with $f \in \{\text{RF}, \text{LF}\}$ and M_h with $h \in \{\text{RH}, \text{LH}\}$ are related to a *single* foot or hand, respectively.

Parameter	Description	Parameter	Description	Parameter	Description
$l_1 = 48 \text{ mm}$...toe height	$l_8 = 108.5 \text{ mm}$...heel height	$l_{15} = 482.3 \text{ mm}$...backbone length
$l_2 = 14.5 \text{ mm}$...toe shift	$l_9 = 46.5 \text{ mm}$...heel shift	$l_{16} = 446 \text{ mm}$...shoulder width
$l_3 = 130.1 \text{ mm}$...mid foot length	$l_{10} = 85 \text{ mm}$...toe length	$l_{17} = 383 \text{ mm}$...upper arm length
$l_4 = 60.5 \text{ mm}$...mid foot height	$l_{11} = 85 \text{ mm}$...heel length	$l_{18} = 345 \text{ mm}$...lower arm length
$l_5 = 430 \text{ mm}$...lower leg length	$l_{12} = 276.1 \text{ mm}$...foot length	$d_h = 100 \text{ mm}$...hand diameter
$l_6 = 440 \text{ mm}$...upper leg length	$l_{13} = 220 \text{ mm}$...foot width	$M_f = 11 \%$...foot mass proportion
$l_7 = 252.9 \text{ mm}$...distance hip to RMT	$l_{14} = 246 \text{ mm}$...hip width	$M_h = 2 \%$...hand mass proportion

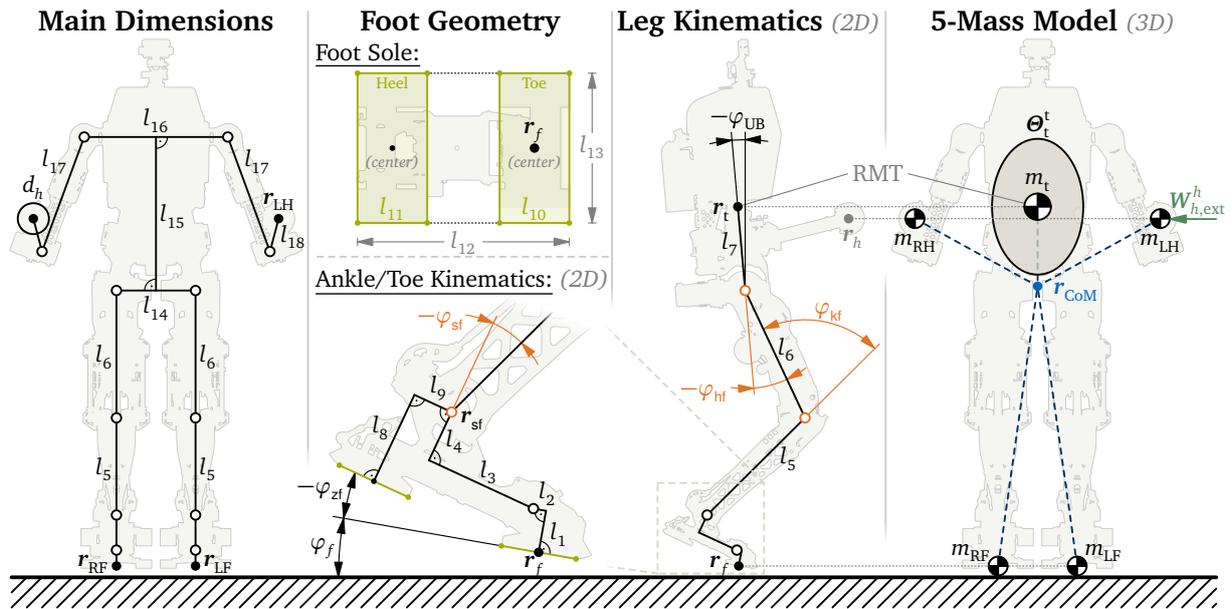


Figure 4.7: Reduced models of LOLA’s kinematics and dynamics as used by the WPG. From left to right: overview of characteristic dimensions; foot sole model approximating the contact pads by a rectangular toe and heel segment (green); simplified 2D (sagittal plane) ankle and toe kinematics; simplified 2D (sagittal plane) leg kinematics; five-mass model approximating the robot’s 3D dynamics. With l_i , the corresponding 3D dimension is meant and *not* the projected length. Angles φ_i highlighted in orange indicate that the corresponding rotational DoF is not (explicitly) defined by the task-space vector \mathbf{x} . The CoM of the five-mass model (blue) defines also the (desired) CoM of the full robot. The kinematics and dynamics are linked through the positions \mathbf{r}_e of the EE masses m_e (with $e \in \{\text{RF}, \text{LF}, \text{RH}, \text{LH}\}$) and the position \mathbf{r}_t of the “virtual” torso mass m_t alias *Reduced Model Torso (RMT)*. Multi-contact dynamics are incorporated through the external contact wrench $\mathbf{W}_{h,\text{ext}}^h$. Constant parameters are specified in Table 4.4.

Kinematics The approximation of the robot’s main dimensions and foot geometry allows efficient checks for reachability (multi-contact) and state feasibility (footholds) within contact planning. The simplified model of the leg is mainly used by the motion generator to compute an upper and lower bound for the vertical RMT position ${}^w r_{t,z}$ during challenging maneuvers. The

2D leg mechanism is derived from a projection⁵⁶ of the corresponding joints onto the sagittal plane. Indeed, the kinematic model is almost identical to the one introduced by HILDEBRANDT et al. in [8]. However, this thesis uses different approaches for computing the vertical RMT boundaries (see Appendix E) and applying them to ${}^W r_{t,z}$ (see Section 6.14). Note that junctions and endpoints in the presented kinematic model indicate special physical locations, e. g. a joint axis or a TCP. An exception is r_t , which denotes the position of the purely “virtual” RMT. As a consequence, we cannot derive l_7 (the constant distance from the hip to the RMT) from the CAD model of the robot. Instead, l_7 is computed by resolving the kinematic chain from r_f to r_t in the idle pose (with ${}^W r_{t,z}$ obtained from Equation F.9 using the CoM height in idle).

Dynamics The five-mass model shown in Figure 4.7 is an extension of the three-mass model originally introduced by BUSCHMANN et al. in [98] (see Section 2.4). In particular, the five-mass model adds

- hand masses m_h to incorporate the dynamic effects of fast arm motion,
- a dynamic mass distribution to allow blending the hands between null- and task-space,
- a mass moment of inertia tensor Θ_t linked to the “virtual” torso (centered at the RMT) to incorporate the dynamic effects of upper body rotation, and
- arbitrary (known) external contact wrenches $W_{h,\text{ext}}^h$ to incorporate multi-contact effects.

In case the hand $h \in \{\text{RH}, \text{LH}\}$ is blended into null-space, the actual position r_h of m_h is a result of the (local) null-space optimization performed by the SIK module and thus, not known to the WPG in advance. In order to avoid wrong contributions to the overall dynamics, we set $m_h = 0$ in this case. In particular, we use the dynamic mass distribution

$$m_f := M_f m = \text{const.}, \quad m_h(t) := \xi_h(t) M_h m, \quad m_t(t) := m - \sum_e m_e(t) \quad (4.2)$$

with $f \in \{\text{RF}, \text{LF}\}$, $h \in \{\text{RH}, \text{LH}\}$, $e \in \{\text{RF}, \text{LF}, \text{RH}, \text{LH}\}$, where $m := 67.3 \text{ kg}$ denotes the total mass of the robot (derived from the CAD model) and M_f and M_h are constant mass proportions for a single foot and hand, respectively. With $\xi_h(t)$, we introduce a time-dependent *task-space selection factor* for each hand which is bound to $\xi_h \in [0, 1]$ where $\xi_h = 0$ means that the corresponding hand is entirely in null-space and $\xi_h = 1$ means that it is entirely in task-space. In order to avoid discontinuities, ξ_h will be planned as a C^2 -continuous blending function (see Section 6.11 for details). Note that changes of the hand masses m_h are compensated by m_t such that the total mass

$$m = m_t(t) + \sum_e m_e(t) = \text{const.} \quad (4.3)$$

remains constant and there is no mass transport over system boundaries. A minor drawback of this approach is that one has to pay special attention to the time-dependent masses during analysis. A comprehensive derivation of the EoM of the five-mass model is given in Appendix F.

The constant mass proportions M_f and M_h are not chosen arbitrarily but instead are the result of a parameter optimization which has been performed by LEYERER within the context of his term paper [29]. In his work, LEYERER investigated two planar projections of the five-mass model (sagittal and frontal plane). These planar models are extended by foot-ground contacts realized through a linear spring/damper pair located at both ends of each foot sole (sagittal plane: toe/heel; frontal plane: inside/outside). Except for the hand masses, this is equivalent to the prediction model introduced by WITTMANN et al. [439], which has proven to deliver a reasonable approximation of LOLA’s dynamics – in particular with regard to disturbance rejection. Hence, the same values for the (initial) mass distribution, the mass moment of inertia, and the stiffness/damping of the foot-ground contacts have been used.

⁵⁶For walking on curved paths, the 3D motion of the robot is projected onto a “mean” sagittal plane determined by the orientation of the left and right foot, see also Section 5.1.

The actual optimization is given by the following steps:

1. Simulate a standard locomotion sequence (straight walking) with the full multi-body model of the robot using the existing simulation framework of LOLA (as “ground truth”).
2. Specify the initial mass distribution, i. e., the set of masses for the feet, hands, and torso.
3. Simulate the five-mass model in the sagittal and frontal plane separately. The “internal” motion of the robot (task-space trajectories defining the relative position of the masses) is copied from the results of Step 1, while the “external” motion of the robot (floating-base displacement and inclination) is to be computed based on the effects of gravity, inertia, and the foot-ground contact.
4. Compute a global cost value accounting for the discrepancy between the motion of the five-mass model and the full multi-body system (vertical position and planar rotation of upper body). The cost further includes penalties to avoid negative masses and to prefer a total mass close to the actual weight of the robot.
5. Select a new mass distribution based on the governing optimization scheme and repeat starting from Step 3 or stop upon convergence. Due to the complex dynamics of legged locomotion, local solvers tend to get stuck in local minima. Thus, global optimization techniques based on PSO [282] and genetic algorithms [105] were used instead.

The optimization resulted in the mass distribution specified in Table 4.4. It has been shown in [29] that the five-mass model is indeed a reasonable approximation of the full multi-body dynamics – even for a relatively long time horizon of 10 s. A summary of LEYERER’s term paper providing more details on the conducted parameter optimization has been published in [4].

Note that the resulting optimum hand masses are rather small which can be attributed to the rather simple realization of the arms. If fully articulated hands are added in a future revision of LOLA, the optimum M_h is expected to be higher. Moreover, the hand masses of the five-mass model can also be used to incorporate the dynamic effects of carrying a payload in a loco-manipulation use-case (by adding the object’s mass to m_h). For controlling the trajectory of the payload, the hands have to be assigned to the task-space (i. e. $\xi_h = 1$) such that the gravito-inertial effects of carrying the object properly contribute to the overall dynamics.

After specifying M_f and M_h , it is left to define the mass moment of inertia tensor of the virtual torso Θ_t , which is meant to account for the dynamic effects of upper body rotation. For simplicity, it is derived from the inertia tensor of the physical torso segment, i. e.,

$${}_{\text{UB}}\Theta_t^t := \begin{bmatrix} 1.03 & 0.0 & -0.27 \\ 0.0 & 1.1 & -0.01 \\ -0.27 & -0.01 & 0.27 \end{bmatrix} \text{kgm}^2. \quad (\text{cf. } {}_t\Theta_t^t \text{ from Table H.1}) \quad (4.4)$$

Note that ${}_{\text{UB}}\Theta_t^t$ is constant because it is specified with respect to the RMT (“t”) and the upper body frame (“UB”) which together define the pose of the virtual torso. It has to be mentioned that the mass moment of inertia has only a rather small contribution to the overall dynamics since the current gait of LOLA is characterized by an upright upper body. Thus, integrating a mass moment of inertia into the five-mass model can be seen as preparation for future investigations which may use a non-constant upper body inclination during locomotion. In this case, one may also find a better parametrization by performing an optimization similar to the one for the mass distribution. For now, Θ_t only affects walking along curved paths (rotation around vertical axis) and special actions such as taking a bow (rotation around horizontal axis).

Finally, it has to be mentioned that there are also other works which approximate the dynamics of a humanoid robot by a five-mass model. An example is [295] by LUO et al., where the masses for the legs and arms however are not located in the EEs (TCPs) but instead in the center of the corresponding limbs (knee/elbow). Moreover, LUO et al. use a different approach for motion generation based on the ZMP preview control by KAJITA et al. [233].

4.5.3 State Estimation

A locomotion system typically needs information about the current state of the robot. While joint angles can simply be obtained from rotary encoders, the six passive DoF of a mobile robot are much harder to determine. As already explained in Section 3.4, even the high-end IMU located in the torso of LOLA is subject to drift and does not (directly) provide a sufficiently accurate 6D pose. In the following, a rather simple method for determining the current state is presented. It is based on a fusion of different data sources and meant to provide only a rough estimate with moderate short-term accuracy. For better results, one might switch to a model-based approach instead (e. g. [394] or [450]). Note that high long-term accuracy is achieved separately through a visual-inertial localization performed by the CV system, see Section 4.4.

In order to estimate the current state on position and velocity level given by \mathbf{q}_{cur} , $\dot{\mathbf{q}}_{\text{cur}}$, \mathbf{x}_{cur} , and \mathbf{v}_{cur} , the following steps are performed within each cycle of the main WPG loop (*Thread 5* in Figure 4.1):

1. Initialize the position ${}_{\text{W}}\mathbf{r}_{\text{t,cur}}$ and translational velocity ${}_{\text{W}}\dot{\mathbf{r}}_{\text{t,cur}}$ of the torso segment with the planned position/velocity from the previous WPG cycle. Note that the IMU provides a signal for translational acceleration, however, integration would lead to significant drift.
2. Compute the orientation ${}_{\text{W}}\mathbf{s}_{\text{t,cur}}$ of the torso segment as combination of the orientation measured by the IMU (horizontal components only) and the planned orientation from the previous WPG cycle (vertical component only). Note that the used IMU is capable of canceling drift of rotation around the horizontal axes (gravity compensation) which is not possible for the rotation around the vertical axis. The angular velocity ${}_{\text{W}}\boldsymbol{\omega}_{\text{t,cur}}$ of the torso segment is simply set to the drift-free measured angular velocity of the IMU.
3. Assemble the current state in joint-space \mathbf{q}_{cur} and $\dot{\mathbf{q}}_{\text{cur}}$ through concatenating the measurements of the joint encoders and the 6D torso pose from Step 1 and 2.
4. Compute the current state in task-space \mathbf{x}_{cur} and \mathbf{v}_{cur} through evaluating $\text{FK}(\mathbf{q}_{\text{cur}}, \dot{\mathbf{q}}_{\text{cur}})$.
5. Apply a translational shift to the entire robot (\mathbf{q}_{cur} , $\dot{\mathbf{q}}_{\text{cur}}$, \mathbf{x}_{cur} , and \mathbf{v}_{cur}), such that the position and translational velocity of the foot TCPs (“mean” frame) match the planned values from the previous WPG cycle. We use the “mean” of the RF and LF frame – even in the SS phase where only one foot is in contact – since using the SF frame would result in discontinuities whenever the stance foot switches.

Through incorporating inclination measurements in Step 2 and performing the translational shift described in Step 5, we indirectly realize an inverted pendulum model for the robot (entire robot rotates around the foot-ground contact acting as pivot point). However, the presented state estimator considers kinematics only (no dynamics involved).

From the estimated state of the robot we can compute the current pose of the VTCP and IMU frame, which are both transmitted to the CV system, see Table 4.2. Currently, the contact planner and motion generator of LOLA do not make use of the state estimation. This is mainly because the WPG considers a certain planning horizon while the presented sensor data fusion only provides the current state without any extrapolation or prediction. Moreover, a drift-compensated localization of the robot within its environment is indirectly realized through the continuously updated transform ${}_{\text{PW}}\mathbf{H}_{\text{VW}}$, see Section 4.4.

In order to get an idea of the long-term accuracy, i. e., the magnitude of drift to be compensated by the visual-inertial localization of the CV system, the output of the state estimator has been evaluated within simulations and real-world experiments. Figure 4.8 shows the translational ($\Delta r_{x|y}$) and rotational ($\Delta \varphi_z$) drift for a scenario where LOLA is commanded to walk along an S-shaped path. Within the simulation, we use the simulated (full multi-body system) 6D-motion of the torso as reference. For the real-world experiment, the reference motion is obtained from optical tracking (external motion capture system by *Vicon Motion Systems*). Since the simulation cannot consider the full spectrum of real-world disturbances, the drift observed

within experiments is significantly higher (here: $\max(|\Delta r_{x|y}|) \approx 60$ mm and $\max(|\Delta \varphi_z|) \approx 3.5^\circ$ within the first 12 s). The drift is caused by slippage in the foot-ground contact which becomes worse for walking on curved paths, i. e., with foot rotation around the vertical axis. Apart from this, the drift is also strongly affected by the step length and duration (walking speed) and the surface properties of the ground.

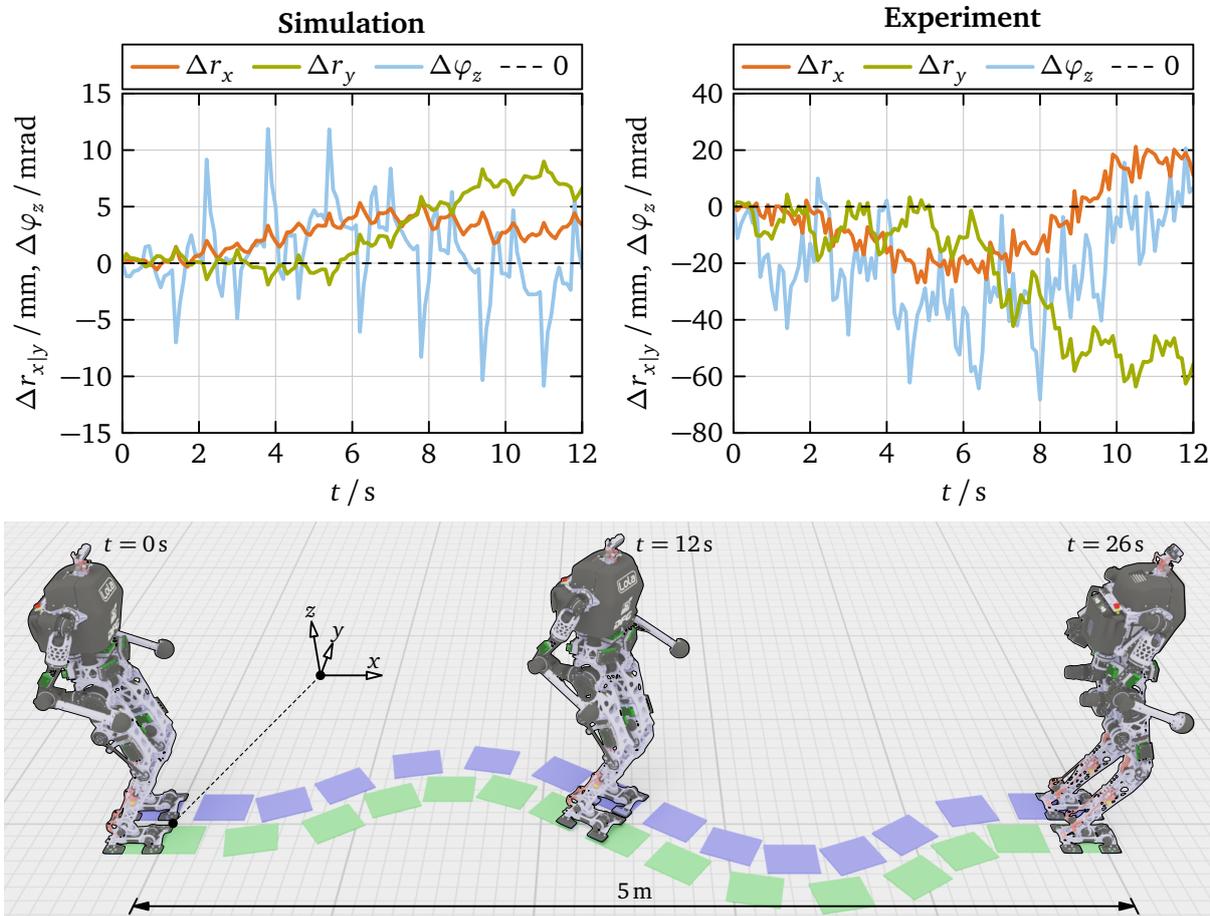


Figure 4.8: Drift of odometry for walking on a curved path. The plots show the translational and rotational error of the (estimated) torso pose relative to the reference (simulation: full multi-body system; experiment: external motion capture). At $t = 0$, all frames are aligned to start with $\Delta r_{x|y}(0) = \Delta \varphi_z(0) = 0$.

4.5.4 Solution Strategy

A motion generator for legged locomotion is mainly characterized by its strategy for accomplishing kinematic and dynamic feasibility. There exists a variety of approaches which have already been briefly summarized in Section 2.4. For LOLA, the main focus lies on robust and (comparatively) fast, gaited locomotion which is planned in real-time. Moreover, versatility is preferred over locomotion efficiency as a general premise. In accordance with these boundary conditions, the motion generator presented in this thesis uses a “classical” approach based on the combination of the ZMP concept with a simplified model of the robot. This section gives an overview of the solution strategy, i. e., the main workflow for planning a corresponding CoM motion. Details on the actual generation of task-space trajectories are given in Chapter 6. The relations to the state of the art – including advantages and limitations of the presented approach – have already been discussed in Section 2.4.

Contact Wrench and Zero-Moment Point (ZMP) The main idea of the ZMP and how it can be used to evaluate dynamic feasibility in legged locomotion has already been discussed in Section 2.4. It is left to formally relate the ZMP to the contact wrench so that it can be used in combination with the five-mass model. For this purpose, we assume a biped robot with either one foot (SS phase) or both feet (DS phase) in contact with level ground, see Figure 4.9. Indeed, the following considerations also hold true for robots with more than two feet (e. g. quadrupeds). Moreover, the feet do not have to be of rectangular shape. The only requirement is a non-degenerated SA ($A_{SA} > 0$) such that a ZMP can exist.

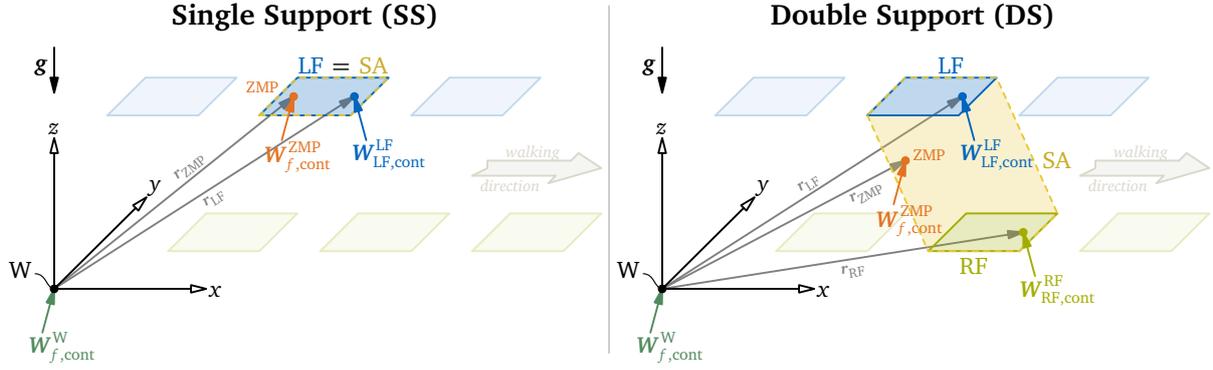


Figure 4.9: Formulation of the foot-ground interface of a biped robot in the SS (left) and DS (right) phase using the Zero-Moment Point (ZMP). The contact wrench acting on the robot can be described either with respect to the feet ($W_{RF,cont}^{RF}$ and $W_{LF,cont}^{LF}$), the world frame ($W_{f,cont}^W$), or the ZMP ($W_{f,cont}^{ZMP}$) for which the horizontal torque components vanish. The footholds are assumed to lie in a plane parallel to the x - y -plane of the world frame (perpendicular to g).

Within the scope of this thesis, we require the gait to be dynamically balanced such that the ZMP stays within the SA at all times and coincides with the CoP and FRI point [428]. The foot-ground contact wrench *acting on the robot* can be described either by

$$\begin{aligned} W_{RF,cont}^{RF} &:= [F_{RF,cont}^T, T_{RF,cont}^T]^T \text{ and } W_{LF,cont}^{LF} := [F_{LF,cont}^T, T_{LF,cont}^T]^T, \text{ or } \textit{(individual wrenches)} \\ W_{f,cont}^W &:= [F_{f,cont}^T, T_{f,cont}^T]^T, \text{ or } \textit{(combined wrench at world frame origin)} \\ W_{f,cont}^{ZMP} &:= [F_{ZMP}^T, T_{ZMP}^T]^T. \textit{(combined wrench at ZMP)} \end{aligned} \quad (4.5)$$

The relationship between these wrenches is given through the equality of forces and torques:

$$\begin{aligned} F_{f,cont} &= F_{ZMP} &= F_{RF,cont} + F_{LF,cont}, \\ T_{f,cont} &= T_{ZMP} + r_{ZMP} \times F_{ZMP} &= T_{RF,cont} + r_{RF} \times F_{RF,cont} + T_{LF,cont} + r_{LF} \times F_{LF,cont}, \end{aligned} \quad (4.6)$$

where $r_{RF} = r_{W-RF}$, $r_{LF} = r_{W-LF}$, and $r_{ZMP} = r_{W-ZMP}$ denote the position of the foot TCPs and the ZMP relative to the world frame, respectively. By inserting the definition of the ZMP

$${}_W T_{ZMP} := [0, 0, {}_W T_{ZMP,z}]^T \textit{(horizontal torque components vanish at ZMP)} \quad (4.7)$$

into Equation 4.6, we further find the relation

$${}_W T_{f,cont} = {}_W T_{ZMP} + {}_W r_{ZMP} \times {}_W F_{f,cont} = \begin{bmatrix} 0 \\ 0 \\ {}_W T_{ZMP,z} \end{bmatrix} + \begin{bmatrix} {}_W r_{ZMP,x} \\ {}_W r_{ZMP,y} \\ {}_W r_{ZMP,z} \end{bmatrix} \times \begin{bmatrix} {}_W F_{f,cont,x} \\ {}_W F_{f,cont,y} \\ {}_W F_{f,cont,z} \end{bmatrix} \quad (4.8)$$

$$\begin{bmatrix} {}_W T_{f,cont,x} \\ {}_W T_{f,cont,y} \\ {}_W T_{f,cont,z} \end{bmatrix} = \begin{bmatrix} {}_W r_{ZMP,y} {}_W F_{f,cont,z} - {}_W r_{ZMP,z} {}_W F_{f,cont,y} \\ {}_W r_{ZMP,z} {}_W F_{f,cont,x} - {}_W r_{ZMP,x} {}_W F_{f,cont,z} \\ {}_W T_{ZMP,z} + {}_W r_{ZMP,x} {}_W F_{f,cont,y} - {}_W r_{ZMP,y} {}_W F_{f,cont,x} \end{bmatrix}. \quad (4.9)$$

The first two rows of Equation 4.9 will be combined in Section 6.14 with the EoM of the five-mass model to formulate a decoupled BVP describing the horizontal RMT motion.

Obviously, the assumption of level ground required by the ZMP does not apply for scenarios with non-coplanar foot-ground contacts such as climbing stairs. Since the kinematic capabilities of LOLA allow only moderate step heights anyway (potentially less than 15 cm per step), the error introduced through violation of this requirement is assumed to be small and thus, considered as minor disturbance. In contrast, considering hand-environment interactions as planned (known) external wrenches acting on the five-mass model (see $W_{h,ext}^h$ in Figure 4.7) allows to incorporate multi-contact effects without violating any ZMP requirements, see also Figure 2.20.

Planning the CoM Motion Similar to most other ZMP-based WPGs, the motion generator presented within this thesis solves the so-called *inverse problem* [233], i. e., it first plans a ZMP reference trajectory which is then used to derive the corresponding (desired) CoM motion using the reduced model introduced in Section 4.5.2. The particular workflow of the WPG for planning a kinematically and dynamically feasible CoM motion is visualized in Figure 4.10. Note that the idealized task-space trajectories generated by the WPG are modified by the subsequent SIK module such that the real (executed) CoM and ZMP trajectories differ from their reference.

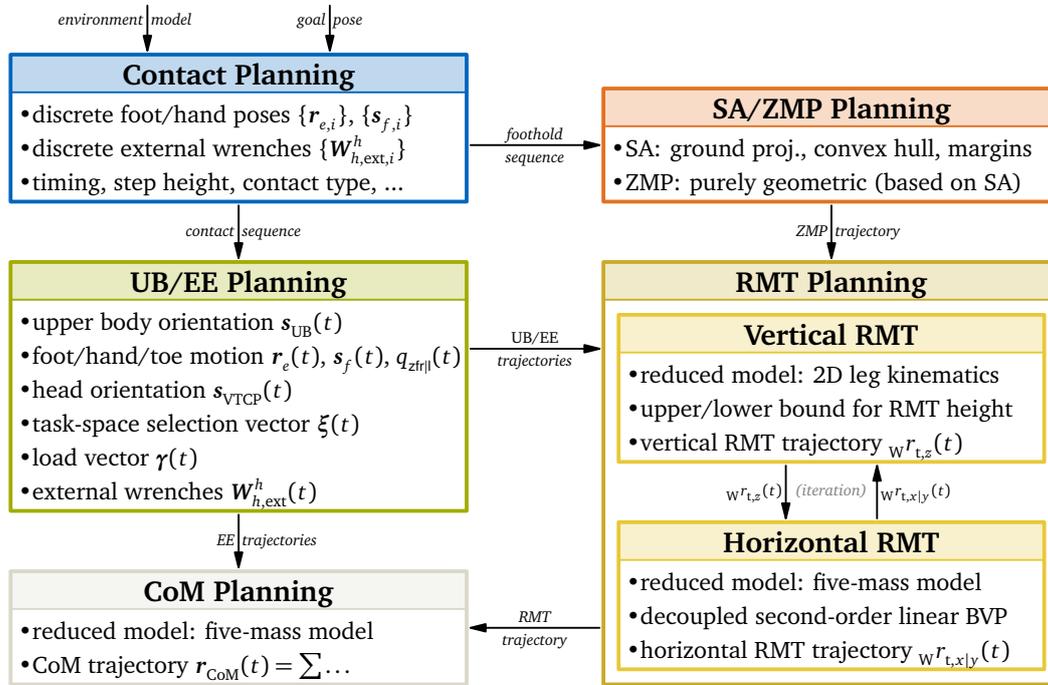


Figure 4.10: Schematic showing the main workflow for planning a feasible CoM motion within an autonomous multi-contact locomotion scenario. The kinematic and dynamic feasibility is achieved through incorporating the simplified (2D) leg kinematics and the five-mass model (cf. Section 4.5.2) within the planning of the vertical and horizontal RMT trajectory. The CoM is simply a result of the combined EE and RMT motion.

The entire process starts with the contact planner which uses the environment model to generate a feasible contact sequence guiding the robot from its current location to a user-specified goal. The sequence represents a chain of discrete poses for the feet and hands ($\{r_{e,i}\}, \{s_{f,i}\}$), which is extended in multi-contact situations by a desired (planned) external contact wrench $\{W_{h,ext,i}^h\}$ acting at the corresponding hand. Moreover, auxiliary parameters for specifying the motion in-between two discrete poses (e. g. timing, step height for stepping over obstacles, etc.) and the type of foot-ground contacts (full/partial/tiptoe, cf. Section 5.1) are added. The contact sequence is formulated as a list of *Quasi-Planar Walking Transitions (QPWTs)* which will be explained in more detail in Section 5.3.

The contact sequence is then passed to the generation of upper body and EE motion. In particular, this section generates trajectories for the upper body orientation $s_{UB}(t)$ (upright in most scenarios), the foot and hand poses $(r_e(t), s_f(t))$, the toe joint angles $q_{zfr||}(t)$ (for large step lengths, tiptoe walking, etc.), the head orientation $s_{VTCp}(t)$ (controlling the field of view of the CV sensors), the task-space selection vector $\xi(t) = [\xi_{RH}(t), \xi_{LH}(t)]^T$ (enabling/disabling the hands for multi-contact), the load vector⁵⁷ $\gamma(t) = [\gamma_{RF}(t), \gamma_{LF}(t), \gamma_{RH}(t), \gamma_{LH}(t)]^T$, and the external wrenches $W_{h,ext}^h(t)$ (multi-contact). These components are computed using rather simple heuristics connecting the discrete states of the contact sequence by smooth trajectories.

The sequence of footholds (as subset of the contact sequence) is further used to compute the SAs for each DS and SS phase of the walking pattern. Based on geometric considerations, a ZMP trajectory is planned such that the ZMP lies inside the current SA at all times (dynamic balance). The ZMP motion is fed – together with the upper body and EE trajectories – into the RMT planner. The computation of the RMT position $w_{Rt}^r = [w_{Rt,x}^r, w_{Rt,y}^r, w_{Rt,z}^r]^T$ is separated into subroutines for generating the vertical $w_{Rt,z}^r(t)$ and horizontal $w_{Rt,x|y}^r(t)$ motion. At this point, the reduced model from Section 4.5.2 comes into play: while the simplified 2D leg kinematics are used to determine an upper and lower bound for $w_{Rt,z}^r(t)$ (kinematic feasibility), a decoupled second-order BVP for $w_{Rt,x|y}^r(t)$ is formulated based on the five-mass model and the ZMP motion (dynamic feasibility). Since the computation of the vertical RMT component requires knowledge about the horizontal RMT components and vice versa, both subroutines are embedded within an iteration. Finally, the CoM trajectory $r_{CoM}(t)$ is computed by combining the EE and RMT motion using the five-mass model. Note that special attention has to be paid to the time-dependency of masses (blended through $\xi_h(t)$, cf. Equation 4.2 and Appendix F).

4.5.5 Planning Pipeline

Following the presentation of the solution strategy in the previous section, it is left to discuss the realization of the WPG and its integration into the governing locomotion framework presented in Figure 4.1. As already explained in Section 2.2, legged robots need a carefully thought-out software design in order to satisfy the hard real-time requirements. This applies in particular to interfaces between lower and higher levels of the locomotion system. Indeed, the WPG represents such an interface as it connects the CV system (large data, low update rate) with the SIK module (small data, high update rate). Moreover, the WPG involves heavy-duty tasks on its own (e. g. contact planning), which must not jeopardize the real-time capabilities of the locomotion system. For this reason, the WPG is split into sub-routines running either on the main WPG loop (*Thread 5* in Figure 4.1, cyclic processing @1 kHz) or in dedicated asynchronous threads (*Thread 6+8* in Figure 4.1, event-triggered acyclic processing). These domains are connected through thread-safe data containers representing

- the (*Planning*) *Context* as a knowledge database holding all relevant information and
- the (*Motion*) *Plan* as an analytic description of the resulting task-space motion which will be specified in the Sections 5.2 and 6.2.

The actual contact planning and motion generation are performed within a sequential planning pipeline which makes use of the *Context* (=input) in order to create a new or update an existing *Plan* (=output). The proposed software architecture is visualized in Figure 4.11.

The *Context* represents a collection of thread-safe data containers (bullet points in Figure 4.11), which are cyclically updated by a *Context Manager* responsible of pre-processing received data (from shared memory, etc.) and performing the state estimation described in Sec-

⁵⁷The load factors γ_e represent an auxiliary input to the SIK module indicating “the fraction of the maximum contact wrench capability of an EE, which may be used by contact force control to generate the desired total CoM wrenches and stabilize the robot.”[399, p. 51] The factors are bound to $\gamma_e \in [0, 1]$ and the sum of the foot load factors is constrained by $\gamma_{RF} + \gamma_{LF} := 1$. In contrast, the hand load factors γ_{RH} and γ_{LH} are not linked to each other.

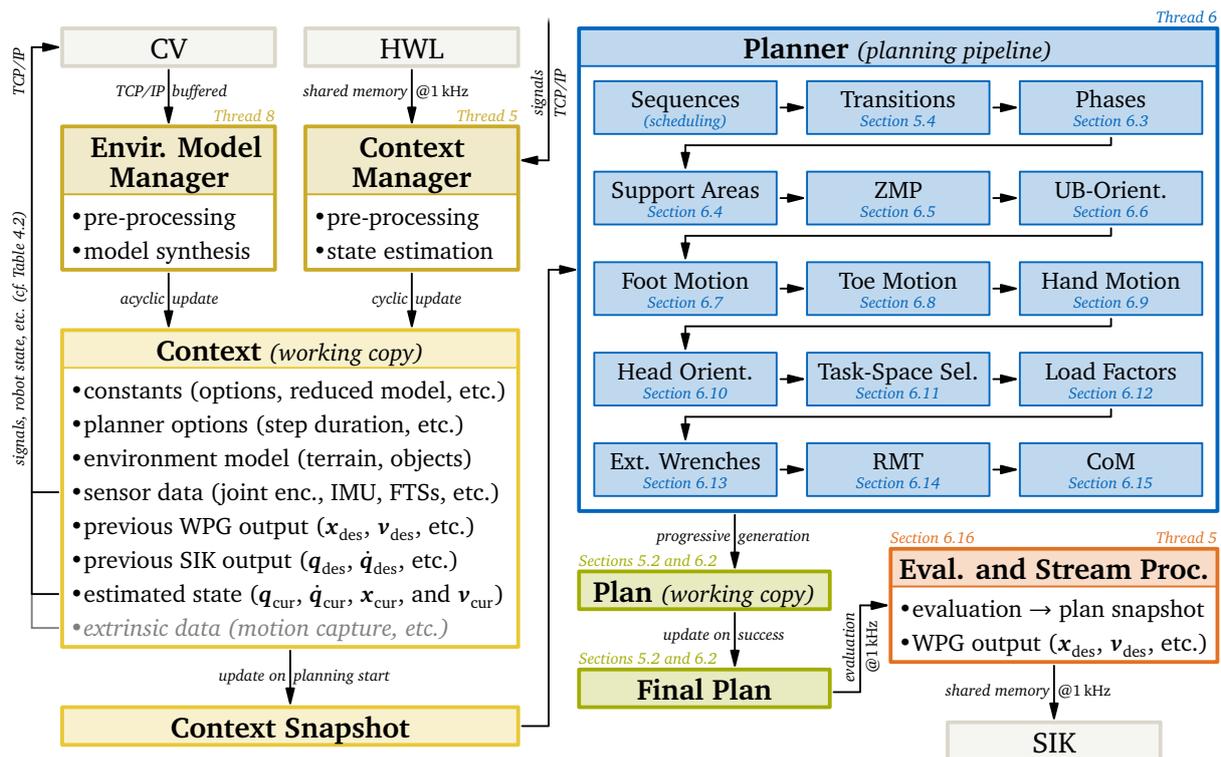


Figure 4.11: Realization of the WPG as part of the locomotion framework presented in Figure 4.1. The planning pipeline is represented by the *Planner* (blue) which consists of 15 consecutive planning stages. It uses the most recent snapshot of the *Context* (yellow) in order to generate a new *Plan* (green) which is cyclically evaluated and post-processed (orange) in order to obtain the task-space samples required by the SIK module. The *Environment Model Manager* and the *Planner* involve heavy-duty workloads, thus, they are executed within asynchronous threads. The context container for extrinsic data (gray) is used exclusively for synchronized logging and forwarding motion capture data to the CV system (for calibration/logging).

tion 4.5.3. An exception is the environment model, which is updated by a dedicated *Environment Model Manager* (Thread 8) performing the heavy-duty pre-processing of buffered environment model updates received from the CV system (see Section 5.5.2). For simulating autonomous locomotion, the *Environment Model Manager* is additionally responsible for synthesizing the environment model from a corresponding multi-body scene description (see Section 7.4).

The creation of a new *Plan* starts with taking a snapshot of the current *Context*. The snapshot is kept constant during the whole planning process such that all sub-routines of the *Planner* have access to the same input data, even if the working copy of the *Context* changes meanwhile. The *Planner* (Thread 6) encapsulates the planning pipeline consisting of 15 consecutive stages. The first stage is given by the *Sequence Planner* which defines the execution order of the remaining 14 stages. The entire pipeline has a modular design, such that individual stages can be easily added, removed, or re-ordered. The second stage, the *Transition Planner*, realizes the contact planning presented in Chapter 5. In contrast, the remaining stages (from *Phase Planner* to *CoM Planner*) are part of the motion generation task and will be described in detail in Chapter 6. Apart from managing the planning pipeline, the *Planner* additionally implements the logic for deciding if and when a new planning event has to be triggered. For the purpose of dynamic replanning, i. e., for updating a currently executed *Plan*, it has to predict the runtime of the pipeline such that its result is obtained in time. Finally, the *Planner* is also responsible of “garbage collection”, i. e., removing past/unused parts from the *Plan* to accelerate evaluation.

Each stage of the planning pipeline has full access to the *Context Snapshot* and an internal working copy of the *Plan*. The thread-safe container holding the *Final Plan* is updated once the pipeline succeeded. The *Final Plan* is cyclically evaluated to obtain the *Plan Snapshot*, i. e.,

a snapshot of the planned motion at the current sample time t_{cur} . Subsequently, the *Stream Processor* post-processes this snapshot and assembles the final output data container which is transmitted to the SIK module. The plan evaluation and *Stream Processor* are executed within the main WPG loop (*Thread 5*). Thus, the data container holding the *Final Plan* has to be accessible and consistent at all times. This is guaranteed by updating the *Final Plan* only if the planning pipeline has finished successfully.

The particular components of the data container sent from the WPG to the SIK module are specified in Table 4.5. Most components of the container are directly obtained from the *Plan Snapshot*. In contrast, the total contact wrench ${}_{\text{W}}\mathbf{W}_{\text{cont}}^{\text{CoM}}$ is a result of the post-processing performed by the *Stream Processor*, see Section 6.16. Note that the SIK module implements its own method to compute an optimal contact wrench distribution from ${}_{\text{W}}\mathbf{W}_{\text{cont}}^{\text{CoM}}$. Indeed, the planned external wrenches ${}_{\text{W}}\mathbf{W}_{h,\text{ext}}^h$ are used by the SIK module only to extract the orientation of the environmental surface getting in contact with the spherical hand. In contrast, the feet have a clear (planar) contact surface with orientation ${}_{\text{W}}\boldsymbol{\theta}_f$. Finally, the data container includes the contact state of the feet to pass information on the contact area (toe-only or full foot sole).

Table 4.5: Data sent from the WPG to the SIK module through the shared memory interface (cf. Figure 4.1). Except for the current sample time t_{cur} which is passed through from the HWL, all components have to be understood as planned/desired quantities at the particular time t_{cur} , which may be modified by the SIK module.

Symbol	Description
t_{cur}	current sample time according to HWL “master” clock (<i>Thread 1</i> in Figure 4.1)
\mathbf{x}, \mathbf{v}	task-space vector and task-space velocity vector
$\boldsymbol{\xi}$	task-space selection vector with $\boldsymbol{\xi} = [\xi_{\text{RH}}, \xi_{\text{LH}}]^T$
$\boldsymbol{\gamma}$	load vector with $\boldsymbol{\gamma} = [\gamma_{\text{RF}}, \gamma_{\text{LF}}, \gamma_{\text{RH}}, \gamma_{\text{LH}}]^T$
${}_{\text{W}}\mathbf{W}_{h,\text{ext}}^h$	external (multi-contact) wrench acting on the hand h at the hand h
${}_{\text{W}}\mathbf{W}_{\text{cont}}^{\text{CoM}}$	total contact wrench (feet and hands) acting on the robot at the CoM
–	contact state (open/closed, full/partial/tiptoe cf. Section 5.1) for each foot

The numerous experiments conducted so far have shown that the presented WPG architecture is very robust and fault tolerant. This is also a result of the countless distributed validity and plausibility checks and their corresponding fallback strategies. In contrast to the previous implementation of the WPG before the multi-contact revision, the interfaces between the individual components have a clear specification while still avoiding unnecessary copy operations for maintaining real-time performance. Moreover, the modular design of the planning pipeline makes future modifications and extensions significantly easier.

4.6 Excursus: Stabilization and Inverse Kinematics (SIK)

The SIK module receives the planned motion as specified in Table 4.5 and modifies it (still using the task-space description) to compensate disturbances and maintain balance. It further implements a velocity-level IK to generate the corresponding joint-space trajectories to be sent to the HWL, see Figure 4.1. The original stabilization system of LOLA has been developed mainly by BUSCHMANN [100]. In the past years, it has been completely reimplemented and extended by SYGULLA, who documented the new SIK framework in his dissertation [401]. Indeed, [401] represents an excellent complementary work to this thesis. This section gives only a brief summary of the main processing steps of the SIK module in order to explain how the data from Table 4.5 is used. Some remarks on related works relevant to the contents of this thesis have already been given in Section 2.6. For a much more detailed presentation of the state of the art with regard to stabilization, the literature review given in [401] is recommended.

Sensor Data Pre-Processing Apart from the planned motion transmitted by the WPG, another important input of the SIK module are the current sensor measurements provided by the HWL. Primary data sources are the torso IMU, the FTSS located in the feet and hands, and the four binary contact switches per foot. A core indicator for balance is the floating base inclination, i. e., the rotational error in the sagittal and frontal plane, which is computed from the IMU measurements and the planned upper body rotation (${}_{\text{W}}\boldsymbol{\vartheta}_{\text{UB}}$ and ${}_{\text{W}}\boldsymbol{\omega}_{\text{UB}}$ as part of \boldsymbol{x} and \boldsymbol{v}). In another pre-processing step, contact transitions (from opened to closed and vice versa) are detected. In particular, the planned (“ideal”) contact transitions given by the load vector $\boldsymbol{\gamma}(t)$ provided through the WPG are compared against the actual measurements of the foot FTSS and binary contact switches in order to detect early- or late-contact situations. Based on this, activation factors for blending an EE between position- and force-control are computed. In order to mitigate noise and increase robustness, the sensor data pre-processing module also applies rate limitations and low-pass filtering to various signals. This further guarantees C^1 -continuity which is required by subsequent control laws.

Reactive Trajectory Adaption Based on the discrete events detected in the previous step, this module implements feedforward strategies to counteract discrepancies between the planned and real motion. This includes an early-contact reflex which modifies the reference trajectories of the affected foot or hand in order to minimize the disturbance on the overall system. In particular, this strategy tries to reduce the velocity of the corresponding TCP as fast as possible. While this applies to all EEs in the same way, further adaptations are applied in particular to the current swing foot. In order to avoid repeating early- or late-contacts due to an unexpected ground height change (imagine stepping onto a carpet not detected by the CV system), the vertical goal position of the swing foot is modified to match the height of the current stance foot. Note that the deformation of the foot sole is compensated independently by introducing a corresponding offset in vertical direction based on an experimentally determined stiffness.

Balance Control In order to stabilize the floating-base dynamics, this module modifies the task-space trajectories \boldsymbol{x} and \boldsymbol{v} and the total contact wrench ${}_{\text{W}}\boldsymbol{W}_{\text{cont}}^{\text{CoM}}$ acting on the robot. For this purpose, the robot is approximated by a flywheel model (non-linear inverted pendulum with a point mass located in the CoM and an additional mass moment of inertia linked to the torso). Based on this model, the floating-base inclination is stabilized in the frontal and sagittal plane by a PD-type controller augmented by a non-linear gravity compensation term. Unfortunately, the performance of this controller is strongly affected by structural resonances, which is the reason for the careful torso design presented in Section 3.5. In addition, this module implements control strategies based on vertical CoM motion. Since a vertical acceleration of the CoM directly causes corresponding contact forces in the feet, it can be used indirectly as a force controller. For this purpose, the robot is approximated by a single point mass located in the CoM (without mass moment of inertia). This model is the basis of a strategy for mitigating late-contact situations where the CoM and the foot expected to make contact are accelerated simultaneously effectively reducing the time to establish the contact. Apart from the vertical CoM acceleration, a further controller tracks the desired vertical CoM position. This is necessary to avoid an undesired CoM drift caused by foot-height modifications, e. g. due to the previously mentioned early-contact reflex. The tracking controller is realized through a virtual spring-damper pair “pulling” the actual CoM to its reference height.

Optimal Contact Wrench Distribution After the total contact wrench ${}_{\text{W}}\boldsymbol{W}_{\text{cont}}^{\text{CoM}}$ acting on the robot has been modified by the balance controller, it has to be distributed to the individual EEs in contact. Originally, BUSCHMANN used a heuristics based approach to compute the contributions of the right and left foot (hand contacts were not considered at that time). Within the scope of the multi-contact revision of LOLA, SYGULLA developed an optimization based method

which respects constraints such as unilateral interaction (pushing only) and linearized friction cones with individual friction coefficients ($\mu_f = 0.8$ and $\mu_h = 0.05$) [401, p. 80ff]. Since the hands have a spherical shape, point contact is assumed such that only forces can be transmitted. As mentioned earlier, the contact normal is extracted from ${}_{\text{W}}\mathbf{W}_{h,\text{ext}}^h$ provided by the WPG (direction of force component which is perpendicular to the environment model surface). The interaction of the foot with the ground is modeled as surface contact such that also torques can be transmitted. The torque limits depend on the foot geometry (components lying in contact plane) and the friction coefficient (component around contact normal) where the contact area is obtained from the contact state sent from the WPG. The contact normal of the foot is defined by the foot orientation ${}_{\text{W}}\boldsymbol{\theta}_f$ as part of \mathbf{x} . The contact wrench distribution problem is formulated as a convex QP which involves costs to minimize the contact forces/torques at the EEs and the residual of the desired total contact wrench provided by the balance controller.

Contact Force Control For each EE, the SIK module implements an individual contact force controller which tries to establish the previously computed wrench distribution. For the feet, only the components ${}_f F_{f,\text{cont},z}$, ${}_f T_{f,\text{cont},x}$, and ${}_f T_{f,\text{cont},y}$ are force-controlled. The remaining directions of the local TCP frame are position-controlled. The controller uses an explicit contact model featuring an infinite number of linear springs between the four contact pads of the foot and the ground. The actual contact surface is estimated based on the input signal of the binary contact switches which is filtered with a second-order low-pass for smoothness. The binary switches may be replaced in future with a tactile skin such as the one developed by SYGULLA et al. in [399]. The controller computes primarily the foot velocity while the corresponding modification of the foot position is obtained by integration. In order to deal with changing ground stiffness (e. g. for stepping on a soft carpet), SYGULLA also implemented an adaptive control law based on a *Model-Reference Adaptive Control (MRAC)* [322] scheme which includes an online estimation of the mechanical properties of the foot-ground contact. In comparison to the feet, the contact force control for the hands is much simpler: due to the assumption of point contact (modeled as 3D linear spring), only the position needs to be controlled. Apart from this, the same hybrid force/motion control approach as for the feet is applied.

Inverse Kinematics (IK) As a final step, the SIK module computes the joint-space trajectories \mathbf{q} and $\dot{\mathbf{q}}$ which correspond to the desired (modified) task-space trajectories \mathbf{x} and \mathbf{v} . For this purpose, a velocity-level IK scheme based on ASC [283] is applied. In particular, we compute

$$\dot{\mathbf{q}} := \mathbf{J}^\# (\mathbf{v} + K \Delta \mathbf{x}) - \alpha_N (\mathbf{1} - \mathbf{J}^\# \mathbf{J}) \left(\frac{\partial H}{\partial \mathbf{q}} \right)^\text{T}, \quad \mathbf{J} := \left(\frac{\partial \mathbf{v}}{\partial \dot{\mathbf{q}}} \right), \quad \mathbf{J}^\# := \mathbf{J}^\text{T} (\mathbf{J} \mathbf{J}^\text{T})^{-1} \quad (4.10)$$

where \mathbf{J} and $\mathbf{J}^\#$ denote the task-space Jacobian and its pseudoinverse and $K \in \mathbb{R}$ and $\alpha_N \in \mathbb{R}$ are scalar weights. The joint-space vector \mathbf{q} is obtained by numerical integration of $\dot{\mathbf{q}}$. Special attention has to be paid to the computation of the task-space error $\Delta \mathbf{x} = f(\mathbf{x}, \mathbf{x}_{\text{cur}})$ since it involves orientations described by rotation vectors, see [401, p. 66ff] for details. The current (actual) task-space vector \mathbf{x}_{cur} is obtained by evaluating the FK using the joint-space vector \mathbf{q} from the previous cycle (instead of using measured joint angles). The IK scheme presented in Equation 4.10 is the solution to the optimization problem

$$\min \left(\frac{1}{2} \dot{\mathbf{q}}^\text{T} \dot{\mathbf{q}} + \alpha_N \left(\frac{\partial H}{\partial \mathbf{q}} \right)^\text{T} \dot{\mathbf{q}} \right) \quad \text{with the constraint} \quad \mathbf{v} = \mathbf{J} \dot{\mathbf{q}}, \quad (4.11)$$

which allows to exploit the kinematic redundancy by specifying secondary objectives through a corresponding gradient of the cost function H . For LOLA, these secondary objectives include a comfort pose, joint limit- and self-collision avoidance (the latter uses the new implementation of the SSV library – see Appendix C), and minimization of the total vertical angular momentum to reduce slippage by rotation around the vertical axis, see [104, 371, 372] for details.

As already explained in Section 4.3, the multi-contact locomotion system of LOLA uses a variable task-space definition. In particular, the position of the right and left hand may be assigned either to the task-space or the null-space. In order to allow a smooth transition between these configurations, the task-space selection vector $\xi = [\xi_{RH}, \xi_{LH}]^T$ has been introduced. With regard to the IK as part of the SIK module, SYGULLA implemented a solution similar to the one proposed by AN and LEE [59]. In particular, four individual IK algorithms are triggered which differ only in the used task-space definition (cf. Table 4.1). The results are then combined through smooth bilinear interpolation using ξ as weights:

$$\begin{aligned} \dot{\mathbf{q}} &= (1 - \xi_{RH})(1 - \xi_{LH}) \dot{\mathbf{q}}_{RH,LH} && \text{(both hands in null-space)} \\ &+ \xi_{RH}(1 - \xi_{LH}) \dot{\mathbf{q}}_{RH,LH} && \text{(right hand in task-space, left hand in null-space)} \\ &+ (1 - \xi_{RH})\xi_{LH} \dot{\mathbf{q}}_{RH,LH} && \text{(right hand in null-space, left hand in task-space)} \\ &+ \xi_{RH}\xi_{LH} \dot{\mathbf{q}}_{RH,LH} && \text{(both hands in task-space)} \end{aligned} \quad (4.12)$$

In order to obtain \mathcal{C}^2 -continuous joint-space trajectories \mathbf{q} , the task-space selection factors have to be at least \mathcal{C}^1 -continuous. Indeed, the WPG generates \mathcal{C}^2 -continuous task-space selection factors since they are also used to control the mass-distribution in the five-mass model (cf. Equation 4.2) and therefore, have direct influence on the continuity of the CoM trajectory.

4.7 Excursus: Hardware Layer (HWL)

The HWL receives the current sample of the joint-space trajectories \mathbf{q} and $\dot{\mathbf{q}}$ (without the “passive” 6D torso pose) from the SIK module and forwards it – after some additional processing – to the 26 individual servo drives. In return, it receives the current sensor measurements (joint encoders, IMU, FTSs, and contact switches) and provides it to the higher levels of the locomotion system. In other words, the HWL represents a low-level abstraction layer of the robot’s hardware. The main focus lies on highest possible real-time performance, i. e., low communication latency (affects control bandwidth) and jitter (standard deviation of control loop cycle time). A brief summary on the state of the art with regard to low-level locomotion control for legged robots has already been given in Section 2.2. For LOLA, an *EtherCAT* [77] real-time bus represents the backbone of the low-level communication system. It was introduced by WITTMANN and SYGULLA as replacement of the previous *Sercos-III* [377] bus. Details on this upgrade – which includes a complete redesign and reimplementations of the HWL of LOLA – have been published in [10]. Since a comprehensive explanation of the present state of the HWL is already given in [401, p. 15ff], the following paragraphs give only a brief summary of the three main subroutines (*Thread 1*, *2*, and *3* in Figure 4.1). Note that SYGULLA has published parts of the HWL source-code as part⁵⁸ of *Broccoli*.

Timing (*Thread 1*) The only purpose of this module is to realize a high-precision “master” clock for the locomotion system. It runs at 4 kHz in a background thread with highest *QNX* real-time priority and defines the timing of all cyclic tasks in the HWL, SIK, and WPG modules (*Threads 1-5* in Figure 4.1). The timing thread is optimized for low latency and jitter to allow a precise synchronization between the different modules.

***EtherCAT* / IO (*Thread 2*)** This module implements an interface to a commercial *EtherCAT* master stack by *Acontis Technologies* [48] through a custom middleware developed by SYGULLA et al. in [10]. The communication between bus nodes is based on *Process Data Objects (PDOs)*

⁵⁸See the module *hw1* of *Broccoli*.

(for synchronous data) and *Service Data Objects (SDOs)* (for asynchronous data) as defined by *CANopen* [107]. For each bus device (servo drives, *CAN-EtherCAT* gateway, IMU, and FTSSs), an abstraction class with its own internal FSM is implemented. The IO operations of the *EtherCAT* bus are triggered by the high-precision timing thread with a frequency of 4 kHz.

Main Loop (Thread 3) The main execution loop of the HWL primarily manages the communication with the higher levels of the locomotion system. In particular, it provides a synchronized shared memory interface to the SIK and WPG modules, a socket to communicate over the publish/subscribe system with the high-level signal broker (*Process 4* in Figure 4.1), and handles the access to the file system, e. g. for logging or loading calibration data (cf. Appendix H.3). Moreover, this module performs pre- and post-processing such as

- target data extrapolation based on target data gradients to upsample the joint-space trajectories provided by the SIK module at 1 kHz to the 4 kHz required by the *EtherCAT* bus,
- conversion of joint- to motor-angles and vice versa by a dedicated joint model (incorporates HD gears and the special kinematics for the knee and ankle mechanisms),
- joint feedforward control on velocity level to reduce position tracking errors (gains obtained through reinforcement learning, see WITTMANN [443, p. 33ff] for details), and
- extensive safety checks (plausibility of sensor data, monitoring tracking error, detecting violations of real-time constraints indicated by working counter mismatches, etc.).

Finally, this module implements a FSM which is responsible of switching between different modes of operation and handling all types of errors.

4.8 Results and Discussion

Through numerous successful experiments, the presented multi-contact locomotion framework of LOLA has proven to be

robust: biped walking with additional hand support has been shown to be stable even under large disturbances in the form of an uncertain ground (uneven terrain / rolling board) and unforeseen external interaction (pushing by human), see [20 📺@t=10s],

versatile: the numerous scenarios shown in [18 📺@t=6m45s] (support against walls / tables / corridors, bypassing / stepping over obstacles, climbing ramps / stairs with full or partial foot contact, etc.) and also the combined scenario shown in [20 📺@t=1m41s] (stepping up and down platform with hand support and partial / tiptoe contact) have been achieved using the same parametrization of the locomotion system, i. e., no scenario dependent tuning is required – these capabilities work “out of the box”, and

efficient: planning and control is performed onboard and in real-time even for complex autonomous locomotion without any prior knowledge of the environment, see [17 📺].

While the qualitative performance has been demonstrated by the aforementioned videos, this section focuses on the quantitative performance of the governing framework (software design). Note that conclusions on the embedded contact planner and motion generator are drawn separately at the end of the Chapters 5 and 6, respectively. Moreover, we focus in the following on the WPG, SIK, and HWL modules as the three main processes of the locomotion system which are subject to hard real-time constraints, cf. Figure 4.1.

Real-Time Performance The presented hierarchical architecture which separates tasks according to their real-time priority makes the locomotion system of LOLA very robust, in particular with regard to maintaining hard real-time constraints. This is shown by the fact that even under

heavy load (e. g. while planning autonomous locomotion in a complex environment), critical tasks still remain within their individual time budget without “skipping a beat”. For the WPG and SIK module, this means that each cycle of their respective “main loops” (*Threads 4 and 5* in Figure 4.1) running at 1 kHz (synchronized through shared memory) does not exceed a maximum runtime of 1 ms. Moreover, the HWL module has to maintain a cycle frequency which should be as close as possible to 4 kHz. Table 4.6 shows the experimentally evaluated real-time performance of the WPG, SIK, and HWL module in different situations.

Table 4.6: Real-time performance of the WPG, SIK, and HWL module in different situations (execution time of a single cycle in the respective “main loop”, see Figure 4.1). The corresponding mean t_{mean} , standard deviation t_{σ} , and maximum t_{max} are evaluated over a time horizon of 10 s (WPG/SIK: 10,000 samples; HWL: 40,000 samples).

Situation	WPG (<i>Thread 5</i>) / μs			SIK (<i>Thread 4</i>) / μs			HWL (<i>Thread 3</i>) / μs		
	t_{mean}	t_{σ}	t_{max}	t_{mean}	t_{σ}	t_{max}	t_{mean}	t_{σ}	t_{max}
idle	112.4	11.5	296.6	378.3	17.3	474.3	249.2	14.2	296.6
motion	121.0	15.5	514.5	396.5	19.5	593.4	249.2	14.0	327.3
planning	102.3	14.3	251.8	364.4	15.5	447.5	249.3	17.5	303.5

The data shown in Table 4.6 is based on a high-precision time measurement routine⁵⁹ which is used for all runtime evaluations presented within this thesis. During “idle”, the robot remains in the static idle pose (see Section 4.5.2) which causes the lowest possible computational load for the WPG module. During “motion”, the robot executes a rather long fixed (no contact planning) walking sequence similar to the one shown in Figure 4.8 such that it is in motion throughout the entire measurement period. For this situation, we observe a moderate increase of the overall system load and a significant increase of t_{max} for the WPG which is due to the evaluation of the more complex motion plan. However, the real-time constraints are still satisfied. For “planning”, the robot is physically at rest (same as “idle”) but computes a complex motion plan. In particular, an autonomous locomotion scenario similar to [18 @ $t=9\text{m}19\text{s}$] is chosen. Since the computationally expensive contact planning and motion generation are offloaded to an asynchronous thread, the cycle times of the WPG, SIK, and HWL remain similar to the “idle” case. A more detailed review of the runtimes for contact planning and motion generation is postponed to the end of the following chapters. A runtime analysis for the different components of SIK has already been given by SYGULLA in [401, p. 139ff]. Note that in contrast to [401], the measurements shown in Table 4.6 cover the entire SIK “application”, hence, it also includes the management of interfaces and the FSM.

Memory Consumption Besides computational effort, another metric which is important to an embedded system is the memory consumption. The maximum per-thread *stack* (used for static allocation) consumption is independent of the situation and amounts to 96 KiB for the HWL, 76 KiB for the SIK, and 336 KiB for the WPG module (*Threads 3, 4, and 5* in Figure 4.1). The total per-process *heap* (used for dynamic allocation) consumption amounts in “idle” to 33 MiB for the HWL, 127 MiB for the SIK, and 519 MiB for the WPG module (*Process 1, 2, and 3* in Figure 4.1). While the heap consumption remains constant for the SIK and HWL module, it increases for the WPG module within “motion” to 531 MiB and within “planning” to 2.3 GiB. The additional 1.8 GiB for the WPG are mainly due to the contact planner, which makes extensive use of pre-computed (scenario independent) buffers for acceleration (see Section 5.5.2) and also implements a custom pool allocator for the D -ary heap used by the A^* search (see Section 5.5.3). The actual complexity of the scene, i. e., the count of terrain patches and objects,

⁵⁹On QNX, time is measured with `ClockCycles()` on the basis of CPU clock cycles. The resolution is 1 ns and the duration of a single time measurement is identified as 152 ns on the used hardware (averaged over 10^6 samples). On Linux, time is measured using `clock_gettime()` using the system-wide realtime clock. See the method `PlatformHelper::currentTime()` in the module `core` of `Broccoli` for details.

has only a minor contribution (typically less than 50 MiB for the scenarios considered within this thesis). Since the control PC is equipped with 32 GiB RAM and runs a 64bit OS, the present memory consumption is far away from the system's limits. Moreover, dynamic memory allocation by the WPG is performed at startup within the scope of a dedicated initialization routine or, alternatively, after triggering the corresponding planning stage for the very first time. Indeed, dynamically allocated memory is re-used whenever possible in order to avoid computational expensive re-allocation.

Bottlenecks and Suggestions Although the HWL module communicates with the hardware at a frequency of 4 kHz, Table 4.6 clearly shows that the WPG and SIK module cannot keep up with this rate. Instead, they are limited to their present 1 kHz cycle, however, providing some reserves for future extensions. Since the planned trajectories are mostly C^2 -continuous with moderate rates of change, raising the update rate of the WPG above 1 kHz is not expected to improve the overall locomotion performance. In contrast, the SIK module may benefit from a 2 kHz cycle as this would lead to lower latencies in the involved reflex strategies. Currently, almost 70 % of the SIK workload is caused by the FK and IK evaluation [401, p. 140]. Runtime optimizations of these components might allow a 2 kHz cycle in future.

A serious issue of the presented locomotion framework is its logging capability. As with most research prototypes, one is typically interested in logging as much data as possible to allow extensive analysis or debugging in the aftermath. Unfortunately, logging all available data of LOLA (environment model, planned motion, internal states, sensor / actuator signals, etc.) typically leads to a violation of the real-time constraints. Although the main workload for logging (buffering data containers, compression, and file IO) is offloaded to asynchronous threads running with the lowest QNX real-time priority (*Threads 9a, 9b, and 9c* in Figure 4.1), the raw amount of data seems to exceed the available bandwidth of the platform: only six CPU cores are available on the control PC (the RTOS requires SMT to be disabled) which share cache and main memory. The current workaround to this problem is to limit the bandwidth by disabling log data which is not in the focus of the particular experiment. For future developments, a better solution might be to offload the task of logging to a separate board. The communication with the real-time locomotion system may then be realized by a high-throughput and low-latency technology such as *Remote Direct Memory Access (RDMA)* (allows transfers between the main memory of separated PCs without CPU interaction). A further step would be to distribute the HWL, SIK, and WPG modules onto separated (potentially less powerful) boards where the currently used shared memory interface may also be realized on top of RDMA. This could further increase robustness with respect to real-time performance.

Software – Part B: Contact Planning

Parts of this chapter have already been published in [1, 7].

This chapter presents a novel contact planner which has been developed by the author of this thesis within the context of the multi-contact revision of LOLA. Indeed, autonomous biped locomotion (without hand contacts) was already possible for LOLA prior to this revision thanks to the excellent work of BUSCHMANN [100, 104], WAHRMANN [430], HILDEBRANDT [201], and many others. The contact planner described within this thesis represents an entirely new system which extends the autonomous locomotion capabilities of the robot not only for multi-contact scenarios, but also for biped walking without hand contacts such as climbing stairs and traversing ramps. Although the contact planner has been redesigned and reimplemented from scratch, some ideas and concepts have been adopted from the previous system. Moreover, the involved search strategies have many similarities to those used for action planning in manipulation. Indeed, the experience gained from pre-studies on cooperative manipulation (which independently led to [7]) also contributed to the planner presented in the following. The chapter starts with Section 5.1 which introduces the reader to some important concepts and models used within the remainder. Subsequently, in Section 5.2 it is explained how the WPG describes abstract motion in the form of a hierarchical tree structure. The explanation will be restricted to the higher levels of the tree which are in particular relevant for contact planning. The specification of the lower levels is postponed to the following chapter describing the motion generation pipeline. In Section 5.3, the so-called *Quasi-Planar Walking Transition (QPWT)* is introduced, which allows to describe discrete contact sequences in a convenient and human readable form. This is followed in Section 5.4 by an overview of the transition planner representing the second stage of the planning pipeline shown in Figure 4.11. In Section 5.5, the actual multi-level search for autonomous locomotion as part of the transition planner is explained. Finally, the chapter is concluded in Section 5.6.

The main concept of the multi-level search used for planning autonomous multi-contact locomotion in real-time has already been presented in [1]. Since the new contact planner was the very last piece in the puzzle for making LOLA ready for multi-contact locomotion, its development finished together with the end of the accompanying DFG project. As a consequence, the details presented within this chapter have not been published before. An exception is the video [18 ●@t=1m54s], which briefly visualizes the main workflow and shows results for selected scenarios demonstrating the new autonomous locomotion capabilities of LOLA. As already mentioned above, the presented contact planner was inspired by the results of multiple pre-studies partially performed by student assistants, see Appendix J for details.

5.1 Preliminaries

The contact planner is the component of the WPG which is responsible for creating a feasible sequence of discrete contact configurations guiding the robot from its current location to a user-specified goal. Within this context, feasibility means that collisions with obstacles are avoided, contacts remain stable, and the resulting motion complies with the robot's kinematic

and dynamic capabilities. Since a primary objective for the new contact planner is real-time performance, feasibility has to be estimated on the basis of rather coarse approximations of the robot and its environment. In addition, generous safety margins have to be applied which, unfortunately, also prevent the contact planner from exploiting the full potential of the hardware.

Note that the fundamentals and state of the art with regard to contact planning for autonomous legged locomotion have already been summarized in Section 2.3. This section focuses on core concepts and models used by the particular contact planner proposed within this thesis.

Contact Model: Full, Partial, and Tiptoe In accordance with the reduced model of the robot presented in Section 4.5.2, the interactions of the feet with the ground are modeled as surface contacts while support with the spherical hands is approximated by point contacts. For the feet, the WPG additionally distinguishes between *full*, *partial*, and *tiptoe* contact, see Figure 5.1. For *full* contact, the interfacing areas of the toe and heel segment are combined to a single enclosing rectangle ($l_{12} \times l_{13}$ in Figure 4.7) representing the entire foot sole. It implies $q_{zfr||} = 0$ and represents the default contact type for the feet. With *partial* contact, the interface is restricted to the toe segment only ($l_{10} \times l_{13}$ in Figure 4.7). Same as before, we assume a flat foot, i. e., $q_{zfr||}$ to be zero. This variant is used for example to climb stairs with short treads where a full contact is not possible. Finally, there is the *tiptoe* contact, which uses also only the toe segment but explicitly requires a bent foot, i. e., $q_{zfr||} \neq 0$. A typical scenario is stepping down a platform (or stairs), where the heel is lifted in order to increase the kinematic capabilities of the robot (less abrupt lowering of the CoM) and additionally to avoid collision of the heel with the platform.

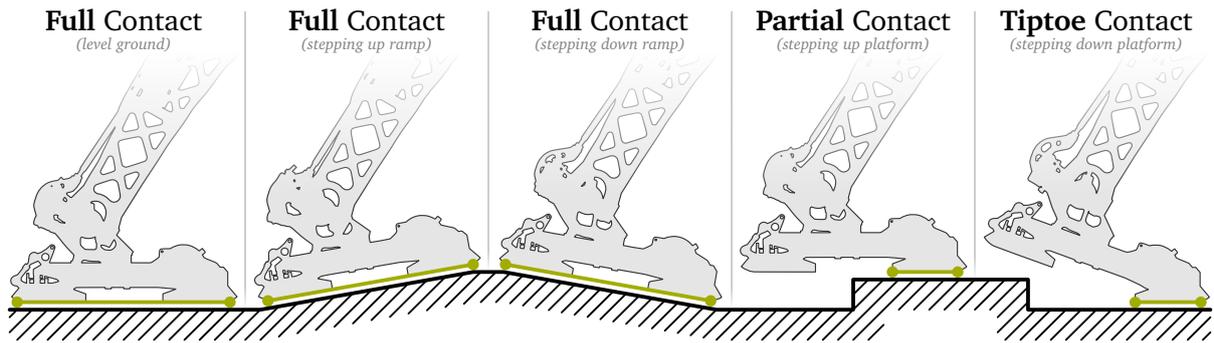


Figure 5.1: Types of contact for the foot-ground interface as used by the WPG. From left to right: *full* contact shown for level ground and stepping up/down a ramp, *partial* contact shown for stepping up a platform, and *tiptoe* contact shown for stepping down a platform. The corresponding contact area is highlighted in green. Note that for full contact we use the convex hull of the toe and heel segment area (cf. Figure 4.7) which includes the “foot arch” in between.

Mean Foot TCP Frame The contact planner presented within this chapter is based on a hierarchical search on different levels of detail. For finding a rough guiding path (coarsest level), there won’t be an explicit consideration of individual feet. Instead, their mean pose will be used to roughly specify the location of the robot. For this reason, we formally define the mean foot TCP frame “MF” extending the CoSy definitions from Section 4.2:

$$\mathbf{r}_{MF} := 0.5(\mathbf{r}_{RF} + \mathbf{r}_{LF}) \quad \text{and} \quad \mathbf{s}_{MF} := \begin{cases} \text{SLERP}(\mathbf{s}_{RF}, +\mathbf{s}_{LF}, 0.5) & \text{if } \mathbf{s}_{RF} \odot \mathbf{s}_{LF} \geq 0, \\ \text{SLERP}(\mathbf{s}_{RF}, -\mathbf{s}_{LF}, 0.5) & \text{else} \end{cases} \quad (5.1)$$

where we simply use the arithmetic mean for the position \mathbf{r}_{MF} and the great circle arc mean obtained through a SLERP operation for the orientation \mathbf{s}_{MF} . The case differentiation in Equation 5.1 guarantees minimum rotation, see Appendix B.3.3 for details. Note that we already used a simplified version of the MF frame (position only) for Step 5 of the state estimation described in Section 4.5.3.

Since the contact planner discretizes the robot’s location in the search space through a two-dimensional grid aligned with the VW frame to match the terrain discretization, footholds are predominantly assumed to be horizontal (except for the post-processing step where footholds are adapted to the terrain inclination). Thus, instead of evaluating s_{MF} through SLERP, the average of the foot rotation angles around the vertical axis can be used. Again, special care has to be taken to interpolate the shortest rotation (“wrapping” angles within $[0, 2\pi[$).

Task-Space Collision Model For efficient evaluation of the (minimum) distance between two bodies, the locomotion framework of LOLA uses SSV approximations, see Appendix C. Distance queries are mainly used for avoiding collisions where we distinguish between self-collisions and collisions of the robot with its environment. The former is realized through a *joint-space SSV model* of the robot which defines an individual SSV segment for each physical segment of the robot. Instead of testing each segment against all others, only selected pairs⁶⁰ are evaluated which drastically reduces overall runtime. Self-collisions are then avoided by formulating a corresponding cost gradient⁶¹ within the velocity-level IK algorithm presented in Section 4.6, see also [372, p. 81ff] for details. Since the WPG describes motion in task-space, it does not have (explicit) access to joint-space data. Hence, self-collision avoidance is restricted to the SIK module. However, it is still possible for the WPG to avoid collisions of the robot with environmental objects. For this purpose, the contact planner defines a *task-space SSV model* of the robot, see Figure 5.2.

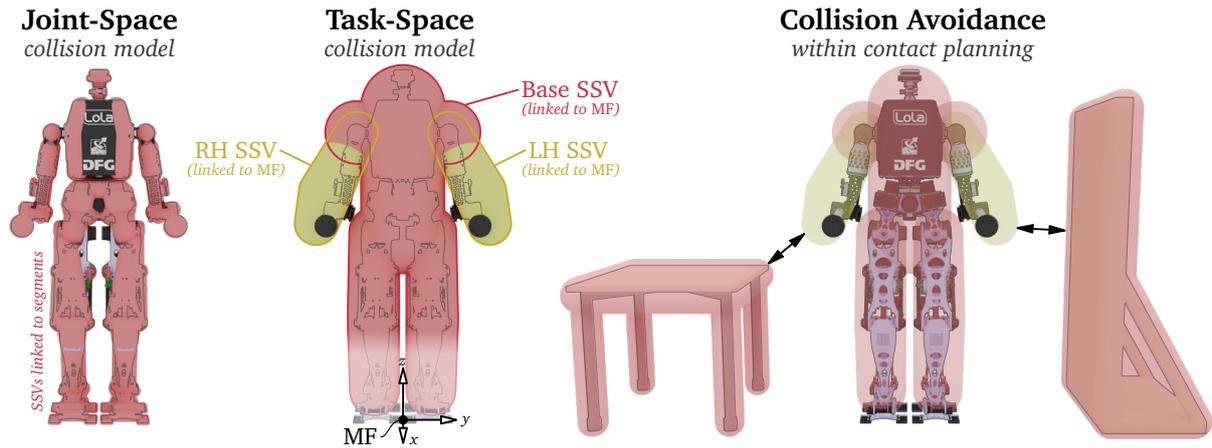


Figure 5.2: Collision avoidance within the locomotion framework of LOLA. From left to right: joint-space SSV model used by the SIK module for self-collision avoidance; task-space SSV model used by the WPG module for avoiding collisions with the environment (split into components for the base (red) and each arm (yellow) – all linked to MF frame); illustration of collision avoidance within contact planning by evaluating the distance between the (task-space) SSV model of the robot and environmental objects. See the video [18 @t=3m13s] for an animated 3D visualization.

The task-space SSV model is much less detailed and consists of only three individual SSV segments: one for the legs and torso (“base”) and one for each arm. While the representation of the legs through the “base” SSV is very coarse, it is sufficient to prevent collisions of the

⁶⁰Currently, only the segments `efr|l` are tested against `torso`, `ba`, `hrr|l`, and `hfr|l`. As a further optimization, the segments `torso` and `ba` each define an SSV segment for their left and right side which are only tested against the corresponding elbow on the same side. Note that collisions of the right and left leg are not possible since the WPG plans the corresponding foot TCP frame motions accordingly.

⁶¹Although the (scalar) minimum distance between two SSV segments is continuous, the 3D vector connecting the closest points may jump (e. g. in case two pairs of SSV elements have the exact same distance). In order to avoid discontinuities in the joint-space velocities, the cost gradient originally formulated by SCHWIENBACHER (which uses the connection vector in the form of the translational Jacobians of the closest points, cf. [372, p. 81ff]) has been extended by a subsequent first-order low pass filter with 6 Hz cutoff frequency. This modification has been made by the author of this thesis during the integration of the new SSV library (see Appendix C) into the SIK module.

(yet unknown) knee and hip motion with close-by obstacles. Note that collisions of the feet with environmental objects will be handled separately based on the terrain (see Section 5.5.4 for details). By detaching the arms, the contact planner can choose between four possible configurations depending on the current multi-contact situation (without arm SSVs / with right or left arm SSV / with both arm SSVs). The arm SSVs are only active if the corresponding hand is *not* meant to make contact such that collisions due to the (unknown) null-space motion of the arm are avoided. All three SSV segments are defined relative to the MF frame such that they move with the robot through the discretized search space.

Multi-Contact Target Volume The WPG uses SSV approximations not only for collision avoidance, but also for efficient proximity tests. In particular, the contact planner defines a so-called *multi-contact target volume*, which represents a rough approximation of the set of “in-motion” reachable hand TCP positions. The target volume is given by two line-SSVs for each hand as shown in Figure 5.3 which are derived from the workspace computed in Section 3.3. In order to check if a certain environmental object is close enough for a potential hand contact, the object’s SSV model is tested for intersection with the multi-contact target volume of the corresponding hand. Note that this represents only a very rough (but conservative) check for *proximity* since the object’s volume model delivered by the CV system might be very coarse. Indeed, testing for actual *reachability* requires a more accurate representation of the object’s geometry by its surface model which will be explained later in Section 5.5.

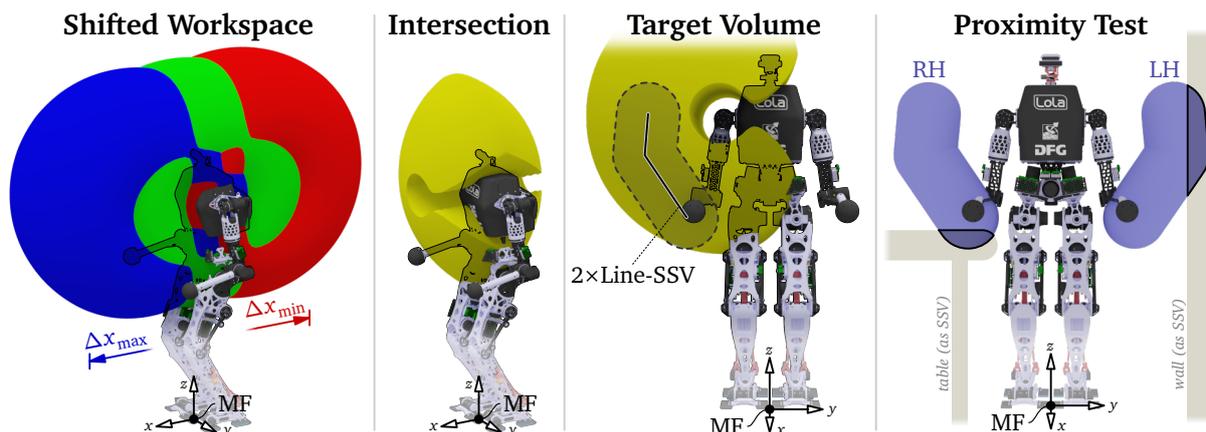


Figure 5.3: Construction and usage of the multi-contact target volume. From left to right: workspace of the right arm from Section 3.3 shifted in walking direction by $\Delta x_{\min} = -0.35$ m (red), $\Delta x = 0$ m (green), and $\Delta x_{\max} = 0.35$ m (blue); intersection volume (yellow) of shifted workspaces from Δx_{\min} to Δx_{\max} with step size 5 cm; multi-contact target volume as SSV segment fitted into intersection volume; illustration of proximity tests within contact planning to identify objects for potential hand contact. See the video [18 @t=3m25s] for an animated 3D visualization.

For computing the workspace of LOLA’s arm, we assumed in Section 3.3 that the torso segment is fixed in space. However, for most multi-contact scenarios considered within this thesis, the torso is in motion. Obviously, once a hand gets in contact, it has to be guaranteed that the contact point remains reachable for the entire duration of the contact. To take this into account, we duplicate the workspace volume to generate a series of instances which are shifted along the typical walking direction. For the shift, we choose an interval of 0.7 m (rough upper bound for foot step length) which is split symmetrically ($\Delta x_{\min} = -0.35$ m and $\Delta x_{\max} = 0.35$ m). We then compute the intersection of the entire series through successive application of the corresponding Boolean operator of *Blender*. The resulting volume is reachable by the hand while walking on a straight path (neglecting lateral torso motion) with a maximum step length of 0.7 m (as seen from the middle of the SS phase). This intersection volume is finally replaced by an inner approximation with the aforementioned line-SSVs. Same as with the task-space collision model, the multi-contact target volume is linked to the MF frame such that it moves with the robot.

5.2 Motion Plan: Higher Level Structure

The WPG stores the output of the planning pipeline (cf. Figure 4.11), i. e., the planned task-space motion, in an analytical form alias (*Motion*) *Plan*. In order to allow efficient online modifications of the plan by replacing outdated future motion with an alternative solution, e. g. as reaction to changes of the environment or to abort the currently executed motion, the data is organized as the hierarchical tree structure shown in Figure 5.4.

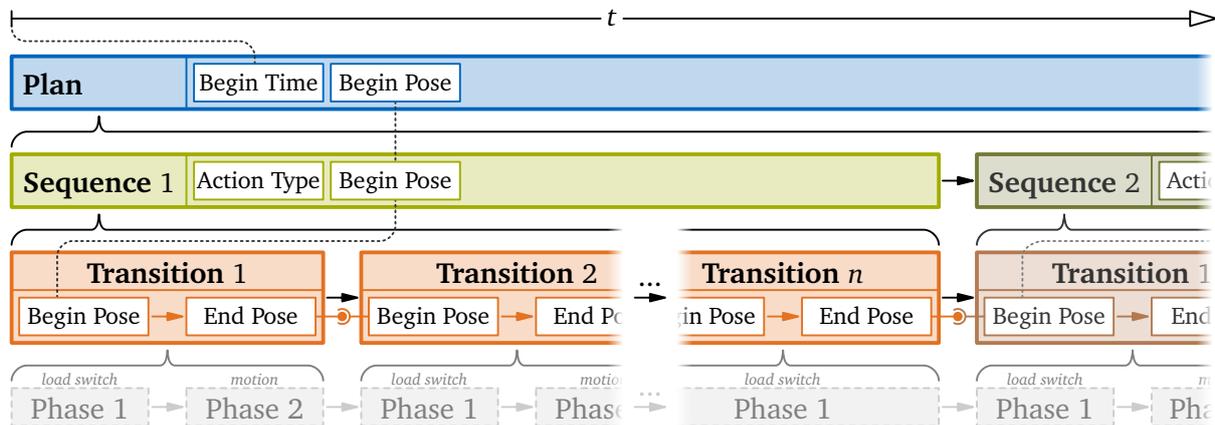


Figure 5.4: Higher level structure of the (*Motion*) *Plan*. The *Plan* (blue) represents the root element and contains a sorted list of *Sequences* (green) which in turn have consecutive *Transitions* (orange) as children. The *Robot Pose* at the beginning of the plan (at *Begin Time*) is identical to the *Begin Pose* of the very first sequence and its very first transition. Moreover, the *End Pose* of each transition defines the *Begin Pose* of its successor. This illustration focuses on the higher levels of the plan while the lower levels (e. g. *Phases*) will be described in Section 6.2.

Hierarchy The upmost level of the tree structure is given by the *Plan* itself. It specifies a *Begin Pose*, i. e., the initial *Robot Pose* to begin the planned motion with. A *Robot Pose* primarily specifies the pose and contact state of all EEs (details follow in a dedicated paragraph). The plan contains an arbitrary count (greater zero) of consecutive *Sequences* which are assigned a certain *Action Type* describing the actual type of motion (e. g. autonomous / teleoperated walking, balancing, etc.). Each sequence is split up into an arbitrary count (greater zero) of consecutive *Transitions* which each describe the motion between their respective *Begin Pose* and *End Pose* (a single footstep in biped walking). The begin and end pose of two consecutive transitions are required to be equivalent such that switching from the end of a transition to the beginning of its successor does not cause any discontinuities in the executed motion. The begin pose of the plan, the very first sequence, and the very first transition are equivalent (dotted line in Figure 5.4). Finally, each transition contains either one or two *Phases*. The description of phases is postponed to Section 6.2 together with the introduction of other lower level components of the motion plan.

If an entirely new sequence is to be planned, its begin pose is copied from the end pose of the very last transition in the previous sequence, or, if there is none, from the begin pose of the plan. Independent of the action type, each sequence ends with an end pose where the robot is in an idle configuration as specified in Section 4.5.2. This guarantees that each plan terminates with a safe stop. In case an already planned motion has to be modified, e. g. to react to an updated environment, the presented tree structure allows to efficiently “cut” a sequence to remove outdated future transitions and replace them with new ones. Cutting sequences can also be used to remove already executed transitions from the front (in Section 4.5.5 referred to as “garbage collection”) effectively reducing the total size of the plan, hence, accelerating its evaluation. This requires updating the begin pose (and begin time) of the corresponding sequence (and plan) such that it matches the new first transition.

Timing Besides the begin pose, the plan also specifies a corresponding *Begin Time* as the starting time of the entire plan measured relative to the HWL clock. The duration of the plan is simply given as the accumulated duration of its sequences which in turn inherit their duration from the corresponding transitions. The duration of a transition, i. e., the time spent for the motion between the respective begin and end pose $t_{\text{tra,dur}} = t_{\text{tra,end}} - t_{\text{tra,beg}}$, is either defined by a global parameter (alias step duration; typically 0.8 s) or is set individually. Customized transition durations are primarily used to (locally) slow down motion, e. g. for stepping over obstacles, stepping up or down platforms/stairs, or in multi-contact situations.

Action Types The proposed WPG supports a variety of actions which gives a clue on the current motion capabilities of LOLA and simultaneously demonstrates the generality and versatility of the presented planning framework. The most important action types are

idle: static standing in the idle configuration (cf. Section 4.5.2) as the default action typically used for early tests of new low-level control settings or methods (cf. [21] @ $t=20s$),

fixed sequence walking (straight): biped walking (no multi-contact) on a straight path configured by user-defined parameters such as the total distance, the minimum / maximum step lengths, and the step length gradient (cf. [21] @ $t=51s$),

fixed sequence walking (input file): biped walking (optionally with multi-contact) by executing a user-defined contact sequence described as list of QPWTs (see Section 5.3) read from an input file in TOML [351] format (cf. [20] @ $t=1m41s$),

teleoperated walking: biped walking (no multi-contact) with sagittal step length l_x , lateral step length l_y , and vertical step angle φ_z (cf. Table 5.2) controlled in real-time through a *Human Interface Device (HID)* such as a joystick (e. g. used for live-demonstrations),

autonomous walking: biped walking (optionally with multi-contact) with autonomously planned contact sequence towards a custom goal (cf. [17] and [18] @ $t=6m46s$), and

special motion: typically non-gaited motion such as bowing, standing on one foot, turning the head for exploring the environment, etc. (cf. [20] @ $t=47s$).

The action type is primarily used to select the particular submodule of the transition planner which is responsible for handling this type of sequence. Moreover, it allows to estimate the execution time of the entire planning pipeline. Obviously, planning *idle* causes minimum cost while autonomous locomotion is the computationally most expensive operation. The predicted runtime is used for *dynamic replanning*, i. e., for updating the currently executed sequence by replacing future transitions. In particular, it has to be guaranteed that transitions are replaced in time (before their planned execution). Finally, the action type also specifies if a sequence allows dynamic replanning (teleoperated and autonomous walking only) and if it supports *stopping* (all actions except for idle and special motions). Stopping can be seen as special form of dynamic replanning where all future transitions of the currently executed sequence are discarded and replaced by a minimum set of transitions which bring the robot to a safe stop, i. e., a static standing pose in the idle configuration from Section 4.5.2.

Robot Pose In order to describe the task-space configuration of the robot at a certain point in time, the WPG uses the so-called *Robot Pose* as data container. Robot poses are primarily used to describe the robot's state at the beginning and end of a transition. The sequence of robot poses obtained from chaining transitions as shown in Figure 5.4 actually represents what was previously called (*discrete*) *contact sequence*, i. e., the main output of the contact planner (cf. Figure 1.2 and Figure 4.10). Based on the definition of the task-space vector \mathbf{x} (cf. Table 4.1), a robot pose is defined by the components listed in Table 5.1.

Although the SF and $\overline{\text{SF}}$ TCP frames have already been introduced in Section 4.2, this is actually the first time we explicitly use the concept of a *stance* and *swing* foot. Indeed, almost all

Table 5.1: Components of a *Robot Pose* representing the task-space configuration (cf. Table 4.1) of the robot at a certain point in time. Within a robot pose, the task-space selection factors ξ_h are of Boolean type ($\xi_h \in \{0, 1\}$) while in general they are formulated as floating point numbers within the interval $[0, 1]$.

Symbol	Description
SF, $\overline{\text{SF}}$	type of the stance foot $\text{SF} \in \{\text{RF}, \text{LF}\}$ and swing foot $\overline{\text{SF}} := \{\text{RF}, \text{LF}\} \setminus \text{SF}$
${}^W r_{\text{SF}}, {}^W s_{\text{SF}}$	pose of the stance foot TCP frame described in the world FoR
${}^{\text{SF}} r_{\overline{\text{SF}}}, {}^{\text{SF}} s_{\overline{\text{SF}}}$	pose of the swing foot TCP frame described in the stance foot TCP FoR
ξ_h	task-space selection factor for each hand $h \in \{\text{RH}, \text{LH}\}$ (here: $\xi_h \in \{0, 1\}$)
${}^{\text{SF}} r_h$	position of the TCP frame for each hand described in the stance foot TCP FoR
–	contact state (open/closed) for each foot and hand
–	contact type (full/partial/tiptoe cf. Section 5.1) for each foot
${}^{\text{SF}} W_{h,\text{ext}}^h$	external wrench at each hand described in the stance foot TCP FoR
${}^{\text{SF}} s_{\text{UB}}$	orientation of the upper body frame described in the stance foot TCP FoR
${}^{\text{SF}} s_{\text{VTCP}}$	orientation of the head (VTCP frame) described in the stance foot TCP FoR
${}^W r_{\text{SF-CoM},z}$	(desired) height of the CoM relative to the stance foot TCP frame

components of a robot pose are described with respect to the stance foot TCP as FoR, while SF itself is described relative to the world frame. This greatly simplifies the procedural generation of contact sequences since new poses can be constructed by copying an already existing one and transforming only its stance foot (all other quantities are moved automatically). Note that changing the stance foot type also requires to update the FoR for components described relative to SF. For biped walking, a robot pose typically describes the state at the beginning of a DS phase (details follow in Section 6.3). Since in DS both feet are in contact, either foot can be considered as current “stance”. For this reason, the WPG distinguishes between *equality* and *equivalence* of two robot poses: while *equality* requires all components to be strictly identical, *equivalence* allows the stance foot to be swapped (if both feet are in contact). Indeed, the *Begin Pose* of a transition or sequence has to be equivalent but not necessarily equal to the *End Pose* of its predecessor in order to avoid discontinuities in the executed motion.

For planning multi-contact locomotion, we also include the task-space selection factors ξ_h which, however, have to be either zero or one, i. e., each hand must be either entirely in null- or task-space. Furthermore, a robot pose also includes the current contact state / type for each foot and hand as well as the planned external wrenches for multi-contact situations. Note that by specifying both, the task-space selection factors and contact states, it is possible to blend a hand into task-space without making contact (currently used to realize “waving” as special motion).

Transition: Additional Parameters Besides the begin and end pose, each transition stores additional parameters which roughly characterize the motion in between. An example is the already mentioned (optional) custom duration $t_{\text{tra,dur}}$. In addition, each transition specifies an (optional) custom step height h_{step} and a swing foot timing factor $\tau_{\overline{\text{SF}}}$. Both modify the swing foot trajectory which is used for traversing obstacles and stepping up and down. Details are postponed to Section 6.7 which explains how the swing foot motion is planned.

Limitations Although the description of motion through the presented tree structure supports a very broad spectrum of gaited and even non-gaited motions, it comes with certain restrictions. Most importantly, foot and hand contacts are forced to be synchronized which is due to the fact that they use the same timing specified by the partitioning of the motion plan into sequences and transitions. Since this thesis focuses on *gaited* locomotion where foot and hand contacts execute a similar cyclic pattern, this constraint does not represent a severe restriction. However, if more complex non-gaited locomotion or loco-manipulation are considered in future, one should consider to separate the description of the foot and hand motion into individual containers.

5.3 Quasi-Planar Walking Transition (QPWT)

In order to describe a robot pose relative to its predecessor in a convenient and intuitive way, this thesis introduces the *Quasi-Planar Walking Transition (QPWT)*. It represents an extension of the step parameters used by BUSCHMANN to describe walking along a “standard circular path” [100, p. 57ff]. Note that there exist numerous similar approaches for describing biped locomotion through a limited set of parameters, cf. [114, 324]. While the original formulation of BUSCHMANN was restricted to *planar* walking on level ground, QPWTs extend this by parameters for climbing platforms / stairs (varying ground height and partial / tiptoe contact) and walking on ramps (inclined ground) which actually motivates the prefix *quasi-planar* in the name. Furthermore, QPWTs allow a full specification of the robot pose, thus, they contain also parameters for the upper body and head orientation, the CoM height, and multi-contact interaction. A formal definition of the QPWT is given in Table 5.2.

Table 5.2: Components of a *Quasi-Planar Walking Transition (QPWT)* as intuitive specification of the *End Pose* relative to the *Begin Pose* of a *Transition* (cf. Figure 5.4 and Table 5.1). The first six components $l_{x|y|z}$ and $\varphi_{x|y|z}$ are referred to as *Step Parameters* as they describe the relative transform between the previous and next stance foot pose, see Appendix D for details.

Symbol	Default	Description
l_x	0 m	step length in sagittal plane (<i>equiv. to L_x in [100, p. 60], cf. Appendix D</i>)
l_y	0 m	step length in lateral plane (<i>equiv. to L_y in [100, p. 60], cf. Appendix D</i>)
l_z	0 m	step length in vertical direction (<i>see Appendix D</i>)
φ_x	0°	step angle around local x -axis (<i>see Appendix D</i>)
φ_y	0°	step angle around local y -axis (<i>see Appendix D</i>)
φ_z	0°	step angle around vertical axis (<i>equiv. to φ_{step} in [100, p. 60], cf. Appendix D</i>)
$t_{\text{tra,dur}}$	0.8 s	custom duration of transition ($t_{\text{tra,dur}} = t_{\text{tra,end}} - t_{\text{tra,beg}}$)
h_{step}	3 cm	custom step height (<i>see Section 6.7</i>)
τ_{SF}	0	swing foot timing factor $\tau_{\text{SF}} \in [0, 1]$ (<i>see Section 6.7</i>)
–	false	flags to force partial/tiptoe contact for the stance foot (<i>at end pose</i>)
–	false	flags to force blending the hands into task-space (<i>at end pose</i>)
–	false	flags to force contact for each hand (<i>at end pose</i>)
${}_{\text{SF}}r_h$	(<i>see Section 6.9</i>)	position of the TCP frame for each hand $h \in \{\text{RH}, \text{LH}\}$ (<i>at end pose</i>)
${}_{\text{SF}}W_{h,\text{ext}}^h$	0	external (multi-contact) wrench at each hand (<i>at end pose</i>)
${}_{\text{SF}}S_{\text{UB}}$	(<i>see Section 6.6</i>)	custom orientation of the upper body frame (<i>at end pose</i>)
$q_{\text{vp}}, q_{\text{vt}}$	0°, –25°	custom joint angles for vp and vt (<i>at end pose</i>)
$\Delta_W r_{\text{CoM},z}$	0 m	custom vertical shift applied to (desired) CoM height (<i>at end pose</i>)

Since QPWTs describe biped walking, it is not necessary to specify the stance foot type (the stance switches with each step). The relative transform between the previous (begin pose) and next (end pose) stance foot pose is determined by the first six components alias *Step Parameters*, i. e., $l_{x|y|z}$ and $\varphi_{x|y|z}$. The formulas for computing the relative transform from the step parameters and vice versa are presented in Appendix D. Note that all components listed in Table 5.2 are optional, i. e., for defining a QPWT only parameters have to be specified which should differ from their default. If no component is specified at all, the robot performs a “stamping” motion (walking with zero step length). Moreover, the Boolean flags in Table 5.2 are meant to “force” a certain behavior which is not necessary if the intention is clear from the other parameters. As an example, a hand is automatically blended into task-space if its position ${}_{\text{SF}}r_h$ is specified. Similarly, a hand is automatically considered to be in closed contact if a non-zero contact wrench ${}_{\text{SF}}W_{h,\text{ext}}^h$ is given. The main advantage of QPWTs over a direct specification of a sequence of robot poses is their human readable and compact description. In Table 5.3, two exemplary TOML files are shown which demonstrate the simple syntax for regular walking (left) and the

compact description of complex scenarios (right). Although QPWTs allow to specify all components of a robot pose, they are limited to biped walking with cyclically switching stance foot. Hence, they are not suitable for describing non-gaited (“special”) motion.

Table 5.3: Exemplary TOML files specifying a sequence of QPWTs (cf. Table 5.2). Left: biped walking on a curved path as shown in Figure 4.8. Right: stepping up and down platform with hand supporting against a wall as shown in [20 @t=1m41s] (includes partial and tiptoe contact). The flag *StartWithRightLeg* in line 1 allows to select the leg which moves first, i. e., the swing foot \overline{SF} of the first transition. After this, SF and \overline{SF} switch with each transition.

#	Sequence: Curve (cf. Figure 4.8)	#	Sequence: Multi-Contact Platform Wall (cf. [20 @t=1m41s])
1	StartWithRightLeg = true	1	StartWithRightLeg = true
2	[Transitions]	2	[Transitions]
3	[Transitions.0] ↓ <i>straight walk</i>	3	[Transitions.0] ↓ <i>stamping</i>
4	StepLengthX = 0.19 ← l_x / m	4	StepLengthX = 0.0 ← l_x / m
5	[Transitions.1]	5	[Transitions.1]
6	StepLengthX = 0.19	6	StepLengthX = 0.0
7	[Transitions.2] ↓ <i>turn left</i>	7	[Transitions.2] ↓ <i>straight walk</i>
8	StepLengthX = 0.19	8	StepLengthX = 0.2
9	StepAngleZ = 0.2 ← φ_z / rad
...	...	17	[Transitions.7]
13	[Transitions.4]	18	StepLengthX = 0.2
14	StepLengthX = 0.19	19	[Transitions.8] ↓ <i>step up platform (hand / partial contact)</i>
15	StepAngleZ = 0.2	20	CustomDuration = 1.0 ← $t_{\text{tra,dur}} / s$
16	[Transitions.5] ↓ <i>straight walk</i>	21	StepLengthX = 0.3
17	StepLengthX = 0.19	22	StepLengthZ = 0.125 ← l_z / m
18	[Transitions.6] ↓ <i>turn right</i>	23	ForcePartialContact = true
19	StepLengthX = 0.19	24	LeftHandPositionXYZ = [0.0, 0.7875, 1.25] ← ${}_{\text{SF}}r_{\text{LH}} / m$
20	StepAngleZ = -0.2	25	LeftHandExternalForceXYZ = [0.0, -50.0, 0.0] ← ${}_{\text{SF}}F_{\text{LH,ext}} / N$
...	...	26	[Transitions.9]
33	[Transitions.11]	27	CustomDuration = 1.0
34	StepLengthX = 0.19	28	StepLengthX = 0.2
35	StepAngleZ = -0.2	29	LeftHandPositionXYZ = [-0.2, 0.5125, 1.25]
36	[Transitions.12] ↓ <i>straight walk</i>	30	LeftHandExternalForceXYZ = [0.0, -50.0, 0.0]
37	StepLengthX = 0.19	31	[Transitions.10] ↓ <i>straight walk (slowed down)</i>
...	...	32	CustomDuration = 1.0
44	[Transitions.16]	33	StepLengthX = 0.25
45	StepLengthX = 0.19
46	[Transitions.17] ↓ <i>turn left</i>	40	[Transitions.13]
47	StepLengthX = 0.19	41	CustomDuration = 1.0
48	StepAngleZ = 0.2	42	StepLengthX = 0.20
...	...	43	[Transitions.14] ↓ <i>step down platform (hand / tiptoe contact)</i>
61	[Transitions.22]	44	CustomDuration = 1.0
62	StepLengthX = 0.19	45	StepLengthX = 0.4
63	StepAngleZ = 0.2	46	StepLengthZ = -0.125
64	[Transitions.23] ↓ <i>straight walk</i>	47	LeftHandPositionXYZ = [0.0, 0.7875, 1.25]
65	StepLengthX = 0.19	48	LeftHandExternalForceXYZ = [0.0, -50.0, 0.0]
66	[Transitions.24] ↓ <i>turn right</i>	49	TipToeContact = true
67	StepLengthX = 0.19	50	[Transitions.15]
68	StepAngleZ = -0.2	51	CustomDuration = 1.0
...	...	52	StepLengthX = 0.2
72	[Transitions.26]	53	LeftHandPositionXYZ = [-0.2, 0.5125, 1.25]
73	StepLengthX = 0.19	54	LeftHandExternalForceXYZ = [0.0, -50.0, 0.0]
74	StepAngleZ = -0.2	55	[Transitions.16] ↓ <i>straight walk</i>
75	[Transitions.27] ↓ <i>straight walk</i>	56	StepLengthX = 0.2
76	StepLengthX = 0.19
77	[Transitions.28]	59	[Transitions.18]
78	StepLengthX = 0.19	60	StepLengthX = 0.2

5.4 Transition Planner

The task of contact planning is realized through the so-called *Transition Planner*. It represents the second stage in the planning pipeline (cf. Figure 4.11) and is responsible of filling a sequence with given begin pose and action type by a list of consecutive transitions (cf. Figure 5.4). In particular, the transition planner computes the begin and end pose of each transition as well as the additional parameters characterizing the motion in between, i. e., $t_{\text{tra,dur}}$, h_{step} , and τ_{SF} . Since the workflow of contact planning depends on the particular action type, the transition planner is split into corresponding submodules as shown in Figure 5.5.

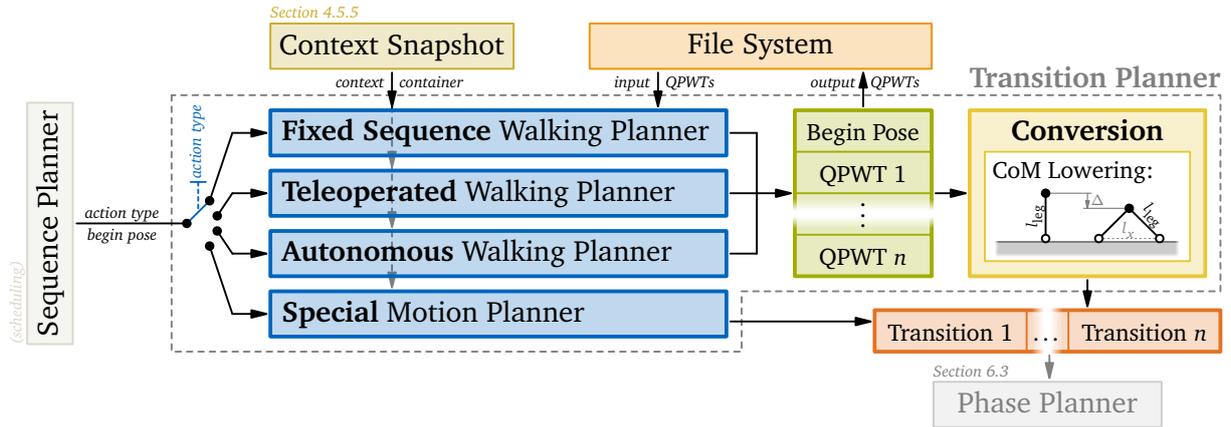


Figure 5.5: Overview of the *Transition Planner* as second stage of the planning pipeline (cf. Figure 4.11). Depending on the action type of the sequence, the corresponding submodule (blue) is triggered. The submodules for fixed sequence, teleoperated, and autonomous walking generate a sequence of QPWTs (green) which is then converted (with optional CoM lowering based on the step length l_x) to a chain of transitions (orange) using the formulas given in Appendix D. In contrast, the submodule for special (non-gaited) motions directly generates the sequence of transitions. The file system allows to read user-defined QPWT sequences from TOML files (cf. Table 5.3). Similarly, the QPWTs generated by the autonomous walking planner may be saved to the file system (e. g. for later playback).

The desired behavior for fixed sequence and teleoperated walking has already been explained in Section 5.2. Contact planning for this type of actions boils down to specifying QPWTs either from a parameterized (straight) path, an input (TOML) file, or HID controlled step parameters. For autonomous walking, the QPWTs are generated by a multi-level search presented in Section 5.5. The QPWTs are then converted to a chain of transitions which is stored in the data structure holding the motion plan. The submodule for planning special motions generates transitions directly, which allows to plan non-gaited motion not describable by QPWTs (e. g. bowing). Note that the described workflow of the transition planner is the same for dynamic replanning, i. e., the responsibility for properly updating or stopping the currently executed sequence is assigned to the corresponding submodule depending on the action type.

CoM Lowering As part of the conversion of a QPWT to the corresponding end pose of a transition, the transition planner decreases the (desired) CoM height depending on the current step length in the sagittal plane l_x , cf. Table 5.2. This is meant to keep the CoM height roughly in the center of the kinematically feasible region (a much more accurate consideration of the kinematic limits follows in Section 6.14). For the vertical shift, we use

$$\Delta_W r_{\text{SF-CoM},z} = \frac{1}{2} \left(l_{\text{leg}} - \sqrt{l_{\text{leg}}^2 - \frac{4}{25} l_x^2} \right) \quad \text{with} \quad l_{\text{leg}} := l_5 + l_6 \quad (\text{leg length, see Table 4.4}) \quad (5.2)$$

which is subtracted from ${}_W r_{\text{SF-CoM},z}$ (cf. Table 5.1) at the end pose of the transition. This heuristic has been adopted from the original WPG implementation by BUSCHMANN. The WPG proposed within this thesis uses the CoM lowering as optional feature which is enabled by default.

5.5 Autonomous Locomotion

This section describes the *Autonomous Walking Planner* (cf. Figure 5.5) as the component of the transition planner which is responsible of contact planning for autonomous (multi-contact) locomotion – also referred to as “path finding” or “navigation”. As input data, this module takes

- the begin pose of the governing sequence (for dynamic replanning: the end pose of the last transition to be kept unchanged) representing the initial state of the robot (“start”),
- the pre-processed environment model as part of the context snapshot, and
- a user-specified “goal” formulated as (horizontal) position ${}_{PW}r_{MF,x|y}$ and (vertical) rotation ${}_{PW}\varphi_{MF,z}$ of the mean foot TCP frame MF described in the planning world FoR PW.

For an intuitive specification of the goal, we use the 2D pose of the mean foot TCP frame while the remaining “passive” DoF of the robot, i. e., the vertical position and horizontal rotation, are automatically derived from the terrain. Based on this information, the contact planner generates a series of QPWTs reaching from the start to the goal which are then converted into a chain of transitions as described in the previous section.

Numerous contact planners for autonomous locomotion of legged robots have been proposed so far. A brief summary of the state of the art has already been given in Section 2.3. Most of the presented works adapt and extend well-known search algorithms such as A* or RRTs in order to find a (close to) optimal solution in a reasonable amount of time. In particular for complex problems with high-dimensional search space (such as multi-contact locomotion of humanoids), real-time planning is an active area of research. This thesis presents a discrete planner based on the A* algorithm, where real-time performance is achieved through a hierarchical approach using multiple levels of detail. In particular, the contact planner is organized as a pipeline of the five consecutive stages shown in Figure 5.6.

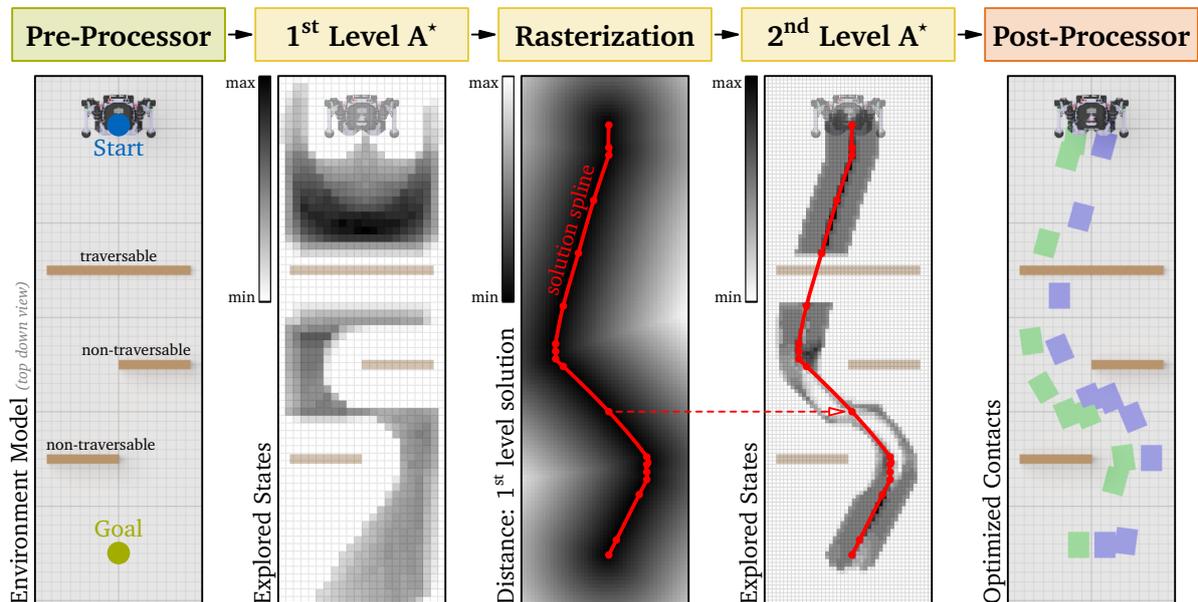


Figure 5.6: Illustration of the workflow of the contact planner for autonomous locomotion. The pipeline consists of five consecutive stages (from left to right): pre-processing (bottom: environment model in top down view with traversable (step over) and non-traversable (bypass) obstacles); 1st level A* search (bottom: map of explored states – each pixel corresponds to a terrain cell); rasterization of the solution spline (alias “guiding path”) of the 1st level search (bottom: distance map); 2nd level A* search (accelerated by rasterized solution spline); post-processing (bottom: sequence of refined footholds). See [18 @t=1m54s] and [18 @t=7m54s] for an animated visualization.

The main idea is to start with a rough search using a rather coarse search space discretization in the following denoted as the 1st level. The sequence of discrete states representing the solution

of the first level are connected by a *solution spline* acting as “guiding path”. The solution spline gets *rasterized*, meaning the creation of a special 2D map (aligned with the terrain). The cells of the map store local metrics such as the distance to the solution spline or its orientation, the remaining path length, and the remaining rotation at the closest point on the solution spline. These metrics take effect on the state costs and heuristics of A^* in order to accelerate the fine-grained 2^{nd} level search. Furthermore, both, the first and second level search, are accelerated through extensive use of pre-computed (scenario independent) buffers which are setup once during initialization of the WPG. The solution of the second level is passed to the post-processor which refines the 6D pose of footholds and finds optimal contact points for the hands in multi-contact scenarios.

In the following subsections, the contact planning pipeline is explained in more detail. In Section 5.5.1, the discretization of the search space for the first level (“coarse”), the second level (“medium”), and the post-processing (“fine”) is specified. This is followed in Section 5.5.2 by a description of the pre-processing which includes the setup of buffers for acceleration as well as a pre-evaluation of environmental surfaces. In Section 5.5.3, the A^* algorithm is introduced together with remarks on an efficient implementation on real-time platforms. The formulation of the first and second level search as well as their coupling through the rasterized solution spline follows in Section 5.5.4. The last stage of the contact planning pipeline, the post-processor, is finally described in Section 5.5.5. For a better understanding of the overall workflow, the animated visualization given in [18] @ $t=1m54s$ is recommended.

5.5.1 Discretization

Before setting up a graph search for contact planning, we first have to specify a discretized form of the robot’s state. Since the computational effort increases drastically with each additional dimension of the search space, a very abstract description is necessary in order to achieve real-time performance. For the first and second level search, an approach similar to the one proposed by CHESTNUTT [114, p. 21] is used. In particular, only the 2D pose of the robot, i. e., the horizontal position and vertical rotation, and the stance foot type (second level only) are used which is obviously much less detailed when compared to the *Robot Pose* from Section 5.2. For describing the horizontal position, we discretize the x - y -plane of the vision world frame VW such that the resulting 2D grid is aligned with the terrain specified by the environment model (cf. Figure 4.5). The vertical position of the robot (in direction of ${}_{VW}\mathbf{e}_z$) is implicitly given by the height value stored in the corresponding terrain cell. The entire process of contact planning takes place in the VW FoR where we use ${}_{PW}\mathbf{H}_{VW}$ provided by the CV system (cf. Section 4.4) to transform (lightweight) robot poses during the pre- and post-processing, while the (heavyweight) data structures of the environment model do not need any transformation.

With a maximum resolution of $1\text{ cm} \times 1\text{ cm}$ provided by the terrain specification, a rather fine discretization of the location is possible, e. g. when compared to GRIFFIN et al. [178] with $5\text{ cm} \times 5\text{ cm}$. To the author’s best knowledge, only NISHIWAKI et al. [326] use a similar precision with a horizontal resolution of $2\text{ cm} \times 2\text{ cm}$. The same holds true for discretizing the rotation around the vertical axis, for which we use a maximum resolution of 3.75° while related work typically uses a much greater step size (e. g. 10° in GRIFFIN et al. [178]). However, the contact planner proposed within this thesis uses the maximum resolution only for post-processing while the previous stages work with coarser grids. At this point, we benefit from describing terrain patches by quadtrees which provide ground information on different levels of detail.

1st Level Within the first level search, the robot’s state is described by a horizontal position ${}_{VW}r_{MF,x|y}$ discretized with a resolution of 8 cm and a discrete vertical rotation ${}_{VW}\varphi_{MF,z}$ around the positive ${}_{VW}\mathbf{e}_z$ axis with a step size of 7.5° . Since the first level search does not consider

individual feet, the 2D pose is related to the mean foot TCP frame MF. Moreover, the robot is assumed to be in an idle configuration with parallel feet (cf. Section 4.5.2). Thus, the contact with the ground is assumed to be of rectangular shape (convex hull of both footholds with default foot separation). Both feet are assumed to be in full contact.

2nd Level For the second level, the resolution of ${}_{\text{VW}}r_{\text{SF},x|y}$ is increased to 4 cm while ${}_{\text{VW}}\varphi_{\text{SF},z}$ uses the same step size of 7.5° . In contrast to the first level, the 2D pose is no longer related to MF, but the stance foot TCP frame SF instead. For this reason, the robot’s state additionally provides a binary variable indicating the stance foot type (right / left). Moreover, the contact area with the ground is given by a single foothold (current stance) which adapts its size depending on the contact type ($l_{12} \times l_{13}$ for full and $l_{10} \times l_{13}$ for partial / tiptoe contact, cf. Table 4.4). In order to keep the dimension of the search space low, we do not include the contact type within the state description. Instead, it affects the state costs within the related graph search making full contact the preferred option. The same applies to collision detection and multi-contact reachability (both evaluated using the SSV models introduced in Section 5.1), which are considered in all three stages: the first and second level search, and the post-processing. Transforming a 2D pose from the first level to the corresponding 2D pose on the second level is straightforward, however, one has to consider the relation between the MF and SF frame (translational shift by half foot separation in idle configuration).

Post-Processing During post-processing, the stance foot pose is refined using the maximum resolution of 1 cm for the horizontal position ${}_{\text{VW}}r_{\text{SF},x|y}$ and 3.75° for the vertical rotation ${}_{\text{VW}}\varphi_{\text{SF},z}$. Remaining quantities (horizontal foot rotation, hand positions for multi-contact, etc.) are directly derived from environmental geometry and have continuous values.

State Hashing A frequently occurring operation in graph-based search algorithms is the comparison of states. For the A* algorithm presented in the following, this primarily applies to checks for equality in order to find out if a certain state is contained in a given set. In order to test equality between two data structures, a simple solution is to compare their members individually. Under certain circumstances, it can be more efficient to encode each container into a single integer using a certain hash function such that testing equality between two data structures boils down to comparing two integers. A drawback of this approach is that – depending on the complexity of the structure – a hash collision might occur (differing containers get encoded to the same hash). In our case, the state descriptions on the first and second level are small enough to fit in a single 64-bit integer (stance foot type: 1 bit, horizontal position: 2×2 bytes, vertical rotation: 1 byte). This allows us to formulate an extremely efficient and guaranteed collision-free hash function by simply concatenating the memory of the components. Moreover, it is possible to efficiently recover the full state description from a given hash value without loss of information.

5.5.2 Pre-Processing

The task of pre-processing can be split up into the computation of scenario independent acceleration buffers (performed only once within an initialization routine of the WPG) and the preparation of environmental surfaces for finding optimal hand contact points (performed by the environment model manager, cf. Figure 4.11, together with building terrain patch quadtrees). The acceleration buffers might seem like a minor implementation detail, however, they have a significant influence on the overall runtime of the contact planning pipeline. Since a detailed explanation would go beyond the scope of this thesis, the following paragraph represents only a brief summary.

Acceleration Buffers In order to maximize the performance of the contact planning pipeline, the individual stages make extensive use of pre-computed internal buffers acting as lookup tables for certain quantities. Prime examples are buffers storing

- 2D rotation matrices (combined with a constant scaling factor to account for the step size of translational coordinates) for all possible (discrete) orientations,
- potential state successors and terrain cells representing the foot-ground interface for all possible orientations and contact types,
- the transformed SSV models used for collision detection (cf. Figure 5.2) and multi-contact reachability tests (cf. Figure 5.3) for all possible orientations, and
- terrain cells traversed by the swing foot for all possible (straight) swing foot paths (limited by maximum step size in sagittal and lateral plane).

For a given discrete 2D pose of the robot, these buffers allow to lookup the corresponding element related to the robot’s orientation and shift it afterwards according to the robot’s position. This gives a significant performance boost, since rotation is typically an expensive operation while translation is cheap for most quantities, e. g. SSV models⁶². Moreover, 2D areas (foot-ground interface, swing foot traversal, etc.) are described as clusters of terrain cell coordinates, which can be moved efficiently across the grid (like a “stamp”) by simply applying the robot’s position as translational offset. The buffers for storing clusters of cell coordinates are generated by rasterizing a continuous geometry onto the 2D grid (cf. Figure 5.13 left-bottom).

Surface Preparation – Adaptive Subdivision The proposed WPG models the surface of environmental objects by an indexed triangle mesh, cf. Section 4.5.1. While it is assumed that the triangles provided by the CV system are non-degenerated, they still can have an arbitrary size and shape. Since the surface evaluation method presented in the following paragraph benefits from a fine-grained grid with only slightly distorted triangles (ideally equilateral), an adaptive subdivision is applied. Instead of methods which simultaneously smooth the mesh (e. g. the classical subdivision algorithm by CATMULL and CLARK [112]), a rather simple approach was chosen which does not alter the original geometry. In particular, a maximum edge length $l_{\max} = 10\text{ cm}$ is introduced on the basis of which either none, one, two, or all three edges of a triangle are split into half, see Figure 5.7. Whenever an edge is split, it has to be ensured that the neighboring triangles remain connected, i. e., that the topological information is preserved. Depending on the maximum edge length of the original mesh, the subdivision algorithm might require multiple iterations. The adaptive subdivision algorithm has been published as part of the library *am2b-vision-interface* [16] (see the method `CGMeshExtended::adaptiveSubdivision`).

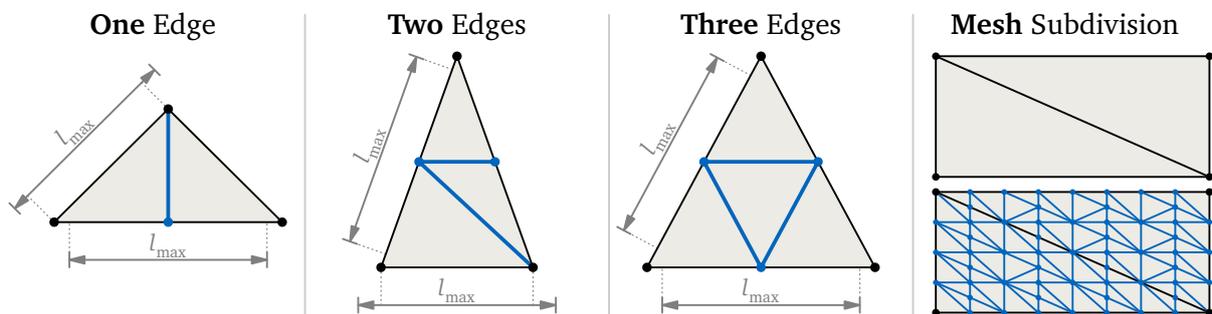


Figure 5.7: Adaptive subdivision applied to the triangles of an environmental surface mesh. Each edge exceeding the maximum allowed length of l_{\max} is split into half effectively increasing the total count of triangles. Left: subdivision of a triangle with one, two, or three edges exceeding the maximum length. Right: exemplary subdivision of a large rectangle (represented by two triangles) into a fine-grained grid of small-sized triangles (3 iterations).

⁶²For this purpose, the SSV library shipped with *Broccoli* [15] implements special, optimized methods for efficient translation-only transformations (see for example the method `SSVSegment::translateSegmentAndUpdate`).

Surface Preparation – Multi-Contact / Surface Confidence Subsequent to the adaptive subdivision, the surface’s local suitability for potential hand contacts is evaluated. In order to quantify the fitness of a certain point on the surface, we introduce the *multi-contact / surface confidence* c_{surf} given as the product

$$c_{\text{surf}}(c_{\text{vis}}, c_{\text{dist}}, c_{\kappa}) := c_{\text{vis}} c_{\text{dist}} c_{\kappa} \in [0, 1] \quad (5.3)$$

where $c_{\text{vis}} \in [0, 1]$ denotes the visual perception confidence and $c_{\text{dist}} \in [0, 1]$ and $c_{\kappa} \in [0, 1]$ represent confidence values related to the distance to the mesh boundary and the local curvature, respectively. While c_{vis} is provided by the CV system (similar to the terrain confidence, see Section 4.5.1), we derive c_{dist} and c_{κ} purely from geometry. Same as before, a confidence value of 1 indicates that the considered location is an ideal contact point while a value of 0 is assigned to regions which are not suitable for hand support at all.

In a first step, the topology of the triangle mesh is analyzed to allow an efficient computation of the n -ring neighborhood of a vertex (see Figure 5.8, left). For “open” meshes, this includes the identification of the boundary of the mesh, i. e., the set of edges which are connected to a single triangle. Once the boundary is identified, we can move towards the interior of the mesh by repeatedly “growing” along the 1-ring neighborhood until all vertices of the mesh have been visited. For each interior vertex, the connection to the boundary is stored in the form of the minimum distance $l_{\text{dist},\text{min}}$ (see Figure 5.8, center) which is measured along the traversed edges taking the individual edge length into account. While this represents only a rather coarse approximation, it can be implemented very efficiently within the aforementioned process of growing rings. The corresponding confidence value c_{dist} is then given by

$$c_{\text{dist}}(l_{\text{dist},\text{min}}) := \begin{cases} \frac{l_{\text{dist},\text{min}}}{l_{\text{dist},\text{max}}} & \text{if } l_{\text{dist},\text{min}} \leq l_{\text{dist},\text{max}} \\ 1 & \text{else} \end{cases}, \quad \text{with } l_{\text{dist},\text{max}} := \frac{3}{4} d_h \quad (d_h \text{ from Table 4.4}) \quad (5.4)$$

such that regions in the close vicinity of the boundary (parameterized by the hand diameter d_h) are penalized. In the case of a “closed” mesh (each edge is connected to two triangles such that no boundary exists), we assume $l_{\text{dist},\text{min}} \rightarrow \infty$ and consequently $c_{\text{dist}} = 1$.

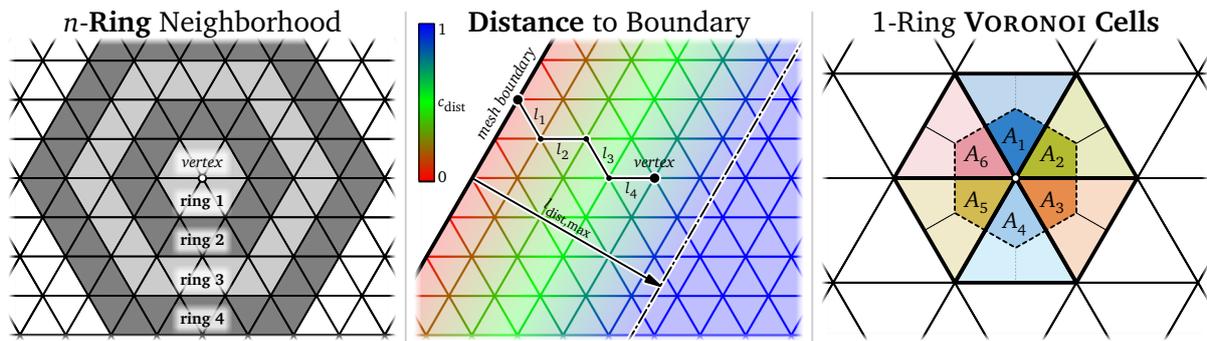


Figure 5.8: Illustration of concepts used during evaluation of environmental surfaces. From left to right: n -ring neighborhood of a vertex; minimum distance of a vertex to the mesh boundary (here: $l_{\text{dist},\text{min}} = l_1 + l_2 + l_3 + l_4$) and its influence on c_{dist} ; areas A_{1-6} of 1-ring VORONOI cells around a vertex used to compute the local curvature affecting c_{κ} .

For computing the local curvature of a surface represented by a triangle mesh, numerous methods have been proposed in literature. Within this thesis, the method presented by MEYER et al. in [306, p. 35ff] is used since it is consistent (exact for tessellation with infinitesimal small triangles), robust (works for irregular meshes), accurate, and efficient. The main idea is to approximate the surface integral around the considered vertex by using a mixed finite element/volume approach based on the VORONOI cells of the triangles in the 1-ring neighborhood (see Figure 5.8, right). The algorithm proposed by MEYER et al. delivers the principal curvatures $\kappa_1, \kappa_2 \in \mathbb{R}$ with $\kappa_1 \geq \kappa_2$, the Gaussian curvature $\kappa_{\text{gau}} := \kappa_1 \kappa_2$, the mean curvature

$\kappa_{\text{mean}} := \frac{1}{2}(\kappa_1 + \kappa_2)$, and the (discrete) mean curvature normal operator $\kappa \in \mathbb{R}^3$ alias LAPLACE-BELTRAMI operator (divergence of the gradient, see [306, p. 44ff]) with $\|\kappa\| = 2\kappa_{\text{mean}}$ for each interior vertex. For vertices lying on the boundary of the mesh, a curvature is not defined. In this case, we use $\kappa_1 = \kappa_2 = 0$ as a fallback solution. Note that this choice does not affect c_{surf} since also $c_{\text{dist}} = 0$ for $l_{\text{dist},\text{min}} = 0$. For a better understanding of these metrics, Figure 5.9 shows the evaluation results for an exemplary (synthetic) surface.

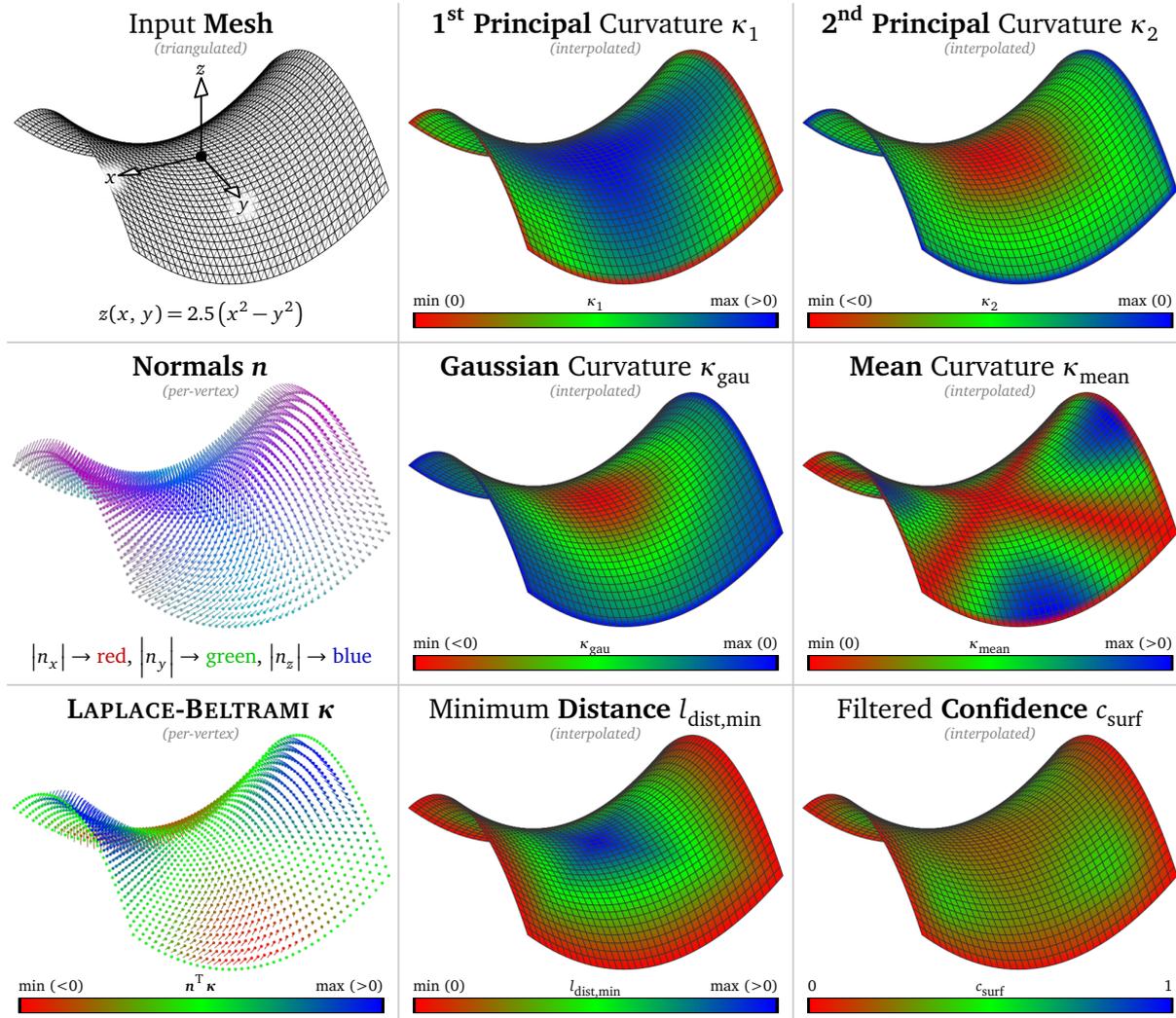


Figure 5.9: Evaluation of a hyperbolic paraboloid as an exemplary surface featuring curvatures in two directions. The corresponding triangle mesh is considered “open” (with $x \in [-0.2, 0.2]$ and $y \in [-0.2, 0.2]$). The per-vertex normals \mathbf{n} (color-coded by direction) are automatically derived using the method proposed by THÜRRNER and WÜTHRICH, see [410]. The principal curvatures κ_1 and κ_2 , the Gaussian curvature κ_{gau} , the mean curvature κ_{mean} , and the LAPLACE-BELTRAMI operator κ are computed according to MEYER et al. [306, p. 35ff]. The minimum distance to the boundary $l_{\text{dist},\text{min}}$ is obtained following the scheme illustrated in Figure 5.8 (center). Finally, Equation 5.3 (with subsequent Gaussian filtering) leads to the depicted *surface* confidence c_{surf} (assuming a constant $c_{\text{vis}} = 1$).

For computing c_{κ} from the local curvature, we introduce the maximum (absolute) principal curvature $\kappa_{\text{max}} := \max(|\kappa_1|, |\kappa_2|)$. Furthermore, we differentiate between three cases characterizing the shape of the surface in the neighborhood of the considered vertex:

$$c_{\kappa}(\kappa_{\text{max}}) := \begin{cases} c_{\kappa,\text{convex}}(\kappa_{\text{max}}) & \text{if } \kappa_{\text{gau}} \geq -\varepsilon \text{ and } \mathbf{n}^T \kappa > 0, \\ c_{\kappa,\text{concave}}(\kappa_{\text{max}}) & \text{if } \kappa_{\text{gau}} \geq -\varepsilon \text{ and } \mathbf{n}^T \kappa \leq 0, \\ c_{\kappa,\text{saddle}}(\kappa_{\text{max}}) & \text{else} \end{cases} \quad (5.5)$$

Here, we use κ_{gau} to determine if the considered vertex represents an elliptic point ($\kappa_{\text{gau}} > \varepsilon$: both principal curvatures have the same sign), a parabolic point ($|\kappa_{\text{gau}}| \leq \varepsilon$: at least one principal curvature is close to or equal to zero), or a saddle point ($\kappa_{\text{gau}} < -\varepsilon$: principal curvatures have different sign). For computing c_{κ} , we handle elliptic and parabolic points in the same way. However, we differentiate between a *convex* and *concave* surface by checking if the normal \mathbf{n} (defines “inside” and “outside” of the object) and the mean curvature normal operator κ point into the same or opposite direction (sign of $\mathbf{n}^T \kappa$). We concretize the three cases presented in Equation 5.5 by defining

$$c_{\kappa, \text{convex}}(\kappa_{\text{max}}) := \begin{cases} c_{\kappa, \text{flat}} \left(1 - \frac{\kappa_{\text{max}}}{\kappa_{\text{thres},1}}\right) & \text{if } \kappa_{\text{max}} \leq \kappa_{\text{thres},1}, \\ 0 & \text{else} \end{cases} \quad (5.6)$$

$$c_{\kappa, \text{concave}}(\kappa_{\text{max}}) := \begin{cases} c_{\kappa, \text{flat}} + (1 - c_{\kappa, \text{flat}}) \frac{\kappa_{\text{max}}}{\kappa_{\text{opt}}} & \text{if } \kappa_{\text{max}} \leq \kappa_{\text{opt}}, \\ 1 - \frac{(\kappa_{\text{max}} - \kappa_{\text{opt}})}{(\kappa_{\text{thres},2} - \kappa_{\text{opt}})} & \text{if } \kappa_{\text{opt}} < \kappa_{\text{max}} \leq \kappa_{\text{thres},2}, \\ 0 & \text{else} \end{cases} \quad (5.7)$$

$$c_{\kappa, \text{saddle}}(\kappa_{\text{max}}) := c_{\kappa, \text{convex}}(\kappa_{\text{max}}) \quad (5.8)$$

where we use the parametrization

$$c_{\kappa, \text{flat}} := \frac{3}{4}, \quad \kappa_{\text{thres},1} := \frac{2}{3d_h}, \quad \kappa_{\text{thres},2} := \frac{2}{d_h}, \quad \text{and} \quad \kappa_{\text{opt}} := \frac{1}{d_h}. \quad (d_h \text{ from Table 4.4}) \quad (5.9)$$

For zero curvature $\kappa_{\text{max}} = 0$ representing a flat surface, we choose $c_{\kappa, \text{flat}}$ as a “default” confidence. Due to the higher risk of slipping on convex- and saddle-shaped areas, the confidence degrades in these cases with increasing curvature until it reaches 0 at the threshold $\kappa_{\text{thres},1}$. In contrast, concave regions promise a higher contact stability, thus, in this case the confidence increases with the curvature finally reaching its maximum of 1 at the optimum κ_{opt} . Beyond κ_{opt} , the confidence decreases again and reaches 0 at the threshold $\kappa_{\text{thres},2}$.

Once, c_{dist} and c_{κ} have been computed, we can derive the overall *surface* confidence c_{surf} by evaluating Equation 5.3. In order to compensate parasitic effects due to the approximation of a continuous surface by a discrete triangle mesh, we blur c_{surf} by applying a Gaussian filter. To limit the computational cost, the kernel size is chosen to the 3-ring neighborhood. Moreover, we use the Euclidean distance between the currently investigated vertex and its neighbors in combination with a standard deviation of $c_{\text{surf}, \sigma} = d_h/2$. Vertices lying on the boundary of the mesh are not filtered in order to keep their confidence at zero.

The parametrization presented in Equation 5.9 has been determined empirically such that a reasonable c_{κ} is obtained for typical geometries occurring in our target scenarios such as the lab table shown in Figure 4.6. In addition to the assumption of $c_{\text{vis}} = 1$ (perfect perception), the table is approximated by a closed mesh ($c_{\text{dist}} = 1$) such that the depicted surface confidence c_{surf} is equivalent to the (filtered) c_{κ} . Note that the flat area on the top (cyan) yields the object’s maximum confidence of $c_{\kappa, \text{flat}} = 0.75$. At the table’s (convex) edges, the confidence rapidly degrades (blurred by Gaussian filter).

The presented method for evaluating environmental surfaces based on c_{vis} , c_{dist} , and c_{κ} has not been published before. To the author’s best knowledge, no other work has proposed a comparable algorithm in the context of multi-contact locomotion. The probably closest match is given by CHESTNUTT in [114, p. 24ff], where the local curvature of the terrain is computed in order to reason about the “stability” of footholds. While level ground is considered optimal, convex areas are penalized more than concave ones. Besides the curvature, CHESTNUTT also proposed other measures such as the roughness of the terrain. A similar (but much simpler) metric based on the maximum height difference within a cluster of terrain cells is used also by the graph search presented within this thesis (details follow in Section 5.5.4).

5.5.3 A* Algorithm

For finding the optimum path from a given start state to a user-specified goal state within the discretized search space, the WPG uses (except for slight modifications) the original A* algorithm introduced by HART et al. in [190]. Among the variety of search algorithms, A* is probably the most prominent one. Consequently, there exists an excessive amount of literature covering its working principle and properties. Instead of repeating well-known fundamentals, this section focuses on the particular implementation of A* within the contact planning pipeline of LOLA. Note that there exist numerous extensions and variations of A*. A brief overview with regard to legged locomotion planning has already been given in Section 2.3. For LOLA, real-time contact planning for multi-contact locomotion represented an entirely new domain. Hence, it seemed reasonable to stick to the original form of A* and focus on the formulation of appropriate models, costs, and the search space instead. Further optimizations through extending or replacing A* as the underlying graph search algorithm are left for future investigations.

As a graph traversal algorithm, A* builds a tree of *states* $\{S_i\}$ which, in our case, represent the robot's discretized state as defined in Section 5.5.1. Each state S_i is assigned to a corresponding *node* \mathcal{N}_i , which is simply a data container holding the state (includes its hash, cf. Section 5.5.1) and related costs. Except for the start node alias “root”, each node additionally stores a link to its predecessor (describes the optimum path) and the total count of predecessors. The nodes are connected through transitions, for which (unidirectional) *transition costs* $c_t(\mathcal{N}_{\text{pre}}, S_{\text{next}})$ are defined. As a slight modification to the original formulation by HART et al., we additionally introduce *state costs* $c_s(S_i)$, which are applied – independently of the predecessor – upon reaching the corresponding state S_i . State costs can be seen as an additional contribution to the transition costs, thus, they do not affect the properties of A* in any way. However, they make the specification of costs clearer and allow a much more efficient⁶³ implementation in our case. In order to limit memory consumption, an implicit graph description is used such that only space for explored nodes needs to be allocated. For each node \mathcal{N}_i of the graph, A* assigns a *score*

$$c_f(\mathcal{N}_i) := c_g(\mathcal{N}_i) + c_h(S_i) \quad \text{with} \quad c_g(\mathcal{N}_{\text{next}}) := c_g(\mathcal{N}_{\text{pre}}) + c_t(\mathcal{N}_{\text{pre}}, S_{\text{next}}) + c_s(S_{\text{next}}) \quad (5.10)$$

where $c_g(\mathcal{N}_i)$ denotes the *minimum path cost* from the start to the current state S_i (as accumulation of transition and state costs) and $c_h(S_i)$ represents the *heuristic* estimating the costs from the current state to the goal. An important requirement for optimality is that the heuristic represents a lower bound, i. e., it must be lesser or equal to the real costs of the remaining path [190]. For finding the shortest path between two locations in Cartesian space, a frequently used approach is to use a heuristic based on the Euclidean distance. Indeed, choosing a heuristic of $c_h = 0$ is also valid (equivalent to DIJKSTRA's algorithm [135]). However, in order to obtain a decent acceleration of the search, one has to find a close approximation of the real costs which is also cheap to compute. The score $c_f(\mathcal{N}_i)$ is used to sort \mathcal{N}_i within a list of currently investigated nodes, the so-called *open list*. Upon each iteration of A*, the node with the lowest score c_f is extracted and expanded (evaluation of potential successors). The iteration continues until the goal is reached. Same as with the original formulation of A*, we allow the definition of multiple⁶⁴ goal states. Once the goal is reached, the optimum path is obtained by recursively following the predecessor links stored within each node. Algorithm 5.1 provides a more detailed description of the particular implementation of A* used within the WPG of LOLA.

⁶³The state costs $c_s(S_i)$ are evaluated only once upon first occurrence of the state S_i . In contrast, the transition costs $c_t(\mathcal{N}_{\text{pre}}, S_{\text{next}})$ towards the state S_{next} depend on the previous node \mathcal{N}_{pre} , i. e., they have to be evaluated for each potential predecessor. Thus, splitting the costs for moving from node \mathcal{N}_{pre} to state S_{next} into separated transition and state costs allows to avoid duplicate calculations which depend on S_{next} only. Since the state costs (specified in the following section) are rather expensive to compute, this extension leads to a significant performance boost.

⁶⁴The proposed WPG allows the user to select a goal through specifying the horizontal position ${}_{\text{PW}}r_{\text{MF},x|y}$ and the vertical rotation ${}_{\text{PW}}\varphi_{\text{MF},z}$ of the mean foot TCP frame. This leads to a single goal state within the first level and two (equivalent) goal states (left or right stance) in the second level A* search of the contact planning pipeline. The heuristic has to be chosen such that it does not overestimate the remaining path costs for any goal state.

Algorithm 5.1: Implementation of the A* search algorithm within the contact planning pipeline of LOLA.

Input: Start state $\mathcal{S}_{\text{start}}$ and finite set of goal states $\{\mathcal{S}_{\text{goal},i}\}$
Output: Optimum path from the start state $\mathcal{S}_{\text{start}}$ to one of the goal states $\{\mathcal{S}_{\text{goal},i}\}$

```

begin
   $c_{s,\text{start}} \leftarrow \text{stateCost}(\mathcal{S}_{\text{start}})$  // state cost of start state
   $c_{s,\text{goal}} \leftarrow \min(\{\text{stateCost}(\mathcal{S}_{\text{goal},i})\})$  // minimum state cost of all goal states
  if  $c_{s,\text{start}} = \infty$  or  $c_{s,\text{goal}} = \infty$  then // check if start state or all goal states are invalid
    | return Error // start state and/or all goal states are invalid
  end
   $c_{g,\text{start}} \leftarrow c_{s,\text{start}}$  // initialize minimum path cost with state cost of start state
   $c_{h,\text{start}} \leftarrow \text{heur}(\mathcal{S}_{\text{start}}, \{\mathcal{S}_{\text{goal},i}\}) + c_{s,\text{goal}}$  // heuristic of start state ( $c_{s,\text{goal}}$  is not included in operator heur())
   $\mathcal{N}_{\text{start}} \leftarrow \text{node}(\mathcal{S}_{\text{start}}, \emptyset, c_{s,\text{start}}, c_{g,\text{start}}, c_{h,\text{start}})$  // create start ("root") node (no predecessor:  $\text{pred}(\mathcal{N}_{\text{start}}) = \emptyset$ )
  openList.insert( $\mathcal{N}_{\text{start}}, c_{f,\text{start}} = c_{g,\text{start}} + c_{h,\text{start}}$ ) // initialize open list with root node

  while openList.isEmpty() = false do // repeat unless open list becomes empty
     $\mathcal{N}_{\text{cur}}, \mathcal{S}_{\text{cur}}, c_{g,\text{cur}} \leftarrow \text{openList.extractMin}()$  // extract node with lowest score  $c_f$  from open list
    if  $\mathcal{S}_{\text{cur}} \in \{\mathcal{S}_{\text{goal},i}\}$  then // check if we reached the goal
      optimumPath.insert( $\mathcal{S}_{\text{cur}}$ ) // initialize optimum path with reached goal state
      while  $\text{pred}(\mathcal{N}_{\text{cur}}) \neq \emptyset$  do // follow chain of predecessors until root node is reached
        |  $\mathcal{N}_{\text{cur}}, \mathcal{S}_{\text{cur}} \leftarrow \text{pred}(\mathcal{N}_{\text{cur}})$  // switch to the predecessor of the current node
        | optimumPath.insert( $\mathcal{S}_{\text{cur}}$ ) // add predecessor state to the optimum path
      end
      return optimumPath.invertOrder() // output optimum path (in reverse order)
    end
    closedSet.insert( $\mathcal{S}_{\text{cur}}$ ) // add current state to closed set such that it does not get investigated again
     $\{\mathcal{S}_{\text{succ},i}\} \leftarrow \text{succList}(\mathcal{N}_{\text{cur}})$  // compute list of potential successors of current node
    for  $\mathcal{S}_{\text{succ}} \in \{\mathcal{S}_{\text{succ},i}\}$  do // iterate over all potential successors
      if closedSet.contains( $\mathcal{S}_{\text{succ}}$ ) = false then // skip successors which are in the closed set
         $\mathcal{N}_{\text{succ}} \leftarrow \text{openList.find}(\mathcal{S}_{\text{succ}})$  // try to find corresponding node in open list
        if  $\mathcal{N}_{\text{succ}} \neq \emptyset$  then // check if successor was found in open list
           $c_t \leftarrow \text{tranCost}(\mathcal{N}_{\text{cur}}, \mathcal{S}_{\text{succ}})$  // transition cost from current node to successor state
          if  $c_t \neq \infty$  then // skip successor if transition is invalid
             $c_{s,\text{succ}}, c_{g,\text{succ,pre}}, c_{h,\text{succ}} \leftarrow \mathcal{N}_{\text{succ}}$  // extract previously computed costs
             $c_{g,\text{succ,new}} \leftarrow c_{g,\text{cur}} + c_t + c_{s,\text{succ}}$  // min. path cost of successor assuming  $\mathcal{N}_{\text{cur}}$  as predecessor
            if  $c_{g,\text{succ,new}} < c_{g,\text{succ,pre}}$  then // check if we found a less expensive path
               $c_{g,\text{succ}} \leftarrow c_{g,\text{succ,new}}$  // replace minimum path cost for using  $\mathcal{N}_{\text{cur}}$  as new predecessor
               $\mathcal{N}_{\text{succ}} \leftarrow \text{node}(\mathcal{S}_{\text{succ}}, \mathcal{N}_{\text{cur}}, c_{s,\text{succ}}, c_{g,\text{succ}}, c_{h,\text{succ}})$  // update successor (new predecessor)
              openList.update( $\mathcal{N}_{\text{succ}}, c_{f,\text{succ}} = c_{g,\text{succ}} + c_{h,\text{succ}}$ ) // reorder successor node in open list
            end
          end
        else // successor is not in open list
           $c_{s,\text{succ}} \leftarrow \text{stateCost}(\mathcal{S}_{\text{succ}})$  // state cost of successor state
          if  $c_{s,\text{succ}} \neq \infty$  then // check if successor state is valid
             $c_t \leftarrow \text{tranCost}(\mathcal{N}_{\text{cur}}, \mathcal{S}_{\text{succ}})$  // transition cost from current node to successor state
            if  $c_t \neq \infty$  then // skip successor if transition is invalid
               $c_{g,\text{succ}} \leftarrow c_{g,\text{cur}} + c_t + c_{s,\text{succ}}$  // minimum path cost of successor
               $c_{h,\text{succ}} \leftarrow \text{heur}(\mathcal{S}_{\text{succ}}, \{\mathcal{S}_{\text{goal},i}\}) + c_{s,\text{goal}}$  // heuristic of successor state
               $\mathcal{N}_{\text{succ}} \leftarrow \text{node}(\mathcal{S}_{\text{succ}}, \mathcal{N}_{\text{cur}}, c_{s,\text{succ}}, c_{g,\text{succ}}, c_{h,\text{succ}})$  // create successor node
              openList.insert( $\mathcal{N}_{\text{succ}}, c_{f,\text{succ}} = c_{g,\text{succ}} + c_{h,\text{succ}}$ ) // insert successor node to open list
            end
          end
        else // successor state is invalid
          | closedSet.insert( $\mathcal{S}_{\text{succ}}$ ) // add successor state to closed set (do not investigate again)
        end
      end
    end
  end
end
return Error // no solution found (none of the goal states is reachable from the start state)
end

```

Within Algorithm 5.1, the node graph is represented by two containers, the *open list* and the *closed set*. The open list stores all currently investigated nodes in a special way such that it allows an efficient extraction of the element with the lowest score c_f (best-first). Since its particular realization has a significant influence on the overall runtime of A^* , a custom (optimized) implementation is used which will be described in the following. In contrast, the closed set represents a much simpler container storing finally evaluated and invalid states such that they do not get considered again. Since the closed set only needs to provide methods for inserting elements and checking containment, it is realized through a `std::unordered_set` container (elements are hash values of states) as provided by the C++ standard library.

Note that Algorithm 5.1 is not restricted to LOLA or even legged locomotion but instead represents a generic implementation of A^* . Indeed, the application-specific functionality is introduced with the operators `succList()`, `stateCost()`, `tranCost()`, and `heur()`, which are defined individually for the first and second level search of the contact planning pipeline, see Section 5.5.4 for details. The presented implementation of A^* minimizes the count of calls for these operators by reusing previous results whenever possible. Moreover, by returning an infinite cost, the operators `stateCost()` and `tranCost()` can label certain states and transitions as *invalid* such that they do not get considered again. This way, states and transitions can be excluded on the basis of the local environment (e. g. the terrain) right after expansion of potential successors (similar to the “adaptive node expansion” proposed by KARKOWSKI et al. in [244]).

Efficient Implementation of the Open List Besides the four application-specific operators, a major impact on the overall performance of A^* is given by the realization of the open list. In addition to guaranteeing consistency, i. e., correct order of the contained elements, Algorithm 5.1 requires the open list to provide the methods

- isEmpty()**: returns flag indicating if elements are contained,
- insert($\mathcal{N}_i, c_{f,i}$)**: inserts the node \mathcal{N}_i and sorts it according to its score $c_{f,i}$,
- extractMin()**: returns the node with minimum score and removes it from the container,
- update($\mathcal{N}_i, c_{f,i}$)**: replaces the node \mathcal{N}_i and sorts it according to the new score $c_{f,i}$, and
- find(\mathcal{S}_i)**: returns the node related to the state \mathcal{S}_i (without removing it from the container).

The first three operations are characteristic for a *priority queue*, while `update()` and `find()` represent less common extensions. A classic implementation of a priority queue is the *binary heap* originally introduced by WILLIAMS in [435] as data structure for the famous *heap sort* algorithm. Indeed, numerous other realizations of priority queues have been proposed so far. In [266], LARKIN et al. compared the performance of different implementations for the specific application as an open list within DIJKSTRA’s algorithm [135]. For their path finding scenario (full USA road map), LARKIN et al. identified the *implicit D-ary heap*⁶⁵ with branching factor 4 as the optimum choice effectively being more than three times faster than the (often for A^* suggested) FIBONACCI heap introduced by FREDMAN and TARJAN in [159]. For the open list used in Algorithm 5.1, we adopt the implicit *D-ary heap* proposed in [266] and extend it for our purposes. The source code of the resulting open list container has been published as part⁶⁶ of *Broccoli*. In order to avoid dynamic memory allocation at runtime, our heap further integrates a custom pool allocator⁶⁷ responsible for storing the actual node data.

⁶⁵An implicit *D-ary heap* is a generalization of a binary heap ($D = 2$), where each node has $D > 1$ child nodes. The data is stored as an *implicit heap*, which means that it is encoded as a level-order traversal in an array (in contrast to an *explicit heap* which stores heap-allocated nodes and pointers). This typically has better caching behavior since all nodes are stored in a continuous container. Moreover, it allows an efficient iteration over all nodes in the tree.

⁶⁶See the class `ImplicitDaryHeap` in the module `memory` of *Broccoli*.

⁶⁷See the class `PoolAllocator` in the module `memory` of *Broccoli*.

5.5.4 Hierarchical Graph Search

This section briefly describes the first level search, the rasterization of its solution, and the second level search which together represent the stages two, three, and four of the contact planning pipeline shown in Figure 5.6. As already discussed in Section 2.3, accelerating a (heavyweight) fine-grained search by some kind of guiding path computed by a preceding (lightweight) coarse-grained search is a common approach in locomotion planning. Indeed, the novelty of the contact planner proposed within this thesis lies in the used models for evaluating foot contacts, collisions, and multi-contact situations as well as the specification of corresponding A^* costs and the way how the first and second level search are coupled – all together leading to a versatile multi-contact planner with real-time performance. Since a full specification of the used cost functions would go beyond the scope of this document, only a brief summary of the main contributions together with the most important parameters is given instead. Indeed, the proposed contact planning pipeline is configured through more than 120 parameters which have been tuned such that reasonable results are obtained for the scenarios shown in the video [18 @t=6m46s]. It has to be highlighted that the *same* set of parameters is used for all scenarios, thus, no scenario-dependent tweaking is necessary. In the following, we make use of the models which have been introduced in Section 5.1. In particular,

- the contact area between the foot and the ground is represented by a cluster of terrain cells whose shape depends on the contact type (full or partial / tiptoe, cf. Figure 5.1),
- the task-space SSV model (with optional SSVs to model the left / right arm’s null-space motion depending on the multi-contact situation, cf. Figure 5.2) is used to detect collisions with environmental objects and label corresponding A^* states as invalid, and
- the multi-contact target volume (cf. Figure 5.3) is used to efficiently evaluate reachability⁶⁸ and reward corresponding A^* states which allow hand support.

Furthermore, we filter the environment model provided by the context snapshot in order to generate a list of objects which are relevant for multi-contact locomotion (the ones classified as walls, tables, etc. and with a minimum surface confidence $c_{\text{surf,min}}$ of 1/3) and a list of objects which should be considered during collision detection (the ones not classified as floor).

1st Level – States and Successors Within the first level A^* search, each state S_i describes a position ${}_{\text{VW}}r_{\text{MF},x|y}$ and an orientation ${}_{\text{VW}}\varphi_{\text{MF},z}$ which are discretized according to Section 5.5.1. The operator $\text{succList}(\mathcal{N}_i)$ provides a set of potential successors⁶⁹ which lie in front of the 2D pose described by S_i . The set is parameterized through the step parameters $l_{x,\text{min}} = 8\text{ cm}$, $l_{x,\text{max}} = 64\text{ cm}$, $\varphi_{z,\text{min}} = -15^\circ$, and $\varphi_{z,\text{max}} = 15^\circ$. Since $l_{x,\text{min}} > 0$, walking backwards is not considered.

1st Level – State Costs In a first step, the operator $\text{stateCost}(S_i)$ checks if the terrain cell related to the state S_i has a non-zero confidence. This allows to quickly discard unrevealed regions. Then, the cluster of terrain cells describing the foot-ground interface is analyzed. The cluster is determined through rasterization of the (minimum) rectangle enclosing parallel footholds (in full contact) with default foot separation. We formulate the state costs $c_s(S_i)$ as linear function of the maximum height difference (lower is better) and the mean confidence value (higher is better) of all cells in the cluster. A height difference of more than 4 cm (very bumpy terrain) or a mean confidence $c_{\text{vis,mean}}$ of less than 2/3 (uncertain regions) make the state invalid. Next, the multi-contact target volumes for the left and right hand are used to check reachability of environmental objects. If no multi-contact capable object is reachable, the state is further penalized

⁶⁸Similar to KUMAGAI et al. [262], we assume that a hand contact is feasible if it is geometrically reachable during the considered step. However, this thesis uses a different approach for modeling volumes and surfaces and also considers geometric properties of the object’s shape.

⁶⁹Since the successors of an A^* state do not necessarily have to be in its “physical” vicinity, the often equivalently used term “neighbors” can be misleading and is therefore avoided in this thesis.

with additional costs. Left and right hand are considered individually such that a configuration with both hands in contact is preferred over single-handed support. Finally, a collision check is performed where the configuration of the task-space SSV model (activation of left / right arm SSV) depends on the previously identified multi-contact feasibility. In case of a collision, the state is labeled as invalid. Note that the order of execution of validity checks is chosen according to the likelihood of the corresponding event such that – in case of an invalid state – calculations are aborted as soon as possible.

1st Level – Transition Costs The operator $\text{tranCost}(\mathcal{N}_{\text{pre}}, \mathcal{S}_{\text{next}})$ evaluates the traveled path length (Euclidean distance between \mathcal{S}_{pre} and $\mathcal{S}_{\text{next}}$), the step length in vertical direction (difference of mean heights between terrain cells related to \mathcal{S}_{pre} and $\mathcal{S}_{\text{next}}$), and the cluster of terrain cells traversed by the robot (foot-ground interface extruded along line connecting \mathcal{S}_{pre} and $\mathcal{S}_{\text{next}}$). The traveled path length is used to limit the step length in walking direction ($l_{x,\text{max}} = 32$ cm) for the very first⁷⁰ and last⁷¹ transition of a sequence. This effectively limits the walking speed at the beginning and end of the sequence leading to a more decent acceleration and deceleration. For the (absolute) step length in vertical direction, we set a maximum of 13 cm which complies with the kinematic capabilities of LOLA. The cluster of terrain cells traversed by the robot allows to identify the height of obstacles for which we choose a maximum of 15 cm. After these validity checks, the transition costs $c_t(\mathcal{N}_{\text{pre}}, \mathcal{S}_{\text{next}})$ are computed as the sum of: a constant penalty (such that sequences with fewer steps are preferred), a linear cost for the traveled path length (lower is better; to reward short paths), a linear cost for the step length in walking direction (optimum at $l_{x,\text{opt}} = 40$ cm representing the “desired” step length), a quadratic cost for the vertical rotation (optimum at $\varphi_{z,\text{opt}} = 0$ to prefer straight walking), a linear cost for the step length in vertical direction (lower is better), and a linear cost for the obstacle height (lower is better). Obviously, the individual weighting of cost terms has a significant influence on the “decisions” the A^* algorithm makes, e. g. if an obstacle is traversed or bypassed or – considering a multi-contact scenario – if the robot leaves the shortest path in order to allow hand support against a close-by object. Indeed, the “optimum” path strongly depends on the application and has no universal definition.

1st Level – Heuristic The minimum path cost from the given state \mathcal{S}_i to one of the goal states $\{\mathcal{S}_{\text{goal},i}\}$ is given by the accumulation of state and transition costs assigned to the remaining section of the (unknown) optimum path. The operator $\text{heur}(\mathcal{S}_i)$ returns a lower bound for these costs. Note that this excludes the minimum state cost of the goal states $c_{s,\text{goal}}$ which is directly added by Algorithm 5.1 instead. For this purpose, we first compute a lower bound for the remaining path length (minimum Euclidean distance between \mathcal{S}_i and $\{\mathcal{S}_{\text{goal},i}\}$) and the remaining rotation (minimum relative rotation between the \mathcal{S}_i and $\{\mathcal{S}_{\text{goal},i}\}$). Together with the previously declared maximum step length in walking direction and the maximum vertical rotation per step (see the operator $\text{succList}(\mathcal{N}_i)$), this allows us to compute the minimum count of steps the robot has to make in order to reach one of the goal states. In addition, we determine the minimum height difference between \mathcal{S}_i and $\{\mathcal{S}_{\text{goal},i}\}$ by comparing the mean height values stored in the corresponding terrain cells. The heuristic $c_h(\mathcal{S}_i)$ is then computed as the sum of: the minimum step count times the constant penalty for transitions, the linear cost for the path length, the minimum step count required for the remaining rotation times the quadratic cost for rotation with the smallest possible step angle, and the linear cost for the height difference. For computing $c_h(\mathcal{S}_i)$, we use the same parametrization as for the transition costs such that the heuristic is guaranteed to represent a lower bound. In contrast to the transition costs, we do not formulate a contribution for the step length in walking direction since we assume that every step has optimal length. Similarly, we assume that there are no obstacles on the remaining path such that there is no contribution related to the obstacle height.

⁷⁰To identify the very first transition, we check if the node \mathcal{N}_{pre} lacks of a predecessor, i. e., if it is the root node.

⁷¹To identify the very last transition, we check if the state $\mathcal{S}_{\text{next}}$ is contained in the set of goal states $\{\mathcal{S}_{\text{goal},i}\}$.

1st Level – Results Figure 5.10 presents exemplary results for a scenario where the robot traverses and bypasses obstacles. It has to be highlighted that the proposed first level search is much more general when compared to continuous 2D path planners (e. g. the one proposed by KARKOWSKI and BENNEWITZ in [243]) which typically do not allow stepping over obstacles or climbing stairs. Hence, although we do not consider individual feet in the first level, the discrete nature of legged locomotion is still accounted for.

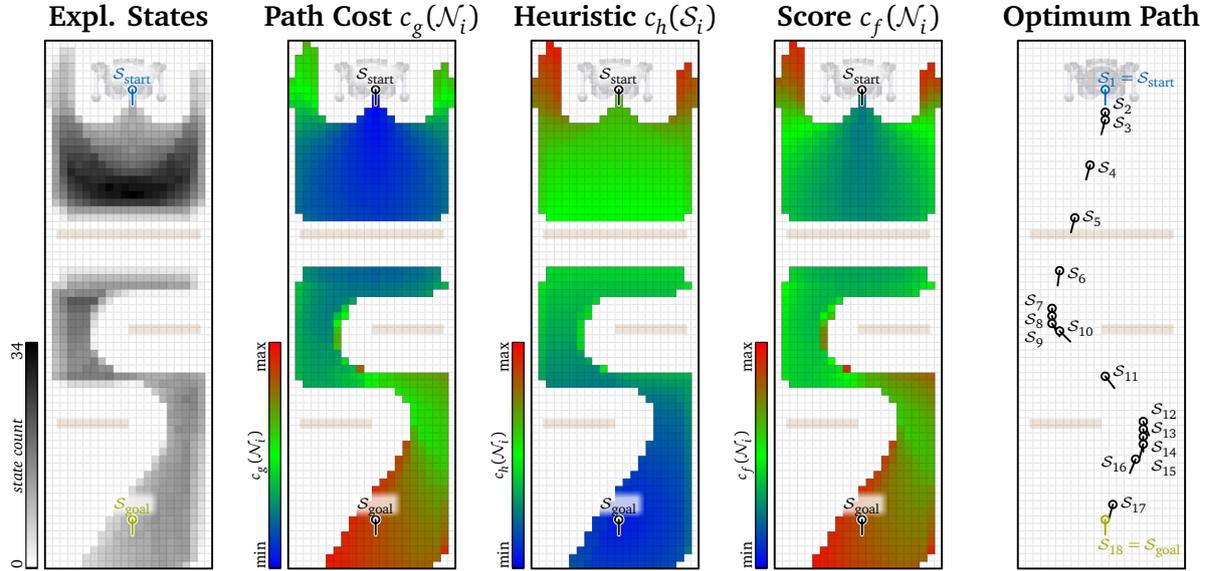


Figure 5.10: Exemplary results of the first level A^* search for the scenario shown in Figure 5.6 and [18 @t=7m54s]. From left to right: map showing the count of explored (valid) A^* states for each terrain cell; path costs $c_g(\mathcal{N}_i)$, heuristic $c_h(\mathcal{S}_i)$, and score $c_f(\mathcal{N}_i)$ (minimum of corresponding terrain cell); optimum path described as sequence of states $\{\mathcal{S}_i\}$ connecting \mathcal{S}_{start} and \mathcal{S}_{goal} . According to the search space discretization specified in Section 5.5.1, each terrain cell has a size of $8\text{ cm} \times 8\text{ cm}$ and represents $48 (= 360^\circ / 7.5^\circ)$ individual A^* states. For an animated visualization showing the progress of the node expansion, see [18 @t=2m5s].

For the depicted example scenario, the solution of the first level search is obtained in about 42 ms ($\sigma = 165\ \mu\text{s}$) on average⁷². In particular, the A^* search involves 9,646 explored nodes, 17,444 invalid states, and 22 invalid transitions. Moreover, we count $> 10^5$ calls to the openList container ($9,646 \times \text{insert}()$, $8,654 \times \text{extractMin}()$, $3,851 \times \text{update}()$, $98,341 \times \text{find}()$) and $> 3 \cdot 10^5$ calls to the closedSet container ($26,097 \times \text{insert}()$, $340,160 \times \text{contains}()$).

Rasterization In order to convert the discrete solution of the first level search into a continuous “guide path”, we compute a C^2 -continuous cubic spline (with zero second-order derivatives at the boundaries) interpolating the first level solution states shown in Figure 5.10 right. The resulting path is called *solution spline* (Figure 5.11, left) and will be used to accelerate the second level A^* search by affecting the state costs $c_s(\mathcal{S}_i)$ and heuristics $c_h(\mathcal{S}_i)$. For this purpose, we define four individual *local* metrics which depend on the *test* point \mathbf{r}_{test} and its *closest* point \mathbf{r}_{clo} on the solution spline. Both, \mathbf{r}_{test} and \mathbf{r}_{clo} , lie in the horizontal plane of the vision world FoR. In order to efficiently determine \mathbf{r}_{clo} , we approximate the solution spline by splitting each segment of the cubic spline into ten interconnected line segments.

⁷²For analyzing the runtime of WPG components, we perform 200 (identical) simulations of the considered scenario. After trimming the slowest and fastest 10%, we compute the arithmetic mean of the remaining 160 runtime samples. The simulations are run on the *Operator PC* where we assign the highest OS priority to the single-threaded simulation process and pin it to a single CPU core to minimize cache misses. Moreover, SMT of the CPU is disabled to maximize reproducibility. The simulations are executed one after the other while the remaining system load is kept as low as possible. Time is measured using the method explained in Footnote 59. We stick to simulations since obtaining reliable measurements for the “real-world” runtime (using the onboard PC during walking experiments) is difficult due to the strong interference with other real-time applications running on the same platform.

The local metrics at \mathbf{r}_{test} are then given by

- the Euclidean *distance* between the test point \mathbf{r}_{test} and the closest point \mathbf{r}_{clo} ,
- the *direction* of the solution spline (tangent) at the closest point \mathbf{r}_{clo} ,
- the remaining *arc length* (accumulated line segment lengths) at the closest point \mathbf{r}_{clo} , and
- the remaining *rotation* (accumulated absolute values of relative angles between consecutive line segments) at the closest point \mathbf{r}_{clo} .

Although computing these metrics for every expanded A^* state of the second level search is possible, it would be very inefficient. Indeed, each terrain cell in the second level search represents $48 (= 360^\circ / 7.5^\circ) \times 2$ (left/right stance) = 96 individual A^* states which are all linked to the same discretized position \mathbf{r}_{test} . Since the metrics depend on \mathbf{r}_{test} but not on the orientation or stance, they are identical for all 96 states. Therefore, it is sufficient to evaluate the metrics only once for each terrain cell. For this purpose, the solution spline is *rasterized*, i. e., we setup a two-dimensional map storing the local metrics in each cell. The map can be interpreted as image, where each “pixel” represents a terrain cell of the second level search. The dimensions of the map are chosen such that they enclose the entire solution spline with an inner margin of 1.5 m. If an A^* state outside this area needs to be evaluated, the metrics have to be computed directly from the solution spline, i. e., without accelerated map lookup. However, since states which are too far away from the solution spline will be labeled invalid (details follow with the description of state costs), it is guaranteed that node expansion does not exceed the map boundaries.

Similar to rasterized images in computer graphics, the resulting map suffers from artifacts caused by the finite resolution of the map. In particular, segments of the solution spline which are not parallel to the x - or y -axis of the grid will have a staircase-like appearance in the map. In order to counteract this effect, we perform spatial anti-aliasing through supersampling. In particular, each map cell is split uniformly into four square subcells. The metrics are then evaluated at the centroids of this 2×2 pattern and simply combined by forming the arithmetic mean. An exception is made for the component storing the *direction* of the solution spline, for which we have to use the circular mean instead (proper wrapping of angles). Exemplary results for the acceleration map are presented in Figure 5.11.

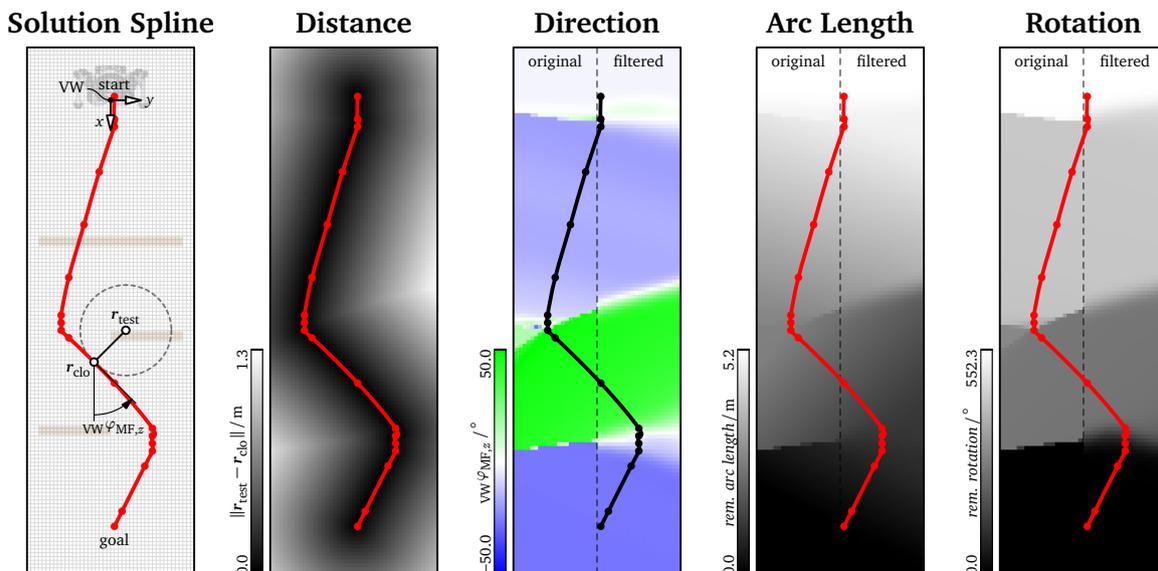


Figure 5.11: Rasterization of the solution spline (left) which interpolates the sequence of first level states forming the optimum path (cf. Figure 5.10 right). The generated map stores local metrics for each test point \mathbf{r}_{test} (discretized by $4\text{ cm} \times 4\text{ cm}$ cells alias “pixels”) such as the *distance* to the solutions spline and the *direction*, remaining *arc length*, and remaining *rotation* of the closest point \mathbf{r}_{clo} on the solution spline. The last three images are split into half where the left side shows the original data and the right side shows the final output after Gaussian filtering.

Independent of the shape of the solution spline, the distance field is guaranteed to be smooth. In contrast, the remaining three metrics (direction, arc length, and rotation field) show discontinuities at locations where the closest point on the solution spline is not unique. This leads to visible edges in the corresponding data fields. In order to blur discontinuities, we apply a Gaussian filter (standard deviation: $\sigma = 8$ cm, normalized kernel: $6\sigma \times 6\sigma$) to the direction, arc length, and rotation field. Since the direction field stores orientations, its Gaussian filter is modified to use the (weighted) circular mean of the samples. For the shown example scenario, the entire rasterization process takes 41 ms on average (map size: 100×190 cells).

Unfortunately, the interpolation by a cubic spline can lead to bumpy spline segments, especially between control points which lie close to each other (e. g. S_7 - S_{10} and S_{12} - S_{16} in Figure 5.10 right). While this has only a minor effect on the computed arc length, it can lead to a considerable overestimation of the remaining rotation. Within a potential future revision, one might consider to replace the cubic spline by a simple piecewise linear path instead.

2nd Level – States and Successors Within the second level A^* search, each state S_i is characterized by the current stance foot type (right / left) as well as its position ${}_{VW}r_{SF,x|y}$ and orientation ${}_{VW}\varphi_{SF,z}$ (discretized according to Section 5.5.1). Note that it is simple to derive the corresponding 2D pose of the MF frame used by the first level search and the acceleration map (rasterized solution spline). The operator $\text{succList}(\mathcal{N}_i)$ provides a set of potential successors to the second level state S_i which is defined through the following constraints:

- the stance foot type has to switch (restriction to biped walking),
- the step length in the sagittal plane is limited by $l_{x,\min} = 0$ and $l_{x,\max} = 64$ cm,
- the step length in the lateral plane (stepping “outside”) is limited by $l_{y,\max} = \pm 24$ cm,
- the step angle around the vertical axis is limited by $\varphi_{z,\min} = -15^\circ$, and $\varphi_{z,\max} = 15^\circ$, and
- the joint-space SSV models (cf. Figure 5.13 center) of the right foot (sfr+zfr for $q_{zfr} = 0$) and the left foot (sfl+zfl for $q_{zfl} = 0$) must have a minimum distance of 2 cm.

In order to forbid large diagonal steps (e. g. with $l_x = l_{x,\max}$ and $l_y = l_{y,\max}$ at the same time), we further setup the constraint $|l_x/l_{x,\max}| + |l_y/l_{y,\max}| \leq 1$ such that the permitted area for placing successors is triangle-shaped. It has to be highlighted that evaluating the aforementioned constraints (which involves costly collision detection between the rather complex SSV models of the left and right foot) comes with zero runtime cost since the operator $\text{succList}(\mathcal{N}_i)$ extracts the list of potential successors from pre-computed buffers (cf. Section 5.5.2).

2nd Level – State Costs Same as in the first level, the operator $\text{stateCost}(S_i)$ first checks if the terrain cell related to the state S_i has been revealed. Subsequently, the cluster of terrain cells resembling the foot-ground interface (single footprint of current stance assuming full contact) is analyzed. The state costs $c_s(S_i)$ are formulated as a linear function of the maximum height difference (lower is better) and the mean confidence value (higher is better) of all cells in the cluster. If the height difference exceeds 4 cm (bumpy terrain) or the mean confidence is less than 2/3 (uncertain regions), full contact is considered to be infeasible. In contrast to the first level, the state S_i might still be valid for partial (or tiptoe) contact. To find out, another cluster of terrain cells representing the foot-ground interface is assembled – now using the toe segment rather than the full foot sole as contact surface (cf. Figure 5.1). Again, we check the maximum height difference and the mean confidence value against the same validity thresholds. In order to avoid collisions of the heel with the ground, we additionally require the maximum cell height of the partial contact cluster to be greater or equal to the maximum cell height of the full contact cluster. If all constraints are satisfied, the contact is considered to be valid and the state costs are computed as linear function of the maximum height difference and the mean confidence with regard to the partial contact cluster. Furthermore, a constant penalty for partial contact is added such that full contact is preferred in general.

If the first level search was successful, the operator $\text{stateCost}(\mathcal{S}_i)$ additionally extracts the metrics *distance* and *direction* (using the position of the MF frame) from the acceleration map, cf. Figure 5.11. If the distance to the solution spline exceeds 15 cm or the error in the orientation exceeds 30° , the state is drawn invalid. The start state $\mathcal{S}_{\text{start}}$ and the goal states $\{\mathcal{S}_{\text{goal},i}\}$ are excluded from this check. Discarding states which deviate too much from the first level solution leads to a significant acceleration. Unfortunately, we also lose the guaranty of optimality. However, for most scenarios considered within this thesis, we still obtain the optimum while a suboptimal path – which still represents a valid solution – occurs only in rare cases. Note that in very complex scenarios, e. g. where the robot has to walk sideways to pass a narrow corridor, the first level search will fail such that the second level search delivers the optimum, however, without acceleration. See also [18] @t=9m20s for another example.

Feasibility of multi-contact configurations and collisions are evaluated similar to the first level. However, we further constrain that support with one or both hands is only allowed if the hand on the opposite side of the current stance foot is involved. This prevents situations where only the left or right side of the robot is in contact which is potentially less stable than without hand support. Multi-contact configurations are rewarded in the same way as within the first level. Similarly, collisions again draw the state invalid.

2nd Level – Transition Costs The operator $\text{tranCost}(\mathcal{N}_{\text{pre}}, \mathcal{S}_{\text{next}})$ starts with an evaluation of the traveled path length and the step length in vertical direction. Same as for the first level, the traveled path length is used to limit the step length in walking direction to 30 cm for the very first and last transition of a sequence. The step length in vertical direction is (again) limited to ± 13 cm. Within the second level search, we additionally limit the step length in walking direction to 50 cm for vertical step lengths exceeding 4 cm. This leads to shorter steps whenever the robot is about to step up or down (e. g. for climbing platforms or stairs).

The transition cost operator additionally analyses the terrain cells which are traversed by the swing foot. For assembling the corresponding cluster of terrain cells, one might be tempted to use the predecessor $\text{pred}(\mathcal{N}_{\text{pre}})$ as starting point of the swing foot motion. However, the A^* algorithm forbids transition costs which depend on predecessor states since predecessor links change whenever a less expensive path is found. For this reason, we consider only half of the swing foot motion, however, for the left and right foot simultaneously. In particular, we analyze the motion from \mathcal{S}_{pre} to $\bar{\mathcal{S}}_{\text{next}}$ and the motion from $\bar{\mathcal{S}}_{\text{pre}}$ to $\mathcal{S}_{\text{next}}$. Here, we introduced the “conjugate” A^* state $\bar{\mathcal{S}}_i$ which is equivalent to \mathcal{S}_i but with opposite stance foot and ${}_{\text{VW}}r_{\text{SF},x|y}$ shifted accordingly (assuming default foot separation in lateral direction). For the considered walking patterns, this approach delivers a sufficiently accurate approximation. The identified terrain cell clusters (one for each half) are then used to determine the height of traversed obstacles. Same as within the first level, we define a maximum obstacle height of 15 cm. Moreover, we limit the step length in walking direction to 50 cm for obstacles taller than 4 cm.

Once these validity checks are passed, the transition costs $c_t(\mathcal{N}_{\text{pre}}, \mathcal{S}_{\text{next}})$ are computed in (almost) the same manner as for the first level. Notable specifics are the use of the mean foot TCP frame MF to determine the traveled path length, the introduction of linear costs depending on the step length in the lateral plane (walking sideways was not considered in the first level), and individual obstacle traversal costs for each half of the swing foot motion.

2nd Level – Heuristic The operator $\text{heur}(\mathcal{S}_i)$ begins with an estimation of the remaining path length and rotation for the state \mathcal{S}_i . If the first level search was successful, we directly extract the corresponding metrics *arc length* and *rotation* (using the position of the MF frame) from the acceleration map, cf. Figure 5.11. Otherwise, we proceed like in the first level: the remaining path length is estimated through the minimum Euclidean distance between \mathcal{S}_i and $\{\mathcal{S}_{\text{goal},i}\}$ (again using the MF frame) while the remaining rotation is approximated by the minimum relative rotation between \mathcal{S}_i and $\{\mathcal{S}_{\text{goal},i}\}$. Apart from this, $c_h(\mathcal{S}_i)$ is computed in the exact same manner

as within the first level search. By using the pre-computed metrics from the acceleration map, we get a (in most cases) better approximation of the remaining path costs which further accelerates the second level search. Unfortunately, it is not guaranteed that this enhanced heuristic is a lower bound for the true remaining optimum path costs under all circumstances. However, same as for the definition of second level state costs, we prefer real-time performance over strict optimality – especially because the “optimum” strongly depends on the (empirically chosen) weighting of costs.

2nd Level – Results Figure 5.12 presents the results of the second level search for our exemplary scenario. It is clearly visible how the proposed acceleration technique limits node expansion to the close vicinity of the solution spline. Note that even if the first level fails, the second level still might find a valid solution. However, except for very simple scenarios, real-time performance will be lost.

While the first level search always plans the full path towards the user-specified goal, the second level search can be limited to a certain “planning horizon” – in our case implemented as a user-defined maximum count of future steps. For this, we simply set the goal of the second level to the corresponding state in the first level’s optimum path. For computing the heuristic, we additionally need to subtract a constant offset from the metrics *remaining arc length / rotation* (cf. Figure 5.11) to account for the discarded trailing part of the solution spline. The planning horizon is then cyclically updated through our dynamic replanning approach (demonstrated in the video [18 ●@t=10m28s]). Restricting the second level search to a certain planning horizon allows to maintain real-time performance even for planning over long distances.

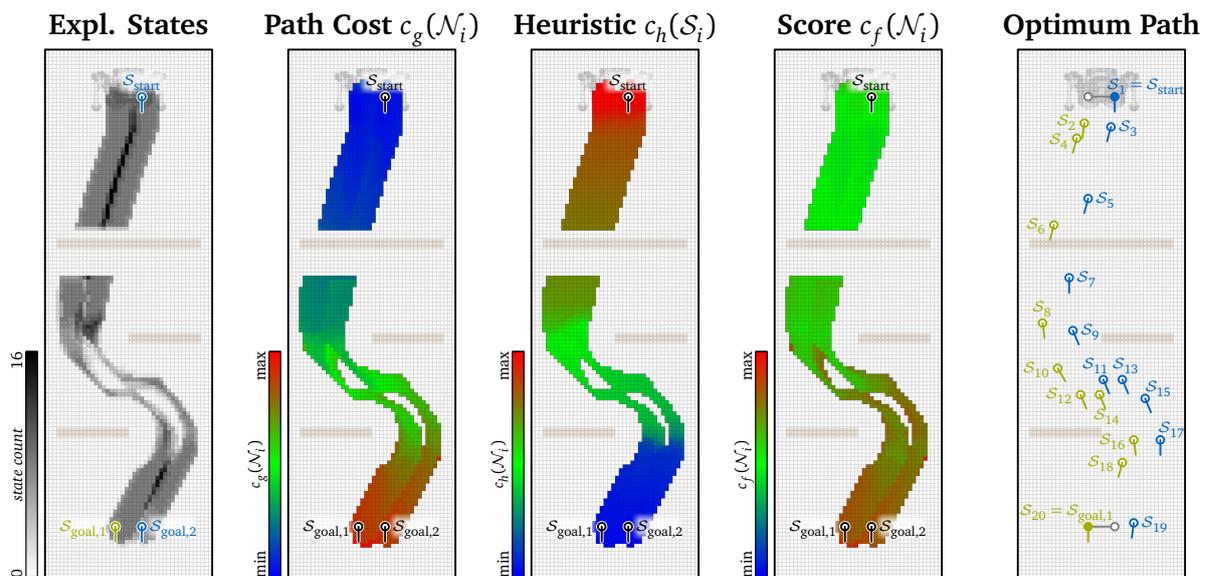


Figure 5.12: Exemplary results of the second level A* search for the scenario shown in Figure 5.6 and [18 ●@t=7m54s]. From left to right: map showing the count of explored (valid) A* states for each terrain cell; path costs $c_g(\mathcal{N}_i)$, heuristic $c_h(S_i)$, and score $c_f(\mathcal{N}_i)$ (minimum of corresponding terrain cell); optimum path described as sequence of states $\{S_i\}$ connecting S_{start} and $\{S_{\text{goal},i}\}$ (right / left stance). According to the search space discretization specified in Section 5.5.1, each terrain cell has a size of 4 cm \times 4 cm and represents 48 ($= 360^\circ / 7.5^\circ$) \times 2 (left/right stance) = 96 individual A* states. For an animated visualization showing the progress of the node expansion, see [18 ●@t=2m26s].

For the depicted example scenario, the solution of the second level search is obtained in 632 ms ($\sigma = 1.19$ ms) on average. In particular, the A* search involves 9,863 explored nodes, 42,952 invalid states, and 4,343 invalid transitions. Moreover, we count $> 4 \cdot 10^5$ calls to the openList container ($9,863 \times \text{insert}()$, $9,320 \times \text{extractMin}()$, $18,038 \times \text{update}()$, $365,278 \times \text{find}()$) and $> 2 \cdot 10^6$ calls to the closedSet container ($52,271 \times \text{insert}()$, $2,235,124 \times \text{contains}()$). The acceleration map storing the rasterized solution spline is queried 53,508 times.

If the second level is triggered *without* acceleration by the first level, the search takes 23.7 s ($\sigma = 45.3$ ms) on average. Consequently, the proposed hierarchical contact planning approach leads to a more than 33 times speedup in this particular scenario. With disabled acceleration, the second level A* search involves $> 2 \cdot 10^5$ explored nodes, $> 10^7$ calls to the openList container and $> 4 \cdot 10^7$ calls to the closedSet container. The maximum count of (simultaneously) stored nodes in the openList container is 28,268. The presentation of these figures is meant to highlight the importance of an efficient implementation of the node containers. Moreover, the greatest count of calls is related to the methods find() or equivalently contains(), which strongly benefit from an efficient state hash function, cf. Section 5.5.1.

5.5.5 Post-Processing

The last stage of the contact planning pipeline (cf. Figure 5.6) is given by the post-processor, which is itself organized as chain of three consecutive subroutines responsible for

- computing the final 6D pose of footholds based on local terrain information using the maximum available resolution (1 cm \times 1 cm discretization),
- finding optimal contact points for hand support based on the pre-processed surface model of environmental objects and the multi-contact target volume, and
- converting the refined sequence of states into a corresponding chain of QPWTs.

The generated sequence of QPWTs is processed further by the *Transition Planner* in the exact same way as it is done for fixed sequence or teleoperated walking, see Figure 5.5.

Foothold Refinement Each state S_i of the second level's optimum path $\{S_i\}$ stores the type and planar pose (${}_{\text{VW}}r_{\text{SF},x|y}$, ${}_{\text{VW}}\varphi_{\text{SF},z}$) of the corresponding stance foot. Within foothold refinement, this 2D pose is handled as rough initial guess which shall be refined to a full 6D pose specifying the final position and orientation of the foot's TCP frame. This is done by aligning the foot with the terrain, where we allow a small shift in the horizontal plane to find an optimal fit. In particular, we run an alignment procedure which can be summarized by the following steps:

1. Setup a search space which is specified by the parameters $\Delta_{\text{VW}}r_{\text{SF},x|y} \in \{-2, -1, 0, 1, 2\}$ cm and $\Delta_{\text{VW}}\varphi_{\text{SF},z} \in \{-3.75, 0, 3.75\}$ ° describing the offset from the initial 2D pose provided by the state S_i , see Figure 5.13 top-left. The parameter ranges have been chosen such that the input pose is refined using the maximum resolution as specified in Section 5.5.1.
2. Pick a not yet evaluated parameter set ($\Delta_{\text{VW}}r_{\text{SF},x|y}$, $\Delta_{\text{VW}}\varphi_{\text{SF},z}$) from the search space and compute the corresponding refined 2D pose.
3. Detect self-collisions between the feet using the joint-space SSV models of the right foot ($\text{sfr} + \text{zfr}$ for $q_{\text{zfr}} = 0$) and the left foot ($\text{sfl} + \text{zfl}$ for $q_{\text{zfl}} = 0$), see Figure 5.13 center. The SSVs are positioned according to the refined stance foot pose of the currently investigated state S_i and its predecessor S_{i-1} . If the distance is less than 1 cm, the configuration is considered as invalid and we switch to the next parameter set (repeat starting from Step 2).
4. Assemble the cluster of terrain cells representing the foot-ground interface under assumption of full contact (like it was done during the second level A* search, see Figure 5.13 bottom-left). If full contact is not feasible, i. e., if the maximum height difference is greater than 4 cm or the mean confidence is less than 2/3, then assemble the cluster of terrain cells for partial contact and perform another feasibility test with additional check for heel-ground collision (maximum cell height for partial contact must be greater or equal than for full contact). If even partial contact is not feasible, the configuration is considered as invalid and we switch to the next parameter set (repeat starting from Step 2).

5. Based on the cluster of terrain cells, compute costs depending on the maximum height difference and the mean confidence similar to the state costs of the second level A^* search. Same as before, a constant penalty is added in case partial contact is necessary (to prefer full contact). Moreover, we add costs which depend on the (absolute) offset from the initial pose (to prefer small shifts $\Delta_{VW} r_{SF,x|y}$, $\Delta_{VW} \varphi_{SF,z}$).
6. If the search space has not been fully evaluated yet, repeat starting from Step 2. Otherwise, determine the optimum based on the computed costs and focus on this parameter configuration for the rest of the alignment procedure while discarding all other candidates.
7. Based on the previously computed terrain cell cluster (full or partial contact as identified in Step 4) related to the optimum parameter set, generate a point cloud where each point corresponds to a cell of the cluster. Each point's horizontal position is given by the centroid of the related cell while the vertical position is specified by the cell's height value. Afterwards, perform a *Principal Component Analysis (PCA)* (through an SVD) to approximate the point cloud by the *contact plane* $\mathcal{P}_{\text{cont}}$, see Figure 5.13 right.
8. Compute the final 6D pose of the stance foot's TCP frame using the refined 2D pose (initial pose with applied optimum offset) for the horizontal position and vertical rotation. The vertical position is obtained by projecting the refined 2D position onto the plane $\mathcal{P}_{\text{cont}}$ (such that the TCP lies on $\mathcal{P}_{\text{cont}}$). The rotation around the horizontal axes is chosen according to the normal \mathbf{n}_{cont} of $\mathcal{P}_{\text{cont}}$ (such that the x - and y -axis of the TCP frame lie in $\mathcal{P}_{\text{cont}}$).
9. In addition to the 6D pose of the stance foot, we also store its contact type (full or partial) from Step 4. Moreover, in order to reduce the likelihood of hitting kinematic limits, we check if the robot is about to take a large step down (current stance's TCP more than 10 cm below previous stance's TCP) and force *tiptoe* contact in this case (cf. [20] @ $t=1m54s$).

This procedure is repeated for every state S_i of the second level's optimum path $\{S_i\}$ – except for $S_1 = S_{\text{start}}$, which will be replaced anyway by the end pose of the last unchanged transition in the motion plan. Note that the involved optimization (Step 1 to 6) uses a brute-force approach since the search space contains only $5 \times 5 \times 3 = 75$ possible parameter configurations. For the sequence of 20 footsteps in our example scenario (cf. Figure 5.12 right), the entire footstep refinement subroutine takes less than 14 ms on average. Similar to the first and second level A^* searches, we make extensive use of pre-computed, scenario independent buffers, e.g. for efficiently assembling terrain cell clusters. Among the scenarios considered within this thesis, foothold alignment is most relevant for climbing ramps as shown in the video [18] @ $t=8m27s$.

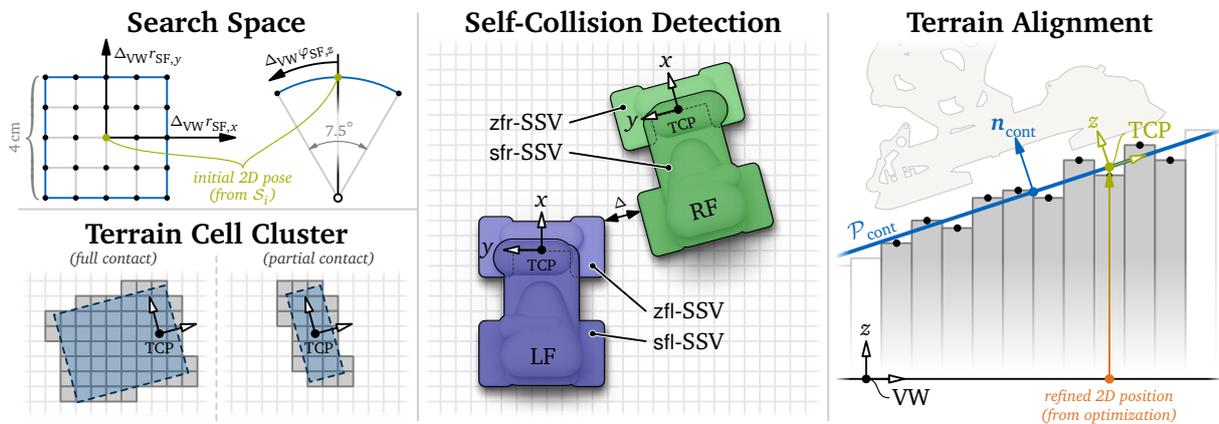


Figure 5.13: Foothold refinement as part of the post-processing stage of the contact planning pipeline. From left to right: search space describing admissible planar offsets $\Delta_{VW} r_{SF,x|y}$, $\Delta_{VW} \varphi_{SF,z}$ during optimization; assembly of terrain cell clusters for full and partial contact (rasterization of foot geometry); self-collision detection based on the joint-space SSV models of the feet (assuming $q_{z|tr|} = 0$); alignment of the foot's TCP frame to the contact plane $\mathcal{P}_{\text{cont}}$.

Optimum Hand Contact Points Within the previous stages of the contact planning pipeline, multi-contact situations have been considered through the definition of corresponding cost terms. In particular, feasibility of hand support has been rewarded by a decrease of the state costs of the first and second level’s A^* search. This way, paths have been preferred which bring the robot close to environmental objects suitable for multi-contact. However, concrete contact points for hand support have not been considered yet. Indeed, the post-processor is responsible of finding optimal contact points depending on the robot’s pose and the geometry of close-by environmental objects. Moreover, the post-processor makes the final decision if a hand contact should be made or not.

First of all, we have to formally define the “optimum”. For this purpose, we specify a cost function, which depends on the local surface confidence $c_{\text{surf}} \in [0, 1]$ of the considered object (evaluated during pre-processing, cf. Section 5.5.2) and the distance l_{dist} to the multi-contact target volume (cf. Figure 5.3). In particular, we introduce the multi-contact cost c_{mult} as

$$c_{\text{mult}}(c_{\text{surf}}, l_{\text{dist}}) := \begin{cases} \frac{3}{2}(1 - c_{\text{surf}}) + 5l_{\text{dist}} & \text{if } c_{\text{surf}} \geq \frac{1}{3} \text{ and } l_{\text{dist}} \leq l_{\text{dist,max}} , \\ \infty & \text{else} \end{cases} \quad (5.11)$$

where the second branch is used to discard individual triangles or entire objects which do not fulfill the minimum requirements for multi-contact. The length l_{dist} is computed as the minimum distance between two SSVs: the multi-contact target volume and either the entire volume model (cf. Figure 4.6 right) or a single point/triangle (represented by a corresponding point/triangle-SSV, see Figure C.1) of an object. Note that l_{dist} becomes negative in the case of penetration. Since the multi-contact target volume approximates the reachable workspace of the TCP, i. e., the center of the hand, we choose $l_{\text{dist,max}} = d_h/2$ (hand radius) to check if the surface of the hand is capable of touching an object’s volume model. Similarly, if we evaluate the distance of the multi-contact target volume to an individual point/triangle, we use $l_{\text{dist,max}} = 0$ and set the SSV radius of the point/triangle to $d_h/2$ instead. According to Equation 5.11, a contact point is ideal (minimum cost) if $c_{\text{surf}} = 1$ and it lies on the base geometry of the multi-contact target volume (the two lines shown in Figure 5.3). Conversely, a (valid) contact point has maximum cost if $c_{\text{surf}} = 1/3$ and it is just about reachable ($l_{\text{dist}} = l_{\text{dist,max}}$).

Based on the cost c_{mult} , the post-processor tries to find optimal contact points for each state \mathcal{S}_i of the second level’s optimum path $\{\mathcal{S}_i\}$. Again, we skip $\mathcal{S}_1 = \mathcal{S}_{\text{start}}$ and perform the same procedure for each state \mathcal{S}_i until we reach $\mathcal{S}_{\text{goal}}$. For locating the robot and its attached multi-contact target volume relative to environmental objects, we use the 6D stance foot poses computed during foothold refinement (without rotation around the horizontal axes since we assume an upright torso). For each state \mathcal{S}_i , we evaluate contact with the right and left hand individually. However, same as for the second level A^* search, we stipulate that support with one or both hands is only allowed if the hand on the opposite side of the current stance foot is involved. In addition, once a hand gets in contact, its contact configuration (position, force, etc.) is copied to the subsequent state \mathcal{S}_{i+1} . This ensures, that the hand remains in contact during the entire transition from \mathcal{S}_i to \mathcal{S}_{i+1} . Note that for the feet this behavior is implicitly dealt with through introducing the concept of a “stance” foot.

Depending on the complexity of the scene, the environment model may contain a large number of objects. Instead of a costly “full” evaluation of all objects, we first compute the minimum distance between an object’s volume model and the multi-contact target volume. Since we assume that the entire surface of an object is encapsulated by its volume model (cf. Section 4.5.1), this delivers a cheap lower bound for l_{dist} . Together with the object’s global maximum of c_{surf} (determined during pre-processing), Equation 5.11 allows us to compute a lower bound of c_{mult} for each object. In case $c_{\text{mult}} = \infty$, the entire surface is guaranteed to be infeasible for multi-contact, thus, the corresponding object can be safely ignored. Indeed, this limits our search to a very small subset of objects which are in the close vicinity of the robot. Furthermore, the remaining objects are arranged in a list which is sorted according to the computed lower bound

of c_{mult} . The iteration of this list starts with the most promising object which further accelerates the search: once a valid contact point is found, we compare its c_{mult} against the lower bound related to the next object in the list and – in case c_{mult} undercuts this lower bound – we can safely stop the iteration without risking to miss the optimum.

In order to find the optimum contact point for a given object, we iterate over all triangles of the corresponding surface model. In a first step, we check if the triangle's normal points into a feasible direction, i. e., towards the upper body of the robot. For this purpose, we specify valid ranges for the direction of the normal in the frontal and transverse plane of the UB FoR, see Figure 5.14 left. While this represents a rather simple classification into valid and invalid triangles, one may consider to introduce a cost term depending on the normal as part of a future revision (which would allow to penalize or reward certain directions).

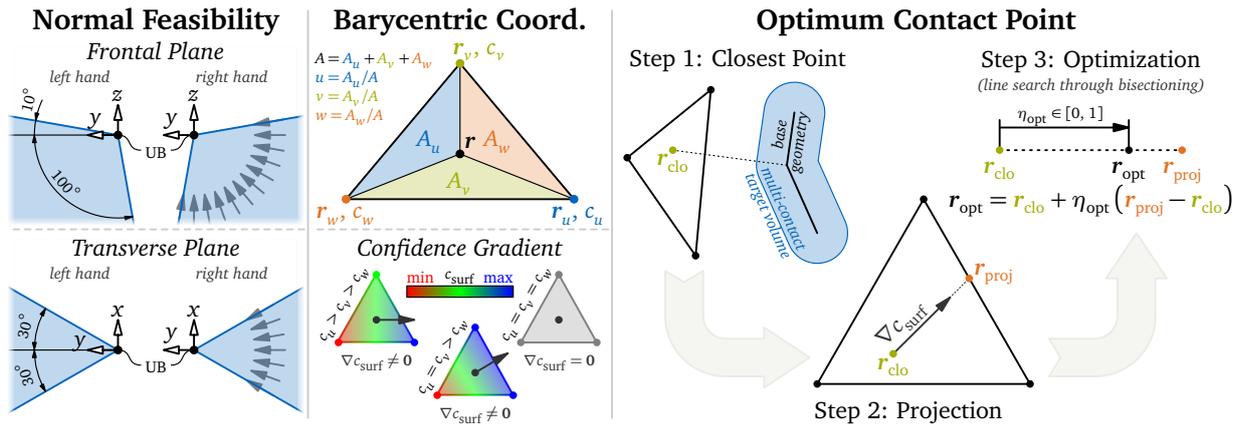


Figure 5.14: Searching for optimum hand contact points as part of the post-processing stage of the contact planning pipeline. From left to right: feasible directions (blue) of the triangle's normal in the frontal (top) and transverse (bottom) plane; linear interpolation of c_{surf} using barycentric coordinates (top) and the corresponding confidence gradient ∇c_{surf} (bottom) with $\{r_{c,\text{max}}\}$ as the set of points with maximum confidence for three example configurations; finding the optimum through computing (1) r_{clo} as the point closest to the multi-contact target volume, (2) r_{proj} as the projection of r_{clo} onto the triangle's boundary (along ∇c_{surf}), and (3) r_{opt} as the optimum contact point lying on the connection line between r_{clo} and r_{proj} . For an animated visualization, see the video [18] @t=3m34s].

For each triangle with feasible normal direction, we search for the point on the triangle which minimizes the multi-contact cost $c_{\text{mult}}(c_{\text{surf}}, l_{\text{dist}})$ as specified in Equation 5.11. For describing the location of points on the triangle, a barycentric coordinate system is used:

$$\mathbf{r}(u, v) := u \mathbf{r}_u + v \mathbf{r}_v + w \mathbf{r}_w \quad \text{with} \quad u, v, w \geq 0 \quad (\in [0, 1]) \quad \text{and} \quad u + v + w = 1, \quad (5.12)$$

where $\mathbf{r}_{u|v|w} \in \mathbb{R}^3$ represent the vertices of the triangle and $u, v, w \in \mathbb{R}$ the corresponding barycentric coordinates as ratios of the triangle's total area A , see Figure 5.14 center-top. In order to compute $c_{\text{mult}}(u, v)$, we have to evaluate the local surface confidence $c_{\text{surf}}(u, v)$ and the distance to the multi-contact target volume denoted by $l_{\text{dist}}(u, v)$. For c_{surf} , we choose a linear interpolation scheme based on the barycentric coordinates:

$$c_{\text{surf}}(u, v) := u c_u + v c_v + w c_w = c_w + \nabla c_{\text{surf}}^T \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{with} \quad \nabla c_{\text{surf}} := \begin{bmatrix} \frac{\partial c_{\text{surf}}}{\partial u} \\ \frac{\partial c_{\text{surf}}}{\partial v} \end{bmatrix} = \begin{bmatrix} c_u - c_w \\ c_v - c_w \end{bmatrix} \quad (5.13)$$

where $c_{u|v|w}$ are the per-vertex confidence values and ∇c_{surf} is the confidence gradient which is constant throughout the entire triangle. Both, $c_{u|v|w}$ and ∇c_{surf} , are computed during the pre-processing stage, cf. Section 5.5.2. Since c_{surf} is linear, the set of points with maximum confidence $\{r_{c,\text{max}}\}$ either contains a single vertex, an edge, or the entire triangle depending on the particular values of $c_{u|v|w}$, see Figure 5.14 center-bottom. In any case, the global maximum of c_{surf} is simply given by $\max(\{c_u, c_v, c_w\})$ and has to be greater or equal to the threshold $1/3$ (cf. Equation 5.11), otherwise the entire triangle is discarded.

For evaluating $l_{\text{dist}}(u, v)$, we consider a point-SSV element with radius $d_h/2$ at the position $\mathbf{r}(u, v)$ and compute the minimum distance to the multi-contact target volume. Since the multi-contact target volume represents an SSV segment, i. e., a compound of multiple SSV elements, the distance l_{dist} is highly non-linear and only C^0 -continuous (shortest connection might jump). Instead of formulating a non-linear optimization problem, we stick to a rather pragmatic process to find an approximate solution for the optimum contact point \mathbf{r}_{opt} which minimizes c_{mult} (see Figure 5.14 right):

1. Compute \mathbf{r}_{clo} as the point on the triangle which is closest to the base geometry of the multi-contact target volume (represents the global optimum with regard to l_{dist}). If the corresponding distance exceeds $l_{\text{dist,max}}$ (cf. Equation 5.11), the entire triangle is discarded.
2. Compute \mathbf{r}_{proj} as the projection of \mathbf{r}_{clo} along ∇c_{surf} onto the triangle's boundary. While \mathbf{r}_{proj} has a greater confidence than \mathbf{r}_{clo} , it does not necessarily have maximum confidence. Instead, one could compute \mathbf{r}_{proj} by projecting \mathbf{r}_{clo} onto $\{\mathbf{r}_{c,\text{max}}\}$, however, the resulting point would be further away from \mathbf{r}_{clo} (potentially leading to a higher l_{dist}). In the special case of $c_u = c_v = c_w$ such that $\nabla c_{\text{surf}} = \mathbf{0}$, the optimum is simply given by $\mathbf{r}_{\text{opt}} = \mathbf{r}_{\text{clo}}$.
3. In case $\nabla c_{\text{surf}} \neq \mathbf{0}$, compute \mathbf{r}_{opt} through bisecting the connection line between \mathbf{r}_{clo} and \mathbf{r}_{proj} (line search). For each sample, evaluate c_{mult} as specified above (with c_{surf} from Equation 5.13 and l_{dist} from the SSV distance) and abort the iteration upon convergence.

The presented scheme delivers a computationally efficient approximation of the optimum contact point. Due to the bounded triangle size (ensured by the adaptive subdivision, cf. Section 5.5.2), the error remains small. From the optimum contact point \mathbf{r}_{opt} and the triangle's normal \mathbf{n} , we directly obtain the optimum position of the hand's TCP frame $\mathbf{r}_h = \mathbf{r}_{\text{opt}} + (d_h/2)\mathbf{n}$. To minimize the risk of slipping, we assume point-contact without tangential force components such that the desired external contact wrench is given by $\mathbf{W}_{h,\text{ext}}^h = [\mathbf{F}_{h,\text{ext}}, \mathbf{T}_{h,\text{ext}}]^T$ with $\mathbf{F}_{h,\text{ext}} = F_{h,\text{ext}}\mathbf{n}$ and $\mathbf{T}_{h,\text{ext}} = \mathbf{0}$. Currently, the magnitude of the contact force $F_{h,\text{ext}}$ is controlled by a user-defined parameter. Within the scope of this thesis, a constant value of $F_{h,\text{ext}} = 50\text{ N}$ is used. As a future improvement, one might consider to set $F_{h,\text{ext}}$ automatically depending on the object's classification or not yet considered surface properties such as the (estimated) friction coefficient.

Conversion to QPWTs The very last step of the post-processor is the conversion of the solution to a corresponding sequence of QPWTs. In particular, we create a QPWT for each pair of (refined) states \mathcal{S}_i and \mathcal{S}_{i+1} and fill its components (specified in Table 5.2) according to the computed optimum. In particular, we obtain the step parameters $l_{x|y|z}$ and $\varphi_{x|y|z}$ from the relative transform between the previous and next stance foot pose using the formulas provided in Appendix D. Subsequently, we assemble the cluster of terrain cells (using the maximum terrain resolution such as during foothold refinement) representing the area traversed by the swing foot. In contrast to the second level A^* search, we have full knowledge of all preceding and following stance foot poses, thus, the start and end of the swing foot motion is precisely known (splitting the swing foot motion into halves is not necessary anymore). From the maximum cell height within this cluster, we obtain the height of traversed obstacles which is measured relative to the maximum vertical stance foot position. If this height exceeds 4 cm, we assume that an obstacle is present and has to be traversed by the swing foot. Consequently, we set the custom step height h_{step} to the obstacle height and add a vertical clearance of 4 cm.

From the vertical stance foot positions, we can detect if the robot is stepping up or down (using again a threshold of 4 cm). In either case (and also for stepping over obstacles), we reduce the likelihood of collisions by setting the swing foot timing factor $\tau_{\overline{\text{SF}}}$ to 0.5. Note that the influence of h_{step} and $\tau_{\overline{\text{SF}}}$ on the swing foot trajectory will be explained in Section 6.7. For these challenging maneuvers, we additionally slow down the motion by a factor of 1.5 through setting a corresponding custom transition duration $t_{\text{tra,dur}}$. The same slow-down factor is applied for transitions which involve multi-contact configurations.

The QPWT flags specifying the foot and hand contacts are set according to the results of the foothold refinement and the computation of optimum hand contact points. The same applies to the hand's position ${}_{SF}r_h$ and desired external wrench ${}_{SF}W_{h,ext}^h$. The remaining components of the QPWT, i. e., ${}_{SF}s_{UB}$, q_{vp} , q_{vt} , and $\Delta_W r_{CoM,z}$, are set to their defaults (cf. Table 5.2).

5.6 Results and Discussion

Within this chapter, a novel contact planner for multi-contact locomotion has been presented. The main focus was set on real-time performance which was achieved on the one hand through the use of rather coarse models describing the foot-ground interface, collision volumes, and the reachable workspace of the arms and on the other hand through a hierarchical search on multiple levels of detail. Within the WPG module, the task of contact planning is realized through the *Transition Planner*, which implements semi-autonomous (fixed sequence / teleoperated walking and special motions) and fully-autonomous locomotion. The output of the transition planner is stored in the form of a hierarchical tree structure, the *(Motion) Plan*, which represents an analytic description of the planned task-space motion and allows to describe a very broad spectrum of gaited and non-gaited motions. Moreover, the special structure of the tree allows an efficient online modification of the planned motion in parallel to its execution (dynamic replanning). For gaited motions, this chapter additionally introduced the concept of QPWTs, which are a very powerful tool for describing even highly complex multi-contact locomotion in a compact and human readable form. By loading a sequence of QPWTs from a simple text-based input file, recompilation⁷³ of source code becomes unnecessary in many cases which greatly simplifies the test of new scenarios and locomotion patterns within both, simulation and real-world experiments. The focus of this chapter was clearly set on contact planning for fully-autonomous locomotion for which LOLA's capabilities have been extended significantly. The videos [17] and [18] show multiple scenarios which highlight the new primary skills. While a general discussion of the conducted simulations and experiments is given in Chapter 8, this section focuses on a runtime analysis of the contact planner.

Runtime Analysis For semi-autonomous locomotion, the entire contact planning process takes less than 1 ms which turns out to be marginal when compared to the computational cost of motion generation. Thus, this paragraph focuses on fully-autonomous locomotion where contact planning represents (by far) the computational most expensive part of the entire planning pipeline. In particular, we consider nine individual scenarios where the robot is commanded to traverse a certain environment autonomously (action *autonomous walking*, cf. Section 5.2). With regard to the WPG, all scenarios are handled in the same way, i. e., with the same set of configuration parameters. The only difference is the environment model and the commanded goal position ${}_{VW}r_{MF,x}$ (while ${}_{VW}r_{MF,y} = {}_{VW}\varphi_{MF,z} = 0$). For evaluating the runtime of the contact planning pipeline, we conduct a series of simulations as described in Footnote 72. The results of this analysis are shown in Table 5.4.

As expected, the runtime of the pre-processor scales with the size of the environment model, i. e., the count of terrain patches (building quadtrees) and objects (surface evaluation). Note that the environment model pre-processing is run in parallel by the environment model manager (*Thread 8*, cf. Figure 4.11) and thus, does not directly add to the total runtime of the contact planning pipeline. The main workload is given by the second level A* search which, however, gets significantly accelerated in case the first level search was successful (which applies to all scenarios except for “Stairs – Partial”). The total speedup factor (second level only vs. combined first level, rasterization, and second level) reaches up to 64.5. However, this peak value is

⁷³On the *Operator PC*, compilation of LOLA's source code takes more than 200 s (using 32 parallel threads).

Table 5.4: Analysis of the contact planning pipeline (cf. Figure 5.6) for multiple example scenarios demonstrating the autonomous locomotion capabilities of LOLA. From top to bottom: general parameters describing the scenario; pre-processing as described in Section 5.5.2 (green); hierarchical graph search as described in Section 5.5.4 (yellow); post-processing as described in Section 5.5.5 (orange); overall results (blue) of the contact planning (bold) and motion generation process. Runtimes are determined following to procedure described in Footnote 72. For the 2nd level A* search, the tag “acc.” indicates the runtime with acceleration by the 1st level while the tag “raw” denotes the runtime without acceleration by the 1st level (used to compute the total speedup factor). Note that the 1st level search fails in the particular scenario “Stairs – Partial”, thus, the 2nd level lacks of acceleration in this case.

Scenario \ Parameter	Platform [18 @ t=5ms]s]	Right Wall [18 @ t=6ms]s]	Right Table [18 @ t=7ms]s]	Corridor [18 @ t=7ms]s]	Obstacles [18 @ t=7ms]s]	Ramps [18 @ t=8ms]s]	Stairs – Full [18 @ t=8ms]s]	Stairs – Partial [18 @ t=9ms]s]	Trap [18 @ t=9ms]s]
goal $p_W r_{MF,x}$ / m	4.0	3.0	3.0	3.0	4.5	4.5	3.5	3.5	4.5
terrain patch count	50	49	48	46	50	30	30	30	72
object count	3	1	1	2	3	3	2	2	4
pre-pro.: terrain / ms	2.2	2.1	2.2	1.9	2.2	0.9	1.0	1.0	3.9
pre-pro.: objects / ms	14.5	5.7	2.6	11.3	2.7	1.5	5.0	4.2	4.9
pre-pro.: total / ms	16.7	7.8	4.8	13.2	4.9	2.5	6.0	5.2	8.8
1 st lvl.: expl. nodes	5,496	9,413	8,972	5,397	9,646	3,888	2,122	10,012	37,280
1 st lvl.: A* -search / ms	28.0	54.9	52.1	35.3	42.1	12.9	6.6	27.4	202.2
1 st lvl.: success	yes	yes	yes	yes	yes	yes	yes	no	yes
rasterization / ms	17.8	13.0	13.0	11.9	41.4	20.4	15.1	–	53.7
2 nd lvl.: expl. nodes	7,085	9,026	7,272	7,969	9,863	11,169	6,837	75,580	11,984
2 nd lvl.: A* (acc.) / ms	587.8	1,264.8	817.8	933.3	631.7	1,005.9	490.6	–	682.9
2 nd lvl.: A* (raw) / s	10.2	14.5	11.8	4.8	23.7	7.6	4.8	7.2	60.5
total speedup factor	16.1	10.9	13.4	4.9	33.1	7.3	9.4	–	64.5
post-pro.: feet / ms	3.9	2.5	3.6	3.1	13.4	4.6	4.2	4.4	28.7
post-pro.: hands / ms	0.5	0.2	0.1	0.5	–	–	–	–	–
post-pro.: QPWT / ms	0.9	1.4	1.1	0.7	4.4	1.0	0.9	0.9	6.2
post-pro.: total / ms	5.3	4.1	4.8	4.4	17.8	5.7	5.1	5.2	34.9
contact planner / ms	639.0	1,337.0	887.9	985.0	733.3	1,044.9	517.6	7,225.8	974.0
motion generator / ms	3.0	1.9	1.9	2.1	5.9	3.0	3.4	3.6	7.0
total runtime / s	0.642	1.339	0.890	0.987	0.739	1.048	0.521	7.229	0.981
executed motion / s	12.3	8.7	8.7	9.5	18.7	12.3	13.5	13.1	21.1
transition count	12	9	9	10	22	13	13	13	26
involves multi-contact	yes	yes	yes	yes	no	no	no	no	no

achieved in the scenario “Trap” which is explicitly designed to highlight the benefit of a guiding path. For more realistic scenarios, one can expect a tenfold speedup. The runtime of post-processing mainly depends on the count of transitions and the complexity of environmental objects (triangle count). Table 5.4 also shows that – for autonomous walking – the task of motion generation has an almost negligible runtime when compared to contact planning. Finally, by setting the total runtime of the planning pipeline (typically around 1 s if the first level search is successful) in relation to the duration of the generated motion, the proposed contact planner can be considered to be real-time capable. Under assumption of a sufficiently fast CV system, the proposed WPG is capable of generating new motion plans with a frequency of around 1 Hz which enables the robot to also handle dynamic environments (e. g. with slowly moving obstacles).

Although the overall runtime still depends on the particular scene, it shows a much lower variance when compared to the previous step planner by HILDEBRANDT [201, p. 63ff, 74ff]. This renders the runtime of the planning pipeline much more predictable which – in addition to the significantly improved overall performance – represents a further major enhancement in particular with regard to real-time execution. A rather unintuitive property of the proposed navigation system is that a complex scene featuring a large count of obstacles does not necessarily imply a higher runtime. As an example, contact planning for the scenario “Platform” (1×platform,

2×walls, 2×hand contact, 4 m distance) finishes in less than half of the time when compared to the much simpler scenario “Right Wall” (1×wall, 1×hand contact, 3 m distance). This can be explained by the higher proportion of invalid A* states (terrain around edges of platform) which get evaluated only once and do not spawn any successors.

Limitations and Suggestions Currently, real-time contact planning for autonomous locomotion is limited to rather short paths of about 5 m length. For greater distances, one should either limit the planning horizon of the second level search (as described in Section 5.5.4), or consider to introduce a higher level search where the present contact planning pipeline is used to find the optimum contact sequence between two waypoints of the high-level path. Besides, there exist numerous possibilities to optimize the current implementation. Examples are the acceleration by (further) sacrificing optimality through a weighted A* approach (cf. Footnote 12), switching to an anytime A* variant such as ARA* as used in [210] or its modification ADA* as used in [209] to allow specifying a maximum planning time, or the implementation of a more adaptive action model similar to [114, p. 33ff] which reduces the set of successor states in simple regions with level ground and without obstacles. Another idea is to split the solution of the first level search into multiple segments for which individual (small) second level searches are triggered in parallel. However, since the present locomotion system of LOLA consisting of WPG, SIK, and HWL module already approaches the limits of the control PC – in particular with regard to parallel real-time threads (cf. Section 4.8) – this might require to upgrade the control PC or, even better, to port the contact planner and the environment model manager to the vision PC. With the sequence of QPWTs as primary output of the contact planner, an appropriate exchange format for transmission over the *Ethernet* link already exists.

Besides optimizing real-time performance, future investigations might also further extend the contact planning capabilities. As an example, the search for optimal hand contact points on environmental surfaces does not consider collisions of the hand with other objects. Furthermore, the fitness of a contact point only depends on the perception confidence and pure geometric properties (feet: roughness of terrain; hands: local mesh curvature) but not on mechanical properties such as the (predicted) friction coefficient. Through extending the terrain and surface confidence specification accordingly, it would be simple to include additional local metrics. The current environment model formulation is very generic, i. e., it allows to describe scenes with arbitrary geometry. However, it seems reasonable to additionally introduce parameterized models for common object types. As an example, an object describing a staircase could – additional to its classification and surface / volume model – include a set of parameters which specify its geometry in an abstract way (step width, length, height, and count). In case the contact planner reaches this object, it could switch to a purely geometric-based algorithm specifically designed for stair traversal which is potentially much faster than a generic graph search and most likely will also generate a solution of higher quality. Certainly, the CV system would need to be capable of generating such parameterized models.

Software – Part C: Motion Generation

Parts of this chapter have already been published in [1–4].

Within this chapter, the motion generation system of LOLA v1.1 is described. Similar to the governing locomotion framework, the general workflow for motion generation has undergone more than two decades of evolution. Indeed, the present state is strongly based on the real-time trajectory generation system proposed by BUSCHMANN [100, p. 55ff]. However, within the context of the multi-contact revision of LOLA, the author of the present thesis redesigned and rebuilt the entire WPG module (including the motion generation pipeline) from scratch. Apart from a modular software design with clearly specified interfaces, this also includes methodological changes in particular regarding planning the ZMP, RMT, and CoM motion. Furthermore, the modifications also comprise the remaining stages of the planning pipeline. In order to support multi-contact locomotion, entirely new components have been introduced, e. g. for planning the trajectories of the task-space selection factors or external (multi-contact) wrenches. Putting all changes together, we obtain a novel motion generation system for humanoid robots which is capable of planning kinematically and dynamically feasible multi-contact locomotion in real-time. As an introduction to this chapter, Section 6.1 gives a brief overview of the objectives and core concepts of the proposed motion generator. This is followed in Section 6.2 by a description of the lower levels of the *Motion Plan* (which completes its specification from Section 5.2). Within Sections 6.3 to 6.15, the individual stages of the motion generator are described following their order of execution within the planning pipeline (cf. Figure 4.11). Section 6.16 explains how a particular task-space sample alias *Plan Snapshot* is extracted from the motion plan. Additionally, it describes the *Stream Processor* which converts the plan snapshot into a corresponding WPG output data structure (cf. Table 4.5). The chapter is concluded in Section 6.17.

In order to reason about kinematic and dynamic feasibility, this chapter uses the reduced models presented in Section 4.5.2. The simplified kinematics of the leg as well as a preliminary version of the dynamics model (without hand masses) have already been published in [3]. Moreover, [3] describes the main workflow of generating LOLA’s CoM motion from the (reference) ZMP trajectory and the reduced models which includes solving an overdetermined BVP through quintic spline collocation. The used algorithm for spline collocation has been published separately in [2]. A coarse summary describing the extension of the motion generator from [3] to support multi-contact locomotion has already been given in [1]. Among other changes, this includes the transition from the three-mass model to the five-mass model which is parameterized according to LEYERER [29] (summarized in [4]). A visualization of the present workflow for motion generation has been published independently through the video [18] @t=4m2s].

6.1 Preliminaries

The motion generator represents the second major component of the WPG module. It is responsible of connecting the discrete contact configurations (provided by the contact planner) with smooth trajectories describing the robot’s motion in task-space. While doing so, the primary objective is to establish kinematic and dynamic feasibility such that the planned motion leads to a

stable behavior when executed on the real hardware. At the same time, the computational cost should be kept as low as possible in order to maintain a high responsiveness which is in particular relevant for teleoperated walking and changing environments (dynamic replanning). This requires LOLA's motion generator to operate on a rather high level of abstraction with accordingly generous safety margins (first variant according to Figure 2.16). As already mentioned in Section 2.4 which summarized the state of the art, motion generation on an abstract level represents the most common approach in the field of humanoid robotics. In the following paragraphs, we briefly recall essential models and concepts which have already been introduced as part of the description of the WPG module (see Section 4.5).

Reduced Kinematic and Dynamic Model In order to maintain real-time performance, LOLA's motion generator makes use of the simplifying models presented in Section 4.5.2 (see in particular Figure 4.7). For reasoning about kinematic feasibility, a planar model of the leg is used to find a lower and upper bound for the vertical torso position. The dynamics of the robot are approximated by the five-mass model which represents an extension of the previous three-mass model by point masses located at the hands. Note that the mass distribution changes dynamically depending on the particular task-space configuration which is controlled by the task-space selection factors. For incorporating multi-contact effects, the five-mass model additionally considers external contact wrenches acting at the hands. The kinematics and dynamics are coupled through the position of the feet, the hands, and the virtual torso mass alias RMT. The models are primarily used by the RMT planner (see Section 6.14). In particular, the kinematic limits constrain the vertical position while the dynamics define the horizontal position of the RMT.

Solution Strategy The overall strategy for generating a kinematically and dynamically feasible CoM motion has already been presented in Section 4.5.4. To summarize, we first compute a reference ZMP trajectory passing through the SAs which in turn are defined by the foothold sequence provided by the contact planner. Independent of the SAs and the ZMP, we generate trajectories for the EE motion (feet, toes, hands, head) and the upper body rotation. Same as for the task-space selection factors, the load vector, and the external (multi-contact) wrenches, these trajectories are planned using rather simple heuristics. The data generated so far is then fed into the RMT planner which computes – based on the aforementioned robot models – an RMT motion such that kinematic limits of the leg are respected and the ZMP moves according to its reference trajectory. The CoM is then simply a result of the EE and RMT motion.

Planning Pipeline In favor of a modular and easily extendable software structure, the motion generator is subdivided into individual stages which are executed one after another. Together with the sequence and transition planner, these stages form the planning pipeline shown in Figure 4.11. Since the presented solution strategy defines a certain flow of data, the order of execution is not arbitrary. Nevertheless, some branches (such as the SA/ZMP and the UB/EE planning – see Figure 4.10) are independent of each other and thus, would allow parallel execution. However, since the main workload is given by the RMT planner (which already uses multiple threads internally – details follow in Section 6.14.2), a parallelization of the remaining stages would not lead to a substantial reduction of the overall runtime. Nevertheless, parallel execution branches may be considered within a future revision of the WPG.

6.2 Motion Plan: Lower Level Structure

While the higher level structure of the *Motion Plan* has already been described in Section 5.2, this section focuses on its lower levels. First of all, we introduce the concept of *Phases* which

act as children of the already specified *Transitions*. In particular, each transition is split into either one or two phases. Within the WPG module, phases primarily serve as data containers for storing the planned task-space trajectories in an analytical form, see Figure 6.1.

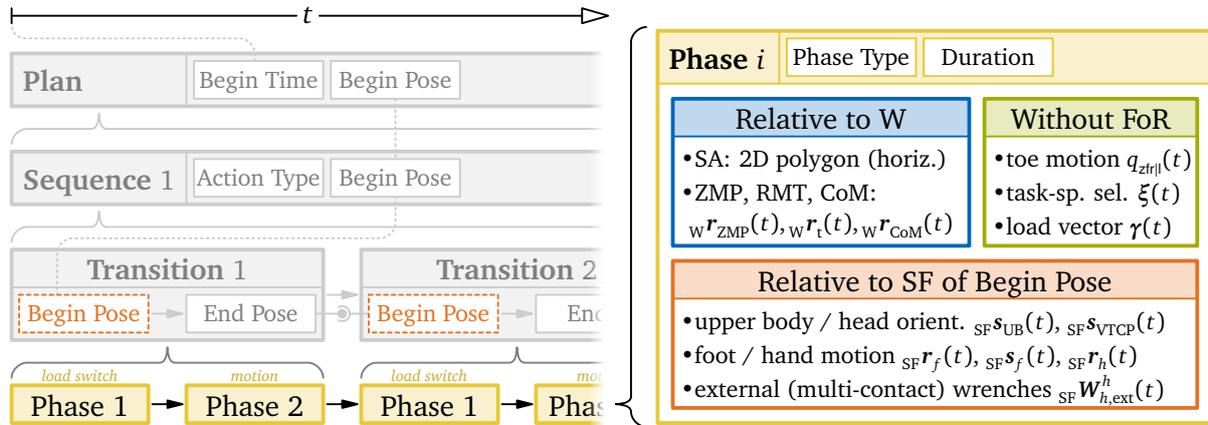


Figure 6.1: Lower level structure of the (*Motion*) Plan. Each *Transition* contains either one or two *Phases* (yellow). Each phase is assigned a certain type (*Load Switch* or *Motion*) and duration. Moreover, it stores quantities described relative to the world frame “W” (blue), quantities described relative to the stance foot TCP frame “SF” related to the *Begin Pose* of the corresponding transition (orange), and quantities without dedicated FoR (green). This illustration focuses on the lower levels of the plan while the higher levels have already been described in Figure 5.4.

Phase Type and Duration Each phase is labeled with an individual *Phase Type* which can be either *Load Switch* or *Motion*. Within a *Load Switch* phase, all EEs residing in the task-space remain static holding their current global position / orientation. Moreover, the (desired) CoM height and upper body orientation are constant. The *Motion* phase represents its counterpart, i. e., a time span where at least one of the aforementioned components is subject to changes. The distinction between these two phase types allows to split a transition into periods with and without task-space motion. Within the context of “regular” biped locomotion, i. e., without hand support, the *Load Switch* phase is equivalent to the DS phase while the *Motion* phase is equivalent to the SS phase. Indeed, this new terminology represents a generalization of the DS/SS concept to the multi-contact case. Moreover, it allows to describe certain non-gaited motions such as bowing (upper body rotation is interpreted as motion while the remaining task-space components are constant). Besides the type, each phase is assigned an individual duration $t_{pha,dur}$ which is computed by the *Phase Planner* based on the duration of the governing transition. Details on how the phase type and duration are determined follow in Section 6.3.

Phase Data and FoR The individual quantities stored within each phase can be clustered according to the used FoR, see Figure 6.1. For practical reasons, the current SA is described as a 2D polygon lying in the x - y -plane of the (planning) world frame “W”. This allows a simple computation by projecting footholds to the horizontal plane. Since the ZMP trajectory ${}_W r_{ZMP}(t)$ is constructed based on the SAs, it is natural to describe it in the same FoR. Moreover, same as for the RMT motion ${}_W r_t(t)$, the ZMP trajectory is split into a horizontal and vertical component which are planned individually. This further motivates the use of W as FoR. Since the CoM motion ${}_W r_{CoM}(t)$ is mainly affected by the RMT trajectory, we use the same FoR which makes comparisons of the CoM and RMT motion more convenient during data analysis. Besides, most other quantities are described relative to the stance foot TCP frame “SF” related to the *Begin Pose* of the corresponding transition. In particular, this applies to the upper body and head orientation trajectories ${}_{SF} s_{UB}(t)$ and ${}_{SF} s_{VTCP}(t)$, the foot and hand position/orientation trajectories ${}_{SF} r_f(t)$, ${}_{SF} s_f(t)$, ${}_{SF} r_h(t)$, and the external (multi-contact) wrench trajectories ${}_{SF} W_{h,ext}^h(t)$. In-

deed, this choice of the FoR minimizes the count of necessary coordinate transformations since most quantities of a *Robot Pose* are already described relative to the SF frame (cf. Table 5.1). However, within motion generation, the computational cost of coordinate transformations is almost negligible so that the FoR is chosen according to convenience during implementation. Finally, a phase also stores FoR-independent quantities such as the toe joint trajectories $q_{zfrll}(t)$, the task-space selection trajectory $\xi(t)$, and the load trajectory $\gamma(t)$.

Describing Trajectories Within the motion plan, all time-varying quantities are described in an analytical form. Compared to a representation by a large series of time step samples, analytic signals use much less memory and can be modified much more efficiently (e. g. by shifting control points). Moreover, depending on the particular type of the underlying function, cheap and exact analytic derivatives may be available. Certainly, the motion generator has to be capable of generating analytic signals which may not be the case depending on the chosen planning approach. However, the solution strategy proposed within this thesis is well suited for an analytic description of signals. Note that trajectories are generated by the asynchronous *Planner* (*Thread 6* in Figure 4.11) while they are evaluated within the WPG’s main loop (*Thread 5* in Figure 4.11). Thus, with regard to real-time performance, we focus in general on an efficient evaluation while the computational cost of generation is less critical.

For the majority of signals, the WPG uses polynomial splines, i. e., a chain of (independent) polynomial segments. With regard to our use case, the main advantages of a polynomial representation are the straightforward superposition of signals (the sum of polynomials is again a polynomial) and the numerous available methods for interpolation and collocation (see also Appendix G). While there exist multiple ways of describing a polynomial spline (with B-splines [129, p. 87ff] being the probably most prominent one), we use in particular the so-called *Piecewise Polynomial (PP)* form [129, p. 69ff] which describes the spline through the coefficients of interconnected, yet independently defined, polynomial segments. The PP form allows an efficient evaluation of the spline and its derivatives by using the method of HORNER and GILBERT (see [208] and [197, p. 94ff]). The implementation used by the WPG of LOLA has been published as part⁷⁴ of the *Broccoli* library.

For describing spatial rotation, the motion generator uses a custom formulation based on quaternions. In particular, we introduce the concept of *quaternion trajectories* which combine a quaternion spline $s(\eta) \in \mathbb{H}^1$ defining the “shape” and a parameter spline $\eta(t) \in \mathbb{R}$ specifying the “timing” of the motion. A quaternion trajectory is simply evaluated through the chained operation $s(\eta(t))$. Augmenting the quaternion spline with a parameter spline gives us full control over the speed of traversal. Within motion generation, this feature is primarily used to enforce zero velocity and acceleration at the start and end of a trajectory which ensures smooth accelerations and C^2 -continuity at junctions (independent of the type of the quaternion spline). The main motivation for using an approach based on quaternions is their robustness (no singularities) and ability to describe rotations in a “natural” way. The fundamentals of quaternion calculus, describing spatial rotation with quaternions, and the interpolation of quaternions are discussed in Appendix B. The implementation of the proposed quaternion trajectories has been published as part⁷⁵ of the *Broccoli* library. This also includes efficient algorithms for evaluating the n -th derivative of a quaternion trajectory as well as the computation of the corresponding angular velocity and acceleration. Among the multiple interpolation methods presented in the appendix, we primarily use quaternion trajectories based on the QBSpline curve (cf. Appendix B.3.6).

Planning on Sequence-, Transition-, or Phase-Level While the contact planner computes the higher levels of the motion plan down to the level of transitions (begin/end pose and additional parameters $t_{\text{tra,dur}}$, h_{step} , and τ_{SF}), the motion generator is responsible of creating phases

⁷⁴See the class `PolynomialCurve|Spline|Trajectory` in the module `curve` of *Broccoli*.

⁷⁵See the class `QuaternionCurve|Spline|Trajectory` in the module `curve` of *Broccoli*.

and filling them with trajectory data. To that end, trajectories may be generated on sequence-, transition-, or phase-level. In order to store a sequence/transition-level trajectory in the motion plan, we simply split it into corresponding phase-level chunks taking the individual phase duration into account. For this purpose, the *Broccoli* library provides methods⁷⁶ to split/join polynomial trajectories (without loss of information). Note that splitting quaternion trajectories has not been implemented yet since the corresponding task-space quantities are (currently) planned exclusively on phase-level. However, a corresponding extension of the library would be straightforward. Indeed, for splitting a sequence/transition-level quaternion trajectory, a simple solution would be to copy the underlying quaternion spline (remains unchanged) and only modify the (polynomial) parameter spline of the corresponding phase-level chunks.

6.3 Phase Planner

The process of motion generation starts with the *Phase Planner* which represents stage 3 in the planning pipeline (cf. Figure 4.11). Based on the data stored in the motion plan so far, it creates phases for each transition and assigns the particular phase type (*Load Switch* or *Motion*, cf. Section 6.2) and duration to it. However, the phase planner does not specify any of the task-space quantities stored within the phase – this is left to the remaining stages of the planning pipeline. For creating the phases of a particular transition, the phase planner considers the entire sequence. While doing so, it guarantees that following constraints are satisfied:

- A sequence has to begin and end with a load switch phase (allows to start and finish the sequence with the same static idle pose specified in Section 4.5.2).
- Load switch and motion phases have to alternate within a sequence (equivalent to alternating DS/SS phases in regular biped locomotion).
- A transition consists of either one or two phases (a transition with more than two phases can always be split into multiple transitions with a maximum of two phases each).
- A transition contains a motion phase if and only if it involves task-space motion (automatically detected by comparing the begin and end pose of the transition).
- In case a transition consists of two phases, the load switch phase has to be executed *before* the motion phase.

Considering these rules, a transition may contain either a single load switch phase, a single motion phase, or a leading load switch and trailing motion phase. Moreover, these constraints imply that the last transition of a sequence always consists of a single load switch phase, see Figure 6.2. As a consequence, the transition planner has to guarantee that the last transition of a sequence does not involve any task-space motion. Indeed, this is satisfied by the transition planner's strategy for finishing/stopping a sequence to bring the robot back to its static idle pose (automatically appends corresponding trailing transitions).

Detecting Motion A motion phase is only generated if the corresponding transition involves task-space motion. In order to detect such motion, we compare the begin and end pose of the transition. In particular, we consider a transition to involve task-space motion if

- the (desired) orientation of the upper body changes, or
- the (desired) height of the CoM changes, or
- the position and/or orientation of at least one EE changes and the corresponding EE is assigned to the task-space at the begin and/or end pose, or
- at least one EE is blended from null-space to task-space or vice versa.

⁷⁶See the class `SplineTrajectorySplitter` in the module `curve` of *Broccoli*.

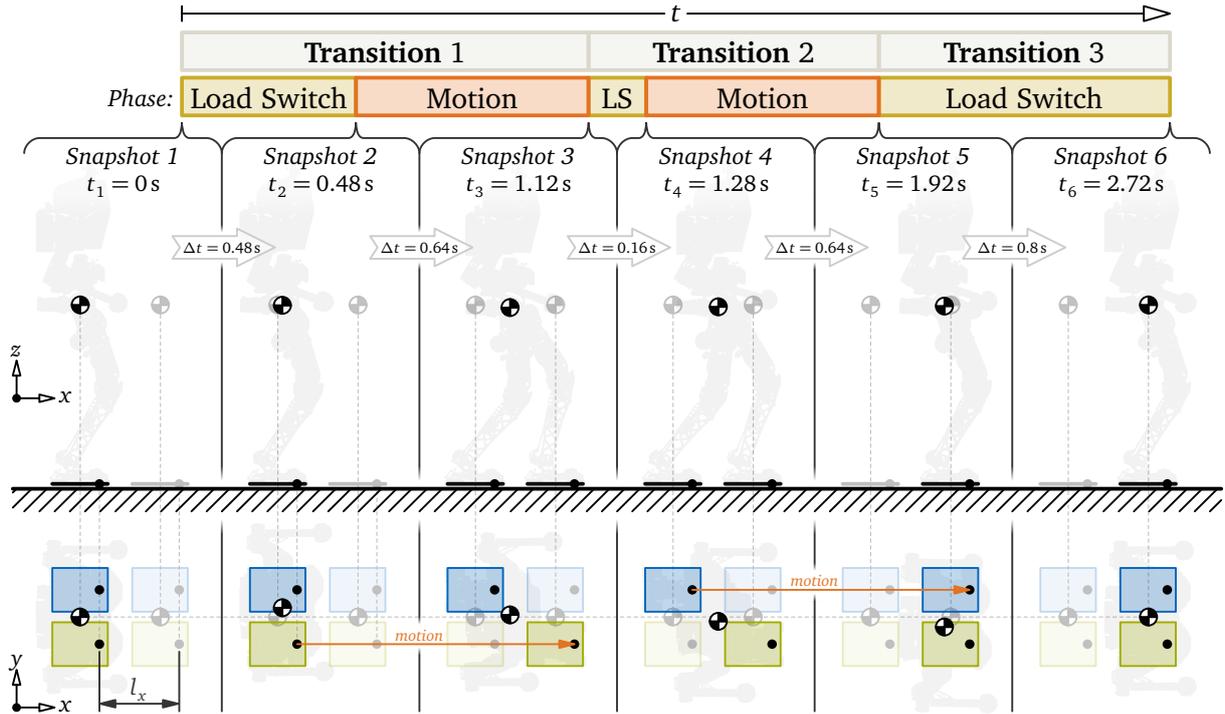


Figure 6.2: Sequence of alternating *Load Switch* and *Motion* phases generated by the *Phase Planner* for an exemplary walking sequence. The transitions are provided by the *Transition Planner* on the basis of a “Fixed Sequence” action featuring a single QPWT ($l_x = 0.4$ m). Transition 2 and 3 are automatically appended by the transition planner to bring the robot back to its static idle pose. The load switch phases are used to shift the CoM (horizontally) and redistribute the EE loads while the motion phases are used to move the EEs.

Note that in terms of an easily understandable presentation, Figure 6.2 focuses on regular biped walking without hand support. However, the phase planner does not distinct between feet and hands as EEs. Thus, the presented workflow is identical for multi-contact locomotion.

Enforcing Motion Although the proposed motion detection allows to automatically generate an appropriate segmentation, it may fail in certain situations. An example is stamping, i. e., walking with zero step length where the begin and end pose of the transitions are equivalent and, hence, seem to involve no task-space motion. To circumvent this problem, the transition planner is further given the capability to *enforce* motion either for a particular task-space component or the entire transition effectively bypassing the automatic motion detection. For the action types “fixed sequence”, “teleoperated”, and “autonomous” walking, the transition planner enforces motion for the swing foot (except the very last transition) which enables walking in place. For special actions such as bowing, the transition planner enforces motion for a virtual transition in order to artificially keep the robot in a (special) static pose for a certain duration.

Phase Duration For determining the duration of a phase, we distinct between the cases of a default and a custom transition duration $t_{tra,dur}$. In case of using the default $t_{tra,dur} = 0.8$ s, we use $t_{pha,dur} = 0.2 t_{tra,dur} = 0.16$ s for the load switch phase (if present) and $t_{pha,dur} = 0.8 t_{tra,dur} = 0.64$ s for the motion phase (if present). Note that these proportions are adopted from the previous trajectory generation system by BUSCHMANN and can be easily changed through a user-specified parameter. For the very first and last load switch phase of a sequence, we additionally slow down the phase by a factor of three (3×0.16 s = 0.48 s) and five (5×0.16 s = 0.8 s), respectively (cf. Figure 6.2). This ensures a decent acceleration and deceleration from/towards the static idle pose. In case of a custom $t_{tra,dur}$, we either set the phase duration to $t_{pha,dur} = t_{tra,dur}$ (one phase) or according to the aforementioned proportions (two phases).

6.4 Support Area (SA) Planner

In the fourth stage of the planning pipeline, the SAs are computed. This happens on phase-level, i. e., for each phase individually. For load switch phases, the SA is computed using the begin pose while for motion phases we use the end pose of the enclosing transition. This is reasoned by our convention that load switch phases always precede motion phases within a transition. In order to decide if a foot contributes to the SA, we check if it is in closed contact throughout the entire phase. As already explained in Section 4.5.4, we assume level ground (as approximation) and thus, project the rectangular outline of the foot sole onto the horizontal plane of the world frame. The dimensions of the contact area are taken from Table 4.4, where we distinct between full and partial/tiptoe contact as indicated by the corresponding robot pose. In addition, we apply symmetric safety margins of 20 % and 30 % to reduce the length and width of the contact area, respectively. The margins take effect on the shape of the ZMP trajectory and are meant to increase robustness (the distance of the ZMP to the SA's border can be seen as metric for the “stability reserve”). Due to the assumption of coplanar contacts in the foot-ground interface, the SA is given by a 2D polygon (cf. Section 2.3) which is computed as the convex hull of all contact points. For constructing the convex hull, we use GRAHAM's scan [121, p. 1030ff] where the particular implementation has been published as part⁷⁷ of *Broccoli*. An animation demonstrating the construction of SAs is given in the video [18 📺@t=4m7s].

6.5 Zero-Moment Point (ZMP) Planner

Once the SAs have been determined, we generate the reference ZMP trajectory ${}_w r_{\text{ZMP}}(t) \in \mathbb{R}^3$. Within this thesis, we require the gait to be dynamically balanced (cf. Section 4.5.4), thus, the ZMP has to reside within the current SA at all times. Apart from this constraint, we can freely choose the path and its speed of traversal. In the following, we use a path consisting of linear segments, however, we could also use other shapes such as cubic splines as proposed by KAJITA et al. in [233]. In fact, the only requirement made by the subsequent planning stages is that ${}_w r_{\text{ZMP}}(t)$ is C^0 -continuous (required for C^2 -continuity of CoM, see Section 6.14.2 for details). In contrast to related work, we propose to use a three-dimensional ZMP where the horizontal and vertical components are processed independently of each other (cf. [18 📺@t=4m38s]).

Horizontal Components The components ${}_w r_{\text{ZMP},x|y}(t)$ which describe the horizontal ZMP motion are planned on sequence-level. In particular, we first generate a series of control-points where each load switch phase spawns a control-point at its beginning and end. In the exemplary walking sequence shown in Figure 6.2, this would result in a total of six control-points (one for each snapshot). The position of a control-point is computed solely based on geometric considerations where we only take the SAs (with applied safety margins) into account. In particular, we use the following scheme (see also Figure 6.3):

- The very first and last control-point of the sequence are placed at the centroids of the corresponding SAs (first and last load switch phase). This particular choice maximizes the “stability reserve” at the beginning and end of a sequence where the robot resides in the static idle pose. In the case of dynamic replanning, the first control-point is instead given by the ZMP position at the end of the last unchanged transition.
- The second control-point is chosen to the point on the SA of the first motion phase which is closest to the very first control-point. Similarly, the second last control-point is chosen to the point on the SA of the last motion phase which is closest to the very last control-point.

⁷⁷See the method `Polygon2D::reComputeConvexHull` in the module `geometry` of *Broccoli*.

- For determining the position of the remaining control-points, we iterate over all “interior” load switch phases. For each of which, we compare the SAs of the preceding and subsequent motion phase where we distinct between two cases:
 - In case the SAs intersect, both control-points related to the currently investigated load switch phase are set to the centroid of the intersection polygon (maximizing stability reserve). In fact, this case can only occur within non-gaited motion – in particular, if the preceding and/or subsequent motion phase does not involve any foot motion. Note that the intersection of two convex polygons is also a convex polygon, which allows an efficient implementation⁷⁸.
 - In case the SAs do not intersect, the control-points of the currently investigated load switch phase are chosen to the end points of the shortest connection between the SAs. Note that the shortest connection line between two non-intersecting (otherwise arbitrary) polygons starts and/or ends at one of the vertices of the polygons – at least if it is unique. For parallel edges, there might be an infinite set of shortest connections. In this case, we use the “mean value” of the set (computed from its “boundaries” which are given by the two shortest connection lines which start/end at a vertex). Since both polygons are convex and have kinks at their vertices (ensured by the algorithm used for computing the convex hull), there can be at most one pair of parallel edges with shortest distance. Also note that parallel edges occur frequently e. g. for straight walking (parallel feet) with small step lengths.

In comparison to more elaborate techniques, e. g. the formulation of an optimization problem as suggested by BUSCHMANN et al. in [98], this purely geometric approach is much simpler, more efficient, and guarantees to find a valid solution in a deterministic time.

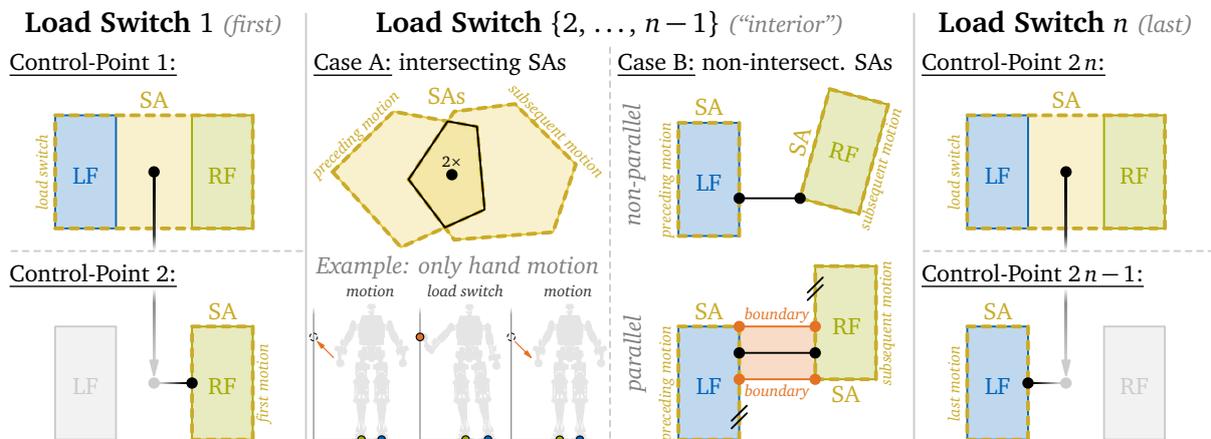


Figure 6.3: Strategy for placing the $2n$ control-points of the horizontal ZMP trajectory ${}^w r_{ZMP,x|y}(t)$ for a sequence containing n load switch phases and $n - 1$ motion phases. From left to right: control-point placement for the first, interior, and last load switch phase. Each load switch phase spawns two control-points.

Same as for the original WPG system by BUSCHMANN [98], we prefer a slowly moving ZMP in order to obtain smoother CoM trajectories. For this reason, we choose to connect the control-points with linear segments (shortest path). Since the SAs are convex polygons, this also guarantees that every point on the path lies within the corresponding SA. Finally, it is left to specify the speed at which the ZMP traverses this path. For the very first and last load switch phase, we choose a cubic interpolation with zero velocity and acceleration at the very first and last control-point. This ensures a decent acceleration and deceleration of the ZMP at the beginning and end of a sequence which, however, only has a rather small impact on the overall locomotion performance of the robot. For all other phases, we use constant velocity, see Figure 6.4.

⁷⁸See the method `Polygon2D::computeAsConvexPolygonIntersection` in the module `geometry` of *Broccoli*.

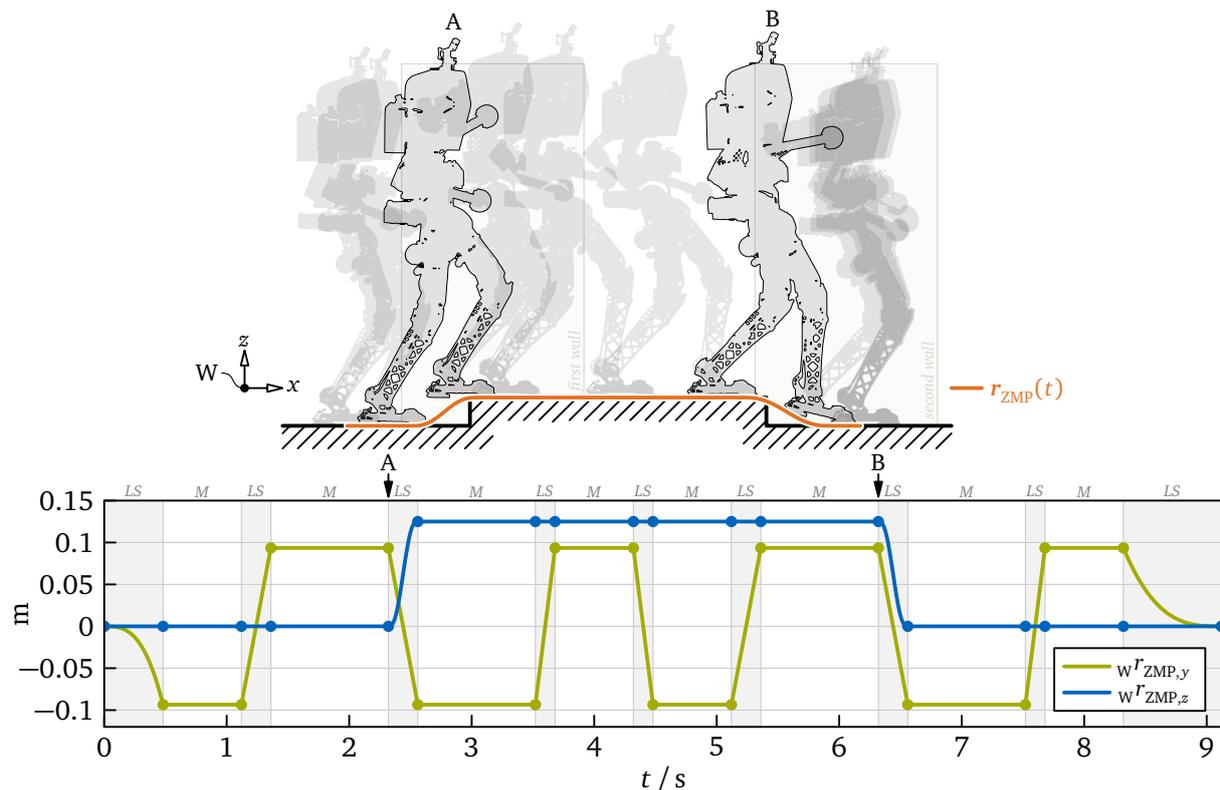


Figure 6.4: Horizontal and vertical components of the ZMP trajectory $w^r r_{ZMP}(t)$ for a multi-contact scenario similar to [20] @ $t=1m40s$ (stepping up and down platform of 12.5 cm height). The background color of the plot shows the partitioning into load switch (“LS”) and motion (“M”) phases. Within the very first and last load switch phase, $w^r r_{ZMP,x|y}$ use cubic interpolation while a constant velocity is set for the remaining phases. Within load switch phases, the vertical component $w^r r_{ZMP,z}$ is formulated as a quintic polynomial which tracks the height of the stance foot (stepping up at “A”, stepping down at “B”). In motion phases, $w^r r_{ZMP,z}$ is kept constant. The dots indicate control-point locations.

Vertical Component According to Equation 4.9, the contact wrench of the foot-ground interface does not only depend on the horizontal components $w^r r_{ZMP,x|y}$, but also the vertical component $w^r r_{ZMP,z}$. For walking on level ground which coincides with the x - y -plane of the world frame, one could simply set $w^r r_{ZMP,z} = 0$ and cancel all related terms. Since the ZMP criterion requires coplanar foot-ground contacts anyways, the vertical component of the ZMP is *de facto* not considered in most related works. Within this thesis, we also allow non-coplanar foot-ground contacts (uneven terrain, stairs, ramps, etc.) and treat the violation of this constraint as (tolerated) modeling error. In order to keep this error as small as possible, we plan $w^r r_{ZMP,z}(t)$ such that it tracks the vertical position of the current stance foot. In particular, we use a quintic polynomial with zero velocity and acceleration at the start and end for load switch phases while $w^r r_{ZMP,z}$ remains constant within motion phases, see Figure 6.4. The polynomial connects the previous control-point with $w^r r_{SF,z}$ of the end pose (last load switch phase) or the begin pose (all other load switch phases) of the governing transition.

Note that the previous version of the ZMP planner presented in [3] did not consider the vertical component but instead used the approximation $w^r r_{ZMP,z} \approx 0$. While the error remains small for the therein presented scenario of stepping up and down a platform of 12.5 cm height, it is not negligible for climbing stairs (vertical displacement accumulates) or in case the robot is initialized at a certain non-zero height. At this point, special thanks go to MORITZ SATTLER for drawing attention to this issue which finally motivated the extension of the ZMP planner to include the vertical component.

6.6 Upper Body Orientation Planner

In the sixth stage of the planning pipeline, the rotational motion of the upper body frame “UB” (cf. Figure 4.2) is generated. In particular, we setup ${}_{\text{SF}}\mathbf{s}_{\text{UB}}(t)$ as a quaternion trajectory combining a quaternion spline and a parameter spline as described in Section 6.2. The trajectory is planned on phase-level where we consider the upper body orientations stored within the begin pose (${}_{\text{SF}}\mathbf{s}_{\text{UB,beg}}$) and end pose (${}_{\text{SF}}\mathbf{s}_{\text{UB,end}}$) of the corresponding transition as keyframes. Within motion phases which involve upper body rotation, we connect these keyframes by a \mathcal{C}^2 -continuous QBSpline of order $k = 4$. Note that a comprehensive description of quaternion interpolation by QBSplines is given in Appendix B.3.6. The control-quaternions are set to $\{{}_{\text{SF}}\mathbf{s}_{\text{UB,beg}}, {}_{\text{SF}}\mathbf{s}_{\text{UB,beg}}, {}_{\text{SF}}\mathbf{s}_{\text{UB,end}}, {}_{\text{SF}}\mathbf{s}_{\text{UB,end}}\}$ where we duplicated the first and last element in order to reach the minimum count of control-quaternions required by this interpolation type. Moreover, we use a clamped, uniform, and normalized knot sequence which ensures that the very first and last control-quaternions are interpolated. The QBSpline is combined with a quintic polynomial as parameter spline to force zero velocity and acceleration at the beginning and end of the phase. This guarantees \mathcal{C}^2 -continuity also at phase boundaries. Within the remaining phases, i. e., for load switch and motion phases without involved upper body motion, ${}_{\text{SF}}\mathbf{s}_{\text{UB}}(t)$ is kept constant. Exemplary results for a bowing motion (rotation around horizontal axis) and walking on a curved path (rotation around vertical axis) are shown in Figure 6.5.

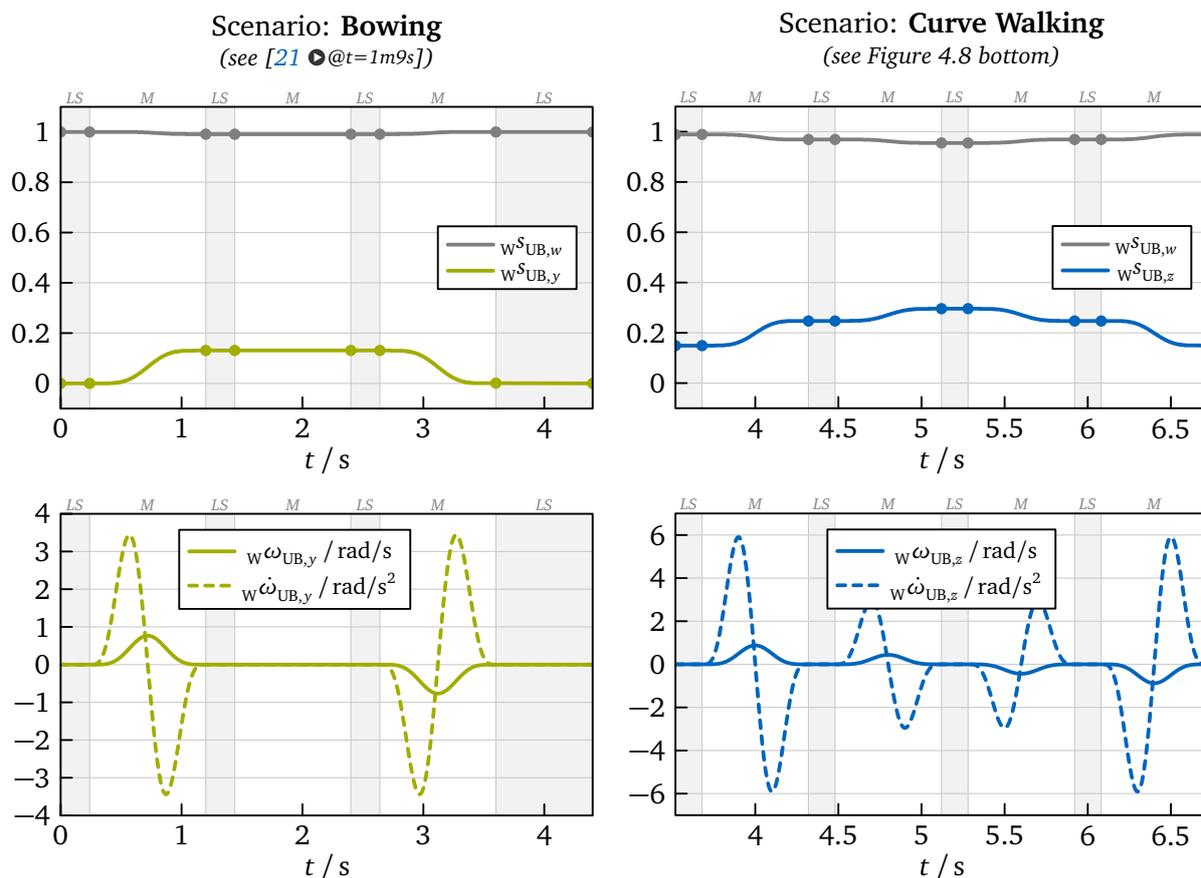


Figure 6.5: Quaternion trajectory describing the rotation of the upper body ${}_{\text{W}}\mathbf{s}_{\text{UB}}(t)$ for two exemplary scenarios: bowing motion (left) and walking on a curved path (right). The plots show relevant components of the orientation ${}_{\text{W}}\mathbf{s}_{\text{UB}}(t)$, angular velocity ${}_{\text{W}}\boldsymbol{\omega}_{\text{UB}}(t)$, and angular acceleration ${}_{\text{W}}\dot{\boldsymbol{\omega}}_{\text{UB}}(t)$. Not shown components are zero in the entire time interval. Although the trajectory is generated with respect to the SF FoR, it is plotted for the W FoR to demonstrate its \mathcal{C}^2 -continuity. The dots indicate control-quaternion locations (left: hard-coded; right: default upper body orientation computed from mean foot orientation).

Control-Quaternions and Default Upper Body Orientation The control-quaternions are derived from ${}_{\text{SF}}\mathbf{s}_{\text{UB}}$ stored in the robot poses at the beginning and end of the governing transition. For special, non-gaited motions, these are typically hard-coded. An example is the bowing motion shown in the left column of Figure 6.5, where the upper body is tilted by an angle of 15° . For action types describing gaited motion (fixed sequence, teleoperated, and autonomous walking), ${}_{\text{SF}}\mathbf{s}_{\text{UB}}$ in the transition’s end pose is set to the custom orientation specified in the corresponding QPWT, see Table 5.2. In case no custom orientation is set, a *default* orientation is computed from the mean foot orientation (only vertical rotation such that torso stays upright even for non-horizontal feet). This case is shown in the right column of Figure 6.5 which is obtained from the curve walking scenario shown in Figure 4.8 (bottom). The depicted plots focus on a time span where the step angle around the vertical axis changes from $\varphi_z = 0.2$ rad (turning left) to $\varphi_z = -0.2$ rad (turning right).

Discussion and Outlook Since planning an ideal upper body rotation is not in the scope of this thesis, a rather simple scheme has been used. Finding a motion which is optimal with respect to a certain criterion (e. g. minimization of total angular momentum) is instead left for potential future investigations. Note that the subsequent stages of the planning pipeline only demand \mathcal{C}^2 -continuity of $\mathbf{s}_{\text{UB}}(t)$ (required for \mathcal{C}^2 -continuity of CoM, see Section 6.14.2 for details). Nevertheless, one should handle the upper body orientation with care: as a task-space component, $\mathbf{s}_{\text{UB}}(t)$ directly affects the resulting joint-space motion. Moreover, $\mathbf{s}_{\text{UB}}(t)$ acts as input to the five-mass model (linked to mass moment of inertia of torso mass), therefore, it has also a certain influence on the planned (horizontal) RMT and CoM trajectory.

6.7 Foot Motion Planner

The task of this submodule is to generate \mathcal{C}^2 -continuous trajectories describing the full 6D motion of the left and right foot’s TCP frame. Same as for $\mathbf{s}_{\text{UB}}(t)$, the construction is performed on phase-level where we focus on the motion phases. Within load switch phases or in case the currently investigated foot is not subject to motion (e. g. the stance foot SF), we simply keep the position and orientation constant. For the (moving) swing foot $\overline{\text{SF}}$, we have to connect the 6D pose of the TCP frame at the beginning (${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{beg}}, {}_{\text{SF}}\mathbf{s}_{\overline{\text{SF}},\text{beg}}$) with the one at the end (${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{end}}, {}_{\text{SF}}\mathbf{s}_{\overline{\text{SF}},\text{end}}$) of the governing transition in an appropriate way. Apart from these BCs, the corresponding motion plan transition also provides the step height h_{step} and the swing foot timing factor $\tau_{\overline{\text{SF}}}$ as additional parameters (cf. Section 5.2). While the step height simply specifies the vertical clearance, the swing foot timing factor allows more substantial modifications of ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}}}(t)$ by blending between a *regular* ($\tau_{\overline{\text{SF}}} = 0$; for level ground) and a *collision-aware* ($\tau_{\overline{\text{SF}}} = 0.5$; for stepping up/down and obstacle traversal, cf. Section 5.5.5) set of timing parameters. See the video [18 ●@t=4m58s] for a 3D visualization of both parametrizations.

Orientation Similar to the trajectory generation system by BUSCHMANN [100, p. 70ff], we consider two components for the foot rotation. The first component represents the transition from ${}_{\text{SF}}\mathbf{s}_{\overline{\text{SF}},\text{beg}}$ at the beginning t_{beg} towards ${}_{\text{SF}}\mathbf{s}_{\overline{\text{SF}},\text{end}}$ at the end t_{end} of the phase. The previous locomotion system assumed a horizontal ground, thus, the first component described a rotation around the vertical axis. Since the new WPG also allows non-horizontal foot-ground contacts (e. g. for climbing ramps), the axis of rotation does not necessarily have to be vertical anymore.

The second component describes a superimposed “interior” ($t_{\text{beg}} < t < t_{\text{end}}$) motion which facilitates a lift-off with the toe right after the beginning (at $t_{\text{beg}} + \varepsilon$) and a strike with the heel just before the end (at $t_{\text{end}} - \varepsilon$) of the motion. While toe-lift-off is meant to push kinematic limits in particular for large steps (similar to the toe joint motion, see Section 6.8), heel-strike

increases overall robustness in the presence of upper body inclination errors [100, p. 72f].

Combining both components yields a multi-axis rotation which we realize by formulating a corresponding quaternion⁷⁹ trajectory. Similar to ${}_{\text{SF}}\mathbf{s}_{\text{UB}}(t)$, we setup a QBSpline of order $k = 4$ as underlying quaternion spline. The control-quaternions are set to $\{{}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}, \mathbf{s}_1, \mathbf{s}_2, {}_{\text{SF}}\mathbf{s}_{\text{SF,end}}\}$ while the knot sequence is again clamped, uniform, and normalized. The first and last control-quaternions (in combination with the clamped knot sequence) ensure that the BCs are satisfied. The interior control-quaternions, \mathbf{s}_1 and \mathbf{s}_2 , are used to establish the second component of foot rotation, i. e., toe-lift-off and heel-strike. In particular, we use

$$\varphi_{\text{SF},y,\text{ref}} := \text{asin}\left(0.1 \text{ m}^{-1} \left({}_{\text{SF}}r_{\text{SF,end},x} - {}_{\text{SF}}r_{\text{SF,beg},x}\right)\right), \quad (\text{reference angle for tilting motion}) \quad (6.1)$$

$$\mathbf{s}_{1,\Delta} := \text{quat}(\mathbf{e}_y, +\varphi_{\text{SF},y,\text{ref}}), \quad (\text{relative rotation describing toe-lift-off}) \quad (6.2)$$

$$\mathbf{s}_{2,\Delta} := \text{quat}(\mathbf{e}_y, -\varphi_{\text{SF},y,\text{ref}}), \quad (\text{relative rotation describing heel-strike})$$

$$\mathbf{s}_{1,\text{virt}} := \begin{cases} \text{SLERP}\left({}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}, +{}_{\text{SF}}\mathbf{s}_{\text{SF,end}}, \frac{1}{3}\right) & \text{if } {}_{\text{SF}}\mathbf{s}_{\text{SF,beg}} \odot {}_{\text{SF}}\mathbf{s}_{\text{SF,end}} \geq 0, \\ \text{SLERP}\left({}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}, -{}_{\text{SF}}\mathbf{s}_{\text{SF,end}}, \frac{1}{3}\right) & \text{else} \end{cases}, \quad (6.3)$$

$$\mathbf{s}_{2,\text{virt}} := \begin{cases} \text{SLERP}\left({}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}, +{}_{\text{SF}}\mathbf{s}_{\text{SF,end}}, \frac{2}{3}\right) & \text{if } {}_{\text{SF}}\mathbf{s}_{\text{SF,beg}} \odot {}_{\text{SF}}\mathbf{s}_{\text{SF,end}} \geq 0, \\ \text{SLERP}\left({}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}, -{}_{\text{SF}}\mathbf{s}_{\text{SF,end}}, \frac{2}{3}\right) & \text{else} \end{cases},$$

$$\mathbf{s}_1 := \mathbf{s}_{1,\text{virt}} \otimes \mathbf{s}_{1,\Delta}, \quad (\text{first interior control-quaternion}) \quad (6.4)$$

$$\mathbf{s}_2 := \mathbf{s}_{2,\text{virt}} \otimes \mathbf{s}_{2,\Delta}. \quad (\text{second interior control-quaternion})$$

With $\varphi_{\text{SF},y,\text{ref}}$, we first compute a *reference* angle for tilting the swing foot around its local y -axis. The heuristic provided in Equation 6.1 is based on the previous implementation by BUSCHMANN and makes $\varphi_{\text{SF},y,\text{ref}}$ dependent on the stride length, i. e., the distance traveled by the swing foot in walking direction (measured in stance foot FoR). Note that the sign of $\varphi_{\text{SF},y,\text{ref}}$ becomes negative for ${}_{\text{SF}}r_{\text{SF,end},x} < {}_{\text{SF}}r_{\text{SF,beg},x}$ such that an equivalent roll-off strategy is obtained for walking backwards. The reference angle is used to compute the relative rotations describing toe-lift-off ($\mathbf{s}_{1,\Delta}$) and heel-strike ($\mathbf{s}_{2,\Delta}$) as elementary rotations around the swing foot's local y -axis, see Equation 6.2. Furthermore, we construct *virtual* control-quaternions $\mathbf{s}_{1,\text{virt}}$ and $\mathbf{s}_{2,\text{virt}}$ which represent snapshots of a virtual SLERP operation from ${}_{\text{SF}}\mathbf{s}_{\text{SF,beg}}$ towards ${}_{\text{SF}}\mathbf{s}_{\text{SF,end}}$ at $\eta = 1/3$ and $\eta = 2/3$, respectively. Note that the case distinctions in Equation 6.3 ensure minimum rotation, i. e., moving along the “shortest path”, cf. Appendix B.3.3. Finally, the interior control-quaternions \mathbf{s}_1 and \mathbf{s}_2 are obtained by superimposing the virtual control-quaternions $\mathbf{s}_{1,\text{virt}}$ and $\mathbf{s}_{2,\text{virt}}$ with the relative rotations $\mathbf{s}_{1,\Delta}$ and $\mathbf{s}_{2,\Delta}$ as shown in Equation 6.4.

An important property of the resulting QBSpline ${}_{\text{SF}}\mathbf{s}_{\text{SF}}(\eta)$ with $\eta \in [0, 1[$ is that it does not pass through \mathbf{s}_1 or \mathbf{s}_2 . However, due to the uniform knot sequence and the symmetric choice of \mathbf{s}_1 and \mathbf{s}_2 , the QBSpline will be close (but not closest) to \mathbf{s}_1 and \mathbf{s}_2 at $\eta = 1/3$ and $\eta = 2/3$, respectively. This motivates the partitioning of the motion phase into the three sub-phases *lift-off* ($\eta \in [0, 1/3]$), *mid-swing* ($\eta \in [1/3, 2/3]$), and *strike* ($\eta \in [2/3, 1]$). In order to control the speed of traversal, we setup a parameter spline $\eta(t)$ consisting of three quintic polynomial segments. The coefficients of the segments are chosen such that $\eta(t)$ forms a smooth (C^4 -continuous) spline with zero velocity and acceleration at t_{beg} and t_{end} (see Algorithm G.1). The control-points are set to $\eta(t_{\text{beg}}) = 0$, $\eta(t_1) = 1/3$, $\eta(t_2) = 2/3$, and $\eta(t_{\text{end}}) = 1$. For simplicity, we choose t_1 and t_2 such that they split the phase duration into three uniform segments. However, t_1 and t_2 are driven by dedicated parameters of the WPG such that the proportions of the three

⁷⁹The previous locomotion system of LOLA used special, custom representations for describing orientations of the feet and the upper body which were originally developed by LÖFFLER for JOHNNIE (see [100, p. 68ff] for details). Within the context of redesigning the WPG, the author of this thesis decided to use a uniform formulation by quaternions instead. Besides the general advantages of quaternions such as robustness and numerical efficiency, its is also easier to convert them to/from other representations – in particular with regard to time derivatives.

sub-phases lift-off, mid-swing, and strike can be customized easily. The resulting quaternion trajectory ${}_{\text{SF}}s_{\text{SF}}(t)$ is C^2 -continuous and has zero angular velocity and acceleration at the phase boundaries, see Figure 6.6.

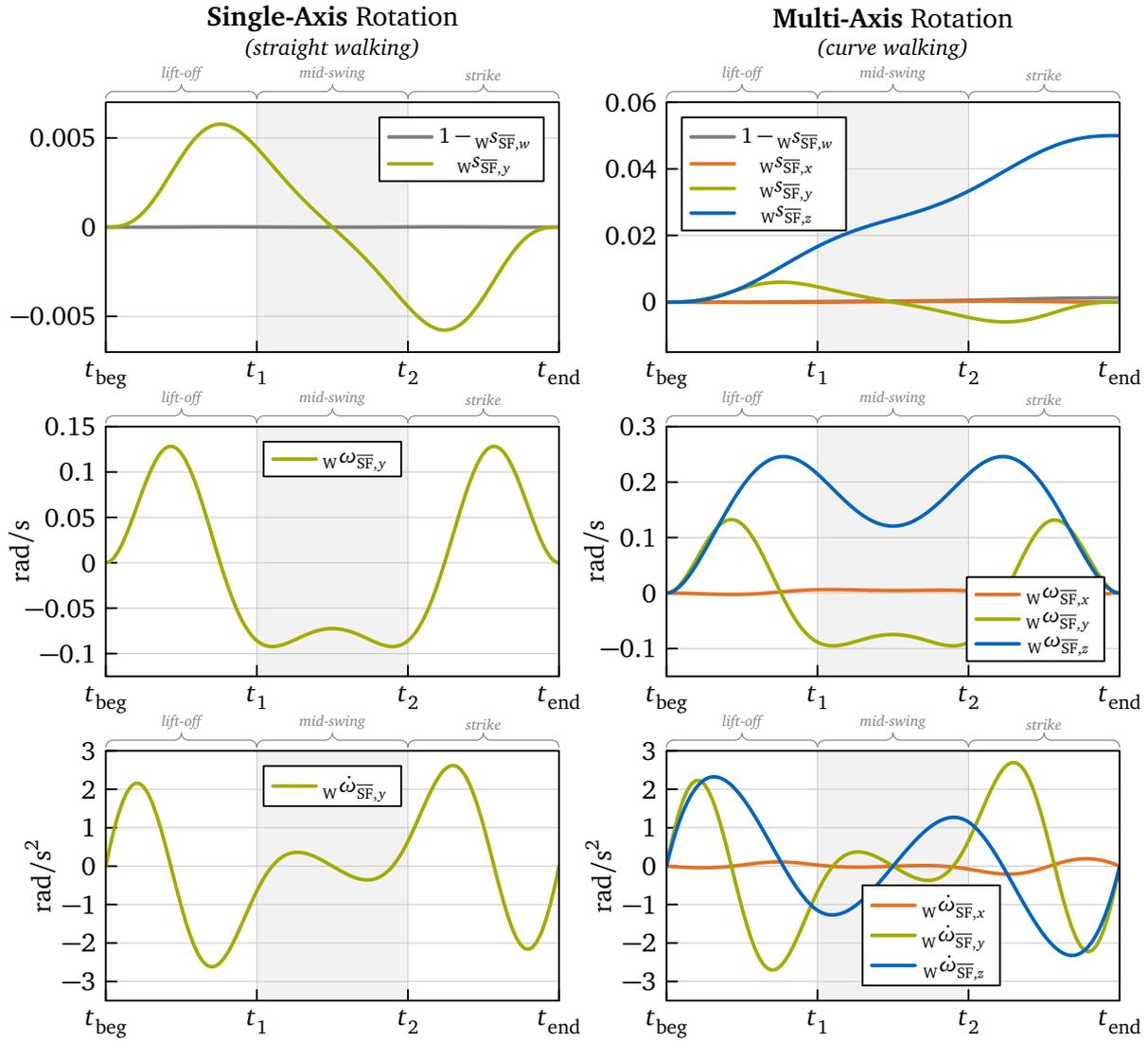


Figure 6.6: Quaternion trajectory describing the orientation of the swing foot ${}_{\text{W}}s_{\text{SF}}(t)$ for the case of single-axis rotation (left) and multi-axis rotation (right). The plots show the orientation ${}_{\text{W}}s_{\text{SF}}(t)$, angular velocity ${}_{\text{W}}\omega_{\text{SF}}(t)$, and angular acceleration ${}_{\text{W}}\dot{\omega}_{\text{SF}}(t)$ for an exemplary motion phase (partitioned into the three sub-phases *lift-off*, *mid-swing*, and *strike*). The plots on the left omit the x - and z -components as they are zero in the entire time interval. The trajectory is C^2 -continuous and has zero angular velocity and acceleration at the phase boundaries at t_{beg} and t_{end} .

Position – Timing In preparation of specifying the trajectories for horizontal⁸⁰ and vertical swing foot position, this paragraph introduces the therein used timing parameters. Same as for ${}_{\text{SF}}s_{\text{SF}}(t)$, the trajectories ${}_{\text{SF}}r_{\text{SF},xy}(t) \in \mathbb{R}^2$ and ${}_{\text{SF}}r_{\text{SF},z}(t) \in \mathbb{R}$ are partitioned into three interconnected segments. However, the duration of these segments is not uniform but instead derived from corresponding proportions which are controlled by the swing foot timing factor τ_{SF} . In

⁸⁰Note that the position of the swing foot ${}_{\text{SF}}r_{\text{SF}}$ is described relative to the TCP frame of the current stance foot. Thus, for inclined ground (e. g. on a ramp), ${}_{\text{SF}}r_{\text{SF},xy}$ and ${}_{\text{SF}}r_{\text{SF},z}$ actually do not represent the “horizontal” and “vertical” position (as defined by the world frame), but rather the components in “tangential” and “normal” direction (with respect to the ground). In the following, we still use the (inexact) terminology “horizontal” and “vertical” since it appears to be more intuitive.

particular, we introduce the proportions τ_{lag} and τ_{lead} specifying the duration of the first and last segment of ${}_{\text{SF}}\mathbf{r}_{\text{SF},xy}(t)$ and the proportions τ_{lift} and τ_{lower} specifying the duration of the first and last segment of ${}_{\text{SF}}\mathbf{r}_{\text{SF},z}(t)$. The proportions of the segments in the middle are simply given by $1 - \tau_{\text{lag}} - \tau_{\text{lead}}$ and $1 - \tau_{\text{lift}} - \tau_{\text{lower}}$. Note that these proportions are related to the duration of the motion phase $t_{\text{pha,dur}} = t_{\text{end}} - t_{\text{beg}}$ and not the duration of the governing transition. In order to define the proportions for both, the regular ($\tau_{\text{SF}} = 0$) and the collision-aware ($\tau_{\text{SF}} = 0.5$) parametrization, we use a linear relationship:

$$\begin{aligned} \tau_{\text{lag}}(\tau_{\text{SF}}) &:= (1 - \tau_{\text{SF}}) 0\% + \tau_{\text{SF}} 25\%, & (\tau_{\text{lag}}(0) = 0\% \text{ and } \tau_{\text{lag}}(0.5) = 12.5\%) \\ \tau_{\text{lead}}(\tau_{\text{SF}}) &:= (1 - \tau_{\text{SF}}) 8.75\% + \tau_{\text{SF}} 25\%, & (\tau_{\text{lead}}(0) = 8.75\% \text{ and } \tau_{\text{lead}}(0.5) = 16.875\%) \\ \tau_{\text{lift}}(\tau_{\text{SF}}) &:= (1 - \tau_{\text{SF}}) 45\% + \tau_{\text{SF}} 25\%, & (\tau_{\text{lift}}(0) = 45\% \text{ and } \tau_{\text{lift}}(0.5) = 35\%) \\ \tau_{\text{lower}}(\tau_{\text{SF}}) &:= (1 - \tau_{\text{SF}}) 45\% + \tau_{\text{SF}} 25\%. & (\tau_{\text{lower}}(0) = 45\% \text{ and } \tau_{\text{lower}}(0.5) = 35\%) \end{aligned} \quad (6.5)$$

The special choice $\tau_{\text{SF}} = 1$ represents a border case where the horizontal and vertical motion are completely decoupled (0 \rightarrow 25%: lift foot, 25 \rightarrow 75%: horizontal shift, 75 \rightarrow 100%: put the foot down). In combination with the terrain-based contact planner presented in Section 5.5, this would indeed guarantee collision-free swing foot motion. However, in favor of a smoother gait with lower EE accelerations, we instead use $\tau_{\text{SF}} = 0.5$ as a compromise for the collision-aware case. Nevertheless, since the swing foot timing factor is a component of the QPWT container (cf. Table 5.2), the user is free to choose an arbitrary $\tau_{\text{SF}} \in [0, 1]$.

Position – Roll-Off Compensation Since we describe the foot motion with respect to the TCP frame which is located in the center of the toe segment’s contact surface (cf. Figure 4.7), a non-zero foot rotation around the local y -axis (as it is used to implement the aforementioned roll-off motion) would possibly lead to a penetration of the ground. For this reason, we compute a compensation vector $\Delta_{\text{SF}}\mathbf{r}_{\text{SF}}(t)$ which can be applied to the TCP position ${}_{\text{SF}}\mathbf{r}_{\text{SF}}(t)$. The compensation vector is derived from simple geometric considerations where we assume straight walking, i. e., single-axis foot rotation around the local y -axis, such that the foot motion is restricted to the sagittal plane. In particular, we demand that the “critical” point on the foot (front of toe segment or rear of heel segment) is at the same 2D position as it would be for zero foot rotation:

$$\begin{aligned} ({}_{\text{SF}}\mathbf{a}_{\text{SF}}, \varphi_{\text{SF}})(t) &:= \text{axAng}({}_{\text{SF}}\mathbf{s}_{\text{SF}}(t)), & (\text{evaluate swing foot orientation as axis-angle pair}) \\ \varphi_{\text{SF},y}(t) &:= {}_{\text{SF}}a_{\text{SF},y}(t) \varphi_{\text{SF}}(t), & (\text{extract angle describing rotation around local } y\text{-axis}) \\ \Delta_{\text{SF}}\mathbf{r}_{\text{SF}}(t) &:= l_{\text{TCP,dist}}(t) \left[1 - \cos(\varphi_{\text{SF},y}(t)), 0, \sin(\varphi_{\text{SF},y}(t)) \right]^T \in \mathbb{R}^3. \end{aligned} \quad (6.6)$$

Here, $l_{\text{TCP,dist}}(t)$ denotes the lever arm, i. e., the (signed) distance of the critical point from the foot’s TCP, which switches between two (constant) values depending on the current tilting angle. Although the restriction to the sagittal plane may introduce some error in certain cases, the presented scheme is still exact for (straight) climbing ramps: all calculations in Equation 6.6 are performed in the SF FoR which is parallel to the (potentially inclined) ground. Note that the compensation vector not only contains a component in local z -direction to avoid penetration, but also a component in local x -direction which is meant to avoid slippage due to foot rotation.

A natural approach for exact compensation of the foot’s roll-off motion would be to add $\Delta_{\text{SF}}\mathbf{r}_{\text{SF}}(t)$ to ${}_{\text{SF}}\mathbf{r}_{\text{SF}}(t)$ for every time step sample. However, since the lever arm $l_{\text{TCP,dist}}(t)$ is subject to abrupt changes, this would result in an only \mathcal{C}^0 -continuous ${}_{\text{SF}}\mathbf{r}_{\text{SF}}(t)$. In order to keep the trajectory smooth, a non-exact approach is chosen where the compensation vector is applied only to the (interior) control-points of the horizontal and vertical swing foot motion, i. e., at

$$t_{\text{lag}} := t_{\text{beg}} + \tau_{\text{lag}} t_{\text{pha,dur}}, \quad t_{\text{lift}} := t_{\text{beg}} + \tau_{\text{lift}} t_{\text{pha,dur}}, \quad (6.7)$$

$$t_{\text{lead}} := t_{\text{end}} - \tau_{\text{lead}} t_{\text{pha,dur}}, \quad t_{\text{lower}} := t_{\text{end}} - \tau_{\text{lower}} t_{\text{pha,dur}}. \quad (6.8)$$

Position – Trajectory For generating ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}}}(t)$, we handle the horizontal and vertical components ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t)$ and ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t)$ individually (see also Footnote 80). As mentioned earlier, the motion of these components is split into three consecutive segments. For each segment, we describe the corresponding chunk of ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t)$ and ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t)$ through a quintic polynomial which has zero velocity and acceleration at the beginning and end of the segment. The polynomials are further specified through the control-points

$$\begin{aligned} {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{beg}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{beg},xy}^{\text{SF}}, & {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t_{\text{beg}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{beg},z}^{\text{SF}}, \\ {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{lag}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{beg},xy}^{\text{SF}} + \Delta_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{lag}}), & {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t_{\text{lift}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z,\text{max}}^{\text{SF}}, \\ {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{lead}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{end},xy}^{\text{SF}} + \Delta_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{lead}}), & {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t_{\text{lower}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z,\text{max}}^{\text{SF}}, \\ {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},xy}(t_{\text{end}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{end},xy}^{\text{SF}}, & {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}(t_{\text{end}}) &:= {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{end},z}^{\text{SF}}, \end{aligned} \quad (6.9)$$

where ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z,\text{max}}^{\text{SF}}$ denotes the maximum vertical position of the swing foot which is given by

$$\begin{aligned} {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z,\text{max}}^{\text{SF}} &:= \max\left({}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{beg},z}^{\text{SF}}, {}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},\text{end},z}^{\text{SF}}\right) \quad (\text{max. swing foot height at begin/end pose}) \\ &\quad + \max\left(\Delta_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}^{\text{SF}}(t_{\text{lift}}), \Delta_{\text{SF}}\mathbf{r}_{\overline{\text{SF}},z}^{\text{SF}}(t_{\text{lower}})\right) \quad (\text{max. roll-off compensation height}) \\ &\quad + h_{\text{step}} \quad (\text{default step height (3 cm) or custom step height provided by transition}) \end{aligned} \quad (6.10)$$

The step height h_{step} is only included in Equation 6.10 if the swing foot makes contact at t_{beg} and t_{end} . Thus, no clearance is added in situations where it is not necessary (e. g. for balancing on a single foot as special action). The resulting C^2 -continuous position trajectory ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}}}(t)$ is presented in Figure 6.7 for both timing parametrizations (regular and collision-aware).

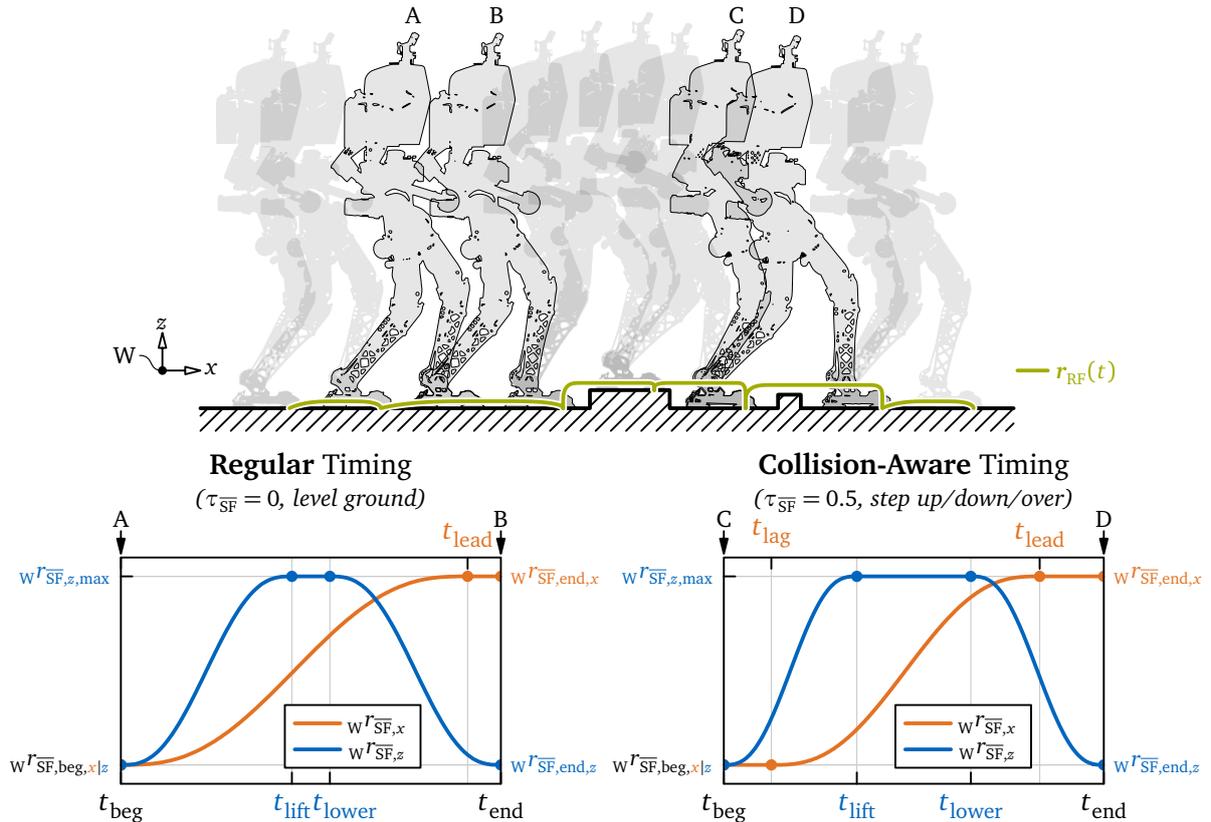


Figure 6.7: Normalized swing foot position trajectory ${}_{\text{SF}}\mathbf{r}_{\overline{\text{SF}}}(t)$ (here: $\overline{\text{SF}} = \text{RF}$) for the regular (left) and collision-aware (right) timing parametrization as used for walking on level ground and stepping up/down/over, respectively. The horizontal and vertical components consist of three consecutive quintic polynomial segments each. Note that $\tau_{\text{lag}}(\tau_{\text{SF}} = 0) = 0\%$ (cf. Equation 6.5), thus, the first segment of the horizontal component vanishes for regular timing. The trajectory is C^2 -continuous and has zero velocity and acceleration at all control-points (indicated by dots).

Discussion and Outlook The presented approach for generating foot motion is rather pragmatic. Despite its simplicity (e. g. when compared to optimization based approaches), it has proven to be effective for numerous maneuvers tested within real-world experiments. Although it is based on the previous trajectory generation system by BUSCHMANN, the proposed foot motion planner comes with several extensions which significantly increase the capabilities of the robot. Examples are the formulation of $s_{\overline{\text{SF}}}(t)$ as a QBSpline leading to a natural multi-axis rotation, the support for stepping on inclined ground enabling the robot to climb ramps, and the introduction of the swing foot timing factor $\tau_{\overline{\text{SF}}}$ which allows to blend between different trajectory shapes. Note that these extensions were developed after the publication of [3], thus, they have not been discussed therein.

A drawback of the current implementation is that collisions with the ground are not checked explicitly. However, in case an environment model is available, the local terrain information could be used to adapt the trajectories accordingly. Note that for autonomous walking the step height h_{step} is already adjusted to the terrain. A further step would be to find a minimum $\tau_{\overline{\text{SF}}}$ which still leads to a collision-free motion. This is left to future investigations.

6.8 Toe Motion Planner

Subsequent to planning the 6D motion of the feet, the toe joint trajectories $q_{\text{zfr}|}(t)$ are generated. As numerous real-world experiments with LOLA have shown, active toe joints can significantly improve walking performance for fast walking (large steps) and for stepping up or down [8, 101]. Especially stepping down represents a challenging maneuver for which the contact planner automatically chooses to make tiptoe contact in order to avoid kinematic limits (see Step 9 in the foothold refinement subroutine of the post-processor). The generation of $q_{\text{zfr}|}(t)$ is performed on sequence-level where we iterate over all motion phases and – depending on the situation – take the preceding or subsequent load switch phase into account. Certainly, if the considered foot is not subject to motion, the corresponding $q_{\text{zfr}|}(t)$ is kept constant.

Without Tiptoe Contact For each considered motion phase, we first check if the swing foot is supposed to make tiptoe contact at the beginning or the end of the governing transition. If neither is the case, a planning approach based on the previous trajectory generation system by BUSCHMANN is used. In particular, we first compute the stride length in x - and z -direction of the stance foot FoR:

$$l_{\text{str},x} := \text{SF}r_{\overline{\text{SF}},\text{end},x} - \text{SF}r_{\overline{\text{SF}},\text{beg},x} \quad \text{and} \quad l_{\text{str},z} := \text{SF}r_{\overline{\text{SF}},\text{end},z} - \text{SF}r_{\overline{\text{SF}},\text{beg},z} . \quad (6.11)$$

Based on $l_{\text{str},x}$, we distinguish between the case of walking forward ($l_{\text{str},x} \geq 0$) and walking backward ($l_{\text{str},x} < 0$). The toe joint trajectory is designed to extend over two phases: the currently investigated motion phase and either the preceding (for walking forward) or subsequent (for walking backward) load switch phase. The related time interval is given by $[t_{\text{beg}}, t_{\text{end}}]$ which (in general) does not correspond to the boundaries of the motion phase's governing transition. Within this time interval, the trajectory $q_{\text{zfr}|}(t)$ is formulated as a piecewise quintic polynomial spline consisting of two segments. The three control-points which split the trajectory into the two segments are chosen to $q_{\text{zfr}|}(t_{\text{beg}}) = 0$, $q_{\text{zfr}|}(t_{\text{max}}) = q_{\text{zfr}|,\text{max}}$, and $q_{\text{zfr}|}(t_{\text{end}}) = 0$ where we use

$$t_{\text{max}} := t_{\text{beg}} + 0.45(t_{\text{end}} - t_{\text{beg}}), \quad (\text{proportion driven by customizable WPG parameter}) \quad (6.12)$$

$$q_{\text{zfr}|,\text{max}} := \underbrace{\begin{cases} \frac{\pi}{18} \frac{|l_{\text{str},x}| - l_{12}}{0.8\text{m} - l_{12}} & \text{if } |l_{\text{str},x}| \geq 2l_{12}, \\ 0 & \text{else} \end{cases}}_{\text{large steps}} + \underbrace{\begin{cases} 3l_{\text{str},z} \text{ rad/m} & \text{if } l_{\text{str},z} > 0, \\ 0 & \text{else} \end{cases}}_{\text{stepping up}} . \quad (6.13)$$

Here, $q_{zfr||,max}$ denotes the maximum toe joint angle which is reached at t_{max} (45% of the entire trajectory). The first term depends on the stride length $l_{str,x}$ (relative to the foot length l_{12} , cf. Table 4.4) and effectively reduces the knee joint velocity for large steps [101]. The second term has been introduced by the author of this thesis and helps to avoid kinematic limits of the leg in the case of stepping up. As mentioned earlier, stepping down is already considered through setting a corresponding tiptoe contact. Depending on the side of the foot, the sign of $q_{zfr||,max}$ is flipped. Moreover, $q_{zfr||,max}$ is bounded by the limits of the joint. Figure 6.8 shows the resulting toe joint trajectory $q_{zfr||}(t)$ for the case of walking forward and backward.

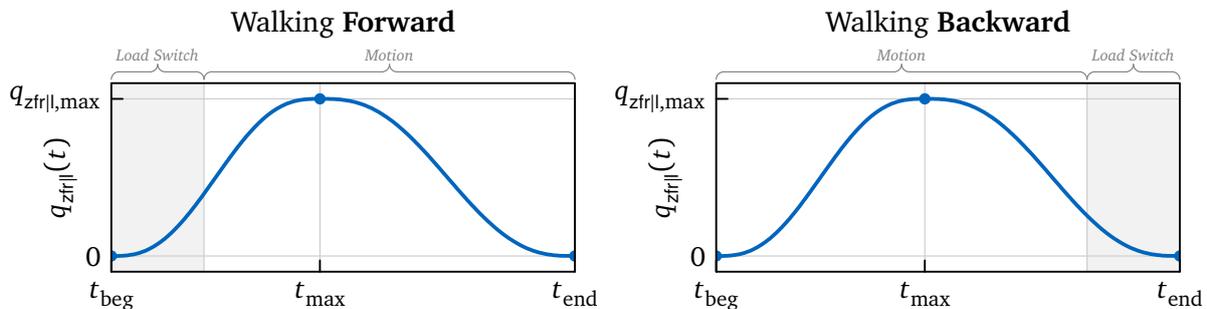


Figure 6.8: Normalized toe joint trajectory $q_{zfr||}(t)$ for walking forward (left) and backward (right) when no tiptoe contact is involved. The trajectory consists of two quintic polynomials with zero velocity and acceleration at all three control-points (indicated by dots). Depending on the walking direction, the trajectory extends to the preceding (walking forward) or subsequent (walking backward) load switch phase.

Note that due to the non-zero toe joint angle in the load switch phase, the contact area changes from the entire foot sole to the toe segment only. However, for the walking patterns considered within this thesis, it is still guaranteed that the planned ZMP remains within the SA.

With Tiptoe Contact If the swing foot is supposed to make tiptoe contact at the beginning and/or end of the governing transition, a much simpler toe joint trajectory is planned. In particular, we formulate $q_{zfr||}(t)$ as a single quintic polynomial with zero velocity and acceleration at its boundaries. In contrast to before, the trajectory covers only the currently investigated motion phase. The polynomial connects the desired toe joint angle at the beginning and end where we use a fixed target angle of 15° in case the foot is in tiptoe contact and 0° otherwise, i. e., for full or partial contact. Note that LOLA's capability of making explicit tiptoe contacts has been introduced within the context of the multi-contact revision. Same as for the heuristics given in the Equations 6.12 and 6.13, the presented parametrization has been derived empirically through numerous simulations and real-world experiments.

6.9 Hand Motion Planner

In stage nine of the planning pipeline, the trajectories ${}_{SF}r_h(t)$ (with $h \in \{RH, LH\}$) describing the position of the TCP frames of the hands are generated. Same as for the feet, planning the hand motion is performed on phase-level. In particular, we focus on the motion phases where the corresponding hand is subject to *task-space* motion. Note that in case the hand is assigned to the null-space throughout the entire phase, the corresponding task-space trajectory stored within the motion plan will not have any influence on the finally executed motion of the robot. Thus, in order to avoid unnecessary computations, the hand motion planner simply uses a constant ${}_{SF}r_h(t)$ in this case (same as for load switch phases). In contrast to the foot motion, we do not distinguish between horizontal and vertical components, but instead handle the x -, y -, and z -component of the trajectory in the exact same way.

Control-Points and Default Hand Position The shape of ${}_{SF}\mathbf{r}_h(t)$ is determined by a set of either two, three, or four control-points: $\{{}_{SF}\mathbf{r}_{h,beg}, {}_{SF}\mathbf{r}_{h,open}, {}_{SF}\mathbf{r}_{h,clo}, {}_{SF}\mathbf{r}_{h,end}\}$ with ${}_{SF}\mathbf{r}_{h,beg}$ and ${}_{SF}\mathbf{r}_{h,end}$ as the *mandatory* boundary control-points and ${}_{SF}\mathbf{r}_{h,open}$ and ${}_{SF}\mathbf{r}_{h,clo}$ as the *optional* interior control-points (cf. Figure 6.10). The boundary control-points describe the position of the hand at the beginning t_{beg} and end t_{end} of the phase. They are derived from ${}_{SF}\mathbf{r}_h$ stored in the corresponding robot pose of the governing transition. For special, non-gaited actions, these positions are typically hard-coded. An example is the multi-contact scenario for testing disturbance rejection shown in [20] @ $t=48s$. For action types describing gaited motion (fixed sequence, teleoperated, and autonomous walking), ${}_{SF}\mathbf{r}_h$ in the transition's end pose is set to the custom location specified in the corresponding QPWT, see Table 5.2. In case no custom location is specified, a *default* position is computed from the footholds and the related mean foot TCP frame MF. In order to realize a counter-swinging arm motion⁸¹, we specify the default hand position relative to the foot on the same (y -component) and the opposite (x - and z -component) side:

$${}_{MF}\mathbf{r}_{RH,def} := [{}_{MF}r_{LF,x} + \Delta x, {}_{MF}r_{RF,y} - \Delta y, {}_{MF}r_{LF,z} + \Delta z]^T \in \mathbb{R}^3, \quad (6.14)$$

$${}_{MF}\mathbf{r}_{LH,def} := [{}_{MF}r_{RF,x} + \Delta x, {}_{MF}r_{LF,y} + \Delta y, {}_{MF}r_{RF,z} + \Delta z]^T \in \mathbb{R}^3. \quad (6.15)$$

The shifts $\Delta x = 0.22$ m, $\Delta y = 0.2025$ m, and $\Delta z = 0.97$ m are derived from the static idle pose specified in Section 4.5.2. The resulting default hand poses are illustrated in Figure 6.9 left.

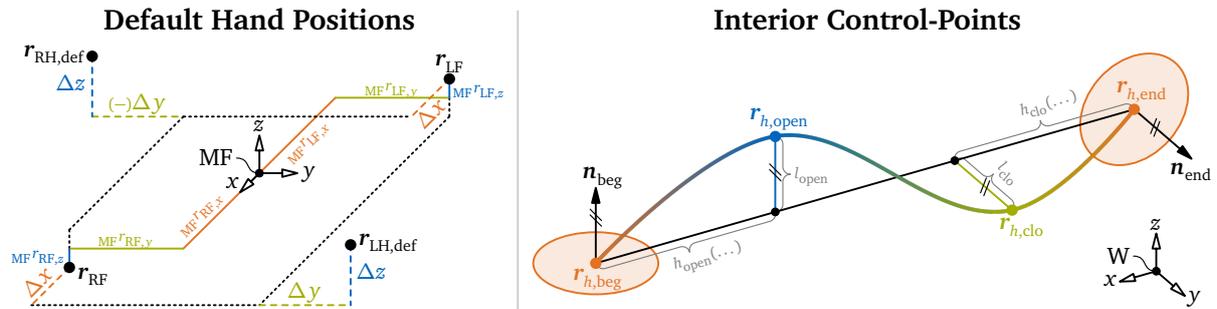


Figure 6.9: Derivation of default hand positions from the foot poses (left) and strategy for computing interior control-points in the case of contact at the beginning and end of the phase (right). The default hand positions are adopted from the static idle pose and realize a counter-swinging arm motion. The position of the interior control-points is primarily determined by the normal of the corresponding contact surface (indicated by orange discs).

In contrast to ${}_{SF}\mathbf{r}_{h,beg}$ and ${}_{SF}\mathbf{r}_{h,end}$, the interior control-points are optional. In particular, ${}_{SF}\mathbf{r}_{h,open}$ is only enabled if the hand is in contact at the beginning of the phase (contact *opens* at $t_{beg} + \varepsilon$). Similarly, ${}_{SF}\mathbf{r}_{h,clo}$ is only active in case the hand is supposed to make contact at t_{end} (contact *closes* at $t_{end} - \varepsilon$). The primary objective of the interior control-points is to prevent undesired contacts (alias “collisions”) during motion, i. e., within $t_{beg} < t < t_{end}$. For this reason, their position is characterized by a shift along the normal of the contact surface:

$${}_{SF}\mathbf{r}_{h,open} = {}_{SF}\mathbf{r}_{h,beg} + h_{open} ({}_{SF}\mathbf{r}_{h,end} - {}_{SF}\mathbf{r}_{h,beg}) + l_{open} \mathbf{n}_{beg}, \quad (6.16)$$

$${}_{SF}\mathbf{r}_{h,clo} = {}_{SF}\mathbf{r}_{h,end} - h_{clo} ({}_{SF}\mathbf{r}_{h,end} - {}_{SF}\mathbf{r}_{h,beg}) + l_{clo} \mathbf{n}_{end} \quad (6.17)$$

In case the hand is supposed to make contact, a non-zero (desired) external contact force ${}_{SF}\mathbf{F}_{h,ext}$ is available (as part of the external wrench ${}_{SF}\mathbf{W}_{h,ext}^h$ stored within the corresponding robot pose of the governing transition, cf. Table 5.1). Thus, we can simply extract the contact surface's normal from the contact force through $\mathbf{n}_{beg|end} = \text{norm}(\mathbf{F}_{h,ext,beg|end})$. The clearance, i. e., the magnitude of the shift in normal direction, is chosen to $l_{open} = l_{clo} = 7$ cm. In order to shape

⁸¹Note that a counter-swinging arm motion allows to reduce the angular momentum around the vertical axis which is induced by the swing leg motion. Besides certain fully-actuated humanoid robots, this strategy is typically also followed by passive-dynamic walkers such as the ones proposed by COLLINS et al. [119].

the path of ${}_{SF}r_h(t)$ as a smooth arc, we further shift the internal control-points along the vector connecting ${}_{SF}r_{h,beg}$ and ${}_{SF}r_{h,end}$ where we choose the proportions $h_{open} = h_{clo} = 1/3$. Figure 6.9 (right) visualizes the proposed strategy for placing the interior control-points.

Trajectory The trajectory ${}_{SF}r_h(t)$ is formulated as a smooth quintic spline (C^4 -continuous, see Algorithm G.1) which passes through the presented control-points and has zero velocity and acceleration at its boundaries. Depending on the contact state at the beginning and end of the phase, this results in a chain of either one, two, or three interconnected segments, see Figure 6.10. While the boundary control-points are linked to t_{beg} and t_{end} , we choose the proportions $\tau_{open} = \tau_{clo} = 1/3$ such that

$$t_{open} := t_{beg} + \tau_{open} t_{pha,dur}, \quad t_{clo} := t_{end} - \tau_{clo} t_{pha,dur} \quad (6.18)$$

define the timing of the optional interior control-points. In case of contact at the beginning and end of the considered motion phase, this leads to a uniform partitioning into the three sub-phases *open*, *move*, and *close* as shown in Figure 6.10 (bottom left).

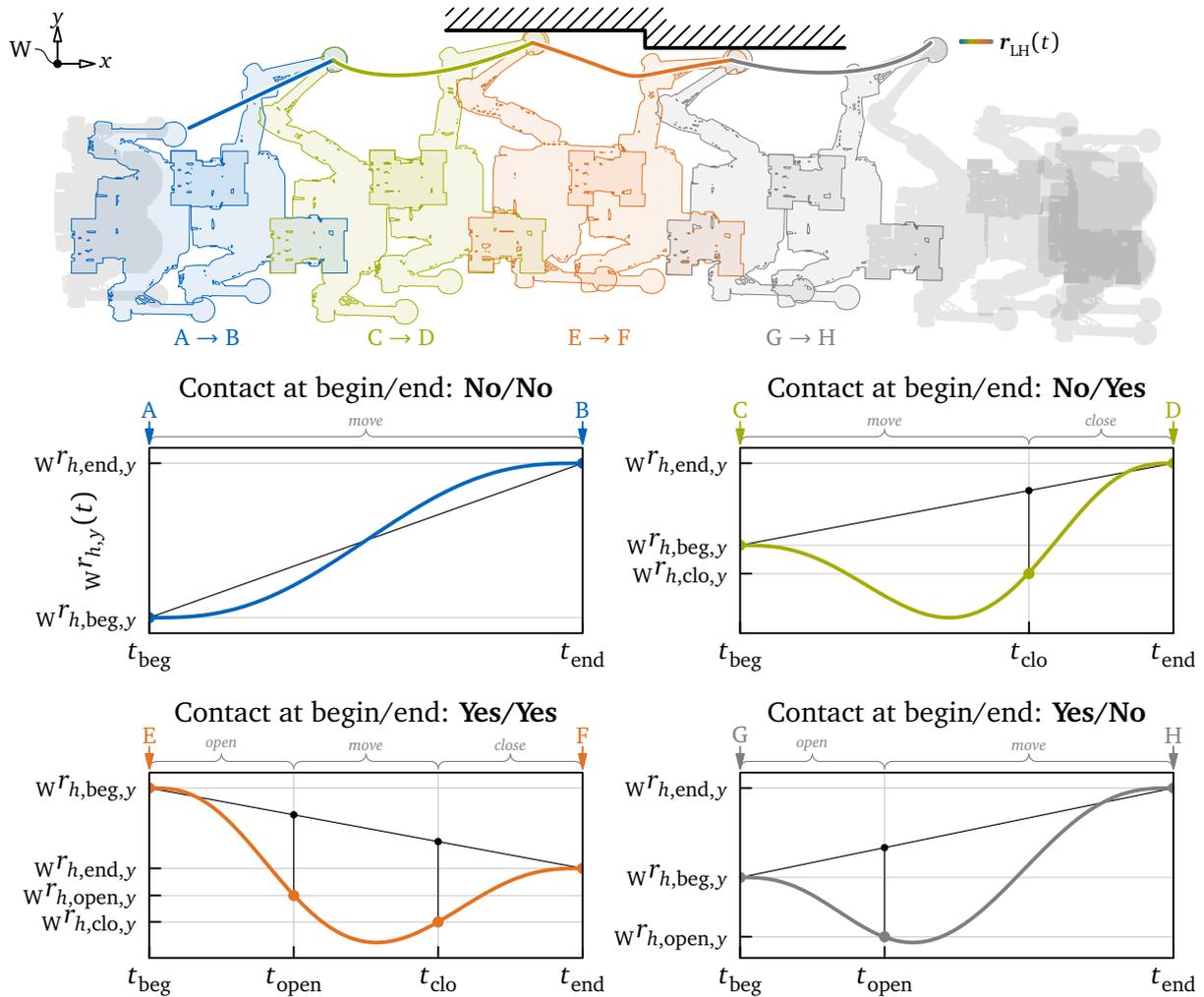


Figure 6.10: Normalized hand motion trajectory ${}_{W}r_h(t)$ describing the position of the TCP frame. The four plots show all possible cases based on the contact state at the beginning t_{beg} and end t_{end} of the phase. The trajectory is described by a smooth (C^4 -continuous) quintic spline passing through either two, three, or four control-points (indicated by dots) depending on the particular contact case. The velocity and acceleration are set to zero at the boundary control-points while they are (in general) non-zero at the interior control-points. For a 3D visualization of the planned hand trajectory, see the video [18 @t=5m2s].

Discussion and Outlook The proposed method for generating ${}_{\text{SF}}\mathbf{r}_h(t)$ appears to be rather simple. Nevertheless, by using the presented parametrization, the resulting hand motion has proven to be effective for all multi-contact scenarios considered within this thesis. Still, the used approach for avoiding (unintended) collisions based on the surface normal only works for rather primitive objects such as planar walls or tables. A reasonable future extension would be to optimize the parameters $l_{\text{open}}, l_{\text{clo}}, h_{\text{open}}, h_{\text{clo}}, \tau_{\text{open}},$ and τ_{clo} on the basis of the environment model. This would enable interactions with more complex objects while still using a parameterized (hence computational efficient) formulation. However, this extension would not be capable of avoiding collisions with other parts of the arm (e. g. the elbow) since only the position of the hand is known within our task-space based planning framework.

6.10 Head Orientation Planner

In stage ten of the planning pipeline, the rotation of the head, i. e., the trajectory ${}_{\text{SF}}\mathbf{s}_{\text{VTCP}}(t)$ describing the orientation of the VTCP frame, is generated. For simplicity, we use the same procedure as for planning the upper body rotation ${}_{\text{SF}}\mathbf{s}_{\text{UB}}(t)$ (cf. Section 6.6). The control-quaternions ${}_{\text{SF}}\mathbf{s}_{\text{VTCP}}$ – which are again extracted from the corresponding begin and end pose of the governing transition – are indeed the only difference. Note that within a robot pose (cf. Table 5.1), we formulate the head’s orientation as a quaternion. In contrast, we directly use the joint angles q_{vp} and q_{vt} within QPWTs which allows a more intuitive specification within (human readable) input files. In order to obtain ${}_{\text{SF}}\mathbf{s}_{\text{VTCP}}$ from q_{vp} and q_{vt} , we use the relationship

$${}_{\text{SF}}\mathbf{s}_{\text{VTCP}}({}_{\text{SF}}\mathbf{s}_{\text{UB}}, q_{\text{vp}}, q_{\text{vt}}) := {}_{\text{SF}}\mathbf{s}_{\text{UB}} \otimes \text{quat}(\mathbf{e}_z, q_{\text{vp}}) \otimes \text{quat}(\mathbf{e}_y, -q_{\text{vt}}). \quad (6.19)$$

Note that the task-space vector \mathbf{x} and the task-space velocity vector \mathbf{v} (cf. Table 4.1) describe the head’s rotation through $q_{\text{vp}}, q_{\text{vt}}$ and $\dot{q}_{\text{vp}}, \dot{q}_{\text{vt}}$, respectively. Thus, samples of the quaternion trajectory ${}_{\text{SF}}\mathbf{s}_{\text{VTCP}}(t)$ have to be converted before they are transmitted to the SIK module. This task is assigned to the *Stream Processor* which will be described in Section 6.16.

6.11 Task-Space Selection Factor Planner

Subsequent to generating the position and orientation trajectories for the EEs, the motion generator plans the task-space selection factors $\xi_h(t)$ (with $h \in \{\text{RH}, \text{LH}\}$) which blend the hands between the task- and null-space. Note that the actual blending is realized by the SIK module and has already been described in Section 4.6. The generation of $\xi_h(t)$ is performed on phase-level. Changes are only allowed within motion phases in which the corresponding hand is subject to task-space motion (similar to the hand motion planner). We formulate $\xi_h(t)$ as a chain of three consecutive quintic polynomials with zero first- and second-order derivatives at the boundaries of each segment resulting in a \mathcal{C}^2 -continuous signal, see Figure 6.11 (top). The control-points are set to $\xi_h(t_{\text{beg}}) = \xi_h(t_{\text{lag}}) = \xi_{h,\text{beg}}$ and $\xi_h(t_{\text{lead}}) = \xi_h(t_{\text{end}}) = \xi_{h,\text{end}}$ where $\xi_{h,\text{beg|end}}$ are provided by the corresponding begin and end pose of the governing transition (cf. Table 5.1). With the relationship

$$t_{\text{lag}} := t_{\text{beg}} + \tau_{\text{lag}} t_{\text{pha,dur}} \quad \text{and} \quad t_{\text{lead}} := t_{\text{end}} - \tau_{\text{lead}} t_{\text{pha,dur}}, \quad (6.20)$$

we implement a timing similar to the horizontal foot motion (cf. Figure 6.7). In case of fading from null- to task-space ($\xi_h: 0 \rightarrow 1$), we use the proportions $\tau_{\text{lag}} = 0$ and $\tau_{\text{lead}} = 0.1$. For blending from task- to null-space ($\xi_h: 1 \rightarrow 0$), we use the parametrization $\tau_{\text{lag}} = 0.1$ and

$\tau_{\text{lead}} = 0.5$. Same as for the other timing parameters of the WPG, these values have been chosen empirically based on numerous conducted simulations and real-world experiments.

The proposed scheme is applied for both hands, i. e., $\xi_{\text{RH}}(t)$ and $\xi_{\text{LH}}(t)$, individually. Note that the resulting task-space selection vector $\xi = [\xi_{\text{RH}}, \xi_{\text{LH}}]^T \in \mathbb{R}^2$ does not only affect the finally executed motion of the hands (Figure 6.11, center), but also the mass distribution of the five-mass model (Figure 6.11, bottom) as described in Section 4.5.2.

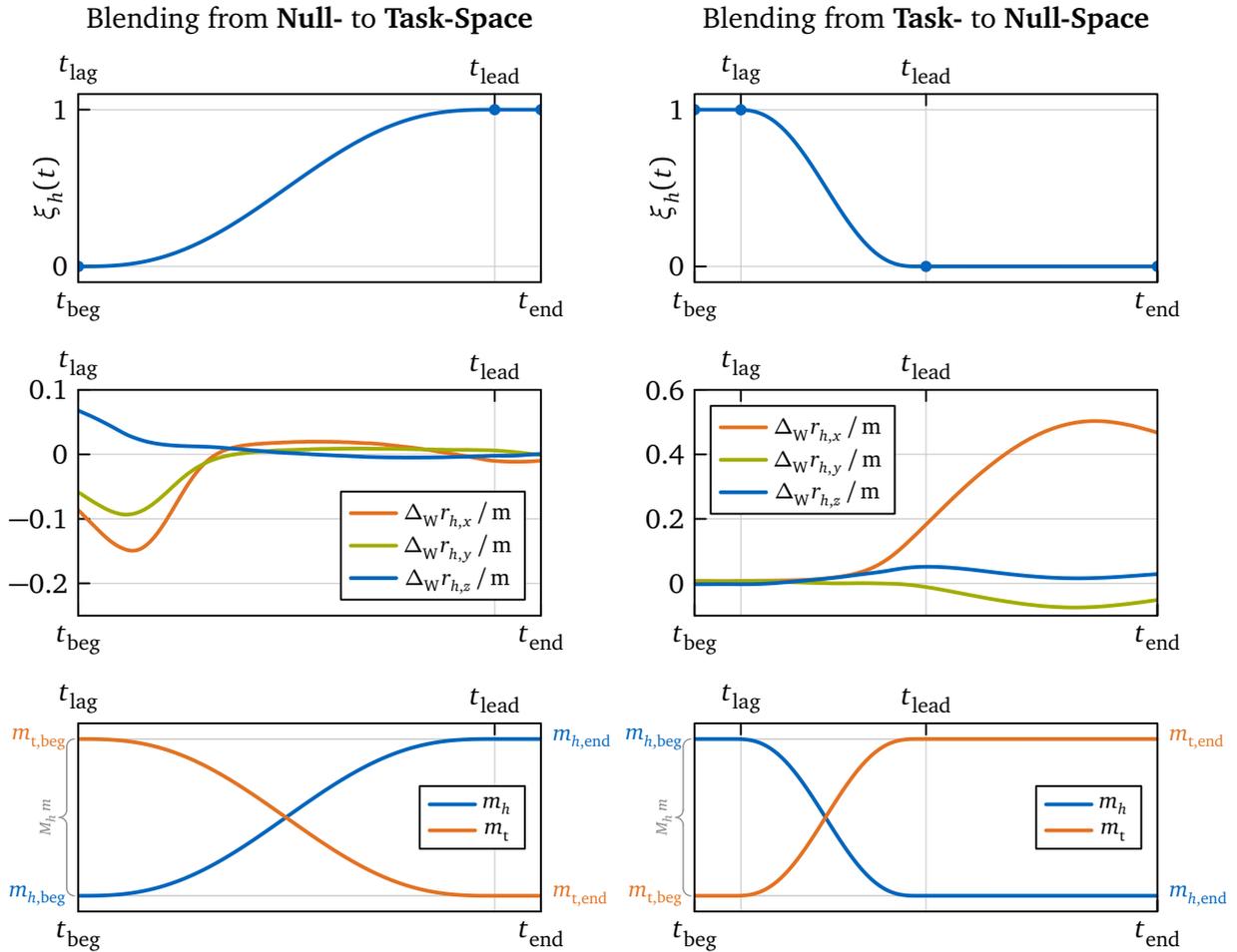


Figure 6.11: Blending the TCP position of the hand $h \in \{\text{RH}, \text{LH}\}$ from null- to task-space (left) and from task- to null-space (right). Top: task-space selection factor $\xi_h(t)$ as \mathcal{C}^2 -continuous signal consisting of three interconnected quintic polynomials. The first- and second-order derivatives are zero at all control-points (indicated by dots). Center: difference vector $\Delta_W r_h(t)$ between the planned (output of WPG) and executed (estimation according to Section 4.5.3) hand motion for the scenario shown in Figure 6.10 (the plots on the left side are related to the section A \rightarrow B). Bottom: normalized hand and torso masses within the five-mass model according to Equation 4.2. Note that the total mass remains constant (cf. Equation 4.3).

6.12 Load Factor Planner

In the next stage of the planning pipeline, the load factor signal $\gamma_e(t)$ is generated for each EE $e \in \{\text{RF}, \text{LF}, \text{RH}, \text{LH}\}$. As already explained in Footnote 57 and Section 4.6, $\gamma_e(t)$ describes the (planned) contact transitions for the particular EE e which is used by the SIK module in order to detect early- or late-contact situations and blend between position- and force-control accordingly. The compound $\gamma = [\gamma_{\text{RF}}, \gamma_{\text{LF}}, \gamma_{\text{RH}}, \gamma_{\text{LH}}]^T \in \mathbb{R}^4$ is called *load vector* [401, p. 51].

The load factors are planned on sequence-level and are (almost) independently of each other. Indeed, apart from $\gamma_e \in [0, 1]$, the only constraint imposed by the SIK module is $\gamma_{RF} + \gamma_{LF} = 1$ which implies that there has to be at least one foot in contact (no flight phases). For switching the load factor of an EE, we make use of the *load switch* phases (hence their name). In particular, we iterate over all load switch phases of the given sequence and generate two control-points $\gamma_e(t_{\text{beg}}) = \gamma_{e,\text{beg}}$ and $\gamma_e(t_{\text{end}}) = \gamma_{e,\text{end}}$ for each. In order to obtain a C^2 -continuous signal representing a smooth contact transition, we connect consecutive control-points by a quintic polynomial with zero first- and second-order derivatives at its boundaries. Moreover, we extend the curve to the preceding and subsequent motion phase if possible. In particular, we use

$$t_{\text{beg}} := t_{\text{pha,beg}} - \frac{7}{16} t_{\text{pha,dur}} \quad \text{and} \quad t_{\text{end}} := t_{\text{pha,end}} + \frac{7}{16} t_{\text{pha,dur}}. \quad (6.21)$$

The special fraction of the load switch phase duration $t_{\text{pha,dur}} = t_{\text{pha,end}} - t_{\text{pha,beg}}$ represents a symmetric offset which ensures that $\dot{\gamma}_e(t_{\text{pha,beg}} + t_{\text{pha,dur}}/2)$, i. e., the “tangent” at the center of the load switch phase, forms the linear connection from $\gamma_{e,\text{beg}}$ at $t_{\text{pha,beg}}$ to $\gamma_{e,\text{end}}$ at $t_{\text{pha,end}}$, see Figure 6.12 left. In other words, the polynomial is designed to smooth-out the “corners” of a virtual, piecewise linear signal. In case there is no preceding or subsequent motion phase, we simply use $t_{\text{beg}} = t_{\text{pha,beg}}$ or $t_{\text{end}} = t_{\text{pha,end}}$ instead. Furthermore, the load factor planner implements the necessary logic to avoid duplicate control-points or overlapping polynomial segments (e. g. in case a very short motion phase lies between two long load switch phases).

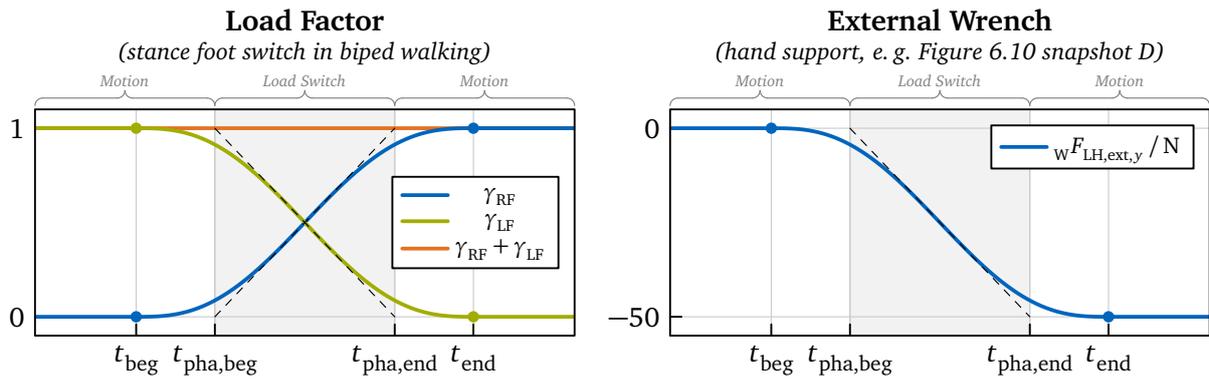


Figure 6.12: Planned load factors of the feet for switching the stance in biped walking (left) and planned external wrench (here: lateral force component) for an exemplary multi-contact scenario (right). The signals are formulated as a quintic polynomial which covers the entire load switch phase and extends to the preceding and subsequent motion phase. For best visual presentation, only the first/last 25% of the motion phases are shown. At the control-points (indicated by dots), the first- and second-order derivatives are zero. The tangents of the signals at the center of the load switch phase are plotted as dashed lines.

The boundary values $\gamma_{e,\text{beg}}$ and $\gamma_{e,\text{end}}$ are derived from the robot pose at the beginning of the corresponding transition. In case the EE is in contact, we check if it is subject to task-space motion in the preceding motion phase. If it is, we set $\gamma_{e,\text{beg}} = 0$. Otherwise (or if there is no preceding motion phase), we use $\gamma_{e,\text{beg}} = 1$. Similarly, we determine if the EE is subject to task-space motion in the subsequent motion phase and set $\gamma_{e,\text{end}} = 0$ if it is or $\gamma_{e,\text{end}} = 1$ if it is not (or if there is no subsequent motion phase). In case the EE is not in contact, $\gamma_{e,\text{beg}} = \gamma_{e,\text{end}} = 0$ is used. In order to satisfy the aforementioned constraint on the load factors of the feet, we check if $\gamma_{RF,\text{beg}} = \gamma_{LF,\text{beg}} = 1$ (e. g. static standing). If this is the case, we simply change the factors to $\gamma_{RF,\text{beg}} = \gamma_{LF,\text{beg}} = 1/2$. The same applies to $\gamma_{RF,\text{end}}$ and $\gamma_{LF,\text{end}}$. Note that the transition planner presented in Section 5.4 guarantees that at least one foot is in contact with the ground such that the invalid cases $\gamma_{RF,\text{beg}} = \gamma_{LF,\text{beg}} = 0$ and $\gamma_{RF,\text{end}} = \gamma_{LF,\text{end}} = 0$ do not occur.

6.13 External Wrench Planner

Since the application of an external (multi-contact) wrench is closely related to the contact state at the corresponding hand $h \in \{\text{RH, LH}\}$, it seems reasonable to design ${}_{\text{SF}}W_{h,\text{ext}}^h(t)$ similar to the load factor $\gamma_h(t)$. Indeed, the signal for the external wrench is generated in the exact same way, however, with $\mathbf{0} \in \mathbb{R}^6$ and ${}_{\text{SF}}W_{h,\text{ext}}^h$ (extracted from the robot pose at the beginning of the corresponding transition, cf. Table 5.1) instead of 0 and 1 as possible control-point values. Apart from this, the timing and logic (checking for contact of the hand and if it is subject to task-space motion in the preceding/subsequent motion phase) is identical. Figure 6.12 (right) presents the resulting signal for an exemplary scenario similar to the one shown in Figure 6.10 (load switch phase starting at snapshot D or equivalently F). While planning the external wrench signals similar to the load factors has shown to be effective in both, simulation and real-world experiments, one might consider to switch to a more sophisticated approach in future (e. g. taking the surface properties and the expected deformation of the object/arm into account).

6.14 Reduced Model Torso (RMT) Planner

Since the ZMP, upper body orientation, and EE trajectories are set, we can finally generate ${}_{\text{W}}r_t(t)$ which describes the motion of the RMT, i. e., the five-mass model's "virtual" torso mass m_t (cf. Figure 4.7). In the following two sections, the vertical and horizontal components are investigated separately. While planning ${}_{\text{W}}r_{t,z}(t)$ involves the simplified kinematic model of the leg in order to maintain kinematic feasibility, the generation of ${}_{\text{W}}r_{t,x|y}(t)$ is primarily based on the five-mass model to accomplish dynamic feasibility. As already mentioned in Section 4.5.4, the computation of ${}_{\text{W}}r_{t,z}(t)$ depends on ${}_{\text{W}}r_{t,x|y}(t)$ and vice versa. In order to resolve this mutual dependency, the subroutines for generating vertical and horizontal RMT motion are embedded within an iteration featuring two cycles, see Figure 6.13. The first cycle delivers an initial solution which is used in the second cycle to compute the final RMT trajectory.

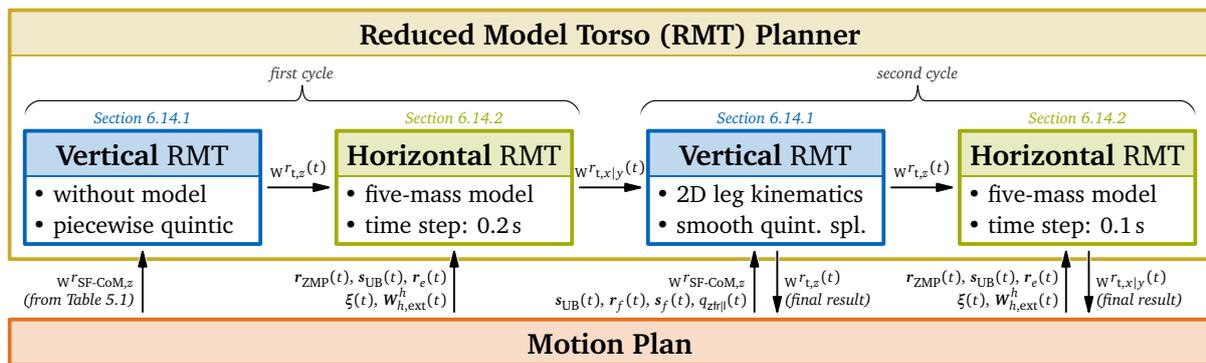


Figure 6.13: Overview of the *RMT Planner* generating the vertical and horizontal components of ${}_{\text{W}}r_t(t)$ through dedicated subroutines which are embedded within an iteration to resolve their mutual dependency. The first cycle creates an initial guess which is then refined in the second cycle. The motion plan provides the input data such as ZMP, upper body orientation, and EE trajectories. Moreover, it stores the final RMT trajectory such that it is available to the subsequent stages in the planning pipeline.

6.14.1 Vertical RMT Planner

Within both cycles of the iteration, the vertical RMT component ${}_{\text{W}}r_{t,z}(t)$ is planned on sequence-level. In particular, we generate a sequence of control-points which are chosen such that a

certain “desired” CoM height is tracked. The desired CoM height has already been determined by the transition planner which stored the corresponding value within the robot poses (see the component $wr_{\text{SF-CoM},z}$ in Table 5.1). Note that for gaited action types, $wr_{\text{SF-CoM},z}$ includes an (optional) automatic CoM lowering based on the step length as described in Section 5.4. In order to obtain the desired RMT height, we make use of the five-mass model:

$$wr_{t,z,\text{des}} := \frac{1}{m_t} \left[m (wr_{\text{SF},z} + wr_{\text{SF-CoM},z}) - m_f (wr_{\text{RF},z} + wr_{\text{LF},z}) - m_{\text{RH}} wr_{\text{RH},z} - m_{\text{LH}} wr_{\text{LH},z} \right] \quad (6.22)$$

where the positions of the EEs and the task-space selection vector (defines the mass distribution, cf. Equation 4.2) are extracted from the corresponding robot pose. Indeed, the vertical RMT planner uses the five-mass model only to derive the desired RMT height from the (more intuitive) $wr_{\text{SF-CoM},z}$. Besides, this submodule does not involve any kind of dynamics.

First Cycle Within the first cycle of the vertical RMT planner, no knowledge on the horizontal RMT motion is available. Since this prevents us from evaluating kinematic limits based on our simplified leg model, we choose a rather simple approach instead where we generate $wr_{t,z}(t)$ solely based on the desired RMT height. In particular, we create a control-point for every robot pose, i. e., at the beginning/end of each transition in the motion plan. Through Equation 6.22, we obtain the corresponding $wr_{t,z,\text{des}}$. Consecutive control-points are then connected through a quintic polynomial with zero velocity and acceleration at its boundaries which ensures C^2 -continuity. Note that since we specified the desired CoM height relative to the vertical position of the current stance foot ($wr_{\text{SF-CoM},z} = wr_{\text{CoM},z} - wr_{\text{SF},z}$), even the initial solution for $wr_{t,z}(t)$ is capable of handling ground height changes (such as in climbing stairs) properly. However, kinematic limits are not considered at all. The resulting initial solution for $wr_{t,z}(t)$ is shown in Figure 6.14 (dashed lines). Indeed, the initial solution is not feasible in the depicted scenario, i. e., the robot falls due to hitting the kinematic limits of the knee joint. This motivates the refinement of $wr_{t,z}(t)$ through a more sophisticated algorithm in the second cycle.

Second Cycle Within the second cycle of the vertical RMT planner, a (preliminary) solution for the horizontal component $wr_{t,x|y}(t)$ is available. From the previous stages of the planning pipeline, we further have access to the upper body orientation, the 6D poses of the feet (TCP frames), and the toe joint angles. Considering our planar model describing the leg kinematics, we can use this information to find the yet unknown (joint-space) angles φ_{hf} , φ_{kf} , and φ_{sf} (highlighted in orange in Figure 4.7) for a given RMT height. For this purpose, we project all 3D quantities to an approximation of the current sagittal plane which is defined by the x -axis of the current upper body frame UB and the z -axis of the world frame W. Note that this also includes a projection of the toe joint angles. While one can easily derive an analytic solution to this inverse kinematics problem (see HILDEBRANDT et al. [8]), it is much more difficult to find an analytic expression for the lower/upper bound of $wr_{t,z}$ which respects the individual joint limits for φ_{hf} , φ_{kf} , and φ_{sf} . For the sake of simplicity, an iterative algorithm was preferred instead to determine the vertical RMT boundaries. In a first attempt, several gradient-based optimization methods have been tested. Due to strong nonlinearities, these algorithms became easily stuck in local optima so that a sampling-based approach based on the equations provided in Appendix E was used in the end to find the global optima. Here we exploit the fact that φ_{hf} and φ_{kf} are implicitly given through φ_{sf} (see Equations E.3 and E.4), hence, only φ_{sf} has to be sampled which is of moderate cost. After evaluating the minimum and maximum RMT heights for both legs separately, they are combined to the “total” lower/upper bound respecting the kinematic limits of both legs simultaneously. The resulting combined boundaries for traversing a platform are shown in Figure 6.14 for two cases. *Case I* describes a walking pattern where every step is subject to full contact and the robot does not adapt its toe joint motion to the ground height change (disabled second term in Equation 6.13). This leads to tight bounds for stepping up and

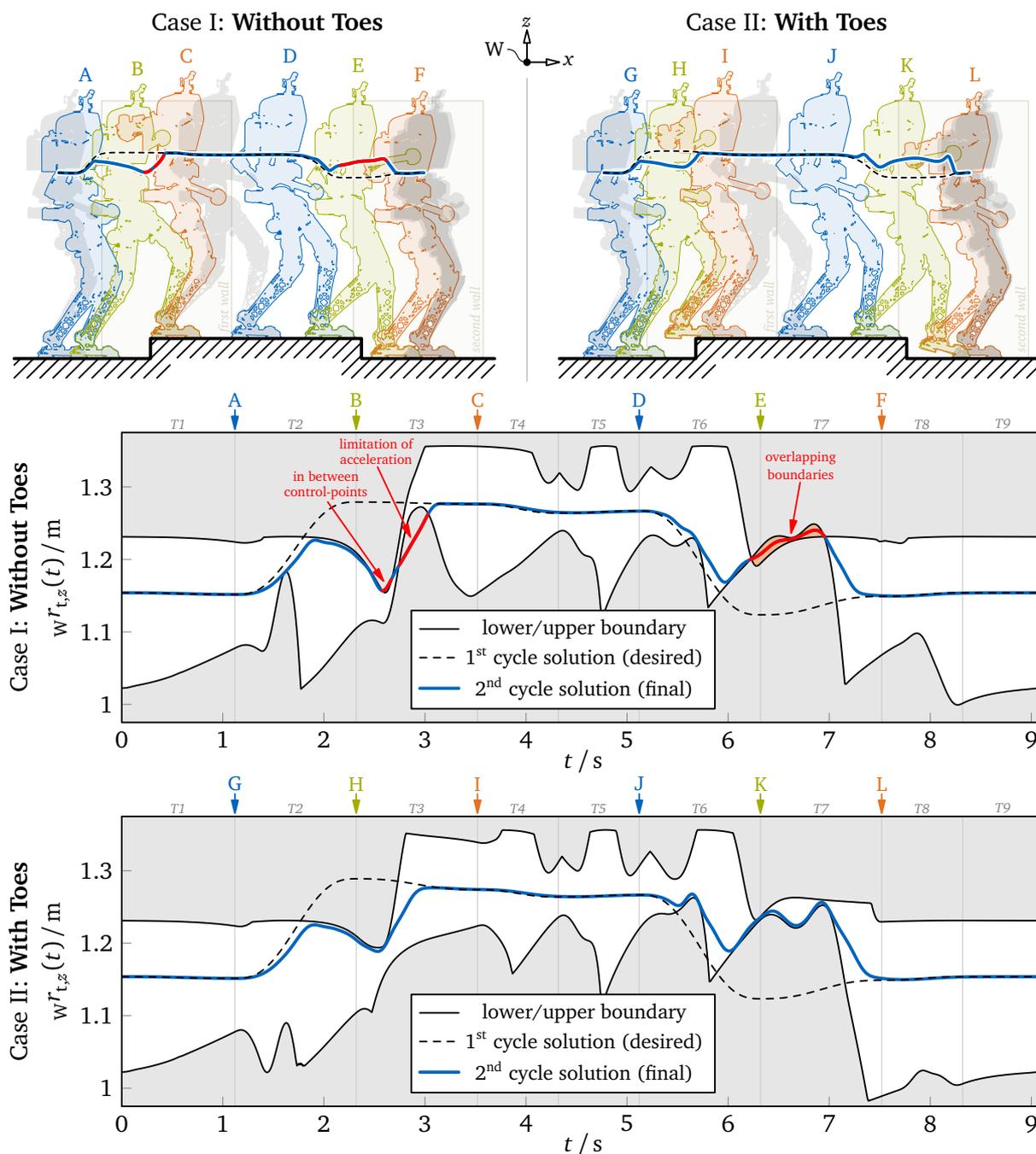


Figure 6.14: Vertical component of the RMT trajectory $w^r_{t,z}(t)$ for a multi-contact scenario similar to [20] (stepping up and down platform of 12.5 cm height). The plots show $w^r_{t,z}(t)$ for a walking pattern without using toes (Case I; hits predicted kinematic limits) and with using toes (Case II; partial contact at snapshot H and tiptoe contact at snapshot K). The lower/upper boundary is derived from the simplified leg kinematics model shown in Figure 4.7. The time axis is partitioned to highlight the transitions $T1$ to $T9$ of the sequence. The 1st cycle solution represents a chain of quintic polynomials (one for each transition) with zero velocity and acceleration at the segment's boundaries. The 2nd cycle solution is described by a C^4 -continuous quintic spline with uniformly distributed control-points ($\Delta t \approx 0.1$ s) and zero velocity and acceleration at the beginning and end of the sequence.

even overlapping limits for stepping down the platform. In Case II, partial contact is made for stepping up while tiptoe contact is triggered for stepping down (as it would be planned by the autonomous contact planner presented in Section 5.5). In combination with the adaptive toe joint trajectory (enabled second term in Equation 6.13), this clearly enlarges the feasible domain

of ${}_W r_{t,z}$ and eliminates overlaps entirely.

In contrast to the first cycle, the control-points of the final trajectory ${}_W r_{t,z}(t)$ are distributed uniformly across the entire sequence using a constant time step size of $\Delta t \approx 0.1$ s. For each time step, the corresponding control-point is chosen to track the desired RMT height from the first cycle. In particular, we implement a first-order low-pass filter⁸² which takes the series of desired vertical RMT positions (sampled at the time steps) as input. The output of the filter is projected to the kinematically feasible domain where we apply an additional (constant) safety margin of 5 mm to the combined lower/upper bound of the RMT height. Note that for challenging maneuvers, the boundaries might overlap (cf. *Case I* at $t = 6.6$ s in Figure 6.14). Although overlapping boundaries indicate that the planned motion is not feasible in theory, it might still be executable in practice. This is because the proposed kinematic model of the leg is conservative in neglecting several DoF (especially the joint ba). In case of an overlap, the control-point is set to the mean value of the minimum and maximum RMT height which is meant to minimize the distance to the “real” limits. Since vertical RMT boundaries are evaluated and checked only at the control-points, it is possible that kinematic limits are hit in between (cf. *Case I* at $t = 2.6$ s in Figure 6.14). Because the chosen time step size Δt is small, (theoretically) infeasible sections of ${}_W r_{t,z}(t)$ are short and thus neglected. As a post-processing step, the control-points are modified such that the vertical acceleration of the RMT is limited to $|\ddot{{}_W r_{t,z}}(t)| \leq g/2$. For rapid changes of the projected RMT height (cf. *Case I* at $t = 2.9$ s in Figure 6.14), this prevents the robot from losing contact with the ground (${}_W \ddot{r}_{\text{CoM},z}(t) < g$). The resulting sequence of control-points is finally interpolated by a C^4 -continuous quintic spline (see Algorithm G.1) with zero velocity and acceleration at the beginning and end.

Discussion Although the simplified model of the leg is almost identical to the previous one proposed by HILDEBRANDT et al. in [8], the vertical RMT planner presented within this thesis uses different approaches for both, evaluating the kinematic limits and using them to plan a feasible trajectory ${}_W r_{t,z}(t)$. Based on the solution of NISHIWAKI and KAGAMI in [325], the method presented in [8] assumes for example a fully stretched leg, i. e., $\varphi_{\text{kf}} = 0$ for estimating the maximum RMT height, which is not exact in general. In contrast, the vertical RMT planner proposed within this thesis evaluates the full kinematic chain of the simplified leg model taking the (preliminary) solution of the horizontal component ${}_W r_{t,x|y}(t)$ into account. Moreover, moving from the (comparatively expensive) optimization suggested in [8] to a simple “projection” to the boundaries (in combination with low-pass filtering of the desired height and limitation of the vertical acceleration) not only reduces the computational cost significantly, but also leads to a smoother trajectory in “unproblematic” regions (compare for example Transition 4 and 5 in Figure 6.14 with the corresponding section of Figure 10 in [8]). The video [23 @t=2m34s] gives a qualitative comparison of the previous solution (from [8]) and the new approach as presented by SEIWALD et al. in [3]. The most recognizable difference is the reduced arm motion (non multi-contact scenario) which indicates that it is easier for the SIK module to track the planned CoM trajectory (less null-space action is required). From a quantitative point of view, the maximum upper body inclination error could be reduced by more than 40% while the kinematic reserves, i. e., the minimum distance to the joint limits, could be increased by 54% for the knee joint kfr| and by 222% for the ankle joint sfr| (see [3] for details). Note that [3] described a preliminary version of the WPG which does not include all extensions presented within this thesis. Since the main difference lies in the usage of the toes, the vertical RMT trajectory from [3] is (roughly) equivalent to *Case I* in Figure 6.14. With the most recent extensions proposed in this thesis, the overall robustness and stability as well as the kinematic reserves during challenging maneuvers have been further improved (see *Case II* in Figure 6.14).

⁸²The amplification and time constant of the filter are chosen to one and 25% of the duration of the very last transition in the sequence (typically $0.8 \text{ s} / 4 = 0.2 \text{ s}$), respectively. This ensures that the output of the filter comes very close to the target value at the end of the sequence, i. e., at the static idle pose.

6.14.2 Horizontal RMT Planner

The proposed method for generating ${}_{W}r_{t,x|y}(t)$ represents an extension of the spline collocation method of BUSCHMANN et al. [98] which was previously used on LOLA. Note that the method of BUSCHMANN et al. itself represents an application of the generic collocation method for solving two-point BVPs with PP functions originally introduced by RUSSELL and SHAMPINE in [362].

The main strategy of the horizontal RMT planner is to combine the EoM of the five-mass model (see Figure 4.7 and Appendix F) with the relations between the ZMP and the contact wrench of the foot-ground interface (see Section 4.5.4) to obtain a decoupled pair of second-order linear ODEs describing the motion of ${}_{W}r_{t,x}(t)$ and ${}_{W}r_{t,y}(t)$ separately. In combination with a set of two-point BCs reflecting the current and target state of the robot (up to the acceleration level – hence over-determined), this leads to a decoupled pair of second-order linear BVPs. Since the BVPs are over-determined, an exact solution does not exist for inconsistent BCs (which represents the regular case in our application). However, we can still try to find a “reasonable” approximation through spline collocation. In particular, we approximate the solution by a spline which satisfies the underlying ODE at a user-defined set of so-called *collocation sites* while simultaneously fulfilling the BCs (see Figure G.2). The particular spline collocation method used in the following has been previously published by SEIWALD and RIXEN in the article [2] which has been integrated into this dissertation in Appendix G. The resulting collocation algorithm is summarized in Algorithm G.2. Note that the source code of the reference implementation which was used to create the results presented in this thesis has been published as part⁸³ of *Broccoli*.

For generating the horizontal RMT components ${}_{W}r_{t,x|y}(t)$, the same algorithm is executed within both cycles of the iteration shown in Figure 6.13. The only difference is the time step size Δt determining the distance between collocation sites which is set to $\Delta t \approx 0.2\text{ s}$ for the first cycle and $\Delta t \approx 0.1\text{ s}$ for the second cycle. Doubling the time step size for the first cycle leads to a significant speedup while still providing a sufficiently accurate initial solution to be used by the second cycle of the vertical RMT planner. Furthermore, the proposed horizontal RMT planner allows to choose between two collocation variants (cf. Algorithm G.2):

Cubic Spline Collocation: similar (but not exactly the same) to the previous method by BUSCHMANN et al. [98]. Requires two virtual control-points in order to satisfy BCs up to the second-order time derivatives. The resulting spline is C^2 -continuous.

Quintic Spline Collocation: extension developed by the author of this thesis which was used to produce the results presented in this document. Satisfies the full set of BCs by construction (no virtual control-points needed). The resulting spline is C^4 -continuous.

While both methods are capable of generating a dynamically feasible horizontal RMT motion, the author of this thesis highly recommends to use the quintic variant since it makes the incorporation of BCs easier. Furthermore, it is more efficient since it achieves a higher approximation quality (measured by the residual of the ODE) for a certain given time budget (cf. Figure G.9).

Problem Formulation In order to setup the decoupled pair of second-order linear BVPs describing the horizontal RMT motion ${}_{W}r_{t,x|y}(t)$, we use the relation between the ZMP and the contact wrench in the foot-ground interface (cf. Equation 4.9)

$${}_{W}T_{f,\text{cont}} = \begin{bmatrix} {}_{W}T_{f,\text{cont},x} \\ {}_{W}T_{f,\text{cont},y} \\ {}_{W}T_{f,\text{cont},z} \end{bmatrix} = \begin{bmatrix} {}_{W}r_{\text{ZMP},y} {}_{W}F_{f,\text{cont},z} - {}_{W}r_{\text{ZMP},z} {}_{W}F_{f,\text{cont},y} \\ {}_{W}r_{\text{ZMP},z} {}_{W}F_{f,\text{cont},x} - {}_{W}r_{\text{ZMP},x} {}_{W}F_{f,\text{cont},z} \\ {}_{W}T_{\text{ZMP},z} + {}_{W}r_{\text{ZMP},x} {}_{W}F_{f,\text{cont},y} - {}_{W}r_{\text{ZMP},y} {}_{W}F_{f,\text{cont},x} \end{bmatrix}, \quad (6.23)$$

⁸³See the classes `Cubic|QuinticSplineCollocator` of the module `ode` of *Broccoli*.

the EoM of the five-mass model related to the change of *linear* momentum (cf. Equation F.3)

$$\mathbf{W}^{F_{f,\text{cont}}} = m_t \mathbf{W} \ddot{\mathbf{r}}_t + \dot{m}_t \mathbf{W} \dot{\mathbf{r}}_t + \mathbf{W} \mathbf{C}_1 = \begin{bmatrix} \mathbf{W}^{F_{f,\text{cont},x}} \\ \mathbf{W}^{F_{f,\text{cont},y}} \\ \mathbf{W}^{F_{f,\text{cont},z}} \end{bmatrix} = \begin{bmatrix} m_t \mathbf{W} \ddot{r}_{t,x} + \dot{m}_t \mathbf{W} \dot{r}_{t,x} + \mathbf{W} C_{1,x} \\ m_t \mathbf{W} \ddot{r}_{t,y} + \dot{m}_t \mathbf{W} \dot{r}_{t,y} + \mathbf{W} C_{1,y} \\ m_t \mathbf{W} \ddot{r}_{t,z} + \dot{m}_t \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{1,z} \end{bmatrix}, \text{ and (6.24)}$$

the EoM of the five-mass model related to the change of *angular* momentum (cf. Equation F.5)

$$\begin{aligned} \mathbf{W}^{T_{f,\text{cont}}} &= m_t \mathbf{W} \mathbf{r}_t \times (\mathbf{W} \ddot{\mathbf{r}}_t - \mathbf{W} \mathbf{g}) + \dot{m}_t \mathbf{W} \mathbf{r}_t \times \mathbf{W} \dot{\mathbf{r}}_t + \mathbf{W} \mathbf{C}_2, \\ \begin{bmatrix} \mathbf{W}^{T_{f,\text{cont},x}} \\ \mathbf{W}^{T_{f,\text{cont},y}} \\ \mathbf{W}^{T_{f,\text{cont},z}} \end{bmatrix} &= \begin{bmatrix} m_t \mathbf{W} r_{t,y} (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) - m_t \mathbf{W} r_{t,z} \mathbf{W} \ddot{r}_{t,y} + \dot{m}_t \mathbf{W} r_{t,y} \mathbf{W} \dot{r}_{t,z} - \dot{m}_t \mathbf{W} r_{t,z} \mathbf{W} \dot{r}_{t,y} + \mathbf{W} C_{2,x} \\ m_t \mathbf{W} r_{t,z} \mathbf{W} \ddot{r}_{t,x} - m_t \mathbf{W} r_{t,x} (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) + \dot{m}_t \mathbf{W} r_{t,z} \mathbf{W} \dot{r}_{t,x} - \dot{m}_t \mathbf{W} r_{t,x} \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{2,y} \\ m_t \mathbf{W} r_{t,x} \mathbf{W} \ddot{r}_{t,y} - m_t \mathbf{W} r_{t,y} \mathbf{W} \ddot{r}_{t,x} + \dot{m}_t \mathbf{W} r_{t,x} \mathbf{W} \dot{r}_{t,y} - \dot{m}_t \mathbf{W} r_{t,y} \mathbf{W} \dot{r}_{t,x} + \mathbf{W} C_{2,z} \end{bmatrix}. \end{aligned} \quad (6.25)$$

Note that $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{R}^3$ are auxiliary vectors defined in the Equations F.3 and F.6 which represent compounds of known quantities (e. g. EE dynamics, upper body rotation, gravitation, etc.). Moreover, \mathbf{C}_1 and \mathbf{C}_2 contain the (known) external wrenches $\mathbf{W}^{h,\text{ext}}(t)$ such that multi-contact effects are properly incorporated, see Appendix F for details. Inserting $\mathbf{W}^{F_{f,\text{cont},x}}$ and $\mathbf{W}^{F_{f,\text{cont},z}}$ from Equation 6.24 and $\mathbf{W}^{T_{f,\text{cont},y}}$ from Equation 6.25 into Equation 6.23 (second row) gives

$$\begin{aligned} 0 &= \mathbf{W} r_{\text{ZMP},z} \overbrace{\left(m_t \mathbf{W} \ddot{r}_{t,x} + \dot{m}_t \mathbf{W} \dot{r}_{t,x} + \mathbf{W} C_{1,x} \right)}^{\mathbf{W}^{F_{f,\text{cont},x}}} - \mathbf{W} r_{\text{ZMP},x} \overbrace{\left(m_t \mathbf{W} \ddot{r}_{t,z} + \dot{m}_t \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{1,z} \right)}^{\mathbf{W}^{F_{f,\text{cont},z}}} \\ &\quad - m_t \mathbf{W} r_{t,z} \mathbf{W} \ddot{r}_{t,x} + m_t \mathbf{W} r_{t,x} (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) - \dot{m}_t \mathbf{W} r_{t,z} \mathbf{W} \dot{r}_{t,x} + \dot{m}_t \mathbf{W} r_{t,x} \mathbf{W} \dot{r}_{t,z} - \mathbf{W} C_{2,y}. \end{aligned}$$

Rearrangement of the terms leads to the second-order linear time-variant ODE describing the RMT motion in x -direction of the world frame

$$\alpha_x(t) \mathbf{W} \ddot{r}_{t,x}(t) + \beta_x(t) \mathbf{W} \dot{r}_{t,x}(t) + \gamma_x(t) \mathbf{W} r_{t,x}(t) = \tau_x(t) \quad (6.26)$$

where the *time-dependent* (but known) coefficients and right-hand side are given by

$$\begin{aligned} \alpha_x(t) &:= m_t (\mathbf{W} r_{\text{ZMP},z} - \mathbf{W} r_{t,z}), \\ \beta_x(t) &:= \dot{m}_t (\mathbf{W} r_{\text{ZMP},z} - \mathbf{W} r_{t,z}), \\ \gamma_x(t) &:= m_t (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) + \dot{m}_t \mathbf{W} \dot{r}_{t,z}, \\ \tau_x(t) &:= \mathbf{W} C_{2,y} - \mathbf{W} r_{\text{ZMP},z} \mathbf{W} C_{1,x} + \mathbf{W} r_{\text{ZMP},x} (m_t \mathbf{W} \ddot{r}_{t,z} + \dot{m}_t \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{1,z}). \end{aligned} \quad (6.27)$$

For deriving the y -component, we follow the same scheme: inserting $\mathbf{W}^{F_{f,\text{cont},y}}$ and $\mathbf{W}^{F_{f,\text{cont},z}}$ from Equation 6.24 and $\mathbf{W}^{T_{f,\text{cont},x}}$ from Equation 6.25 into Equation 6.23 (first row) gives

$$\begin{aligned} 0 &= \mathbf{W} r_{\text{ZMP},y} \overbrace{\left(m_t \mathbf{W} \ddot{r}_{t,z} + \dot{m}_t \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{1,z} \right)}^{\mathbf{W}^{F_{f,\text{cont},z}}} - \mathbf{W} r_{\text{ZMP},z} \overbrace{\left(m_t \mathbf{W} \ddot{r}_{t,y} + \dot{m}_t \mathbf{W} \dot{r}_{t,y} + \mathbf{W} C_{1,y} \right)}^{\mathbf{W}^{F_{f,\text{cont},y}}} \\ &\quad - m_t \mathbf{W} r_{t,y} (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) + m_t \mathbf{W} r_{t,z} \mathbf{W} \ddot{r}_{t,y} - \dot{m}_t \mathbf{W} r_{t,y} \mathbf{W} \dot{r}_{t,z} + \dot{m}_t \mathbf{W} r_{t,z} \mathbf{W} \dot{r}_{t,y} - \mathbf{W} C_{2,x}. \end{aligned}$$

Rearrangement of the terms leads to the second-order linear time-variant ODE describing the RMT motion in y -direction of the world frame

$$\alpha_y(t) \mathbf{W} \ddot{r}_{t,y}(t) + \beta_y(t) \mathbf{W} \dot{r}_{t,y}(t) + \gamma_y(t) \mathbf{W} r_{t,y}(t) = \tau_y(t) \quad (6.28)$$

where the *time-dependent* (but known) coefficients and right-hand side are given by

$$\begin{aligned} \alpha_y(t) &:= m_t (\mathbf{W} r_{t,z} - \mathbf{W} r_{\text{ZMP},z}) = -\alpha_x(t), \\ \beta_y(t) &:= \dot{m}_t (\mathbf{W} r_{t,z} - \mathbf{W} r_{\text{ZMP},z}) = -\beta_x(t), \\ \gamma_y(t) &:= -m_t (\mathbf{W} \ddot{r}_{t,z} + \mathbf{g}) - \dot{m}_t \mathbf{W} \dot{r}_{t,z} = -\gamma_x(t), \\ \tau_y(t) &:= \mathbf{W} C_{2,x} + \mathbf{W} r_{\text{ZMP},z} \mathbf{W} C_{1,y} - \mathbf{W} r_{\text{ZMP},y} (m_t \mathbf{W} \ddot{r}_{t,z} + \dot{m}_t \mathbf{W} \dot{r}_{t,z} + \mathbf{W} C_{1,z}). \end{aligned} \quad (6.29)$$

Note that the pair of ODEs is indeed decoupled, i. e., Equations 6.26 and 6.28 are independent of each other. In order to complete the definition of the BVPs, we need to specify the BCs. Since the horizontal RMT motion is planned on sequence-level, the BCs are related to the begin pose of the very first and the end pose of the very last transition in the currently investigated motion plan sequence. Since every (new) sequence starts and ends with the robot being in the static idle pose defined in Section 4.5.2, we can use the expressions from Appendix F (and in particular Equation F.14) to explicitly compute the corresponding ${}_{\text{W}}r_{t,x}$ and ${}_{\text{W}}r_{t,y}$ while ${}_{\text{W}}\dot{r}_{t,x|y} = {}_{\text{W}}\ddot{r}_{t,x|y} = 0$ (static case) such that the splines match their ODEs at the boundaries. Without external contact wrenches at the hands, this means that the horizontal position of the CoM and ZMP coincide (cf. Equation F.16). In the case of dynamic replanning, the planning horizon does not start in the static idle pose but rather in an arbitrary in-motion configuration. In this case, we simply obtain the BCs (${}_{\text{W}}r_{t,x|y}$, ${}_{\text{W}}\dot{r}_{t,x|y}$, and ${}_{\text{W}}\ddot{r}_{t,x|y}$) at the start by evaluating the already planned RMT trajectory at the end of the last unchanged transition.

Considerations on Continuity Same as for most other signals sent to the SIK module, we require the CoM (and hence RMT) trajectory to be \mathcal{C}^2 -continuous. With regard to \mathcal{C}^2 -continuity of ${}_{\text{W}}r_{t,x|y}(t)$, this means that ${}_{\text{W}}\ddot{r}_{t,x}$ and ${}_{\text{W}}\ddot{r}_{t,y}$ must be \mathcal{C}^0 -continuous, i. e., in Equations 6.26 and 6.28 all ratios $\beta_{x|y}/\alpha_{x|y}$, $\gamma_{x|y}/\alpha_{x|y}$, and $\tau_{x|y}/\alpha_{x|y}$ must be \mathcal{C}^0 -continuous. Note that $1/x$ is only \mathcal{C}^∞ -continuous for $x \neq 0$, thus, we require $m_t \neq 0$ (guaranteed by parametrization of five-mass model) and ${}_{\text{W}}r_{t,z} \neq {}_{\text{W}}r_{\text{ZMP},z}$ (guaranteed by upright walking, i. e., ${}_{\text{W}}r_{t,z} > {}_{\text{W}}r_{\text{ZMP},z}$) such that $\alpha_{x|y} \neq 0$ and therefore $1/\alpha_{x|y}$ is \mathcal{C}^0 -continuous if $\alpha_{x|y}$ is \mathcal{C}^0 -continuous. From demanding \mathcal{C}^0 -continuity of $\alpha_{x|y}$, $\beta_{x|y}$, $\gamma_{x|y}$, and $\tau_{x|y}$, we can derive the following constraints:

- $r_e(t)$ and ${}_{\text{W}}r_{t,z}(t)$ need to be \mathcal{C}^2 -continuous,
- $\xi(t)$ needs to be \mathcal{C}^1 -continuous, and
- $r_{\text{ZMP}}(t)$, $\dot{\omega}_{\text{UB}}(t)$, and $\mathbf{W}_{h,\text{ext}}^h(t)$ need to be \mathcal{C}^0 -continuous.

Although the current implementation of the motion generator easily satisfies these requirements, they still have to be kept in mind whenever modifications of the WPG are to be made in future. Also note that these considerations are related to the “real” (exact) solution of the ODEs. In contrast, our spline collocation algorithm will always, i. e., independent of the continuity of the input signals, return \mathcal{C}^2 (cubic) or \mathcal{C}^4 (quintic) continuous approximations. However, these approximations may become arbitrarily bad in case the aforementioned constraints are violated.

Spline Collocation The BVPs specified through Equations 6.26 to 6.29 are finally solved by spline collocation as described in Appendix G (cf. Algorithm G.2). Note that the Equations 6.26 and 6.28 already show the structure that is expected by our spline collocation algorithm (cf. Equation G.1) such that the coefficients $\alpha_{x|y}$, $\beta_{x|y}$, $\gamma_{x|y}$, and the right hand side $\tau_{x|y}$ can be directly inserted without further transformations. The resulting horizontal RMT trajectory is shown in Figure 6.15 (top) for the same scenario as in Figure 6.14 (*Case II*).

In order to benchmark quintic against cubic spline collocation, we compare their approximation quality. However, since the BVPs are overdetermined and have inconsistent BCs, an exact solution does not exist. Consequently, it is not possible to compute the “error” relative to a reference solution. However, we can still quantify the violation of the underlying ODEs (the BCs are satisfied in both cases). For this purpose, we introduce the residuals $R_x(t)$ and $R_y(t)$

$$R_{x|y}(t) := \alpha_{x|y}(t) {}_{\text{W}}\ddot{r}_{t,x|y}(t) + \beta_{x|y}(t) {}_{\text{W}}\dot{r}_{t,x|y}(t) + \gamma_{x|y}(t) {}_{\text{W}}r_{t,x|y}(t) - \tau_{x|y}(t) \in \mathbb{R} \quad (6.30)$$

and their *Root Mean Square (RMS)*

$$\text{RMS}(R_{x|y}) := \sqrt{\frac{1}{t_{\text{end}} - t_{\text{beg}}} \int_{t_{\text{beg}}}^{t_{\text{end}}} [R_{x|y}(t)]^2 dt} \in \mathbb{R}. \quad (6.31)$$

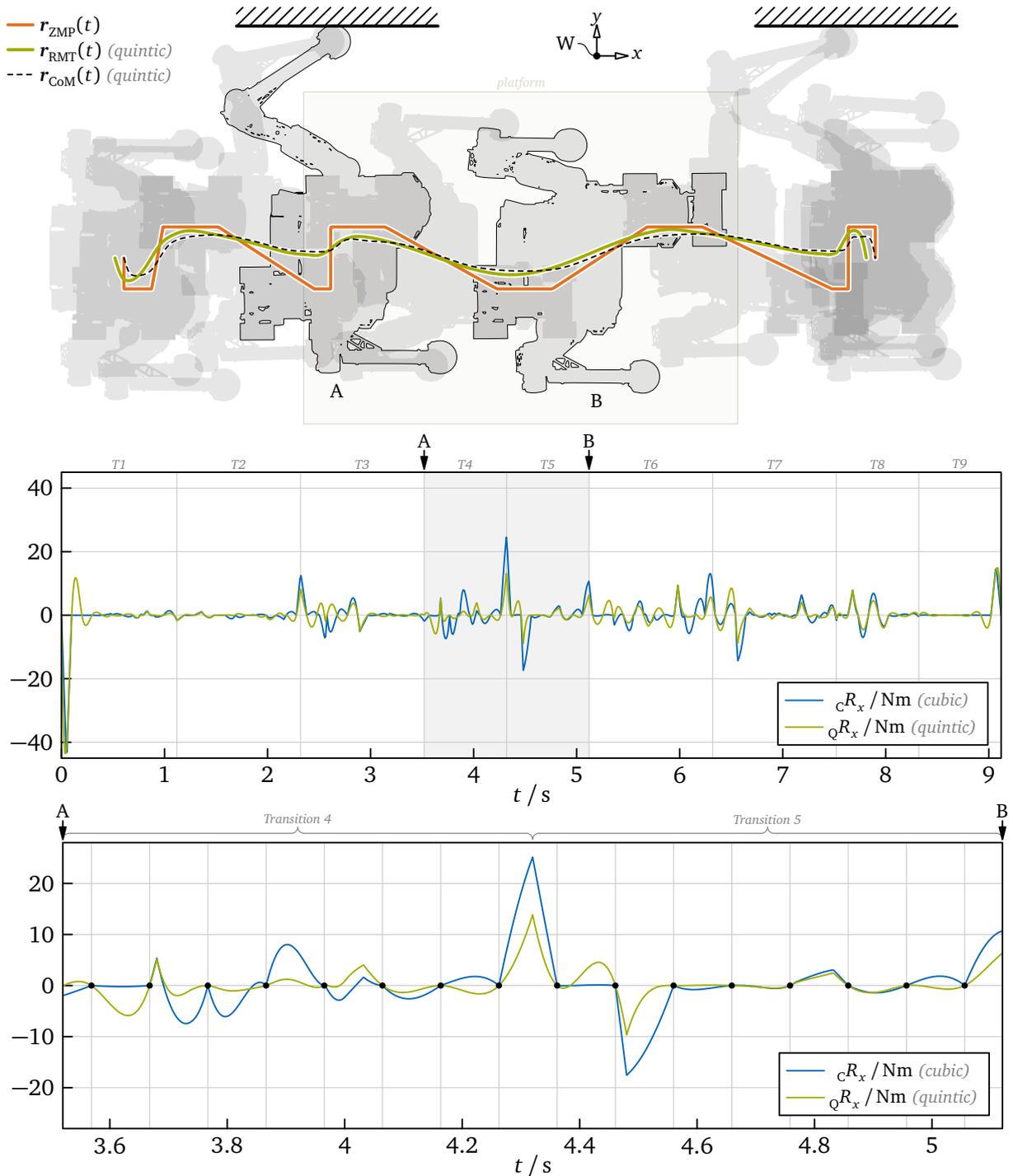


Figure 6.15: Generation of horizontal components of the RMT trajectory ${}_w r_t(t)$ for a multi-contact scenario similar to [20] @ $t=1m40s$] (identical to *Case II* of Figure 6.14). Top: top-down view showing the ZMP (“input”), RMT (“output”), and CoM (subsequent stage) trajectories. The RMT and CoM trajectories are plotted for the case of quintic spline collocation. The corresponding trajectories obtained by cubic spline collocation are very similar and therefore not explicitly plotted. Center: residuals ${}_cR_x(t)$ (cubic) and ${}_qR_x(t)$ (quintic) over the entire sequence. The time axis is partitioned to highlight the transitions $T1$ to $T9$ of the sequence. Bottom: enlarged plot of the same residuals for the time span between snapshot A and B (equivalent to snapshots I and J in Figure 6.14). The vertical grid lines highlight the time step size used during collocation (second cycle: $\Delta t \approx 0.1$ s). Note that the residuals become zero at the collocation sites (black dots). The residuals are only shown for the x -direction, i. e., with regard to collocation of ${}_w r_{t,x}(t)$. The corresponding residuals in y -direction are omitted since they show similar characteristics.

For the considered example scenario, the residual $R_x(t)$ is plotted in Figure 6.15 center. The left hand subscripts C and Q indicate the corresponding collocation method. Below, a further plot provides a better view of the residuals for the particular time span between snapshot A and B. For the same distribution of collocation sites, the quintic variant allows to reduce $\text{RMS}(R_x)$ by 52 % and $\text{RMS}(R_y)$ by 40 % in this time span. For both collocation variants, the residuals are rather high in the vicinity of the spline's boundaries which can be attributed to the (forced) satisfaction of the BCs. However, $R_{x|y}(t)$ become zero at the beginning and end of the spline since we specified the BCs such that they satisfy the underlying ODEs (static idle pose).

Parallelization Since the presented BVPs are decoupled, the horizontal RMT planner can be parallelized by triggering the collocation algorithm for the x - and y -component in two separated threads. However, the WPG proposed in this thesis goes even a step further: since the same distribution of collocation sites is used for ${}_W r_{t,x}(t)$ and ${}_W r_{t,y}(t)$, a significant amount of code of the collocation algorithm needs to be run only once (see Figure G.3 right). In particular, this applies to solving the involved (block-)tridiagonal *Linear System of Equations (LSE)* and computing the spline gradients. The remaining substeps of the algorithm, i. e., solving the (dense) collocation LSE and converting the result to a corresponding trajectory, are run in parallel (*Thread 6* in Figure 4.11 and an additionally spawned thread with the same real-time priority). While the described strategy may seem simple, special care is necessary during implementation such that the overhead introduced by synchronization and additional context switches does not defeat the performance gain through parallel execution.

Discussion The horizontal RMT planner combines most of the signals created in the previous stages of the planning pipeline (ZMP, upper body and EE motion, external contact wrenches, etc.) with the five-mass model to generate a dynamically feasible motion. A notable advantage of the proposed method is that there are no restrictions to the shape of $r_{\text{ZMP}}(t)$ or ${}_W r_{t,z}(t)$ (apart from the aforementioned requirements with regard to continuity). In contrast to the previous approach by BUSCHMANN et al. [98, 100], the new planner is based on a five-mass model which features hand masses, a dynamic mass distribution, rotational inertia of the upper body, and external contact wrenches to incorporate multi-contact effects. Moreover, the use of quintic instead of cubic spline collocation removes the necessity of manually tuned parameters⁸⁴ and simultaneously improves the approximation quality.

Currently, the collocation sites are uniformly distributed over the entire planning horizon. In Appendix G.2.5 it is shown, that this choice is optimal with regard to numerical stability of the involved (block-)tridiagonal LSE (minimum pivotal growth). However, it might be beneficial to switch from smoothest spline collocation to orthogonal spline collocation which means to sacrifice higher-order continuity (C^4 -continuity is not required by our application) and instead use more collocation sites e. g. placed at the Gaussian points of each segment (see also the brief literature review on spline collocation given in Appendix G.1).

6.15 Center of Mass (CoM) Planner

The CoM trajectory $r_{\text{CoM}}(t)$ is finally derived from the masses $m_e(t)$ and $m_t(t)$ and their corresponding positions $r_e(t)$ and $r_t(t)$. An explicit formula accounting for the dynamic mass distribution is given in Equation F.8. Note that both, the masses and their positions, are described by polynomial functions of degree 5, thus, their product is given by a polynomial function of degree

⁸⁴In order to satisfy the full set of BCs during cubic spline collocation, BUSCHMANN et al. proposed in [98] to introduce virtual control-points (similar to how it is done in Appendix G.2.8) and additionally modify the right hand side of the ODEs by manually tuned trapezoidal shape functions.

10. However, in order to simplify the implementation, we formulate the planned CoM trajectory as a spline of piecewise quintic polynomial segments. For each control-point (or collocation site) of the input trajectories, we create a corresponding control-point in the CoM trajectory at which Equation F.8 is evaluated to obtain the “exact” CoM position \mathbf{r}_{CoM} , velocity $\dot{\mathbf{r}}_{\text{CoM}}$, and acceleration $\ddot{\mathbf{r}}_{\text{CoM}}$ according to the five-mass model. Each pair of consecutive control-points is then connected by a quintic polynomial which is uniquely defined by \mathbf{r}_{CoM} , $\dot{\mathbf{r}}_{\text{CoM}}$, and $\ddot{\mathbf{r}}_{\text{CoM}}$ at its boundaries. The resulting CoM trajectory is C^2 -continuous and approximates the “exact” solution from Equation F.8 very well: for the scenario shown in Figure 6.15, the maximum errors in position, velocity, and acceleration are identified as $1.2\mu\text{m}$, $50\mu\text{m/s}$, and 4mm/s^2 , respectively.

6.16 Evaluation and Stream Processor

The CoM planner represents the very last stage of the planning pipeline. Once it finishes, the motion plan holding the analytic description of the primary WPG signals is complete. In order to extract the current sample, i. e., the snapshot of the plan which corresponds to the current point in time, we have to evaluate these analytic expressions. Thanks to the efficient implementation of polynomial and quaternion trajectories in *Broccoli*, the total runtime of creating a plan snapshot is less than $14\mu\text{s}$ (applies to all scenarios considered within this thesis). Note that this includes the evaluation of 66 polynomial (degree 5) and 6 quaternion (QBSpline) trajectories up to the second-order time derivatives (includes signals for experimental purposes).

The continuous stream of plan snapshots is passed to the *Stream Processor* which performs certain post-processing steps for each sample. Most importantly, it computes the task-space vector \mathbf{x} and the task-space velocity vector \mathbf{v} as specified in Table 4.1. Most quantities can be directly extracted from the plan snapshot and may only require a FoR transformation. An exception are the pan and tilt angles defining the orientation of the head (frame VTCP)

$$\text{rotMat}(\bar{\mathbf{s}}_{\text{UB}} \otimes_{\text{SF}} \mathbf{s}_{\text{VTCP}}) = {}_{\text{UB}}\mathbf{A}_{\text{VTCP}}(q_{\text{vp}}, q_{\text{vt}}) = \mathbf{A}_z(q_{\text{vp}})\mathbf{A}_y(-q_{\text{vt}}) \quad (\text{cf. Equation A.2}) \quad (6.32)$$

for which we can find

$$q_{\text{vp}} = \text{atan2}(\mathbf{e}_y^{\text{T}} {}_{\text{UB}}\mathbf{A}_{\text{VTCP}} \mathbf{e}_x, \mathbf{e}_x^{\text{T}} {}_{\text{UB}}\mathbf{A}_{\text{VTCP}} \mathbf{e}_x) \quad \text{and} \quad q_{\text{vt}} = \text{asin}(\mathbf{e}_z^{\text{T}} {}_{\text{UB}}\mathbf{A}_{\text{VTCP}} \mathbf{e}_x). \quad (6.33)$$

The joint velocities \dot{q}_{vp} and \dot{q}_{vt} which are required for \mathbf{v} can be easily derived by applying the chain rule. Once \mathbf{x} and \mathbf{v} have been computed, they are embedded into the data container specified in Table 4.5 which represents the input to the SIK module. Besides the task-space selection vector ξ , the load vector $\boldsymbol{\gamma}$, and the external (multi-contact) wrenches ${}_{\text{W}}\mathbf{W}_{h,\text{ext}}^h$ which are directly extracted from the plan snapshot, the container also includes the total (feet and hands) contact wrench $\mathbf{W}_{\text{cont}}^{\text{CoM}} := [\mathbf{F}_{\text{cont}}^{\text{T}}, \mathbf{T}_{\text{cont}}^{\text{T}}]^{\text{T}}$ acting on the robot at the CoM which is given by

$$\mathbf{F}_{\text{cont}} = \mathbf{F}_{f,\text{cont}} + \sum_h \xi_h \mathbf{F}_{h,\text{ext}}, \quad (6.34)$$

$$\mathbf{T}_{\text{cont}} = \mathbf{T}_{f,\text{cont}} - \mathbf{r}_{\text{CoM}} \times \mathbf{F}_{f,\text{cont}} + \sum_h \xi_h (\mathbf{T}_{h,\text{ext}} + (\mathbf{r}_h - \mathbf{r}_{\text{CoM}}) \times \mathbf{F}_{h,\text{ext}}) \quad (6.35)$$

where $\mathbf{F}_{f,\text{cont}}$ and $\mathbf{T}_{f,\text{cont}}$ are obtained from Equations F.3 and F.5, respectively. The finished data container is then transmitted to the SIK module (see Figure 4.11).

6.17 Results and Discussion

Within this chapter, LOLA’s new motion generation system for planning multi-contact locomotion has been presented. While the generation of most trajectories is based on geometrically

or empirically motivated heuristics, the core component – the RMT planner – uses a more sophisticated approach based on a simplified representation of the robot to guarantee kinematic and dynamic feasibility of the generated task-space motion. For the whole motion generator, a strong focus has been set on efficiency of the involved algorithms such that it meets the hard real-time requirements of our application. The following paragraphs discuss kinematic and dynamic feasibility as well as runtime characteristics in more detail.

Kinematic and Dynamic Feasibility While the reduced models of the robot may seem to be rather coarse, they have proven to approximate the characteristic behavior of the robot during biped gait (including multi-contact scenarios) quite well. This is confirmed by the fact that in case of a “valid” motion plan, i. e., if all parameters of the planning pipeline remain within their designated limits and no warnings are emitted (e. g. due to overlapping boundaries of the vertical RMT position), the planned motion can be successfully executed within simulation and experiments in most cases. Indeed, the most common error source in real-world experiments are misplaced contacts due to an imprecise environment model (low accuracy of CV system) or an outdated localization (slippage in foot-ground interface).

With regard to kinematic feasibility, we currently only consider the lower body of the robot. Although the reachability of contact points for the hands is already (roughly) checked by the contact planner, it should be considered to integrate an explicit kinematic model of the arm as a potential future extension of the WPG. In particular, a prediction of the joint velocities/accelerations seems to be useful, since these are currently prone to hit their corresponding limits in certain multi-contact scenarios (primarily affects the joints $\text{afr}|$ and $\text{arr}|$).

Dynamic feasibility is indirectly checked through combining the multi-body dynamics of the five-mass model with the ZMP concept. Considering external contact wrenches in the EoM allows to properly account for complex dynamic effects in multi-contact scenarios. With the proposed strategy for motion generation, additional hand support primarily affects the horizontal RMT (and hence CoM) position. This becomes clearly visible in multi-contact balancing scenarios such as those shown in the video [18 @ $t=6m5s$]. An important assumption of the presented multi-contact planning approach is that the contact force controller of the SIK module is capable of tracking the reference trajectories for the external wrenches. While this holds true for (quasi-static) balancing scenarios, the tracking error becomes much worse for short periods of contact. In fact, for walking at very high speeds, additional hand support tends to degrade the overall robustness in real-world experiments.

Runtime Analysis The new motion generation system not only extends the capabilities of LOLA, but also comes with a significantly reduced execution time ($\approx -90\%$, [3]). Due to the rather coarse modeling of the robot, the proposed pipeline is also significantly faster than other recent motion generation systems capable of planning multi-contact locomotion (see for example KUMAGAI et al. [262]). In Table 6.1, a detailed runtime analysis of the motion generator is presented. Note that the plan evaluation and stream processor (cf. Section 6.16) are executed within the main WPG loop (*Thread 5*, cf. Figure 4.11), thus, their runtimes are presented separately (last two rows in table). As described in Footnote 72, all runtimes are measured within a single-threaded simulation. Consequently, parallel execution of the horizontal RMT planner as described in Section 6.14.2 is disabled. Thus, execution on the real-hardware benefits from an additional acceleration (performance gain depends on the duration of the planning horizon).

Besides the overall efficiency, another important property of the proposed motion generator is the quite deterministic runtime since no large-scale recursions or optimizations are involved. This makes the planning time predictable (mainly depends on the total count of transitions) which is in particular relevant for dynamic replanning. Finally, it has to be highlighted that in case of fully-autonomous locomotion, the total runtime of the motion generator is almost negligible when compared to the contact planner (see Table 6.1). However, for semi-autonomous

Table 6.1: Runtime analysis of the motion generation pipeline (cf. Figure 4.11) for multiple example scenarios. The scenario *Right Table* is omitted since its runtime profile is almost identical to the scenario *Right Wall*. The individual stages of the motion generation pipeline are sorted in the order of their execution and grouped according to Figure 4.10. From top to bottom: general parameters describing the scenario; phase planner (cf. Section 6.3); SA/ZMP planning (orange, cf. Sections 6.4 and 6.5); UB/EE planning (green, Sections 6.6 to 6.13); RMT planner (yellow, Section 6.14); CoM planner (cf. Section 6.15); overall results (blue) of the contact planning and motion generation (bold) process; plan evaluation and stream processor (cf. Section 6.16). Runtimes are determined following to procedure described in Footnote 72.

Scenario \ Parameter	Platform [18 @t=5ms]s	Right Wall [18 @t=6ms]s	Corridor [18 @t=7ms]s	Obstacles [18 @t=7ms]s	Ramps [18 @t=8ms]s	Stairs – Full [18 @t=8ms]s	Stairs – Partial [18 @t=9ms]s	Trap [18 @t=9ms]s
transition count	12	9	10	22	13	13	13	26
involves multi-contact	yes	yes	yes	no	no	no	no	no
phase planner / μ s	22.6	18.7	20.1	36.6	23.9	23.8	25.4	41.3
SA planner / μ s	24.3	20.3	20.6	43.5	26.4	26.7	28.4	48.3
ZMP-xy planner / μ s	40.7	33.3	36.4	76.4	41.1	43.8	45.2	84.7
ZMP-z planner / μ s	3.5	3.0	3.1	5.4	3.5	3.4	3.7	5.4
\sum SA/ZMP planner / μ s	68.6	56.6	60.1	125.2	71.0	73.9	77.3	138.3
UB orientation / μ s	7.9	6.9	6.8	12.4	7.0	7.3	8.1	13.4
foot motion / μ s	66.3	55.9	59.5	106.1	66.0	67.7	69.9	118.1
toe motion / μ s	16.8	14.8	16.3	26.8	16.4	16.8	17.0	28.2
hand motion / μ s	22.6	19.9	23.3	33.3	20.4	20.0	20.6	38.5
head orientation / μ s	10.3	9.8	10.1	18.6	10.9	10.9	11.0	21.5
task-sp. sel. factor / μ s	16.1	12.6	14.4	27.3	16.2	16.1	16.0	31.5
load factor / μ s	23.9	19.1	20.5	40.7	24.5	24.5	27.3	49.1
external wrench / μ s	103.4	76.0	85.6	200.1	110.9	110.7	136.7	239.4
\sum UB/EE planner / μ s	267.1	215.0	236.5	465.4	272.3	274.0	306.6	539.6
RMT-z 1 st cycle / μ s	12.5	9.6	10.6	21.0	13.1	13.0	13.8	21.1
RMT-xy 1 st cycle / μ s	496.7	310.4	348.0	948.9	488.0	570.1	630.6	1,127.1
RMT-z 2 nd cycle / μ s	582.2	417.9	444.6	845.0	578.7	625.3	606.3	923.2
RMT-xy 2 nd cycle / μ s	1,408.3	781.3	898.1	3,270.1	1,443.5	1,670.4	1,810.4	4,033.1
\sum RMT planner / μ s	2,499.7	1,519.2	1,701.4	5,084.8	2,523.4	2,878.9	3,061.1	6,104.5
CoM planner / μ s	120.0	86.1	94.9	178.4	115.5	121.0	119.5	200.5
contact planner / ms	639.0	1,337.0	985.0	733.3	1,044.9	517.6	7,225.8	974.0
motion generator / ms	3.0	1.9	2.1	5.9	3.0	3.4	3.6	7.0
total runtime / s	0.642	1.339	0.987	0.739	1.048	0.521	7.229	0.981
executed motion / s	12.3	8.7	9.5	18.7	12.3	13.5	13.1	21.1
plan evaluation / μ s	11.7	11.3	11.4	12.8	11.7	11.7	11.7	13.2
stream processor / μ s	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9

locomotion scenarios such as teleoperated walking, the total runtime of the WPG is primarily determined by the motion generator. Thus, a low runtime of the motion generator allows a high update rate for dynamic replanning which in turn reduces the delay between a change in the user-input (e. g. data stream from joystick) and the robot’s reaction.

Chapter 7

Software – Part D: Ecosystem

Parts of this chapter have already been published in [15, 16].

This chapter describes the software ecosystem which is used for developing, testing, and analyzing components of LOLA's locomotion system. While the fundamental framework was primarily set up by BUSCHMANN [100] (partially adopted from the work of LÖFFLER [289] for JOHNNIE), numerous extensions and modifications have been made by the following researchers working on this project (cf. Section 2.7) so that the current software ecosystem has to be seen as joint achievement. The chapter starts in Section 7.1 with a brief overview. This is followed in the Sections 7.2 to 7.6 by a more detailed presentation of the particular components to which the author of this thesis made noteworthy contributions. Section 7.7 concludes the current status of the ecosystem and gives suggestions for future extensions.

7.1 Overview

The main components of the software ecosystem and their connections to the real-time locomotion system have already been introduced in Section 4.1 (see in particular Figure 4.1 right). Except for the *Vicon Tracker* [426] (third-party software), all remaining applications, i. e., the *Simulation* (see Section 7.4), the *Visualization* (see Section 7.5), and the UI alias *Control Panel* (see Section 7.6), are executed on the operator PC running the GPOS *Ubuntu*. Note that for simulations, the entire locomotion system is executed within the dedicated (sequential) simulation application running on the operator PC. Since the WPG, SIK, and HWL module almost⁸⁵ exclusively use system calls which are defined in the *POSIX* standard, they can be executed on the RTOS *QNX Neutrino* and *Linux*-based GPOSs without substantial adaptations to the source code. As a drawback of this hybrid approach, several high-level libraries – in particular related to the field of robotics – are only available for *Linux* and thus, can not be (directly) used in the WPG, SIK, or HWL module. However, with recent versions of *QNX Neutrino*, the count of supported libraries increased.

Programming Languages For performance reasons, the simulation framework and the control panel are implemented – same as for the WPG, SIK, and HWL module – in C++. Since the analysis is predominantly performed in the aftermath of simulations and experiments, data visualization is not considered to be time-critical. Therefore, the corresponding tools are written in Python which significantly accelerates the integration of new functionality and the adaptation to changes in the locomotion framework (e. g. updated format of log files). Finally, source code running on distributed hardware such as the auxiliary controllers interfacing the FTSs and contact switches in the feet (see Figure H.4) is written in plain C.

⁸⁵An exception are specialized functionalities of the platforms such as setting the real-time priority or CPU affinity of a thread or measuring time with a high-precision clock (see for example the class `PlatformHelper` in the module `core of Broccoli`).

Software Projects Almost the entire source code of the LOLA project is contained within the following three software repositories (each of which version controlled through *git* [186]):

am2b: represents the main codebase covering the source code which is explicitly written for the humanoid robot LOLA. Closed source with $\approx 280,000$ lines of code ($\approx 65\%$ contributed by the author of this thesis).

Broccoli: generic C++ library for robotics applications (not restricted to the particular use case of humanoid robots). Open source [15] with $\approx 100,000$ lines of code ($\approx 67\%$ contributed by the author of this thesis). See Section 7.2 for details.

am2b-vision-interface: C++ library implementing the interface between the WPG module of LOLA and an (external) CV system. Open source [16] with $\approx 6,000$ lines of code (100% contributed by the author of this thesis). See Section 7.3 for details.

For counting the lines of code, comment lines are included while empty lines are excluded. Additional to these three repositories, there exist multiple rather small software projects which contain the source code to be executed on distributed / external hardware (e. g. auxiliary controllers, test benches, etc.). The build system is based on *CMake* [206] and uses *Clang* [407] to compile source code for the *Linux* target and *qcc/q++* [83] to cross-compile for the *QNX Neutrino* target. FELIX SYGULLA further added a *Continuous Integration* pipeline with automatic tests, test coverage reports, and code-style checks.

Third-Party Libraries The real-time locomotion system of LOLA uses the following third-party libraries (compatible with *QNX Neutrino*): *Eigen* [182] (linear algebra), *zlib* [165] (compression), *cpptoml* [167] (TOML parsing), and *EC-Master* [48] (EtherCAT master stack). The components of the software ecosystem which are executed only on *Linux* additionally make use of: the *Qt Framework* [408] (graphical UI), the *Simple DirectMedia Layer (SDL)* [265] (joystick and game controller input), and the *Vicon Datastream SDK* [426] (interface to the *Vicon Tracker* application). Among these third-party libraries, *Eigen* is used most frequently. It replaces the previous in-house development *matvec* [388] by SORGE et al., which does not support modern vectorization techniques and suffers from the lack of maintenance. While the majority of source code has already been ported to *Eigen*, some legacy code – mainly related to the multi-body simulation – still uses *matvec*. However, the author of this thesis strongly recommends to replace *matvec* by *Eigen* entirely in the near future.

Runtime Optimization In order to satisfy the hard real-time requirements of our application, each code section is optimized for minimum runtime. Apart from avoiding unnecessary calculations (e. g. by reusing intermediate results), another important measure is to allocate memory statically whenever possible. In case dynamic memory allocation can not be avoided (e. g. for very large chunks of memory), allocation is triggered within an initialization routine directly after startup of the system. Examples are the allocation of internal acceleration buffers of the contact planner as described in Section 5.5.2 and the initialization of the pool allocator for storing node data of the A^* open list as described in Section 5.5.3. Note that in addition to the generally higher costs of dynamic memory allocation, reservation of very large (continuous) memory chunks may cause the OS to trigger defragmentation of the RAM which can drastically degrade the overall system performance affecting also processes of higher real-time priority. Besides dynamic memory allocation, also copying of large data structures is avoided whenever possible. An example is the environment model created by the environment model manager, which is passed to the planning context and handed over to the contact planner through cheap pointer swaps. Finally, whenever calculations can be vectorized, corresponding *Eigen* data types are used. Vectorization on the hardware by using instruction sets such as SSE4.2, AVX2|512, and FMA is automatically handled by *Eigen* for corresponding data types.

Thread Priority and CPU Affinity In addition to optimizing the source code, also the real-time priority is carefully set for each individual thread (cf. Figures 4.1 and 4.11). Unfortunately, even an RTOS such as *QNX Neutrino* can not provide full control over the execution on the underlying hardware. As an example, low-priority threads processing large amounts of data (e. g. operations on the environment model, logging, etc.) continuously flood the CPU cache which in turn can drastically slow down higher-priority threads due to recurring cache misses. Depending on the particular CPU model and how the (multi-level) caches are shared among its individual cores, performance issues related to bad caching behavior may be circumvented by “pinning” critical threads to certain CPU cores. In order to resolve caching issues in the real-time locomotion system of LOLA, *Thread 1* (HWL timing) is pinned to the CPU core “0” (avoids switching cores). Moreover, *Thread 5* (WPG main cycle) and *Thread 6* (WPG planner) are both pinned to the CPU core “1” (both threads access the rather large motion plan data structure).

Testing In order to detect and fix errors in the source code, classical debugging is used together with an automatic static code analysis with *Clang* and – as the circumstances require – a dynamic memory analysis with *Valgrind* [422]. For performing automatic tests within the continuous integration pipeline, the *GoogleTest* [173] framework is used. While the majority of source code in *Broccoli* and *am2b-vision-interface* is covered by comprehensive unit tests, only selected classes of *am2b* are tested individually. Instead, the functionality of the entire locomotion framework is automatically checked through high-level system tests which trigger numerous simulation scenarios (such as those presented within this thesis) and check the resulting behavior of the robot (e. g. falling). The range of test cases extends from simple (static) standing to complex multi-contact locomotion scenarios.

Documentation The main concept, structure, and workflow of the locomotion framework and its algorithms are documented through the numerous scientific publications and dissertations linked to the LOLA project (cf. Section 2.7). Within Chapter 4, a comprehensive overview of the current status of the framework (summarizing the remains of previous works) has been given. In addition, also the implementation is documented. For this purpose, the source code is extensively annotated with specially formatted comments which are parsed by the tool *Doxygen* [424] to generate a comprehensive documentation.

7.2 The Open-Source Library Broccoli

In order to simplify the exchange of source code between the various robotics projects at the *Chair of Applied Mechanics*, TUM, SEIWALD and SYGULLA created the free and open-source library *Beautiful Robot C++ Code Library (Broccoli)* [15]. *Broccoli* is a generic C++ header-only library which provides useful classes and algorithms for robotic applications and has strong focus on real-time performance and portability to RTOSs. Although some parts originate from the LOLA project, it is not restricted to the special case of humanoid robots. Note that *Broccoli* is not an entire middleware such as ROS [330], but can be understood as an independent toolkit. To provide a high level of reusability, strong efforts have been made regarding code style, testing, and documentation. Moreover, multiple example applications demonstrate the usage of more complex components. *Broccoli* depends on the third-party libraries *Eigen*, *zlib*, and *SDL*, however, these dependencies are not mandatory (missing dependencies disable certain functionalities). The following paragraphs give a brief overview of the individual modules of the library. The discussion is limited to the contributions made by the author of this thesis. For a description of the functionality implemented by FELIX SYGULLA, the interested reader is referred to his dissertation [401, p. 137ff]. A full documentation of the entire library is given through the online API reference (see <https://am.pages.gitlab.lrz.de/broccoli/>).

Analysis The module `analysis` provides an efficient, parallelized tool for analyzing the task-space of a robot which is described as a serial kinematic chain (without loops). Within this thesis, the class `TaskSpaceEvaluator` has been used to determine and analyze the workspace of the previous and new arm design of LOLA with respect to certain local and global metrics. Details on the workflow of this tool have already been given in Section 3.3.

Core The module `core` represents a collection of fundamental utility functions. Examples are a precise (absolute) time representation with nanosecond resolution, common string processing, and mathematical operations such as computing factorials, binomial/multinomial coefficients, FAÀ DI BRUNO and generalized LEIBNIZ tuples (each with internal lookup tables for acceleration) as well as efficient solvers for (block-)tridiagonal LSEs. The module `core` also contains numerous utility functions contributed by FELIX SYGULLA.

Curve Efficient evaluation and interpolation methods for analytically described polynomial and quaternion trajectories – as they are used by the motion generator, cf. Section 6.2 – are implemented in the module `curve`. While the class abstracting polynomials supports arbitrary degrees, interpolation of two-point boundary values is currently implemented only up to the degree five. With regard to quaternion interpolation, *Broccoli* supports the types LERP, NLERP, SLERP, QBézier, SQUAD, and QBSpline as described in Appendix B.3. The module distinguishes between a curve (e. g. a single polynomial), a spline (chain of consecutive curves with custom proportions), and a trajectory (combination of a single- or multi-dimensional spline and a custom duration). For polynomial splines, the library supports C^2 -continuous cubic (given second-order derivatives at boundaries) and C^4 -continuous quintic (given first- and second-order derivatives at boundaries) interpolation as described in Algorithm G.1.

Geometry Algorithms which are related to two- and three-dimensional geometry are collected in the module `geometry`. In particular, this module features

- an abstraction of cylindrical, spherical, and barycentric CoSys as well as DH parameters,
- utility functions related to spatial rotation and quaternions (cf. Appendices B.1 and B.2),
- an abstraction of 2D and 3D triangles with efficient point/ray distance and intersection tests (adapted from MÖLLER and TRUMBORE [309]) as well as linear interpolation of per-vertex attributes using barycentric coordinates,
- an abstraction of 2D polygons with algorithms for computing the perimeter, (signed) area, centroid, convex hull (using GRAHAM’s scan [121, p. 1030ff]), and point/ray/polygon to polygon intersections,
- an abstraction of indexed 3D triangle meshes with custom per-vertex normals and colors as well as custom per-triangle materials (e. g. used for surface models, cf. Section 4.5.1),
- efficient methods for creating indexed 3D triangle meshes representing primitives such as planes, boxes, “icospheres” (icosahedron with customizable subdivision), and cones as well as (slices/sectors of) circles, cylinders, and spheres – each of which with exact (analytical) per-vertex normals,
- efficient methods for extruding 2D outlines along 3D paths (e. g. for visualizing trajectories in a 3D animation as a solid path),
- creation of a volumetric SDF from a given 3D triangle mesh (adapted from BAERENTZEN and AANAES [72]) and conversely creation of an indexed 3D triangle mesh representing the surface of a volumetric density grid (as extension of the *Marching Cubes* algorithm of LORENSEN and CLINE [293], cf. Footnote 39), and
- a library for efficient distance evaluation using SSVs as described in Appendix C.

IO This module provides functionality related to *Input/Output (IO)* operations such as human-machine interfaces (console output for RTOSs and game controller input using *SDL* [265]), common filesystem operations, buffered logging for RTOSs, and reading/writing of 2D raster data (PNM format [347]) and 3D triangle mesh and point cloud data (PLY format [416]) in their respective text- and binary-formats. Moreover, the module `io` provides utility functions for serialization of complex data structures as well as compression of data streams (using *Deflate* from *zlib* [165]) on which the enhanced network socket described in Section 4.1 is based.

Memory The module `memory` extends the C++ standard library by data containers which are specifically designed for real-time systems and efficient thread-safe access in parallel workflows. Examples are the classes `SmartVector`, which combines the benefits of `std::array` (static allocation) and `std::vector` (dynamic size), and `CircularBuffer` which represents a thread-safe input/output buffer storing arbitrary objects. With `MultiVector`, a container for n -dimensional arrays is provided. It is used by the class `MultiLevelGrid` which abstracts (dense) n -dimensional grids with multiple levels (e. g. binary tree, quadtree, octree, etc.). Finally, this module implements an implicit D -ary heap (cf. Footnote 65) with integrated pool allocator which is used as data container for the open list of the contact planner's A* search (see Section 5.5.3).

ODE The module `ode` is split into two submodules: one for (time) integration and one for collocation of ODEs. The former is mainly used by the multi-body simulation of LOLA and has been created by FELIX SYGULLA. The latter represents a reference implementation for the cubic and quintic spline collocation algorithm presented in Appendix G (cf. Algorithm G.2) and has been contributed by the author of this thesis.

Parallel This module gathers helper classes related to efficient multi-threading. An example is the class `BackgroundWorker` which abstracts *POSIX* threads and features thread-safe getter and setter functions for custom members, automatic runtime measurements, and high-performance synchronization through condition variables. With the class `SynchronizedWorkerPool`, synchronized execution of a pool of background workers can be managed in a convenient way (e. g. used for parallel evaluation of large SSV scenes).

7.3 The Open-Source Vision Interface

In Section 4.4, the interface between the WPG module and the CV system has been described in detail (see in particular the Tables 4.2 and 4.3). Since the source code of the CV system is part of an external repository managed by the *Chair for Computer Aided Medical Procedures & Augmented Reality*, TUM, it was decided to realize the implementation of the WPG/CV interface through an individual, shared software project. This resulted in the free and open-source C++ library *am2b-vision-interface* [16], which itself depends on *Broccoli* and the (also free and open-source) third-party library *cpptoml* [167]. For *am2b-vision-interface*, special focus was set on a detailed and clear documentation to ensure an unambiguous interface specification and avoid errors due to miscommunication. Moreover, through the strict separation of the real-time locomotion and visual perception system, it is possible to switch between different external CV systems without modifications to LOLA's codebase.

7.4 Simulation

In preparation of using new algorithms or parameters of the locomotion system in real-world experiments, they are extensively tested and analyzed within simulation. For this purpose, LOLA's software ecosystem provides a comprehensive simulation framework which is triggered by calling a dedicated simulation application. In order to specify the particular scenario to simulate, a TOML based configuration file is passed to the application. The configuration file defines meta parameters such as the path to the scene description file (see following paragraph), the initial 6D pose of the robot within this scene (typically drop-off from 2 cm above the ground), and the time after which the simulation is stopped. Moreover, it provides a list of timed signals which can simulate any kind of user-input. Typically, this includes signals to start/stop walking by triggering corresponding WPG actions (cf. Section 5.2). Since the majority of parameters of the locomotion system can be controlled by corresponding signals, parameter studies can be setup and run very efficiently without the need of recompiling the source code. Based on the provided configuration, the framework simulates the execution of all components of the real-time locomotion system, most importantly the WPG, SIK, and HWL module. In order to guarantee reproducibility of the simulation results, the locomotion framework is executed sequentially (parallel execution is currently only tested on the target platform).

The reaction of the hardware (joint and floating-base motion, sensor data, etc.) is computed by an integrated multi-body simulation which has been specifically developed and optimized for the robot LOLA. The multi-body simulation provides two modes of execution: *reduced* and *full*. In the *reduced* mode, an idealized drive system with perfect tracking of the target joint angles and velocities is assumed. In contrast, the *full* mode incorporates an explicit simulation of the drive system (PPI cascade of the commercial servo controllers and motor/gear dynamics, cf. [100, p. 31ff]). Additionally, the *full* mode also considers noise and quantization of sensor signals from the IMU and FTSS. Time integration is performed with the forward EULER method where the step size is chosen to $100\ \mu\text{s}$ for the *reduced* and $10\ \mu\text{s}$ for the *full* mode. Depending on the complexity of the scene, the *reduced* simulation typically takes between 0.6 (only floor) and 1.1 (with platform/walls) times of the simulated period (measured on operator PC according to Footnote 72). Thus, it has “real-time” performance. The *full* simulation is typically 9 to 10 times slower than the *reduced* simulation.

The simulation framework of LOLA was originally developed by BUSCHMANN and SCHWIENBACHER. While the just given overview only briefly summarizes the main workflow, an in-depth description is given in the dissertations of its main creators, see [100, 372]. Over the past years, the simulation framework received multiple modifications. Notable changes were made by FELIX SYGULLA who refactored the high-level interface, integrated a new solver structure, and improved the joint controller and sensor models (see [401, p. 141f] for details). Apart from this, also the author of this thesis refactored and extended numerous components of the simulation framework. The most important modifications are summarized in the following paragraphs.

Scene Description In addition to the aforementioned TOML file for specifying meta parameters, the simulation parses another configuration file describing the scene, i. e., the (virtual) environment of the robot. Due to new requirements related to the multi-contact revision of LOLA (especially with regard to simulation of the CV system), the specification and implementation of the virtual environment has been redesigned from scratch. For consistency, the new scene description file also uses the TOML format. It mainly contains a list of environmental objects, each of which described through a 6D pose (relative to the world frame of the multi-body simulation), a parameter specifying the geometry, and certain flags, e. g. to indicate if the object should be considered for contact detection or if it should be visible in the visualization. The geometry is either loaded from an external PLY file or synthetically generated using the geometry module of *Broccoli*. Currently supported primitives are: rectangle, circle, box, cylinder, cone,

sphere, and icosphere. A special case is given by the geometry type “Plane”, which describes the (infinite) x - y -plane of the object’s local CoSy where the positive z -axis indicates the “outside” of the object. All other geometry types are transformed into a corresponding (finite) triangle mesh which can be further customized through optional parameters specifying the scaling (in local object space) and color. An exemplary scene description file and the resulting virtual environment are shown in Figure 7.1. The following paragraphs describe how the virtual environment is used within the multi-body simulation and the simulation of the CV system.

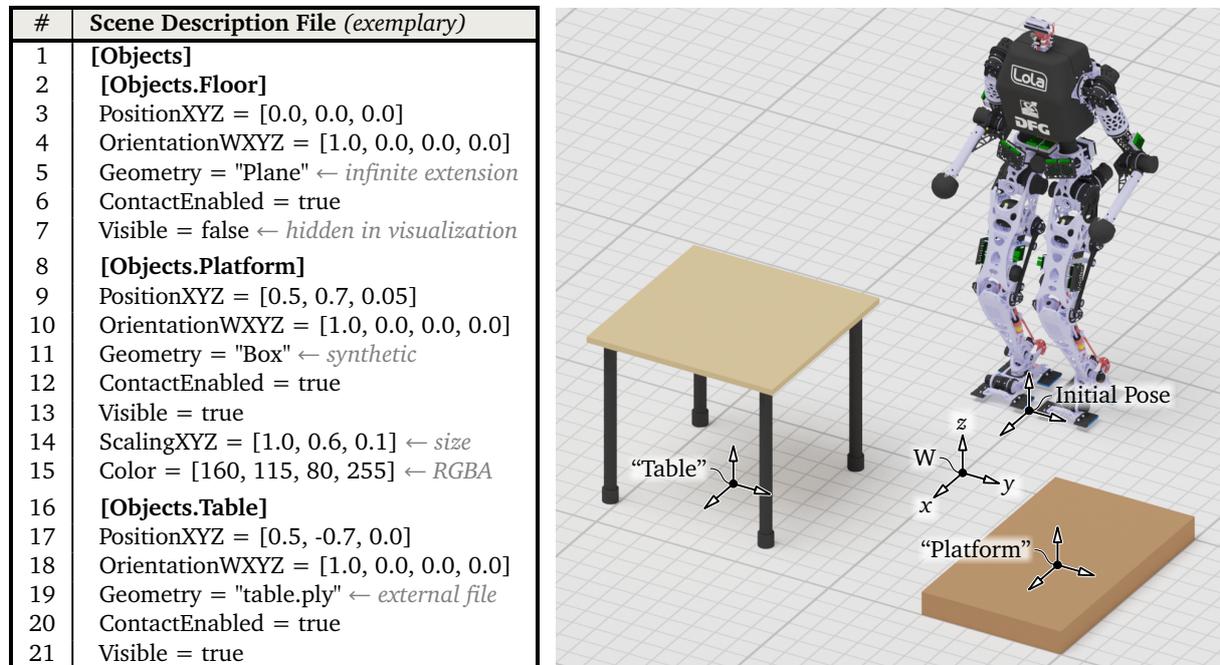


Figure 7.1: Exemplary TOML based scene description file (left) and resulting virtual environment used by the simulation framework of LOLA (right). The object “Floor” is represented by a plane with infinite extension and is not shown (coincides with the x - y -plane of the world frame W). The remaining objects are represented by finite triangle meshes which are either synthetically generated (object “Platform”) or loaded from an external PLY file (object “Table”).

Multi-Body Simulation Within the multi-body simulation of LOLA, only contacts between the robot and the environment are considered. Since the environment is assumed to be static, interactions between environmental objects are not simulated. Instead of using the complex⁸⁶ CAD model of LOLA for detecting collisions, we only allow a limited set of user-defined contact points to interact with the environment (see Figure 7.2 left). In particular, we place a contact point at each corner of the four contact pads of the feet. Moreover, 20 contact points are uniformly distributed over the spherical surface of each hand. Note that a more accurate (and likely also more efficient) representation of the hands through spherical contact surfaces would require extensive modifications to the multi-body simulation’s legacy code. Since the approximation by a finite set of point contacts turned out to be sufficient for our application, this extension is left for a future revision of the simulation framework.

Each contact point of the robot is individually tested for collision with the virtual environment which in turn is represented by a set of planes and triangle meshes. In particular, we compute the signed distance to each object and pick the one with the minimum absolute distance (closest surface). The sign of the distance indicates if the contact is open (positive) or closed (otherwise). While computing the distance of a point to a plane is straightforward, testing against an arbitrary triangle mesh is much more complex – in particular with regard to

⁸⁶The tessellation of LOLA’s CAD model used to create high-quality renderings has more than 18 million triangles. Even the much less detailed version used for real-time visualization has approximately 4 million triangles.

determining if the point lies “inside” or “outside” the not necessarily closed mesh (see Figure 7.2 right). We tackle this issue by adapting the method proposed by BAERENTZEN and AANAES in [72]. In particular, we iterate over all triangles of the mesh to find the set of triangles which have minimum absolute distance to the investigated point and share the same “closest point” (e. g. a shared vertex or edge). If the set contains more than one triangle, we use the local angle weighted pseudonormal (at the closest point) according to THÜRRNER and WÜTHRICH [410] to distinguish between interior and exterior space. If the set contains only one triangle, the “regular” triangle normal is used. Although this approach is rather simple, it delivers correct results for arbitrary triangle meshes with “reasonable” geometry (e. g. no self-intersections, etc.). Compared to the previous implementation by BUSCHMANN, the new method is more robust and handles also (rare) special configurations properly. In case of closed contact, the interaction force is finally determined using a KELVIN-VOIGT (parallel spring-damper) element which is combined with COULOMB friction (see [100, p. 21ff] for details). While the described contact detection algorithm has been contributed by the author of this thesis, the actual contact model (i. e. the computation of the interaction force) dates back to the original implementation by BUSCHMANN which, however, has been recently refactored by SYGULLA [401, p. 142].

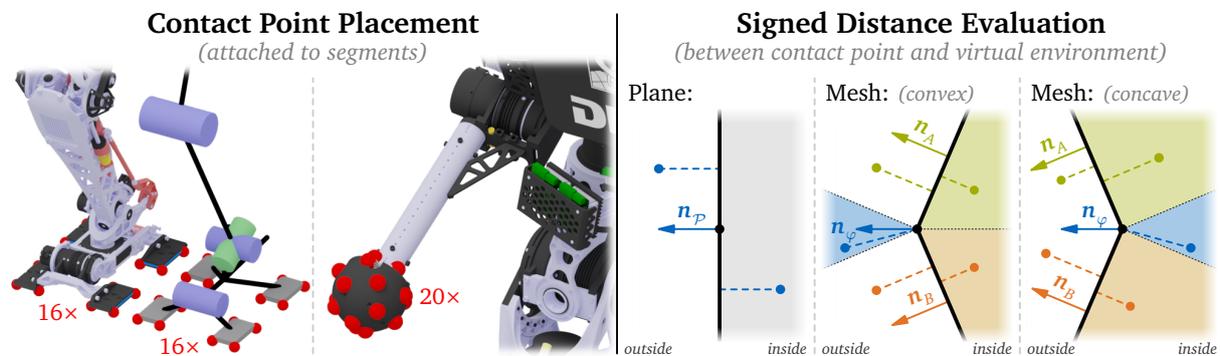


Figure 7.2: Contact detection within the multi-body simulation of LOLA. Left: placement of discrete contact points on the feet (16×2) and hands (20×2) of the robot (red spheres). Right: evaluation of the signed distance (dashed lines) between a contact point (colored dots) and the virtual environment which is represented by planes and triangle meshes (black contours). For testing against planes, the plane normal \mathbf{n}_P is used to determine the sign of the distance. For testing against triangle meshes, either the per-triangle normal $\mathbf{n}_{A|B}$ or the angle weighted pseudonormal \mathbf{n}_φ from [410] is used (depends on location of closest point on mesh).

Since the computational cost of contact detection is proportional to the complexity of the scene (triangle count), the simulation can be significantly accelerated by using rather coarse approximations of the original geometry. As an example, replacing the table shown in Figure 7.1 by its bounding box results in the exact same motion of the robot (assuming only contact of the hand with the top surface). At the same time, the count of evaluated triangles is significantly reduced from more than 1,500 to only 12. However, in certain cases the provided geometry has to be considered in full detail (e. g. for simulating walking on bumpy terrain). In order to accelerate contact detection for large meshes counting more than 100 triangles, the corresponding object is augmented by a volumetric occupancy grid which is automatically generated at the beginning of the simulation. Each cell of the grid stores references to intersecting and close-by triangles (derived from the triangle’s bounding box). During contact detection, the occupancy grid is used as lookup table to find relevant triangles efficiently.

Simulation of the CV System The simulation framework of LOLA integrates most components of the real-time locomotion system such as the WPG, SIK, and HWL module. However, this currently does not include the visual perception system since it does not provide a corresponding

execution mode⁸⁷. Unfortunately, this prevents closed-loop simulations of autonomous locomotion scenarios where the WPG requires knowledge of the robot’s surroundings. As a workaround, the author of this thesis implemented two alternative solutions.

The first solution is to load perception data from a series of files each storing a corresponding `VisionToControlContainer` (cf. Table 4.3) which has been previously recorded during a real-world experiment. For this purpose, a corresponding WPG signal has been defined which can be triggered at any time during the simulation (in the same way as virtual user-input signals). Within the context of LOLA’s multi-contact revision, this approach was mainly used to test the behavior and performance of the new contact planner for realistic input data from the CV system. Since this method relies on data collected within real-world experiments, it is restricted to scenarios which can be realized in LOLA’s laboratory.

The second solution is to synthesize one or more `VisionToControlContainer` from a static scene description as presented above. Instead of simulating the entire perception pipeline, we directly generate an environment model which is similar to what would be provided by the CV system in a real experiment. For this purpose, we extend the scene description file from Figure 7.1 (left) by additional per-object parameters which allows us to customize the conversion of environmental objects to corresponding surface and volume models as well as contributions to the terrain as specified in Section 4.5.1.

Within the proposed scene specification, the geometry of an object is described through a triangle mesh⁸⁸. In case this mesh has been created through tessellation of a parametric model designed within a CAD system, it typically does not show the same characteristics as meshes generated by a point cloud / surfel based CV system. Most importantly, the triangles of a mesh derived from a CAD model are in general strongly distorted, especially within curved surface patches. In contrast, triangles derived from a voxel-based intermediate format are limited in size (voxel dimensions) and have a much more uniform topology. Thus, instead of directly using the provided geometry of the object as its surface model in the environment model, we pre-process the mesh first. In particular, we

- compute a volumetric SDF from the input mesh using *Broccoli*⁸⁹ (method adapted from BAERENTZEN and AANAES [72] – same as for collision detection, cf. Figure 7.2 right),
- simulate errors of the visual perception system by applying Gaussian noise to each distance value stored in the SDF, and
- convert the SDF to a triangle mesh using the *Marching Cubes* implementation of *Broccoli*⁹⁰ (adapted from LORENSEN and CLINE [293], cf. Footnote 39).

The operation is performed in local object space and is customizable through a user-defined per-object voxel size (default: 5 cm) and standard deviation (default: 4 mm) specifying the applied noise. While the input geometry can be an arbitrary compound of (non-intersecting) triangles, the resulting mesh is indexed (provides proper topology information) and has per-vertex normals which are derived from the local density gradient. Note that neither the input nor the output mesh have to be closed. For interpolating colors from the original mesh, different methods are available (see implementation for details). In accordance with similar operators provided by common 3D modeling software, we refer to this (optional) pre-processing step as *remeshing* (see also the method `CGMeshFactory::remesh` from the module `geometry` in *Broccoli*). Cur-

⁸⁷Note that this would require the simulation of the onboard cameras, i. e., the creation of synthetic image streams from the scene description and the simulated motion of the robot. Although a functional solution has not been integrated yet, a pre-study has been conducted at the *Chair of Applied Mechanics* in which the characteristic output of an *Asus Xtion Pro Live* sensor is simulated (see Appendix J for details).

⁸⁸Currently, the simulation of the CV system does not support objects of type “Plane” (infinite dimension). In order to consider such objects for the synthesis of the environment model (e. g. for computing the contribution of the floor to the terrain), they are simply replaced by a corresponding “Rectangle” (finite dimension).

⁸⁹See the method `CGMeshFactory::createVolumeSignedDistanceField` from the module `geometry`.

⁹⁰See the method `CGMeshFactory::createVolumeMarchingCubes` from the module `geometry`.

rently, the proposed remeshing operator does not consider custom (per-vertex) visual perception confidence values so that a uniform value of $c_{\text{vis}} = 1$ is used instead. However, a corresponding extension to interpolate custom per-vertex confidence values would be straightforward. In particular, one could use the same method as for the color interpolation.

Besides the surface model, we also have to assign a volume model to each environmental object. For this purpose, the simulation framework allows to manually specify a corresponding SSV segment within the TOML based scene description file (preferred method). In case no related information has been provided by the user, an SSV segment is automatically generated based on the object's bounding box in local object space. Depending on the relative proportions of the bounding box dimensions, the segment either consists of a single point- or line-SSV element, or a pair of triangle-SSV elements (representing a rectangular shape). While this approach delivers satisfying results for primitive objects such as spheres or boxes, it is not suitable for approximating the volume of more complex geometries. Exemplary surface and volume models obtained through the proposed remeshing and (manual) SSV specification are shown in Figure 7.3.

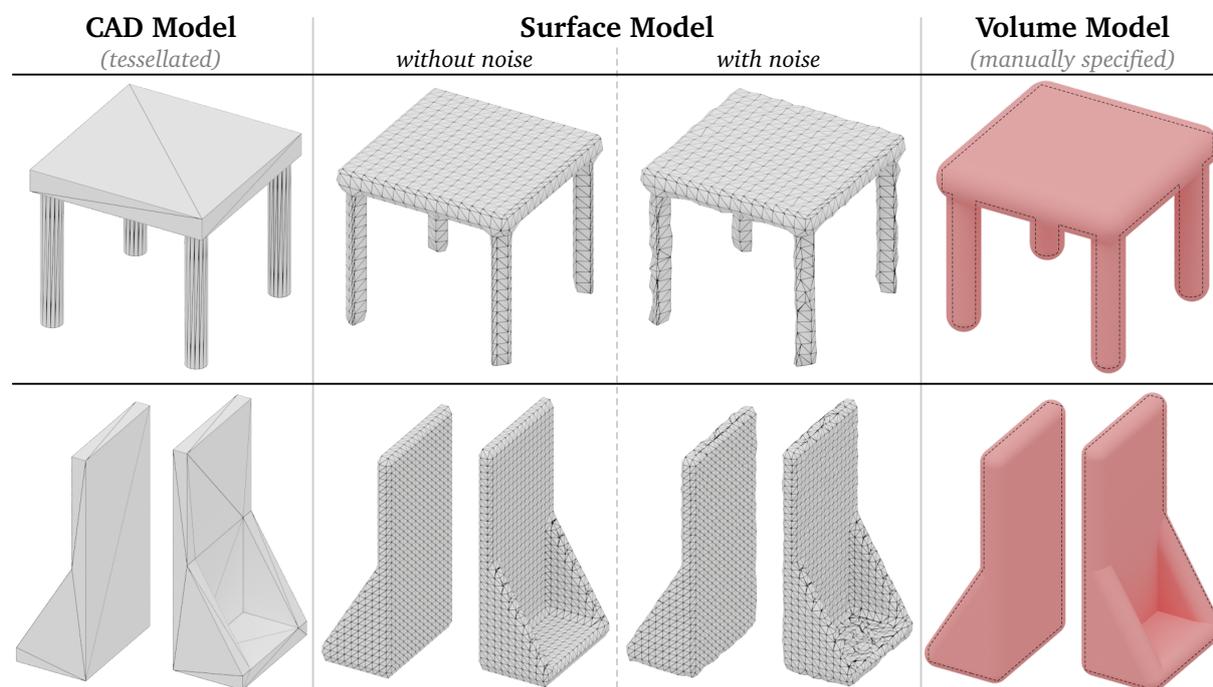


Figure 7.3: Synthesis of an object's surface and volume model from the scene description. From left to right: tessellated CAD model representing the input mesh; surface model obtained through remeshing (without and with applied noise); manually specified SSV segment representing the object's volume. The triangle meshes are rendered without colors and in wireframe mode to highlight the topology.

Once the object database of the environment model is complete, it is left to synthesize the terrain. For this purpose, we merge all surface models into a single triangle mesh. The mesh is then rasterized where we use the x - y -plane of the vision world frame as the projection plane. Each “pixel” represents a terrain cell on the lowest level of the octree which stores the corresponding height and confidence value derived from the triangle mesh. Note that this process passes the artificial noise introduced by the remeshing operator also to the terrain. Similar to the rasterization of the solution spline during contact planning (cf. Section 5.5.4), we perform spatial anti-aliasing through supersampling (uniform 4×4 pattern). Apart from projecting objects to the ground, we further define a WPG signal which allows us to initialize a circular area around the current position of the robot. The height of the related cells is determined by the vertical position of the robot while the confidence is set to the maximum and fades out at the boundaries (see [18] @ $t=1m33s$] for an animated visualization). The same signal is also used at the beginning of real-world experiments (terrain underneath robot is not visible to the cameras).

7.5 Visualization

In order to evaluate, debug, and optimize the locomotion framework of LOLA, the numerous log files generated during simulations and experiments are analyzed. Due to the high overall complexity of the system, the raw amount of data – which is additionally written in different file formats⁹¹ – can be overwhelming. To efficiently manage and post-process this data, several tools have been developed within the LOLA project. An example is the Python tool *Logplot* which uses the *Qt Framework* [408] to provide a convenient graphical UI for browsing signals from column-based log files, see Figure 7.4. After selecting the desired signals, they are plotted using the free and open-source tool *gnuplot* [436]. Originally created by BUSCHMANN [100, p. 120ff], *Logplot* received a substantial redesign and refactoring by the author of this thesis.

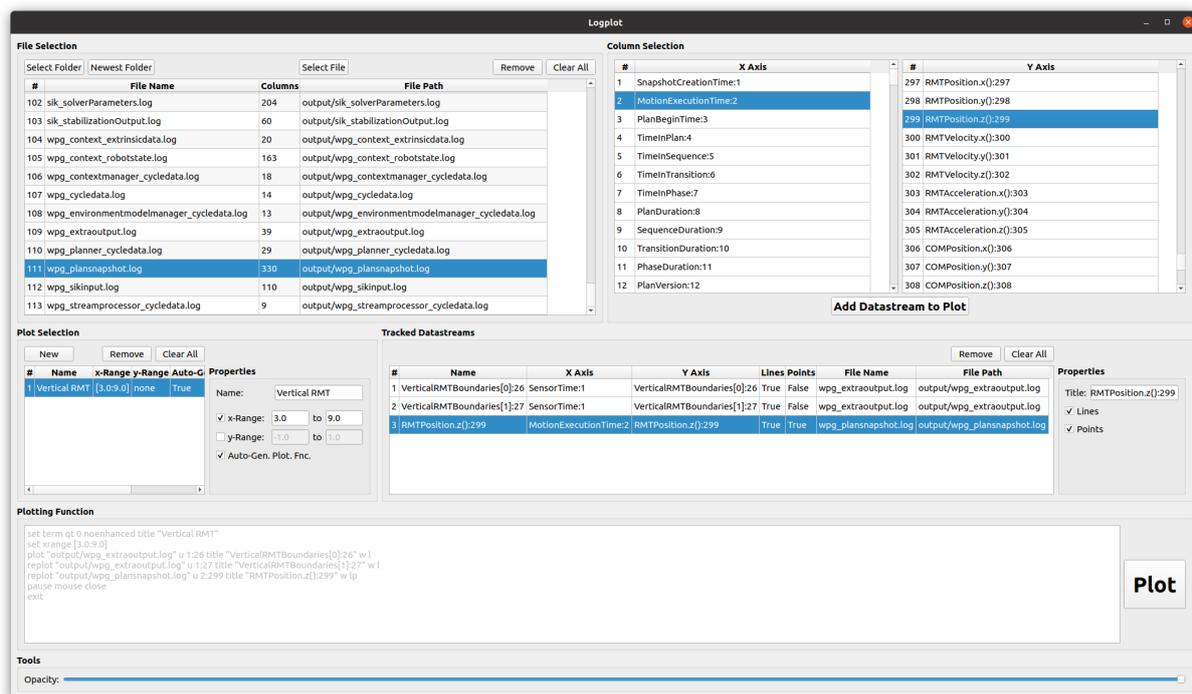


Figure 7.4: Graphical UI of the Python tool *Logplot* (uses *Qt Framework* [408]).

While a quantitative inspection of individual signals is well suited for an in-depth analysis, it is much more intuitive and efficient to verify the qualitative behavior (e. g. the overall motion of the robot) through a 3D visualization. Furthermore, certain types of data can not be expressed as continuous signals (e. g. the discrete contact sequence created by the contact planner) and are best checked through visual inspection. For this purpose, BUSCHMANN and SCHWIENBACHER developed a dedicated 3D viewer application (see [100, p. 120ff] and [372, p. 114ff] for details). Over the years, this application received numerous modifications by different contributors, however, without explicit refactoring or maintenance of its kernel. At the beginning of the author’s work on LOLA, the source code related to the viewer application was somewhat abandoned. Even worse, due to the custom low-level legacy code which is not compatible with modern GPU drivers, the functionality of the application was severely limited. For this reasons, the author of this thesis decided to create an entirely new 3D visualization tool which is referred to as *Blender Viewer*. Instead of writing custom low-level code for interfacing the GPU, the new viewer is based on the free and open-source software *Blender* [85]. *Blender* is a popular 3D graphics suite

⁹¹Depending on the type of data, log files are either written in a column-based (e. g. continuous signals), XML-based (e. g. motion plan), TOML-based (e. g. environment model, SSVs), PLY-based (e. g. point clouds, triangle meshes), or PNM-based (e. g. terrain, intermediate results of contact planner) format.

which is predominantly used for modeling, animating, and rendering complex scenes consisting of triangle-based geometries. Moreover, it provides a powerful Python interface for developing add-ons which allows to control (and thus automate) almost all parts of the software. The new 3D visualization tool of LOLA is designed as such an add-on and allows rapid implementation of new functionality in Python (e. g. parsing new or changed log file structures) while using the highly efficient internal C++ routines of *Blender* for data-intensive operations, low-level communication with the GPU, and high-quality rendering using the integrated physically-based path tracing engine. The add-on provides extensive functionality ($\approx 33,000$ lines of Python code) for visualizing different types of log data obtained from simulations and experiments. The following paragraphs give only a brief overview of its main modules. An exemplary screenshot of the interactive UI is shown in Figure 7.5.

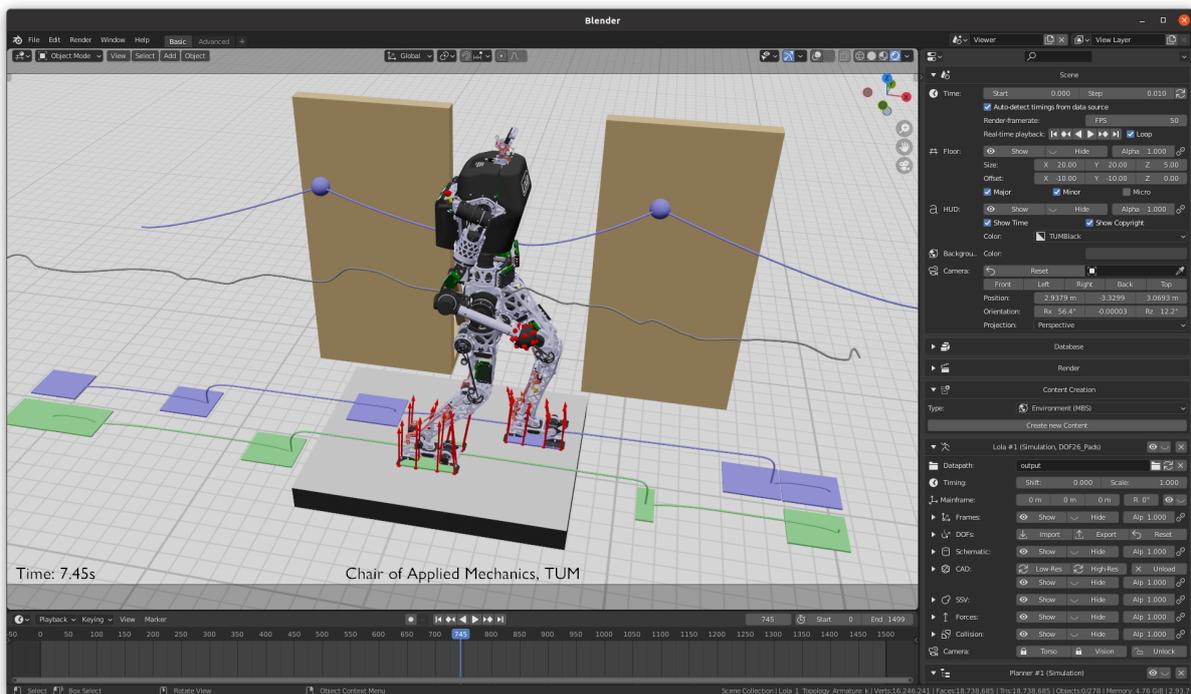


Figure 7.5: Visualization of a multi-contact simulation scenario using the *Blender Viewer* – a custom Python add-on developed for *Blender* [85]. The add-on smoothly integrates into the default graphical UI (custom panels “Scene”, “Database”, “Render”, etc. on right side). The *Blender Viewer* was used to create the numerous renderings presented within this document. The most important capabilities of this tool are indirectly demonstrated through the video [18].

Scene and Render The module Scene provides functionality related to the setup of the 3D scene. This includes timing information used for animation and playback, the generation of the “floor” (grid) and the head-up display, as well as camera parameters. The panel Render collects the most frequently used *Blender* internal settings for rendering and the output format.

Database In order to visualize complex parametric CAD models in the viewer (e. g. the segments of the robot), they first have to be tessellated within the CAD system and subsequently exported to a triangle-based mesh format. The *Blender Viewer* provides a (standalone) Python script which is capable of parsing such meshes (supports different input file formats) and converting them into corresponding *Blender* meshes using *Blender*’s internal triangle mesh and material representation. This significantly accelerates the time required for loading complex models from the file system (which is handled by the Database module).

Lola The module *Lola* allows to create one or more (independent) instances of the robot. It loads the kinematic topology from a TOML file and automatically generates a *Blender* internal representation using parent-child relations between frames for prismatic/revolute joints and armatures for parallel kinematics (e. g. ankle joint actuation). The corresponding panel provides a UI for manual DoF control and displaying the schematic, the CAD model, the joint-/task-space SSV model, and the executed 3D trajectories for all segment/EE frames of the topology. In case log data is loaded from a simulation, this module additionally allows to visualize the contact forces in the feet/hands (red arrows in Figure 7.5) and the (shortest) connection between joint-space SSV segments computed within the IK for collision avoidance.

Planner Elements of the WPG's motion plan data structure (cf. Section 5.2) can be visualized with the module *Planner*. It is capable of displaying the discrete contact sequence, SA polygons, and the planned task-space trajectories (position of EEs, ZMP, RMT, and CoM). For debugging purposes, it also allows to navigate the tree structure of the motion plan starting from the selection of the desired plan version (plan changes over time) down to individual EE poses within the begin/end pose of a transition (allows to show/hide individual elements).

Virtual Environment and Environment Model For displaying the robot's environment, the *Blender Viewer* provides the two modules *Virtual Environment* and *Environment Model*. The former is responsible for the visualization of the virtual environment as specified in the scene description file and used by the multi-body simulation (cf. Figure 7.1). The latter visualizes the pre-processed environment model, i. e., the output of the *Environment Model Manager*, consisting of terrain patches and objects represented by their surface and volume models (cf. Figure 4.4 right). While the *Virtual Environment* is only available in combination with simulation data, the *Environment Model* module also allows to load logged data from real-world experiments and thus, can be used to inspect the output of the CV system.

Motion Capture By using the optical tracking system of LOLA's laboratory, it is possible to record the actual 3D motion of the robot within real-world experiments. For this purpose, reflective markers are attached to certain segments (typically the feet/hands, torso, and head). The module *Motion Capture* allows to load logged tracking data and visualize the corresponding 3D trajectories e. g. for comparison with the planned motion.

3D Model Import/Export and SSV Editor As a comprehensive 3D graphics suite, *Blender* has native support for numerous 3D mesh file formats. The proposed add-on extends this functionality by custom import and export methods for legacy mesh formats (e. g. the one used by our CAD system [125]) as well as customized mesh specifications (e. g. including per-vertex surface confidence values). Moreover, it provides optional post-processing operators for visualizing point cloud data. Finally, the *Blender Viewer* contains the module *SSV Editor*, which allows interactive design of SSV elements (similar to the original SSV modeling tool by SCHWIENBACHER [372, p. 114ff]). The initial implementation of the *SSV Editor* was created by the student assistant REINHOLD POSCHER and has been revised by the author of this thesis.

7.6 Control Panel

For controlling and monitoring the robot during real-world experiments, BUSCHMANN created the application *Control Panel* [100, p. 120ff]. Its main purpose is to provide a clear graphical UI which displays the current state of the system (e. g. sensor/actuator data, internal state of modules, etc.) and enables the human operator to select and trigger signals from an extensive signal

database. This reaches from triggering high-level commands to setting low-level control parameters. Within the context of the multi-contact revision, the author of this thesis refactored large parts of the *Control Panel*. New features are (among others) a condensed UI tab for controlling the most frequently used functions of the WPG, SIK, and HWL modules (see Figure 7.6) and a (preliminary) interface to the autonomous safety frame (see Appendix J). Moreover, the *Control Panel* acts as communication hub for distributing the pre-processed data stream of the external motion capture system. Finally, the joystick / game controller interface has been reimplemented to extend its functionality and to maintain compatibility with modern hardware.

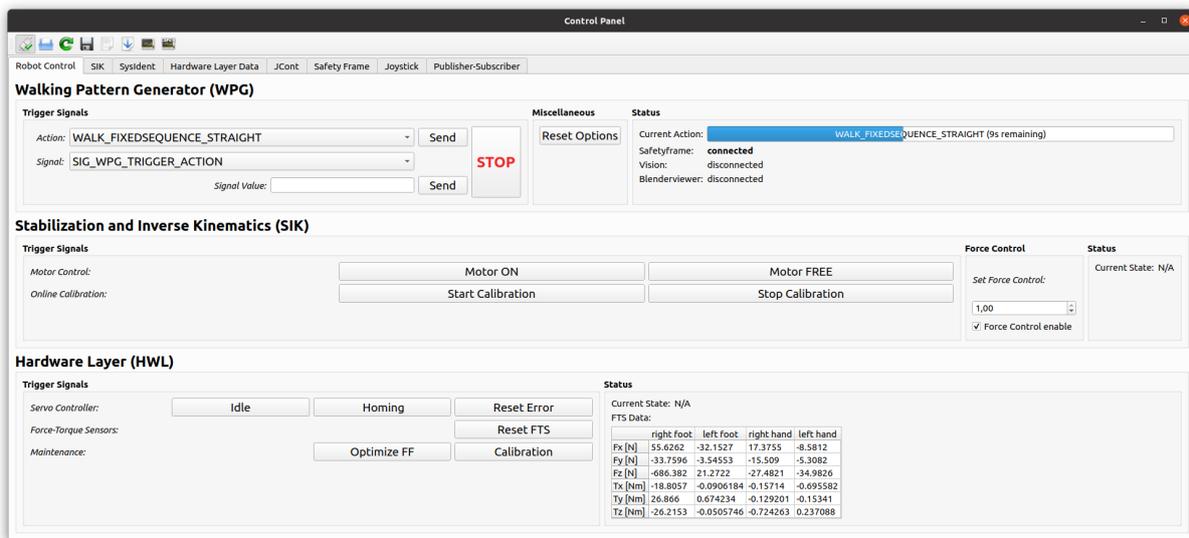


Figure 7.6: Exemplary screenshot of the *Control Panel* – the main graphical UI for conducting and monitoring real-world experiments with the humanoid robot LOLA (uses *Qt Framework* [408]). The main functionality of the application is grouped into eight individual tabs. The currently active tab, “Robot Control”, contains the most frequently used UI elements related to the WPG, SIK, and HWL module.

7.7 Conclusions and Suggestions

Within the context of the multi-contact revision, large parts of LOLA’s source code have been refactored and modernized. For an academic research project where time and manpower are very limited resources, the quality of code – in particular with respect to style, testing, and documentation – is surely exceptional. As a result, the current software is very reliable which in turn allows to conduct experiments and test new methods in a very efficient way. Moreover, the export of general-purpose code to the libraries *Broccoli* and *am2b-vision-interface* makes it available also for other robotics projects.

With regard to the software ecosystem, there is certainly room for improvement. An example is the reimplementing of the multi-body simulation, e. g. to complete the transition from *matvec* to *Eigen* as mentioned earlier. A full redesign of the multi-body simulation would further allow to introduce new features such as support for more complex contact geometries, dynamic environments (e. g. moving obstacles), or simulation of flexible components (e. g. for analyzing structural dynamics). Moreover, integrating functionality for synthesizing input streams from CV sensors would allow to simulate the visual perception pipeline and thus, enable much more realistic closed-loop simulations of autonomous locomotion scenarios.

Chapter 8

Validation – Testing LOLA’s New Capabilities

In the previous chapters, a detailed description of the realized modifications for making LOLA capable of multi-contact locomotion has been given. This included quantitative analyses of performance metrics such as dexterity, structural strength, motor power, and modal dynamics from the hardware perspective as well as solution quality and computational efficiency from the software perspective. Within this chapter, the overall system performance is evaluated from a qualitative point of view. In particular, the new capabilities of the robot are validated, i. e., tested against the original project goals as described in Chapter 1. For this purpose, various locomotion scenarios are investigated within simulation (see Section 8.1) and real-world experiments (see Section 8.2).

8.1 Simulation

In contrast to real-world experiments which underlie certain physical constraints imposed by the robot and its laboratory, (almost) arbitrary locomotion scenarios can be tested within simulation. Moreover, simulations allow to demonstrate the full potential of the new WPG by assuming ideal boundary conditions such as a perfect visual perception system. In the following, we focus on the autonomous locomotion scenarios presented in the video [18]. Snapshots of selected scenarios are shown in the Figures 8.1 and 8.2. Each of the scenarios tests and demonstrates a certain new or enhanced skill of the robot. Note that the same parametrization of the WPG is used for all scenarios which highlights its versatility and universality.

Multi-Contact The primary goal of the project was to make the humanoid robot LOLA capable of autonomous multi-contact locomotion. In accordance with the target scenarios shown in Figure 1.1, corresponding simulation scenarios have been setup for testing additional support with one hand (horizontal force: “Right Wall”; vertical force: “Right Table”) and both hands simultaneously (cf. “Corridor”). Note that the shape of an environmental object considered for hand contact neither has to be flat (indeed concave geometries are preferred – see Section 5.5.2) nor needs to have a horizontal or vertical surface normal (e. g. handrail in Figure 1.1).

The simulations demonstrate that the proposed contact planner is capable of planning a suitable contact sequence based on the provided environment model. Depending on the particular parametrization (e. g. weights in cost specification of A^*), the robot simultaneously searches for the shortest and “safest” path (e. g. close to objects suitable for hand support). The motion generator connects the discrete states by a task-space motion which takes the dynamic effects of the planned multi-contact interaction forces into account (see the video [18]@ $t=6m5s$] for a visual demonstration of the effects on the CoM position).

Collision Avoidance The previous navigation system by HILDEBRANDT et al. modeled the environment through a set of objects represented by SSVs [13]. The contact planner proposed within this thesis additionally makes use of an explicit terrain representation in the form of a height

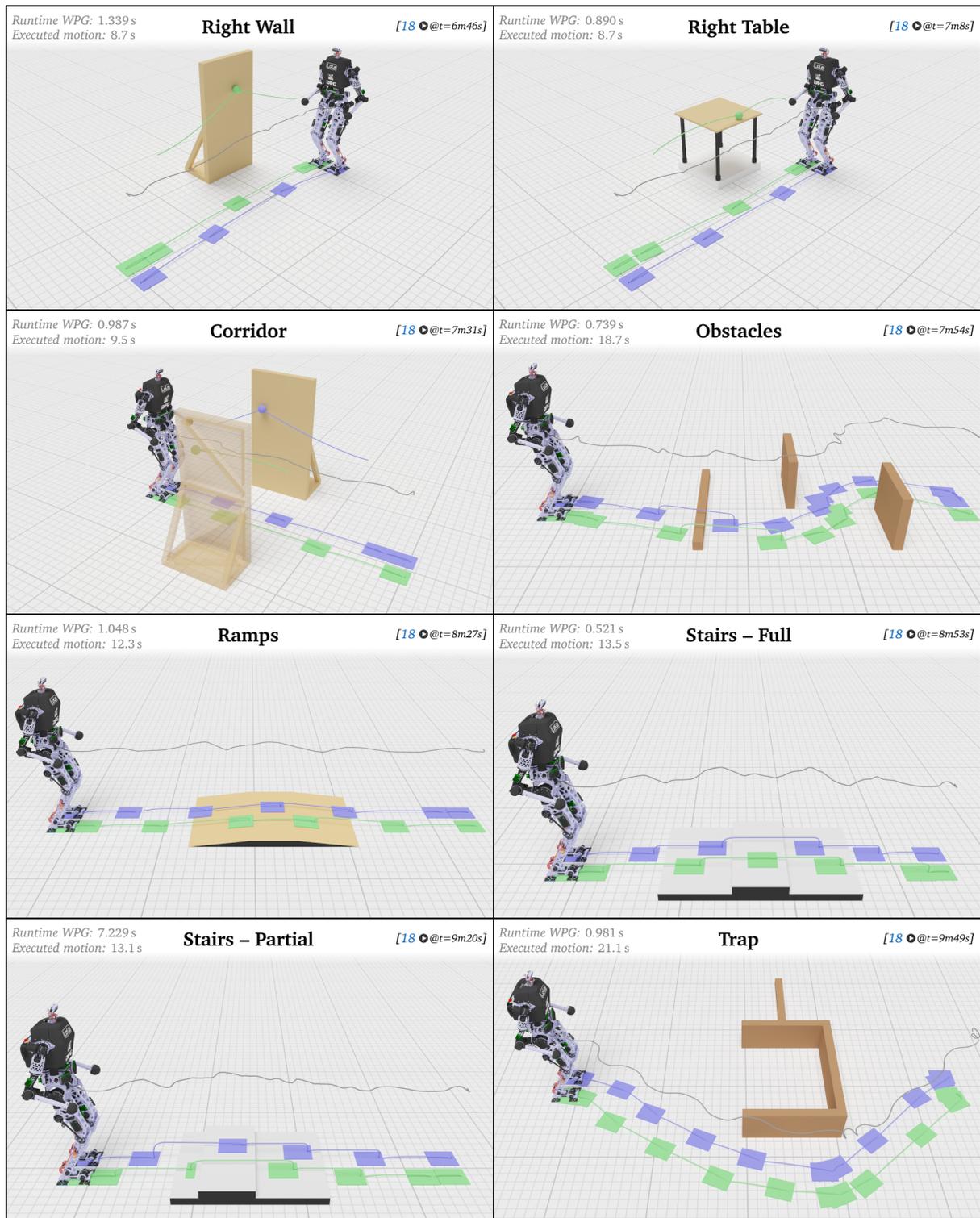


Figure 8.1: Selection of simulation scenarios demonstrating new and enhanced locomotion skills of LOLA. For each scenario, the autonomously planned discrete contact sequence (green/blue rectangles and spheres), EE motion (green/blue trajectories), and CoM motion (gray trajectory) are shown. The visual perception pipeline is not part of the simulation. Instead, the environment model is synthetically generated as described in Section 7.4.

map. This makes collision avoidance by traversing or bypassing obstacles (cf. “Obstacles”) much more general since it is capable of dealing with geometries of arbitrary shape. While the terrain information is mainly used to avoid undesired collisions of the lower body, the new contact planner predominantly uses SSV distance evaluations for avoiding collisions of the upper body.

Foothold Placement Apart from an improved collision avoidance, using an explicit terrain representation further allows native support for non horizontal ground (cf. “Ramps”) and varying ground heights (cf. “Stairs – Full / Partial”). Since ramps, platforms, or stairs are not modeled explicitly but are instead represented by a cluster of terrain cells, they may have an arbitrary shape. Furthermore, the new WPG distinguishes between (planned) full, partial, and tiptoe foot contacts. While partial contacts allow more complex solutions by extending the search space, tiptoe contacts are mainly used to maintain kinematic feasibility during challenging maneuvers.

Hierarchical Search In order to achieve real-time performance, the new contact planner makes use of different levels of detail (coarse-to-fine planning). This significantly accelerates the search and avoids unnecessary exploration of irrelevant states for disadvantageous configurations of the environment (cf. “Trap”). Note that also the previous navigation system by HILDEBRANDT et al. used a hierarchical approach for acceleration [13]. Apart from being more general, the new solution proposed within this thesis is much less dependent on the complexity of the scene which leads to a significantly reduced variance in the total execution time. Besides an increased overall efficiency, this also makes the duration of the contact planning process much more predictable.

Combination of Skills While the scenarios presented so far mainly focus on one particular problem, the new skills of the robot certainly can be performed simultaneously. As an example, Figure 8.2 shows the generated motion plan for a combined scenario featuring multi-contact interaction, stepping up and down a platform, and explicit tiptoe contact. The entire motion plan is generated autonomously solely based on the provided environment model and goal pose.

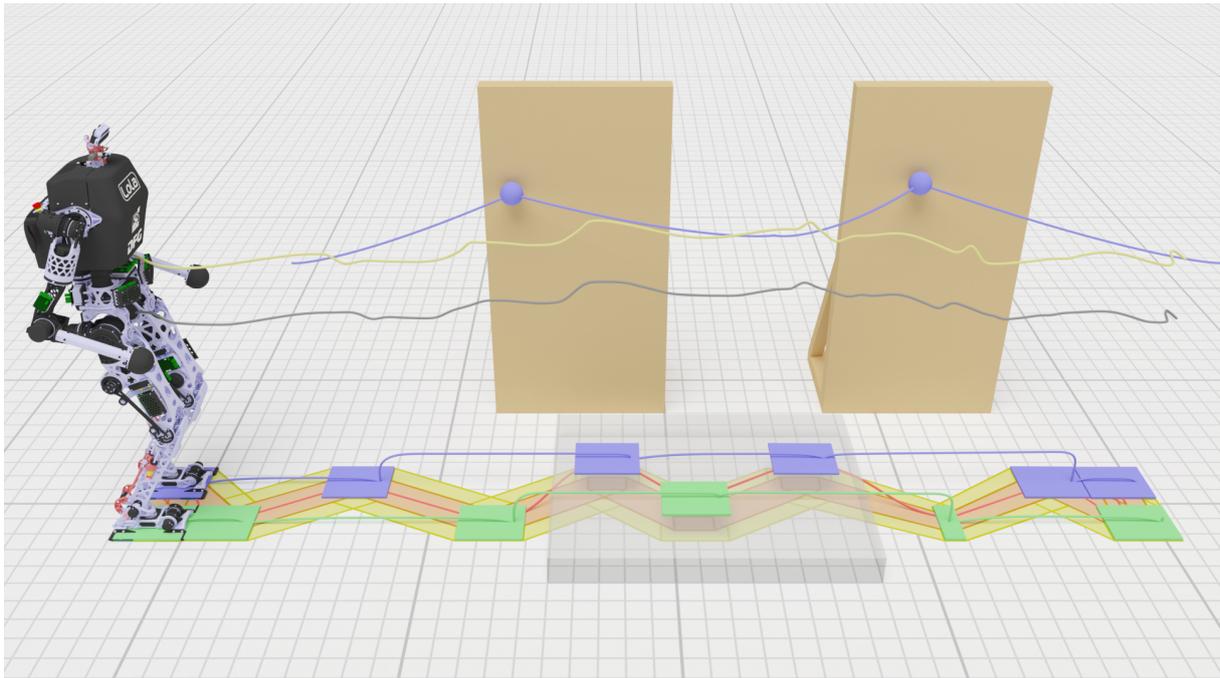


Figure 8.2: Exemplary simulation scenario demonstrating the combination of multiple new and enhanced locomotion skills (cf. [18] @ $t=5m51s$). The visualization shows the autonomously planned discrete contact sequence (green/blue rectangles and spheres), SAs without and with safety margins (yellow/orange polygons), ZMP motion (red trajectory), EE motion (green/blue trajectories), RMT motion (yellow trajectory), and CoM motion (gray trajectory). The total runtime of the WPG for generating the motion plan is 0.642 s in this particular scenario (executed motion: 12.3 s).

Dynamic Replanning For the purpose of dynamic replanning (cf. [18 @t=10m28s]), each individual stage of the WPG’s planning pipeline is capable of performing its operations either on an entirely new or an already existing motion plan. Even if the motion plan is already in execution, it is always guaranteed that valid task-space motion data is available which finally brings the robot to a safe resting pose. With total runtimes of around 1 s for fully-autonomous locomotion, the proposed WPG is capable of reacting to changes in the environment within a reasonable time. In semi-autonomous mode (e. g. teleoperated walking), the planning time drops to less than 10 ms which leads to an almost immediate response to changes of any kind.

8.2 Real-World Experiments

While the simulation scenarios discussed in the previous section have demonstrated the autonomous planning skills of the proposed WPG, it is left to verify if the new hardware configuration of the robot, i. e., LOLA v1.1, is capable of performing such motions. The following paragraphs focus on experiments which reveal the real-world performance of the new hardware and simultaneously demonstrate the effectiveness of the reduced models and methods introduced by the proposed planning pipeline. Note that the WPG uses the exact same parametrization within real-world experiments as in simulations. Thus, no manual tweaking is necessary. Same as for the presented simulations, the results shown in the following could only be achieved through the combination with the new SIK module contributed by FELIX SYGULLA. Indeed, walking without any sensor feedback, i. e., letting SIK pass-through the planned trajectories directly to the HWL, makes the robot fall after very few steps – even without hand contacts.

Hardware Revision Since the modifications to the hardware have been extensive, it first had to be checked if the robot kept its original stabilization and walking capabilities. A selection of initial tests which verify this are shown in the video [21]. In order to test and optimize the stabilization performance for the new multi-contact skills, several balancing experiments have been conducted. An exemplary scenario is shown in Figure 8.3 left, for which a significantly increased robustness against external disturbances is observed when compared to the same motion without hand support. For evaluating the dynamic behavior under more realistic walking conditions, numerous experiments similar to the simulation scenarios from Section 8.1 have been conducted (see Figure 8.3 right).

As the numerous successful experiments confirm, the new hardware fits our target application very well. With regard to the new arms of LOLA, the proposed four DoF design turns out to be a suitable compromise between dexterity (considering only pushing interactions) and mass (allowing fast arm motion). Moreover, as already predicted by the EMA (cf. Section 3.8), the new torso design leads to a clearly visible reduction of the structural oscillations in the upper body. With regard to multi-contact locomotion, this has a positive effect on the accuracy of establishing hand contacts.

Kinematic and Dynamic Feasibility While the new hardware has proven its capabilities with respect to multi-contact locomotion, it is left to verify if the WPG’s concepts for maintaining kinematic and dynamic feasibility indeed lead to a stable real-world execution. For this purpose, a more challenging experiment similar to the simulation scenario “Platform” (cf. Figure 8.2) has been conducted. Exemplary snapshots of the experiment are shown in Figure 8.4. Although not strictly necessary from a kinematic or dynamic point of view, we force the robot to make a partial contact during stepping up which demonstrates that such contacts are realizable in real-world experiments. The tiptoe contact during stepping down increases the kinematic reserves (see also Figure 6.14) while the additional hand support helps to stabilize the robot within SS phases where the stance foot is in partial / tiptoe contact.

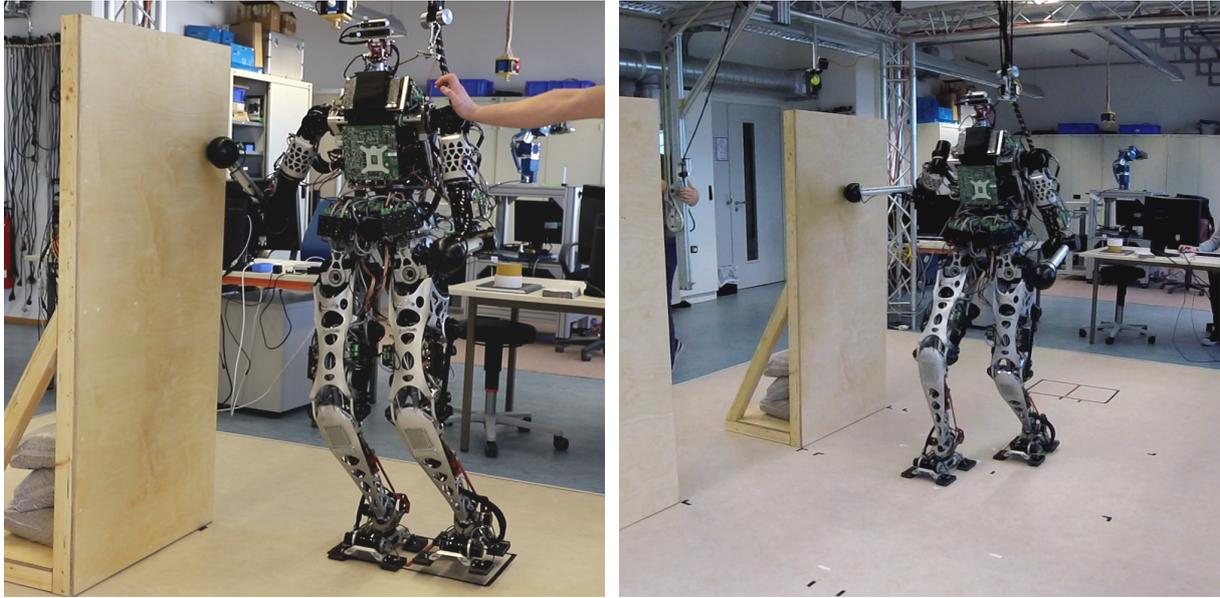


Figure 8.3: Evaluation of the multi-contact performance of the new hardware of LOLA. Left: walking in place with random pushes by a human (cf. [20] @ $t=35s$). Right: exemplary locomotion experiment similar to the simulation scenario “Right Wall” from Figure 8.1 (cf. [20] @ $t=1m17s$). The experiments have been performed in semi-autonomous mode (manually set contact sequence).

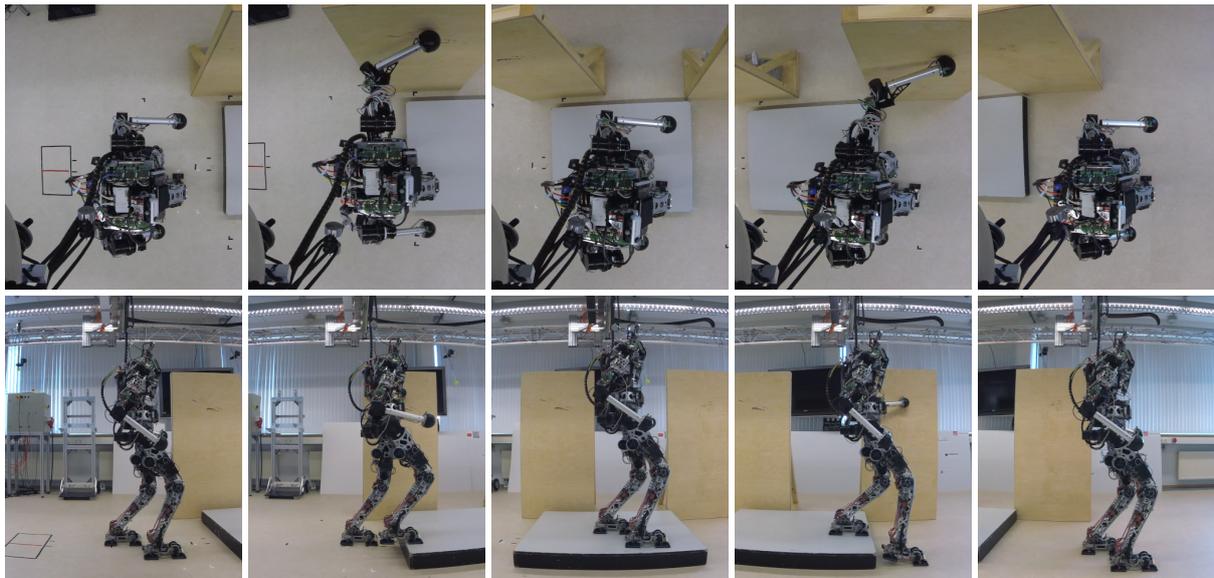


Figure 8.4: Experimental validation of combining multiple new and enhanced locomotion skills (similar to the simulation scenario shown in Figure 8.2). The robot is commanded to step up (with partial contact and hand support) and down (with tiptoe contact and hand support) a platform of 12.5 cm height (cf. [20] @ $t=1m40s$). Exemplary snapshots of the experiment are shown in top-down view (first row) and side view (second row). The experiment has been performed in semi-autonomous mode (manually set contact sequence).

Robustness Since the experiments presented so far can also be performed in “regular” biped mode, additional hand contacts only have an assisting role for increasing overall robustness. However, in very challenging scenarios – e. g. in case of strong disturbances – hand support can become essential. Figure 8.5 shows two such scenarios where the robot is commanded to pass through a corridor while walking on an uneven wooden terrain (left) and stepping on a rolling board (right). The experiment is conducted in semi-autonomous mode, i. e., no input from the CV system is used. Instead, the discrete contact sequence specifying the placement of the feet

and hands is manually set. For placing the footholds, level ground is assumed, i. e., neither the wooden terrain nor the rolling board are known to the robot. Although it has been shown that the wooden terrain can be traversed in regular biped mode, additional hand support significantly increases the rate of success from 2/3 (see [401, p. 155ff]) to almost 100%. The experiment shown on the right side of Figure 8.5 represents a scenario which is only feasible in multi-contact mode. In the authors personal experience, this particular scenario is even challenging for healthy human beings knowing about the disturbance.

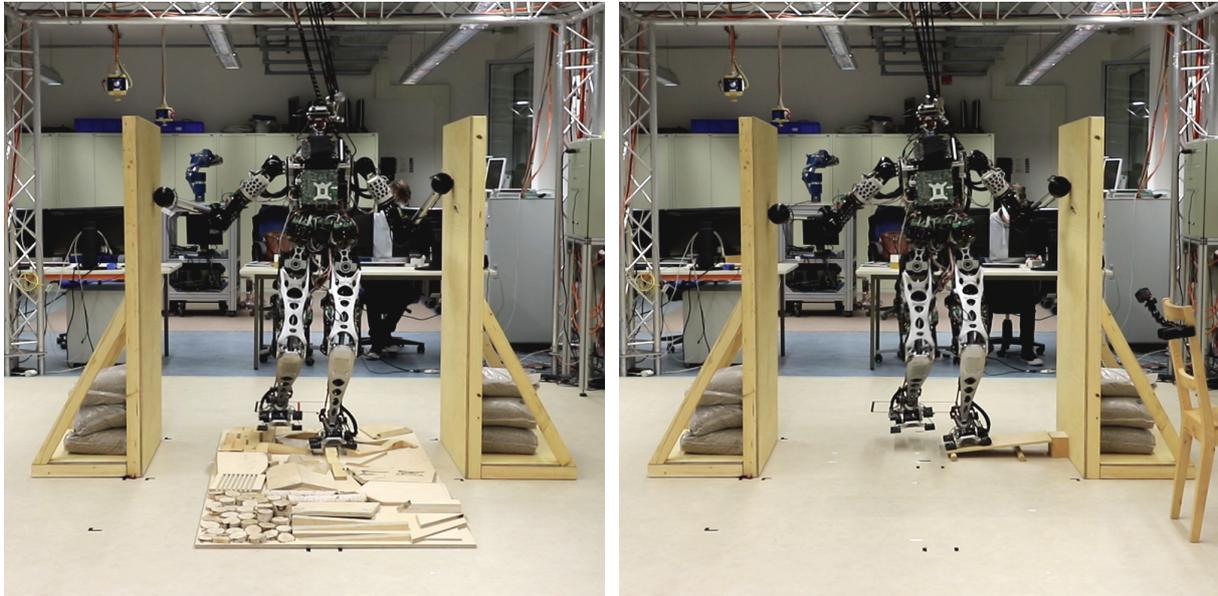


Figure 8.5: Exemplary scenarios demonstrating the importance of hand support in case of strong disturbances. Left: walking on an uneven wooden terrain (cf. [20 ●@t=10s]). Right: stepping on a rolling board (cf. [20 ●@t=23s]). The experiments have been performed in semi-autonomous mode (manually set contact sequence). Both, the wooden terrain and the rolling board are not known to the robot and thus, represent unexpected disturbances.

Vision Guided Locomotion While the full potential of the proposed contact planner has already been demonstrated within simulations, it still has to be verified if similar results can be obtained in real-world experiments, i. e., using actual input data from the CV system. Figure 8.6 shows snapshots of experiments which evaluate the real-world performance for fully-autonomous locomotion. For the complete footage of the experiments, see the video [17 ●]. Although the basic functionality of the closed-loop system could be confirmed, the overall performance is rather poor. This is mainly due to the low accuracy of the used low-cost depth sensor which causes significant distortions in the scene reconstruction (see Section 3.8 for details). In particular, we observe large absolute displacements in the surface models (up to 7 cm) and a bumpy reconstruction of the flat ground (height differences in terrain up to 4 cm). With industrial-grade high-end perception hardware, the real-world performance for fully-autonomous locomotion is expected to improve.

Locomotion Speed For regular biped locomotion, BUSCHMANN et al. reported a maximum walking speed of 3.34 km/h (step length $l_x = 0.65$ m and step duration $t_{\text{tra,dur}} = 0.7$ s) [101]. In order to evaluate the dynamic capabilities of LOLA v1.1, similar tests have been conducted with the new hardware. Despite the slight increase in total mass, a new speed record of 3.38 km/h (step length $l_x = 0.63$ m and step duration $t_{\text{tra,dur}} = 0.67$ s) could be achieved. While this does not represent a noteworthy improvement over the previous system, it clearly shows that the revision of the upper body did not degrade the biped walking performance of LOLA.

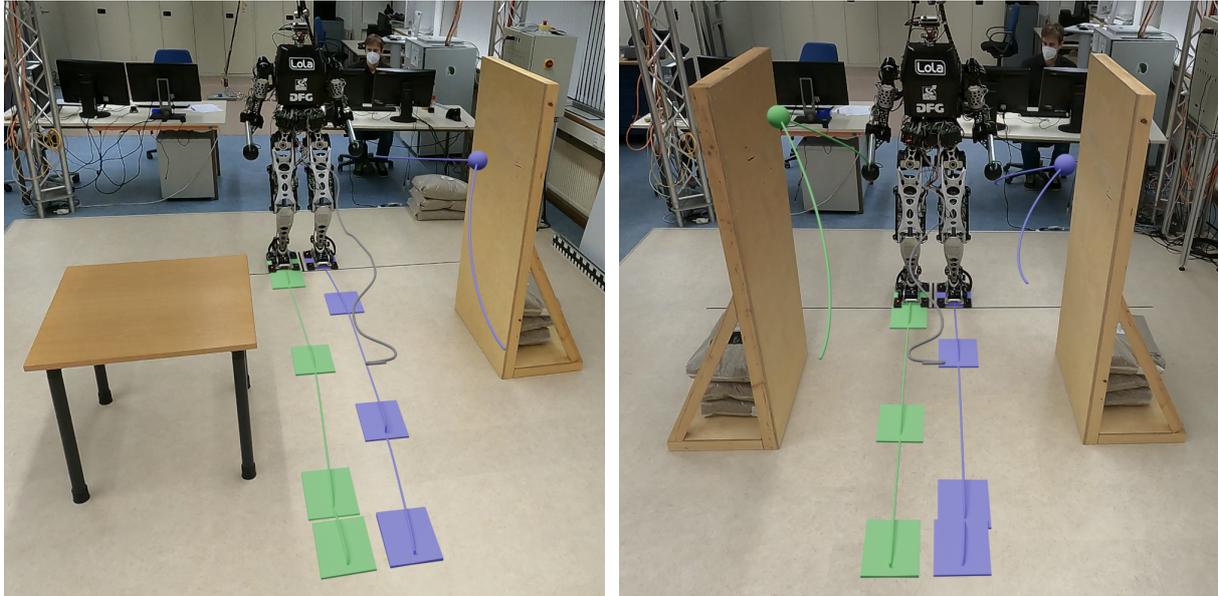


Figure 8.6: Experimental validation of fully-autonomous (“vision guided”) locomotion. Left: single handed support against a wall (cf. [17 ● @ $t=15s$]). Right: double handed support within a corridor (cf. [17 ● @ $t=1m32s$]). The corresponding environment models provided by the CV system are shown in Figure 4.4. The resulting motion plan is visualized as an overlay to the real footage.

To the author’s best knowledge, the current speed limit of the robot can not be traced back to a specific system component. Instead, the upper bound is characterized by a combination of disturbances (e. g. slipping, early / late contact events) which can not be compensated entirely by the stabilization system anymore and, hence, accumulate with every footstep. At some point, these disturbances get large enough such that the system fails (e. g. due to actuators exceeding their limits). In order to make more profound statements on the exact reasons for the current speed limit (and how it could be surpassed with a potential successor of LOLA), this topic has to be investigated in much more detail which, however, was not in the scope of this thesis.

For experiments involving multi-contact situations, a maximum speed of 1.8 km/h was used. Note that this value does not represent an upper bound, but rather an optimum with regard to overall stability and robustness. While multi-contact locomotion is also possible for higher walking speeds, hand contacts with short duration tend to act more like a disturbance rather than providing additional support. Furthermore, the velocity limits of the joints in the arms are more likely to be hit for higher locomotion speeds.

Chapter 9

Closure

Within this chapter, the presented work is concluded. In Section 9.1, the main contents of this thesis are condensed into a brief summary. Section 9.2 highlights the author’s contributions to the field of humanoid robotics in general and the multi-contact locomotion project of LOLA in particular. Concluding thoughts and recommendations for future work are given in the Sections 9.3 and 9.4, respectively.

9.1 Summary

The main motivation for this work was to render the locomotion capabilities of legged robots more versatile and robust. In particular, this thesis focused on additional hand support during (gaited) biped walking alias *dynamic multi-contact locomotion*. While the majority of the presented results are applicable to humanoid robots in general, this thesis is closely related to the particular research platform LOLA. In order to enable multi-contact locomotion for this robot, extensive modifications to the hard- and software had to be made. While a corresponding visual perception (CV) system and low-level control (SIK) module have been contributed by WU et al. [446, 448] and SYGULLA [401, p. 130ff], respectively, the author of the present thesis redesigned the upper body hardware and developed a new contact planning and motion generation system.

Since the original upper body hardware was not made for multi-contact locomotion, it had to be redesigned from scratch. In order to withstand the increased loads caused by additional hand support, LOLA’s torso has been significantly strengthened. This also allowed to resolve issues of the previous design related to structural oscillations which decreased the performance of the low-level controller. Besides the torso, also the arms received a substantial upgrade. This includes a new 4 DoF kinematic topology which has been optimized for our target application under consideration of certain local and global metrics. The new hands are equipped with commercial six-axis FTSs surrounded by a spherical contact surface which allows force-controlled pushing interactions with the environment.

LOLA’s real-time locomotion framework has a hierarchical structure consisting of a planning (WPG), control (SIK), and interface (HWL) module. This thesis focused on the planning layer which had been redesigned from the ground up in order to add native support for multi-contact locomotion. The software architecture of the new WPG module features a modular design with clear interfaces which encapsulates complexity and drastically improves maintainability and extensibility for subsequent researchers working with this platform. Moreover, considerable effort has been made to allow a highly efficient parallelized execution while respecting strict constraints such as the limited onboard computing power.

The primary goal of this thesis was to achieve autonomous locomotion, i. e., based on an environment model provided by a corresponding CV system (which itself is not in the focus of this work), the robot should find an “optimal” path towards a user-defined goal on its own. For this purpose, a novel contact planner has been developed which is capable of generating a feasible contact sequence – even for the complex multi-contact case – in real-time (typically

around 1 s total runtime). In order to achieve this performance, a coarse initial solution is computed which is used to accelerate a subsequent fine-grained search. Apart from introducing multi-contact planning for LOLA, the new contact planner also extends the robot's "regular" biped walking skills by following a more generic modeling and planning approach.

For connecting the discrete states of the contact sequence through smooth task-space trajectories, a new motion generation pipeline has been introduced. The main workflow is adopted from the original real-time trajectory generation system by BUSCHMANN [100, p. 55ff]. In order to maintain dynamic feasibility, a novel five-mass model approximating the multi-body dynamics of the robot is used. Based on a geometrically generated reference ZMP trajectory, this model is used to compute a corresponding CoM motion through quintic spline collocation – an extension of the original cubic spline collocation method by BUSCHMANN et al. [98]. Kinematic feasibility during challenging maneuvers is maintained through a simplified kinematic model of the leg (represents extension of the previous approach by HILDEBRANDT et al. [8]).

Within the context of LOLA's multi-contact revision, also the software ecosystem has been refactored and extended. In particular, the author of this thesis contributed to the simulation framework (synthesis of environment model and contact detection in multi-body simulation) and visualization tools. The latter includes an interactive 3D viewer based on *Blender*.

9.2 Author's Contributions and Innovation

While the previous section summarized the main contents of this document (which describes also work by other researchers), this section is dedicated to highlight the most important contributions and innovations of the author of this dissertation.

Hardware Design – Upper Body Core contributions of the author with regard to the revision of LOLA's upper body hardware include

- the overall concept and strategy for realizing an upper body specifically designed for multi-contact locomotion which combines versatility, robustness, and low mass,
- the mechanical design of a robust and extensible torso minimizing structural oscillations for unfavorable mode shapes (excludes certain non-critical parts such as covers and mounting brackets for electrical components, cf. Section 3.5),
- the mechanical design of robotic arms capable of making unilateral contact with the environment based on the optimal kinematic topology identified by NEUBURGER [28] (excludes joint actuators for which the original modular drive design by LOHMEIER [291, p. 96ff] is used instead), and
- an efficient, parallelized open-source tool for computing and analyzing the workspace of a series kinematic chain with respect to certain local and global metrics.

Software Design – WPG Module In order to allow motions which involve multi-contact interactions, the planning layer of LOLA's locomotion framework has been redesigned from the ground up. The entire WPG module has been created by the author of this thesis. It features

- a modular architecture which encapsulates complexity and integrates extensive error and plausibility checks (with corresponding fallback strategies) – together leading to an unprecedented reliability within real-world experiments,
- an effective clustering of computational workloads according to their real-time priority and distribution among distinct threads in order to satisfy the hard real-time requirements of the system while planning highly complex (and computationally expensive) motions, and

- an efficient formulation of planned (task-space) motion in the form of a hierarchical tree structure allowing to describe a variety of gaited and non-gaited motion patterns which can be modified online, i. e., during its execution (dynamic replanning).

Contact Planning Furthermore, the author developed a novel contact planning system which takes perception data from an arbitrary (yet compatible) CV system, assembles a corresponding environment model, and generates a discrete contact sequence towards a custom user-defined goal. In particular, this includes

- an open-source specification and implementation of the interface between an external CV system and the WPG module which has a strong focus on real-time performance,
- an application-oriented environment model formulation featuring a dynamic map size, multiple-levels of detail, and an automatic pre-evaluation of potential hand contact positions (multi-contact) based on local shape information of environmental objects,
- the introduction of QPWTs as an intuitive specification of almost arbitrary gaited locomotion patterns which supports non-horizontal ground, multi-contact configurations, and customization of the (task-space) motion in between discrete states,
- a strategy for efficient collision checks and multi-contact proximity tests based on SSVs (see Appendix C for details on the revision of the SSV library),
- a novel hierarchical contact planning pipeline featuring two subsequent graph searches (coarse-to-fine planning) which are coupled through a 2D map storing local metrics obtained from the first level (rasterization) used to accelerate the second level,
- a highly efficient graph search based on a custom implementation of the A* algorithm, custom data containers which are optimized for RTOSs, and various pre-computed (but scenario independent) acceleration buffers, and
- rather simple yet effective optimization schemes to compute the final foot / hand contact poses based on local shape information obtained from the environment model.

Motion Generation On the basis of LOLA's original real-time trajectory generation system designed by BUSCHMANN [100, p. 55ff], the author of the present thesis developed a new motion generator from the ground up. It features

- a sequential planning pipeline realized as a chain of consecutive sub-modules – each of which responsible of generating a certain component of the motion plan,
- a generalization of the concept of SS and DS phases known from biped walking to *load switch* and *motion* phases which allow the description of gaited multi-contact locomotion with synchronized foot and hand motion,
- generation of C^2 -continuous and natural (i. e. quaternion based) multi-axis rotations based on QBSplines driven by polynomial parameter splines (cf. Appendix B),
- a highly efficient open-source implementation of analytical polynomial and quaternion trajectories for describing translation and rotation in three-dimensional space,
- a ZMP planner with deterministic runtime generating horizontal and vertical reference trajectories solely based on geometric considerations,
- a foot and toe motion planner which adapts to local terrain information (e. g. for stepping up/down or traversing obstacles) and automatically establishes roll-off motion (heel-strike and toe-lift-off),
- heuristics based yet effective methods for planning hand motion and corresponding interaction forces within multi-contact situations,

- a strategy for blending the hands between null- and task-space (actual blending through smooth bilinear interpolation of parallel calculated IK solutions is realized within the SIK module contributed by FELIX SYGULLA – cf. Section 4.6),
- an effective method for maintaining kinematic feasibility based on a simplified model of the leg (extends the method of HILDEBRANDT et al. [8]), and
- an effective method for maintaining dynamic feasibility based on a novel five-mass model (extends three-mass model of BUSCHMANN et al. [98]) used to formulate an overdetermined BVP which is solved through a new quintic spline collocation algorithm (extends cubic spline collocation of BUSCHMANN et al. [98]).

Ecosystem Apart from the planning layer of the real-time locomotion system, the author of this thesis also made considerable contributions to the software ecosystem of the LOLA project. Most prominently, this includes

- extensions to the existing simulation framework by a more robust contact detection algorithm (multi-body simulation) and a (rather coarse) simulation of the CV system by synthesizing a corresponding environment model from a given scene description and
- useful tools for visualization such as a powerful interactive 3D viewer based on *Blender*.

Beyond that, large parts of the source code contributed by the author of this thesis have been published through the free and open-source C++ libraries *Broccoli* and *am2b-vision-interface*.

9.3 Conclusions

The main objective of this work was to enhance the versatility and robustness of biped locomotion for humanoid robots by making use of additional hand supports. In contrast to related work which is mostly restricted to quasi-static motions, a primary goal was to achieve (relatively) high velocities. To reduce the complexity of this problem, this thesis focused on gaited locomotion which preserves the main characteristics of biped walking. While this certainly represents some kind of restriction, still a large variety of motion patterns can be displayed. Another goal of this work was to achieve a high level of autonomy, i. e., the robot plans its motion autonomously solely based on knowledge of its surroundings and a user-specified goal position. In order to react to changes in the environment or user input within a reasonable amount of time, the total planning time should be kept in the range of the duration of one or two foot steps.

With the proposed modifications to the hard- and software of LOLA, all our original goals have been achieved. This is best demonstrated by the numerous published videos which have been referred to within this thesis. Apart from the videos of simulations and real-world experiments which verify the basic functionality, also the robustness and reliability of the proposed locomotion framework has been demonstrated, e. g. through the live experiment presented within the context of the 2020 *IEEE-RAS International Conference on Humanoid Robots* (July 19-21, 2021) which showcased LOLA's new multi-contact locomotion skills.

In the author's opinion, the reasons for this success are two-fold. First, the thorough design of LOLA's lower body hardware has been continued also for the new upper body. This includes an uncompromising lightweight design and strong efforts to find an optimal solution for our particular target application. With the recent multi-contact upgrade, LOLA's hardware is (at least) on a par with the currently best academic and commercial electrically driven humanoid robots. The second reason for the success is attributed to the hierarchical architecture of the locomotion framework which separates higher level planning based on rather coarse models living within an idealized world and lower level stabilization compensating the errors of these models and rejecting any kind of disturbances. In comparison to holistic frameworks which use

a very detailed model of the robot and its environment on the planning level, the hierarchical approach seems to lead to a higher overall robustness within real-world execution. Moreover, it allows to drastically reduce computational cost which makes real-time planning feasible.

Compared to other electrically driven, fully-actuated, and life-sized humanoids, LOLA's multi-contact locomotion capabilities are surely at the cutting edge – probably even leading current technology. This has been confirmed by national and international media through dedicated press reports (see [27] and the *IEEE Spectrum* article [26]). The most prominent features are the (relatively) high locomotion speed, the robustness even under large disturbances, and the fact that fully autonomous multi-contact planning runs onboard and in real-time. However, compared to the versatility, speed, and robustness of human locomotion, we still observe an enormous gap. Although the electromechanical hardware of LOLA still leaves room for improvement, the author believes that currently the most severe bottlenecks with regards to autonomous multi-contact locomotion lie in the area of visual perception and scene reconstruction.

9.4 Outlook

Considering the given time and manpower, the scope of the multi-contact revision of LOLA was remarkable. Despite the overall success of the project, the presented solution has to be seen as “first shot” which is far from being optimal. Accordingly, there are countless things which can be improved. The following paragraphs present a selection of recommended future extensions.

Visual Perception Since the low accuracy of the environment model is currently the main bottleneck within autonomous walking experiments, it is highly recommended to upgrade the visual perception system. For this purpose, one might replace the currently used low-cost cameras by more potent industrial-grade products. Furthermore, it should be considered to use multiple sensors attached to different segments of the robot (e. g. the knees) which would extend the field of view and simultaneously increase perception accuracy (shorter distances and data fusion for overlapping fields of view). Additionally, the generation of the height map from the surfel reconstruction is suboptimal (bumpy reconstruction of flat ground). Thus, a more application-oriented solution, e. g. by fitting a plane into the point-cloud cluster representing the floor, should be considered in future.

Mechanical Structures Through the multi-contact revision, the stiffness of LOLA's upper body has been significantly increased. However, in order to reduce undesired structural vibrations, one should also try to increase (structural) damping. For this purpose, it is suggested to identify critical parts of the current hardware configuration and optimize them with regards to their damping properties, e. g. by experimenting with compound materials or 3D (metal) printing technology. Bio-inspired designs currently seem to be the most promising approach. The gained insights could then be used as guidance for designing a potential successor of LOLA.

Hand Design Due to its simplicity, the current design of the arms allows rather dynamic motions. However, with the presented hand design, only unilateral interactions with the environment are possible. To remove this restriction, a rather simple mechanism – e. g. involving a single DoF gripper – could be used. Depending on its particular realization, this extension would possibly also require to add additional joints to the arm (e. g. 6 DoF topology). Unfortunately, this inevitably leads to an increase of mass effectively degrading the dynamic performance during multi-contact locomotion. Thus, one might try to find a compromise between dexterity and mass / inertia. Note that a fully articulated hand would further allow to investigate complex manipulation which, however, has not been in the scope of the LOLA project so far.

Torque Control For the actuators of the new upper body we reused the original modular joint drives by LOHMEIER [291, p. 96ff]. Consequently, also the new arms do not feature dedicated sensors for joint torque feedback. However, in order to benchmark position- against torque-control (and potential hybrid schemes), the robot would need to integrate torque sensors. Since this would require extensive changes to the hardware of LOLA, the author of thesis recommends to realize this idea with a potential successor platform instead.

Full (Hardware) Autonomy Within the context of this thesis, fully-autonomous locomotion describes the automatic generation and execution of a motion plan solely based on the provided environment model and without any additional user-input (except for specifying the desired goal position). However, for “real” autonomy of a humanoid robot, also the hardware needs to be autonomous, i. e., “detached”. For LOLA, this means that the high-level communication between the human operator and the robot has to be realized through a wireless technology. Since high-level signals are typically not time-critical (a delay of few milliseconds is not noticeable by humans anyway), an off-the-shelf solution seems to be sufficient. Furthermore, also an onboard power supply (e. g. in the form of a sufficiently dimensioned battery pack) is required. An entirely untethered operation further requires to detach the robot from its safety harness. Certainly, this demands a much more robust hardware which is able to survive falls.

In the author’s personal opinion, an onboard power supply is a reasonable extension for a future revision of LOLA since it affects the total weight and mass distribution which in turn changes the dynamic properties of the system. In contrast, wireless high-level communication is not expected to affect the performance of the robot in any way. Similarly, the safety harness represents only a minimal mechanical disturbance while drastically lowering the risk of damage during the test of new methods or very challenging maneuvers. While an entirely untethered operation makes sense with regard to the development of a commercial product, it seems to have more disadvantages than advantages for an academic research prototype.

Distributed Computation Except for the CV system which is executed on a separate board, the entire locomotion framework of LOLA runs on the same PC. Although an RTOS is used which allows to prioritize individual threads, lightweight low-level tasks still share common resources (CPU cache, RAM, etc.) with heavyweight high-level tasks. Since the current setup is reaching its limits with regard to the hard real-time constraints, the author recommends to distribute the computational workload to different boards. As a first step, it seems advantageous to execute the computational expensive contact planning pipeline on the vision PC.

Contact Planning The (multi-)contact planner presented in Chapter 5 is the most recent and consequently least mature component of the locomotion framework. In Section 5.6, concrete suggestions for future improvements and extensions have been presented.

Optimize Parameters Due to its complexity, the real-time locomotion framework of LOLA depends on a large amount of customizable parameters. Certainly, this holds also true for the new WPG module. Since the currently used parameter set has been determined by manual tuning, it has to be seen as a first “working” configuration which definitively can be optimized through dedicated parameter studies.

Appendix A

Notation

For mathematical formulations, this document adopts the notation of the *Chair of Applied Mechanics*, TUM, which is in large parts equivalent to the common conventions in the field of robotics. This section gives a brief summary of the most important definitions. Although quaternions are introduced in Appendix B, some formulations are presented already in this section to show their equivalence to rotation matrices with regard to notation.

Scalars, Vectors, and Matrices Scalars are represented by regular (non-bold) lower or upper case letters such as a or A . In contrast, a bold letter denotes either an one-dimensional array such as a vector or quaternion (typically lower case, e. g. \mathbf{a}) or a multi-dimensional array such as a matrix (typically upper case, e. g. \mathbf{A}). For vectors and matrices, the dimension is either explicitly defined upon first appearance, or it is implicitly given through the context. If not specified otherwise, vectors are column vectors.

Coordinate Systems (CoSys) and Frames of Reference (FoR) All considered CoSys alias *frames* are defined to be Cartesian and right-handed. Some quantities such as positions and orientations may be described in a certain CoSy acting as FoR. The corresponding FoR is denoted by a left hand subscript, e. g. ${}_X\mathbf{r}_P$ which denotes the position of the point P described in the frame X . Similarly, the rotation matrix ${}_X\mathbf{A}_Y$ or equivalently the (unit) quaternion ${}_X\mathbf{s}_Y$ denote the orientation of the frame Y described in the frame X . A special notation is used for wrenches and the mass moment of inertia tensor, for which (additional to the FoR) a reference point has to be specified. This is done by a right hand superscript, e. g. ${}_X\mathbf{W}^P$ which denotes a wrench acting at the point P or ${}_X\mathbf{\Theta}^P$ which denotes a mass moment of inertia tensor with respect to the point P (both described in the frame X).

Positions and Translations In general, *positions* are specified as vectors starting from the origin of the corresponding FoR, i. e., ${}_X\mathbf{r}_P$ denotes the vector from the origin of the frame X to the point P described in the frame X . In order to write the relative (difference) vector between two points, a second right hand subscript is added, e. g. ${}_X\mathbf{r}_{P-Q} := {}_X\mathbf{r}_Q - {}_X\mathbf{r}_P$ which denotes the vector from the point P to the point Q described in the frame X . Indeed, this notation implies the equivalence ${}_X\mathbf{r}_P = {}_X\mathbf{r}_{X-P} = {}_X\mathbf{r}_P - {}_X\mathbf{r}_X$ where the first variant is preferred for brevity (${}_X\mathbf{r}_X = \mathbf{0}$). A relative position can be understood as a *translation*, e. g. ${}_X\mathbf{r}_Q = {}_X\mathbf{r}_P + {}_X\mathbf{r}_{P-Q}$ represents a translation of the point P by ${}_X\mathbf{r}_{P-Q}$ resulting in the point Q .

Orientations and Rotations An *orientation* describes the alignment (as current state) of one CoSy with respect to another CoSy. A *rotation* typically denotes the action to get from one orientation to the other. Same as with positions and translations, the mathematical description of orientations and rotations is equivalent. As an example,

$${}_X\mathbf{r}_P = {}_X\mathbf{A}_Y {}_Y\mathbf{r}_P = \text{vec}\left({}_X\mathbf{s}_Y \otimes \overset{\text{real/vec}}{\text{quat}}(0, {}_Y\mathbf{r}_P) \otimes {}_X\bar{\mathbf{s}}_Y\right) = \underbrace{\text{rotMat}({}_X\mathbf{s}_Y)}_{{}_X\mathbf{A}_Y} {}_Y\mathbf{r}_P \quad (\text{for } \mathbf{r}_{X-Y} = \mathbf{0}) \quad (\text{A.1})$$

represents a (pure) rotation of the point P originally described in the frame Y (${}_Y\mathbf{r}_P$) by the orientation of the frame Y (${}_X\mathbf{A}_Y$ or equivalently ${}_X\mathbf{s}_Y$) such that it is finally described in the frame X (${}_X\mathbf{r}_P$). Here, we assumed that X and Y have the same origin, i. e., that $\mathbf{r}_{X-Y} = \mathbf{0}$ holds (no translation). Note that rotations can be chained, i. e., ${}_X\mathbf{A}_Z = {}_X\mathbf{A}_Y {}_Y\mathbf{A}_Z$ or equivalently ${}_X\mathbf{s}_Z = {}_X\mathbf{s}_Y \otimes_Y \mathbf{s}_Z$. This unveils the handiness of the chosen notation with left and right subscripts allowing quick feasibility checks.

Elementary Rotation Matrices The symbol $\mathbf{A}_{x|y|z}$, i. e., a rotation matrix lacking a left subscript, is used to denote an elementary rotation matrix either around the x -, y -, or z -axis. In particular, we define

$$\mathbf{A}_x(\varphi) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix}, \mathbf{A}_y(\varphi) := \begin{bmatrix} \cos \varphi & 0 & \sin \varphi \\ 0 & 1 & 0 \\ -\sin \varphi & 0 & \cos \varphi \end{bmatrix}, \mathbf{A}_z(\varphi) := \begin{bmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (\text{A.2})$$

Transforms A combination of rotation and translation can be used to move from one CoSy to the other. For the position of a point P , we can change the FoR from the frame Y to X by

$${}_X\mathbf{r}_P = {}_X\mathbf{r}_Y + {}_X\mathbf{A}_Y {}_Y\mathbf{r}_P \quad \text{or} \quad \begin{bmatrix} {}_X\mathbf{r}_P \\ 1 \end{bmatrix} = {}_X\mathbf{H}_Y \begin{bmatrix} {}_Y\mathbf{r}_P \\ 1 \end{bmatrix} \quad \text{with} \quad {}_X\mathbf{H}_Y := \begin{bmatrix} {}_X\mathbf{A}_Y & {}_X\mathbf{r}_Y \\ \mathbf{0} & 1 \end{bmatrix}, \quad (\text{A.3})$$

where ${}_X\mathbf{H}_Y$ denotes the *homogeneous transform* from the frame Y to the frame X . Same as for rotation matrices, homogeneous transforms can be chained, i. e., ${}_X\mathbf{H}_Z = {}_X\mathbf{H}_Y {}_Y\mathbf{H}_Z$. However, while ${}_X\mathbf{A}_Y^{-1} = {}_X\mathbf{A}_Y^T = {}_Y\mathbf{A}_X$ holds, the inverse of a homogeneous transform is given by

$${}_X\mathbf{H}_Y^{-1} = {}_Y\mathbf{H}_X = \begin{bmatrix} {}_X\mathbf{A}_Y^T & -{}_X\mathbf{A}_Y^T {}_X\mathbf{r}_Y \\ \mathbf{0} & 1 \end{bmatrix} \quad \text{such that} \quad \begin{bmatrix} {}_Y\mathbf{r}_P \\ 1 \end{bmatrix} = {}_Y\mathbf{H}_X \begin{bmatrix} {}_X\mathbf{r}_P \\ 1 \end{bmatrix}. \quad (\text{A.4})$$

Time Derivatives Derivatives with respect to time are indicated by a single or double dot above the symbol while higher order derivatives are denoted by a right hand superscript enclosed within brackets. Thus, $\dot{x} = dx/dt$ denotes the first, $\ddot{x} = d^2x/dt^2$ the second, and $x^{(3)} = d^3x/dt^3$ the third time derivative of x . If not specified otherwise, time derivatives of FoR-dependent quantities such as positions and orientations are defined to be *absolute*, i. e., with respect to an *inertial* (non-moving but probably jumping) FoR.

Partial Derivatives The partial derivative of a (column) vector with respect to another (column) vector follows the common definition of a Jacobian matrix. Most prominently, this applies to the task-space jacobian $\mathbf{J}(\mathbf{q})$ given by

$$\mathbf{J}(\mathbf{q}) := \left(\frac{\partial \dot{\mathbf{x}}(\mathbf{q})}{\partial \dot{\mathbf{q}}} \right) = \begin{bmatrix} \frac{\partial \dot{x}_1(\mathbf{q})}{\partial \dot{q}_1} & \dots & \frac{\partial \dot{x}_1(\mathbf{q})}{\partial \dot{q}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \dot{x}_p(\mathbf{q})}{\partial \dot{q}_1} & \dots & \frac{\partial \dot{x}_p(\mathbf{q})}{\partial \dot{q}_n} \end{bmatrix} \in \mathbb{R}^{p \times n}. \quad (\text{A.5})$$

Appendix B

Quaternion Calculus and Interpolation of Rotations using Quaternions

This section gives an introduction to the fundamentals of quaternions and presents a selection of interpolation methods. Within the context of this thesis, quaternions are used to describe spatial rotation, thus, a strong focus is put on *unit* quaternions. The contents of this section are gathered from various sources, most importantly [88, 115, 123, 155, 248–250, 310, 378], which are recommended references discussing the corresponding topics in much more detail.

B.1 Fundamentals

Quaternions have been first described by HAMILTON in 1843 [187]. They can be understood as an extension of the concept of complex numbers featuring three instead of one imaginary axis. As a tribute to their discoverer, the set of quaternions is typically denoted by \mathbb{H} . A quaternion is defined as the expression [310, p. 3f]

$$\mathbf{s} = s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k} \in \mathbb{H} \quad \text{with} \quad \underbrace{\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1}_{\text{Hamiltonian rule}} \quad \text{and} \quad s_w, s_x, s_y, s_z \in \mathbb{R}. \quad (\text{B.1})$$

Here, \mathbf{i} , \mathbf{j} , and \mathbf{k} denote the so-called *basic quaternions* alias *imaginary axes*, for which additionally the relations

$$\mathbf{ij} = \mathbf{k} = -\mathbf{ji} \quad \text{and} \quad \mathbf{jk} = \mathbf{i} = -\mathbf{kj} \quad \text{and} \quad \mathbf{ki} = \mathbf{j} = -\mathbf{ik} \quad (\text{pairwise anticommutativity}) \quad (\text{B.2})$$

hold [310, p. 4]. The coefficients s_w , s_x , s_y , and s_z represent real-valued scalars, where s_w takes a special role and is called *real* or alternatively *scalar* component of the quaternion. The three remaining components are typically called the *vector* part [310, p. 4]. To extract these parts from a given quaternion $\mathbf{s} \in \mathbb{H}$, we define the operators

$$\text{real}(s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k}) := s_w \in \mathbb{R} \quad \text{and} \quad \text{vec}(s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k}) := [s_x, s_y, s_z]^T \in \mathbb{R}^3. \quad (\text{B.3})$$

Conversely, we define an operator to construct a quaternion from these parts

$$\mathbf{s} = \text{quat}^{real/vec}(s_w, [s_x, s_y, s_z]^T) := s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k} \in \mathbb{H}. \quad (\text{B.4})$$

Arithmetic Operations The addition and subtraction of two quaternions s_1 and s_2 is simply performed component-wise [310, p. 5], i. e.,

$$\begin{aligned} \mathbf{s}_1 + \mathbf{s}_2 &:= (s_{1,w} + s_{2,w}) + (s_{1,x} + s_{2,x}) \mathbf{i} + (s_{1,y} + s_{2,y}) \mathbf{j} + (s_{1,z} + s_{2,z}) \mathbf{k} \in \mathbb{H}, \\ \mathbf{s}_1 - \mathbf{s}_2 &:= (s_{1,w} - s_{2,w}) + (s_{1,x} - s_{2,x}) \mathbf{i} + (s_{1,y} - s_{2,y}) \mathbf{j} + (s_{1,z} - s_{2,z}) \mathbf{k} \in \mathbb{H}. \end{aligned} \quad (\text{B.5})$$

For multiplication, we differentiate between three cases. The first case is a multiplication of a quaternion $\mathbf{s} \in \mathbb{H}$ with a real-valued scalar $\alpha \in \mathbb{R}$. Same as for addition and subtraction, this is a commutative, component-wise operation [310, p. 5]:

$$\alpha \mathbf{s} = \mathbf{s} \alpha := (\alpha s_w) + (\alpha s_x) \mathbf{i} + (\alpha s_y) \mathbf{j} + (\alpha s_z) \mathbf{k} \in \mathbb{H}. \quad (\text{B.6})$$

The second case represents the multiplication of two quaternions, typically referred to as *quaternion multiplication* or *HAMILTON product*, for which we use the special symbol \otimes . This operation is *not* commutative, i. e., in general $\mathbf{s}_1 \otimes \mathbf{s}_2 \neq \mathbf{s}_2 \otimes \mathbf{s}_1$ holds. The HAMILTON product of two quaternions $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{H}$ is defined as [310, p. 5]

$$\begin{aligned} \mathbf{s}_1 \otimes \mathbf{s}_2 := & (s_{1,w} s_{2,w} - s_{1,x} s_{2,x} - s_{1,y} s_{2,y} - s_{1,z} s_{2,z}) \\ & + (s_{1,w} s_{2,x} + s_{1,x} s_{2,w} + s_{1,y} s_{2,z} - s_{1,z} s_{2,y}) \mathbf{i} \\ & + (s_{1,w} s_{2,y} - s_{1,x} s_{2,z} + s_{1,y} s_{2,w} + s_{1,z} s_{2,x}) \mathbf{j} \\ & + (s_{1,w} s_{2,z} + s_{1,x} s_{2,y} - s_{1,y} s_{2,x} + s_{1,z} s_{2,w}) \mathbf{k} \in \mathbb{H}. \end{aligned} \quad (\text{B.7})$$

As third case, we introduce the *dot product* (alias *Euclidean inner product*) for which we use the special symbol \odot . The dot product of two quaternions $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{H}$ is defined by [310, p. 13f]

$$\mathbf{s}_1 \odot \mathbf{s}_2 = \mathbf{s}_2 \odot \mathbf{s}_1 := s_{1,w} s_{2,w} + s_{1,x} s_{2,x} + s_{1,y} s_{2,y} + s_{1,z} s_{2,z} \in \mathbb{R} \quad (\text{B.8})$$

and is commutative. The notation with $\mathbf{s}_1 \otimes \mathbf{s}_2$ and $\mathbf{s}_1 \odot \mathbf{s}_2$ for the HAMILTON and dot product of two quaternions is chosen such that the reader can easily distinct between these two operations. In contrast, related work (e. g. [310]) typically uses $\mathbf{s}_1 \mathbf{s}_2$ and $\mathbf{s}_1 \cdot \mathbf{s}_2$, which (in the author's personal opinion) is less explicit and may let the reader forget to pay attention to the non-commutativity of the HAMILTON product.

Modulus, Unit Quaternions, and Identities The *modulus* alias *absolute value* of a quaternion is given by the Euclidean length of its representation as \mathbb{R}^4 vector [310, p. 10f], i. e.,

$$|\mathbf{s}| = |s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k}| := \sqrt{s_w^2 + s_x^2 + s_y^2 + s_z^2} \in \mathbb{R}. \quad (\text{B.9})$$

A quaternion \mathbf{s} with modulus $|\mathbf{s}| = 1$ is called *unit quaternion*. Within this thesis, the set of unit quaternions is denoted with \mathbb{H}^1 , i. e.,

$$\mathbb{H}^1 := \{\mathbf{s} \in \mathbb{H} \mid |\mathbf{s}| = 1\}. \quad (\text{B.10})$$

A special unit quaternion is the *multiplicative identity quaternion* $\mathbf{1}_{\mathbb{H}}$ [310, p. 7f] given by

$$\mathbf{1}_{\mathbb{H}} := 1 + 0 \mathbf{i} + 0 \mathbf{j} + 0 \mathbf{k} \in \mathbb{H} \quad (s_w = 1, s_x = s_y = s_z = 0) \quad (\text{B.11})$$

for which the important property $\mathbf{1}_{\mathbb{H}} \otimes \mathbf{s} = \mathbf{s} \otimes \mathbf{1}_{\mathbb{H}} = \mathbf{s}$ holds for an arbitrary $\mathbf{s} \in \mathbb{H}$. The *additive identity quaternion* $\mathbf{0}_{\mathbb{H}}$ [310, p. 7f] (alias *zero quaternion*) is given by

$$\mathbf{0}_{\mathbb{H}} := 0 + 0 \mathbf{i} + 0 \mathbf{j} + 0 \mathbf{k} \in \mathbb{H} \quad (s_w = s_x = s_y = s_z = 0) \quad (\text{B.12})$$

which represents the neutral element of addition, i. e., $\mathbf{s} + \mathbf{0}_{\mathbb{H}} = \mathbf{0}_{\mathbb{H}} + \mathbf{s} = \mathbf{s}$ for an arbitrary $\mathbf{s} \in \mathbb{H}$.

Normalization In order to obtain a unit quaternion from an arbitrary (non-zero) quaternion, a *normalization* operation (in [310, p. 23] called *quaternion sign*) is defined. Similar to the normalization of a vector in \mathbb{R}^n , this is done by

$$\text{norm}(\mathbf{s}) := \frac{\mathbf{s}}{|\mathbf{s}|} \in \mathbb{H}^1 \quad \text{for } \mathbf{s} \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}}. \quad (\text{B.13})$$

Conjugate and Inverse The quaternion \bar{s} given by

$$\bar{s} := s_w - s_x \mathbf{i} - s_y \mathbf{j} - s_z \mathbf{k} \in \mathbb{H} \quad \text{with} \quad s = s_w + s_x \mathbf{i} + s_y \mathbf{j} + s_z \mathbf{k} \in \mathbb{H} \quad (\text{B.14})$$

is called *conjugate quaternion* of s [310, p. 9f]. Together with the modulus $|s|$, the conjugate \bar{s} can be used to obtain the *multiplicative inverse* s^{-1} [310, p. 10f] defined as

$$s^{-1} := \frac{\bar{s}}{|s|^2} \in \mathbb{H} \quad \text{for} \quad s \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}} \quad (\text{B.15})$$

for which $s^{-1} \otimes s = s \otimes s^{-1} = \mathbf{1}_{\mathbb{H}}$ holds for an arbitrary $s \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}}$. For unit quaternions the inverse is equivalent to the conjugate, i. e., $s^{-1} = \bar{s}$ if $s \in \mathbb{H}^1$. Finally, the *additive inverse* of a quaternion s is given by $-s$, such that $s + (-s) = \mathbf{0}_{\mathbb{H}}$ holds for an arbitrary $s \in \mathbb{H}$.

Exponential and Logarithm For an arbitrary quaternion $s \in \mathbb{H}$ with corresponding vector part $\mathbf{v} := \text{vec}(s) \in \mathbb{R}^3$, the *quaternion natural exponential function* is given by [310, p. 88ff]

$$e^s = \exp(s) := \sum_{i=0}^{\infty} \frac{s^i}{i!} = \begin{cases} e^{s_w} \text{quat}(\cos(\|\mathbf{v}\|), \sin(\|\mathbf{v}\|) \frac{\mathbf{v}}{\|\mathbf{v}\|}) \in \mathbb{H} & \text{if } \|\mathbf{v}\| \geq \varepsilon, \\ \text{quat}(e^{s_w}, \mathbf{v}) \in \mathbb{H} & \text{else} \end{cases} \quad (\text{B.16})$$

where the second case represents a fallback to avoid division by (almost) zero using the relation $\lim_{x \rightarrow 0} \sin(x)/x = 1$. Its inverse operation, the *quaternion natural logarithm function* is given by

$$\ln(s) := \begin{cases} \text{quat}(\ln(|s|), \text{atan2}(\|\mathbf{v}\|, s_w) \frac{\mathbf{v}}{\|\mathbf{v}\|}) \in \mathbb{H} & \text{if } \|\mathbf{v}\| \geq \varepsilon, \\ \text{quat}(\ln(|s|), \frac{\mathbf{v}}{s_w} \left(1 - \frac{\|\mathbf{v}\|^2}{3s_w^2}\right)) \in \mathbb{H} & \text{if } \|\mathbf{v}\| < \varepsilon \wedge s_w > \varepsilon, \\ \text{undefined} & \text{else} \end{cases} \quad (\text{B.17})$$

where the first case $\|\mathbf{v}\| \geq \varepsilon$ denotes its basic definition [310, p. 92f] and the second case $\|\mathbf{v}\| < \varepsilon \wedge s_w > \varepsilon$ represents a fallback to avoid division by (almost) zero using a TAYLOR series expansion of the arctangent (up to the second term) as suggested in [385, p. 11]. Note that similar to $\ln(0)$, also $\ln(\mathbf{0}_{\mathbb{H}})$ is not defined (third case). Moreover, following its original definition, $\exp(s_1) = s_2$ has infinitely many solutions $s_1 = \ln(s_2)$ for $s_2 \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}}$. Indeed, for an exact definition, we would have to add the term $2\pi n$ with $n \in \mathbb{Z}$ to the result of atan2 in Equation B.17, see [310, p. 92f] for details. Within the scope of this thesis, we use the special case $n = 0$ such that we can use the definition given in Equation B.17.

Power The *quaternion power function* $s_1^{s_2}$ with $s_1, s_2 \in \mathbb{H}$ is defined by [310, p. 97f]

$$s_1^{s_2} := \exp(\ln(s_1) \otimes s_2) \in \mathbb{H} \quad \text{for} \quad s_1 \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}}. \quad (\text{B.18})$$

We obtain a similar expression for the power function s^α with $s \in \mathbb{H}$ and $\alpha \in \mathbb{R}$ by inserting $s_2 = \alpha + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k} = \alpha \mathbf{1}_{\mathbb{H}}$ into Equation B.18:

$$s^\alpha := \exp(\alpha \ln(s)) \in \mathbb{H} \quad \text{for} \quad s \in \mathbb{H} \setminus \mathbf{0}_{\mathbb{H}}. \quad (\text{B.19})$$

If the exponent is a positive integer $n \in \mathbb{N}_0$, it is also possible to use

$$s^n := \prod_{i=1}^n s = \begin{cases} \mathbf{1}_{\mathbb{H}} & \text{if } n = 0, \\ s & \text{if } n = 1, \\ s \otimes \dots \otimes s & \text{else} \end{cases} \quad (\text{B.20})$$

instead. Certainly, for high values of n , Equation B.19 will be more efficient than the repeated multiplication in Equation B.20.

B.2 Spatial Rotation

There are numerous ways of describing orientations and rotations in three-dimensional space. Prominent examples are rotation matrices, the set of EULER (proper EULER and TAIT-BRYAN) angle parametrizations, the axis-angle representation, rotation vectors, and quaternions. Unfortunately, many concepts and relations known from the space \mathbb{R}^3 do not have a direct or equivalent counterpart in $SO(3)$, which often makes the handling of spatial rotations difficult. Indeed, no “superior” representation solving all difficulties is known. Instead, each of the mentioned descriptions has its individual advantages and disadvantages such that the representation should be selected with respect to the particular application. For the WPG module of LOLA, quaternions have been selected as the primary formulation while other parametrizations are used only in special cases. This section introduces the description of spatial rotations with quaternions and provides some useful formulas for converting them to other formulations.

Quaternions Orientations in three-dimensional space can be described by the subset of *unit* quaternions $\mathbb{H}^1 \subseteq \mathbb{H}$. Following the notation introduced in Appendix A, the quaternion ${}_X s_Y \in \mathbb{H}^1$ denotes the orientation of the frame Y described in the frame X . This can be used to rotate a point P through

$${}_X r_P = \text{vec}\left({}_X s_Y \otimes \overset{\text{real/vec}}{\text{quat}}(0, {}_Y r_P) \otimes {}_X \bar{s}_Y\right) \quad (\text{for } r_{X-Y} = 0) \quad (\text{B.21})$$

which is equivalent to a change of the FoR from Y to X assuming pure rotation without translation. In contrast to minimal⁹² representations such as EULER angles, quaternions are free of singularities, thus, they avoid the common problem of *gimbal lock* (loss of DoF).

Same as with rotation matrices, a simple multiplication of quaternions is sufficient to describe a chained rotation, i. e., ${}_X s_Z = {}_X s_Y \otimes {}_Y s_Z$. Since the HAMILTON product involves only 28 operations (16 multiplications and 12 additions) while the multiplication of two rotation matrices accounts for 45 operations (27 multiplications and 18 additions), chaining rotations is more efficient using quaternions. Moreover, a convenient property of unit quaternions is that their inverse is equal to their conjugate, such that ${}_X s_Y = {}_Y s_X^{-1} = {}_Y \bar{s}_X$ holds. Thus, computing the inverse alias conjugate is extremely efficient and boils down to flipping the sign of three coefficients while, in theory, the transposition of a rotation matrix requires 9 operations.

Another notable property of quaternions in the context of spatial rotation is that ${}_X s_Y$ and its additive inverse $-{}_X s_Y$ describe the same orientation. Replacing ${}_X s_Y$ with $-{}_X s_Y$ and ${}_X \bar{s}_Y$ with $-{}_X \bar{s}_Y$ in Equation B.21 gives an immediate proof. This property originates from the fact that any orientation can be represented as a single-axis rotation either by a positive angle $\varphi_1 \in [0, 2\pi[$, or its equivalent negative angle $\varphi_2 = \varphi_1 - 2\pi \in [-2\pi, 0[$. Or, more generally formulated considering multiple revolutions, by any $\varphi_1 + 2\pi n$ with $n \in \mathbb{Z}$. Indeed, quaternions do not (directly) allow to describe rotation by multiple revolutions, i. e., the information about the count of revolutions is lost once we convert to a quaternion. However, by flipping the sign of a quaternion we can reverse the “direction of movement” (still leads to the same final orientation).

For considering motion, the computation of the angular velocity and acceleration from the corresponding ${}_X s_Y$, ${}_X \dot{s}_Y$, and ${}_X \ddot{s}_Y$ is quite simple and efficient, especially when compared to other parametrizations. In particular, the angular velocity ${}_X \omega_Y$ and the angular acceleration ${}_X \dot{\omega}_Y$ of the frame Y described in the frame X are given by [115]

$${}_X \omega_Y := 2 \text{vec}({}_X \dot{s}_Y \otimes {}_X \bar{s}_Y) = -2 \text{vec}({}_X s_Y \otimes {}_X \dot{\bar{s}}_Y) \quad (\text{B.22})$$

⁹²A minimum of three parameters is necessary to describe an arbitrary orientation in three-dimensional space. Obviously, a unit quaternion $\mathbf{s} \in \mathbb{H}^1$ features four parameters $s_w, s_x, s_y,$ and s_z , which, however, are constrained by the requirement $s_w^2 + s_x^2 + s_y^2 + s_z^2 = 1$.

and

$$\begin{aligned} {}_X\dot{\boldsymbol{\omega}}_Y &:= 2 \operatorname{vec}({}_X\ddot{\mathbf{s}}_Y \otimes {}_X\bar{\mathbf{s}}_Y + {}_X\dot{\mathbf{s}}_Y \otimes {}_X\dot{\bar{\mathbf{s}}}_Y) = -2 \operatorname{vec}({}_X\mathbf{s}_Y \otimes {}_X\ddot{\bar{\mathbf{s}}}_Y + {}_X\dot{\mathbf{s}}_Y \otimes {}_X\dot{\bar{\mathbf{s}}}_Y) \\ &= 2 \operatorname{vec}({}_X\ddot{\mathbf{s}}_Y \otimes {}_X\bar{\mathbf{s}}_Y - ({}_X\dot{\mathbf{s}}_Y \otimes {}_X\bar{\mathbf{s}}_Y)^2) = -2 \operatorname{vec}({}_X\mathbf{s}_Y \otimes {}_X\ddot{\bar{\mathbf{s}}}_Y - ({}_X\dot{\mathbf{s}}_Y \otimes {}_X\bar{\mathbf{s}}_Y)^2) \end{aligned} \quad (\text{B.23})$$

where we used the relation $\dot{\mathbf{s}} \otimes \bar{\mathbf{s}} = -\mathbf{s} \otimes \dot{\bar{\mathbf{s}}}$ for an arbitrary $\mathbf{s} \in \mathbb{H}^1$ and $\dot{\mathbf{s}} \in \mathbb{H}$ (obtained from deriving $\mathbf{s} \otimes \bar{\mathbf{s}} = \mathbf{1}_{\mathbb{H}}$ with respect to time resulting in $\dot{\mathbf{s}} \otimes \bar{\mathbf{s}} + \mathbf{s} \otimes \dot{\bar{\mathbf{s}}} = \mathbf{0}_{\mathbb{H}}$). Note that the second row in Equation B.23 involves the product ${}_X\dot{\mathbf{s}}_Y \otimes {}_X\bar{\mathbf{s}}_Y$ which already appeared in Equation B.22. Thus, simultaneous evaluation of both, ${}_X\boldsymbol{\omega}_Y$ and ${}_X\dot{\boldsymbol{\omega}}_Y$, can be implemented very efficiently. Finally, it has to be highlighted that although ${}_X\mathbf{s}_Y$ is a unit quaternion, this does not hold true for its time derivatives. As a consequence, in general ${}_X\dot{\mathbf{s}}_Y^{-1} \neq {}_X\dot{\bar{\mathbf{s}}}_Y$ (and ${}_X\mathbf{s}_Y^{-1} \neq {}_X\bar{\mathbf{s}}_Y$, etc.) holds.

Rotation Matrices In order to convert the unit quaternion ${}_X\mathbf{s}_Y \in \mathbb{H}^1$ into a corresponding rotation matrix ${}_X\mathbf{A}_Y \in \mathbb{R}^{3 \times 3}$, we define the operator [310, p.40ff]

$${}_X\mathbf{A}_Y = \operatorname{rotMat}({}_X\mathbf{s}_Y) := \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(wy + xz) \\ 2(wz + xy) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(wx + yz) & 1 - 2(x^2 + y^2) \end{bmatrix} \in \mathbb{R}^{3 \times 3} \quad (\text{B.24})$$

where we used the abbreviations $w := {}_Xs_{Y,w}$, $x := {}_Xs_{Y,x}$, $y := {}_Xs_{Y,y}$, and $z := {}_Xs_{Y,z}$ for brevity. From Equation B.24, it is also possible to define an inverse operation, i. e., the construction of a quaternion from a given rotation matrix. However, this requires special attention to avoid numerical issues, especially if the provided matrix is not orthogonal due to numerical errors. This unveils another issue of rotation matrices in the context of chained transformations: while the re-orthogonalization of rotation matrices is rather expensive, the normalization of a quaternion by Equation B.13 is relatively cheap (same as normalizing an \mathbb{R}^4 vector). Indeed, the WPG module of LOLA re-normalizes orientations represented as quaternions at meaningful places (e. g. after extensive transformations). Moreover, the conversion of quaternions to rotation matrices always represents the very last step, while the inverse operation is avoided completely.

Within the WPG module of LOLA, rotation matrices are mainly used to rotate vectors in \mathbb{R}^3 such as positions and velocities. This is motivated by the high efficiency of multiplying a 3×3 matrix with a three-dimensional vector involving only 15 operations (9 multiplications and 6 additions) which is less than what is required with Equation B.21. In many cases, multiple vectors have to be rotated by the same ${}_X\mathbf{s}_Y$ such that it is more efficient to convert ${}_X\mathbf{s}_Y$ to ${}_X\mathbf{A}_Y$ first and perform the actual rotation of the vectors with ${}_X\mathbf{A}_Y$.

Axis-Angle Representation There is a close relationship between the formulation of orientations by unit quaternions and the axis-angle representation. In particular, the construction of a unit quaternion from an axis-angle pair (\mathbf{a}, φ) is given by [310, p. 42ff]:

$$\begin{aligned} \mathbf{s} &= \operatorname{quat}^{\text{ax/ang}}(\mathbf{a}, \varphi) := \cos\left(\frac{\varphi}{2}\right) + (a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k}) \sin\left(\frac{\varphi}{2}\right) \in \mathbb{H}^1 \\ &= \operatorname{quat}^{\text{real/vec}}\left(\cos\left(\frac{\varphi}{2}\right), \sin\left(\frac{\varphi}{2}\right) \mathbf{a}\right) \in \mathbb{H}^1 \end{aligned} \quad (\text{B.25})$$

where $\mathbf{a} = [a_x, a_y, a_z]^T \in \mathbb{R}^3$ with $\|\mathbf{a}\| = 1$ represents the (normalized) axis and $\varphi \in \mathbb{R}$ denotes the angle (positive according to the right hand rule). The periodicity of the involved sine and cosine highlights at which point the information about multiple revolutions gets lost within this conversion. Note that this operator is not to be mistaken with the constructor from the real and vector part defined in Equation B.4, which also takes a three-dimensional vector and scalar as argument but in the opposite order.

For the inverse operation, i. e., computing the axis-angle pair (\mathbf{a}, φ) from a given unit quaternion $\mathbf{s} \in \mathbb{H}^1$ with corresponding vector part $\mathbf{v} := \text{vec}(\mathbf{s}) \in \mathbb{R}^3$, one can easily find

$$\text{axAng}(\mathbf{s}) := (\mathbf{a}, \varphi) \quad \text{with} \quad \begin{cases} \mathbf{a} = \frac{\mathbf{v}}{\|\mathbf{v}\|}, & \varphi = 2 \text{atan2}(\|\mathbf{v}\|, s_w) & \text{if } \|\mathbf{v}\| \geq \varepsilon, \\ \mathbf{a} = [1, 0, 0]^T, & \varphi = 0 & \text{else} \end{cases} \quad (\text{B.26})$$

where $\|\mathbf{a}\| = 1$ and $\varphi \in [0, 2\pi]$ due to $\|\mathbf{v}\| \geq 0$. The second case represents a fallback to avoid division by (almost) zero which occurs for the identity rotation with $\mathbf{s} = \pm \mathbf{1}_{\mathbb{H}}$. The combination of a rotation axis and a scalar angle is probably the most intuitive and human understandable way of describing the relative orientation between two CoSys. Thus, within the WPG module of LOLA, this representation is mainly used for constructing quaternions, i. e., to take a user-specified (\mathbf{a}, φ) and convert it to a corresponding quaternion using Equation B.25.

EULER Angles The axis-angle representation may also be used to convert an arbitrary EULER angle parametrization into a corresponding quaternion. For this, the three elementary (single-axis) rotations of the chosen EULER convention are written as axis-angle pairs (e. g. $(\mathbf{e}_x, \varphi_x)$, $(\mathbf{e}_y, \varphi_y)$, and $(\mathbf{e}_z, \varphi_z)$) and individually converted to corresponding quaternions using Equation B.25. The three resulting quaternions are then multiplied using the HAMILTON product, which finally leads to the overall quaternion describing the multi-axis rotation. For an efficient implementation, one can exploit the special structure of \mathbf{e}_x , \mathbf{e}_y , and \mathbf{e}_z . The inverse operation, i. e., computing EULER angles from the corresponding quaternion, is slightly more involved and differs depending on the order of elementary rotations. Within the WPG module of LOLA, EULER angles are used for the definition of *Step Parameters*, see Appendix D.

Rotation Vectors A rotation vector $\boldsymbol{\vartheta}$ is just another form of the axis-angle representation (\mathbf{a}, φ) , where the angle is encoded as length of the vector describing the axis, i. e., $\boldsymbol{\vartheta} := \varphi \mathbf{a} \in \mathbb{R}^3$ with $\|\boldsymbol{\vartheta}\| = \varphi$. Thus, for constructing a unit quaternion from a corresponding rotation vector we can use the axis-angle representation as intermediate step. Indeed, it can be easily shown that this is equivalent to the operator

$${}_X \mathbf{s}_Y = \text{quat}^{\text{rot-vec}}({}_X \boldsymbol{\vartheta}_Y) := \exp\left(\text{quat}^{\text{real/vec}}\left(0, \frac{{}_X \boldsymbol{\vartheta}_Y}{2}\right)\right) \in \mathbb{H}^1 \quad (\text{B.27})$$

where we reuse the quaternion natural exponential function from Equation B.16 which is robust against the case of ${}_X \boldsymbol{\vartheta}_Y \rightarrow \mathbf{0}$, i. e., an (almost) identity rotation. The inverse operation of constructing a rotation vector from a unit quaternion is given by

$${}_X \boldsymbol{\vartheta}_Y = \text{rotVec}({}_X \mathbf{s}_Y) := 2 \text{vec}(\ln({}_X \mathbf{s}_Y)) \in \mathbb{R}^3 \quad \text{for } {}_X \mathbf{s}_Y \in \mathbb{H}^1 \quad (\text{B.28})$$

where we reuse the quaternion natural logarithm function from Equation B.17 which is robust against the case of ${}_X \mathbf{s}_Y \rightarrow \pm \mathbf{1}_{\mathbb{H}}$, i. e., an (almost) identity rotation.

The representation of orientations as rotation vectors is used within the task-space formulation of LOLA, see Section 4.3. Indeed, they simplify the modification of trajectories within the SIK module. In contrast, the WPG module relies on the axis-angle representation whenever an intuitive formulation is required while rotation vectors are only computed in the very last post-processing step to create the task-space vector \mathbf{x} .

Minimum Rotation In some situations it is required to determine the minimum rotation to get from one orientation to the other. Again, the axis-angle representation comes in handy since it explicitly specifies the desired axis (alias “direction”) and angle (alias “distance”). If we consider a frame Y with orientation ${}_X \mathbf{s}_Y \in \mathbb{H}^1$ and a frame Z with orientation ${}_X \mathbf{s}_Z \in \mathbb{H}^1$ (both described in an arbitrary frame X), then the orientation of the frame Z described in the frame Y is given by

${}_Y \mathbf{s}_Z = {}_X \bar{\mathbf{s}}_Y \otimes {}_X \mathbf{s}_Z$. Through $(\mathbf{a}, \varphi) = \text{axAng}({}_Y \mathbf{s}_Z)$, it is possible to determine the rotation axis \mathbf{a} and the angle φ . In case ${}_Y s_{Z,w} < 0$, it follows that $\pi < \varphi \leq 2\pi$ (cf. Equation B.26), which means that the pair (\mathbf{a}, φ) does not represent the minimum rotation (with respect to the rotation angle). However, we can simply use

$$\varphi_{\min} := \begin{cases} \varphi & \text{if } \varphi \leq \pi, \\ \varphi - 2\pi & \text{else} \end{cases} \quad \text{with } \varphi \in [0, 2\pi] \text{ and } \varphi_{\min} \in]-\pi, \pi] \quad (\text{B.29})$$

such that the pair $(\mathbf{a}, \varphi_{\min})$ or equivalently the dual pair $(-\mathbf{a}, -\varphi_{\min})$ describes the minimum rotation between the frame Y and the frame Z .

It can be easily verified that instead of subtracting 2π , it is also possible to flip the sign of one of the input quaternions ${}_X \mathbf{s}_Y$ or ${}_X \mathbf{s}_Z$ such that $\text{axAng}(-{}_Y \mathbf{s}_Z)$ directly gives us the minimum rotation. This property will be exploited during the interpolation of rotations (see Appendix B.3), where the user specifies a sequence of *keyframe* quaternions, which should be traversed using the minimum rotation between two keyframes. Indeed, the sign of one of the input quaternions only has to be flipped in case of ${}_Y s_{Z,w} < 0$ which implies $\varphi > \pi$ (cf. Equation B.26). Note that we can efficiently compute ${}_Y s_{Z,w}$ through the dot product of the input quaternions

$${}_Y s_{Z,w} = \text{real}({}_Y \mathbf{s}_Z) = \text{real}({}_X \bar{\mathbf{s}}_Y \otimes {}_X \mathbf{s}_Z) = {}_X \mathbf{s}_Y \odot {}_X \mathbf{s}_Z = \cos\left(\frac{\varphi}{2}\right). \quad (\text{B.30})$$

Thus, we can avoid computing the HAMILTON product ${}_Y \mathbf{s}_Z = {}_X \bar{\mathbf{s}}_Y \otimes {}_X \mathbf{s}_Z$ (28 operations) by using the much more efficient dot product ${}_X \mathbf{s}_Y \odot {}_X \mathbf{s}_Z$ (7 operations) instead.

B.3 Interpolation

In the following, selected quaternion-based methods for interpolating a user-defined sequence of *keyframe* orientations are introduced. Compared to \mathbb{R}^3 , interpolation in $\text{SO}(3)$ turns out to be much more complex. While translations are combined in \mathbb{R}^3 through *addition*, rotations are chained in $\text{SO}(3)$ through (non-commutative) *multiplication*. As a consequence, linear relationships in \mathbb{R}^3 often become non-linear when transferred to $\text{SO}(3)$. This makes the development of advanced interpolation methods (e. g. enforcing smoothness) difficult, cf. [249].

A rather simple approach for interpolating spatial rotation is to use EULER angles and to interpolate its three parameters using well-known techniques from \mathbb{R} , e. g. linear interpolation by $\varphi(\eta) = \varphi(0) + \eta(\varphi(1) - \varphi(0))$ with $\eta \in [0, 1]$. However, the resulting motion between keyframes strongly depends on the chosen EULER angle convention, i. e., the order of elementary rotations. Additionally, given a certain angular velocity, the rate of change of the EULER angles grows without limitation in the vicinity of singularities. In contrast, quaternions are immune to gimbal lock and allow to interpolate multi-axis rotation between keyframes in a “natural” manner, which makes the resulting motion behave more like a human would expect it to be. The high computational efficiency and numerical stability of quaternions are additional arguments which motivate their use for interpolating rotations in the WPG module of LOLA.

Source Code An efficient implementation of the presented interpolation methods has been published by the author of this thesis as part⁹³ of the free and open-source library *Broccoli* [15] (see Section 7.2). A special focus was set on efficiency and real-time performance. For interpolation methods where an analytic formulation of the derivative is known, the library implements it up to the highest possible order. Furthermore, the *Broccoli* implementation provides methods to efficiently evaluate the current orientation, angular velocity, and angular acceleration through a single function call, which automatically re-uses intermediate results for highest performance.

⁹³See the module curve of *Broccoli*.

The library distinguishes between

- quaternion *curves* interpolating two (or more) keyframe quaternions,
- quaternion *splines* as a concatenation of consecutive quaternion curves, and
- quaternion *trajectories* combining a quaternion spline (\mathbb{H}^1) with a parameter spline (\mathbb{R}) for advanced speed control (see Appendix B.3.8).

Pre-Processing of Keyframe Quaternions The primary input of all interpolation methods is a user-defined sequence of $n > 1$ keyframe quaternions $\{s_i\}$ with $s_i \in \mathbb{H}^1$ and $i = 1, \dots, n$. The very first ($i = 1$) and last ($i = n$) keyframe typically specify the orientation at the start and end of the motion while the interior keyframes ($1 < i < n$) represent *through-quaternions* or alternatively *control-quaternions* similar to through-points and control-points known from interpolation in \mathbb{R} . Since we are interested in a minimum rotation between keyframes (alias “shortest path”), we have to make sure that the sequence of keyframe quaternions is specified accordingly. For this purpose, we use the findings from Appendix B.2 to check each pair of consecutive keyframes (s_i, s_{i+1}) and flip the sign of the second quaternion if necessary. In particular, we execute Algorithm B.1 as a pre-processing step (applied for all presented interpolation methods).

Algorithm B.1: Pre-processing of keyframe quaternions to ensure minimum rotation between keyframes. Note that the unit quaternion s and its additive inverse $-s$ describe the same orientation (see Appendix B.2 and Equation B.21).

Input: Sequence of $n > 1$ keyframe quaternions $\{s_{\text{pre},i}\}$ with $s_{\text{pre},i} \in \mathbb{H}^1$ and $i = 1, \dots, n$

Output: Sequence of n keyframe quaternions $\{s_{\text{new},i}\}$ with $s_{\text{new},i} \odot s_{\text{new},i+1} \geq 0$

```

begin
  {s_new,i} ← {s_pre,i} // initialize keyframe sequence
  for i = 1 to n - 1 do
    if s_new,i ⊙ s_new,i+1 < 0 then // test for minimum rotation according to Equation B.30
      | s_new,i+1 ← -s_new,i+1 // flip sign to obtain minimum rotation
    end
  end
end
end

```

In contrast to formulations which directly encode angles (e. g. EULER angles, the axis-angle representation, or rotation vectors), quaternions and rotation matrices do not allow to describe multiple revolutions. In some cases this can be beneficial, especially when one is only interested in the final orientation rather than the transition. Within the context of this thesis, quaternions are used to describe the orientation of task-space CoSys such as the UB frame or the VTCP frame, which typically do not undergo full revolutions – at least not within a single DS or SS phase. However, for applications where multiple revolutions are possible (imagine an airplane performing multiple loopings), this restriction can be easily lifted by inserting a sufficient count of intermediate keyframes.

Exemplary Keyframe Sequence For demonstration purposes, we define the following exemplary sequence of $n = 7$ keyframe quaternions:

$$\begin{aligned}
 s_1 &:= 1 & +0\mathbf{i} & +0\mathbf{j} & +0\mathbf{k}, & \text{(start with identity rotation)} \\
 s_2 &:= \sqrt{3/4} & +0\mathbf{i} & +0.5\mathbf{j} & +0\mathbf{k}, & \text{(\varphi}_y = 60^\circ) \\
 s_3 &:= \sqrt{3/8} & +\sqrt{1/8}\mathbf{i} & +\sqrt{1/8}\mathbf{j} & -\sqrt{3/8}\mathbf{k}, & \text{(first } \varphi_z = -90^\circ, \text{ then } \varphi_y = 60^\circ) \\
 s_4 &:= 0 & +0.5\mathbf{i} & +0\mathbf{j} & -\sqrt{3/4}\mathbf{k}, & \text{(first } \varphi_z = -180^\circ, \text{ then } \varphi_y = 60^\circ) \\
 s_5 &:= -\sqrt{3/8} & +\sqrt{1/8}\mathbf{i} & -\sqrt{1/8}\mathbf{j} & -\sqrt{3/8}\mathbf{k}, & \text{(first } \varphi_z = -270^\circ, \text{ then } \varphi_y = 60^\circ) \\
 s_6 &:= -\sqrt{3/4} & +0\mathbf{i} & -0.5\mathbf{j} & +0\mathbf{k}, & \text{(first } \varphi_z = -360^\circ, \text{ then } \varphi_y = 60^\circ) \\
 s_7 &:= -1 & +0\mathbf{i} & +0\mathbf{j} & +0\mathbf{k}. & \text{(return to starting orientation)}
 \end{aligned} \tag{B.31}$$

B.3.1 Linear Interpolation (LERP)

The simplest way of interpolating two quaternions is to linearly interpolate the components s_w , s_x , s_y , and s_z . We name this method *Linear Interpolation (LERP)* (also called *straight line in-betweening* by SHOEMAKE [378]). In particular, we define the operator

$$s(\eta) = \text{LERP}(s_1, s_2, \eta) := (1 - \eta)s_1 + \eta s_2 \in \mathbb{H} \quad \text{with } \eta \in [0, 1]. \quad (\text{B.32})$$

The i -th derivative of LERP is given by

$$\frac{\partial^i s(\eta)}{\partial \eta^i} = \begin{cases} s_2 - s_1 & \text{if } i = 1, \\ \mathbf{0}_{\mathbb{H}} & \text{if } i > 1. \end{cases} \quad (\text{B.33})$$

While being simple and efficient, LERP has several drawbacks which makes it inappropriate for many use cases. Most importantly, even if s_1 and s_2 are unit quaternions, the resulting $s(\eta)$ is not (only for $\eta = 0$ and $\eta = 1$). Moreover, a quaternion spline chaining multiple interconnected LERP segments is only C^0 -continuous at most, see Figure B.1.

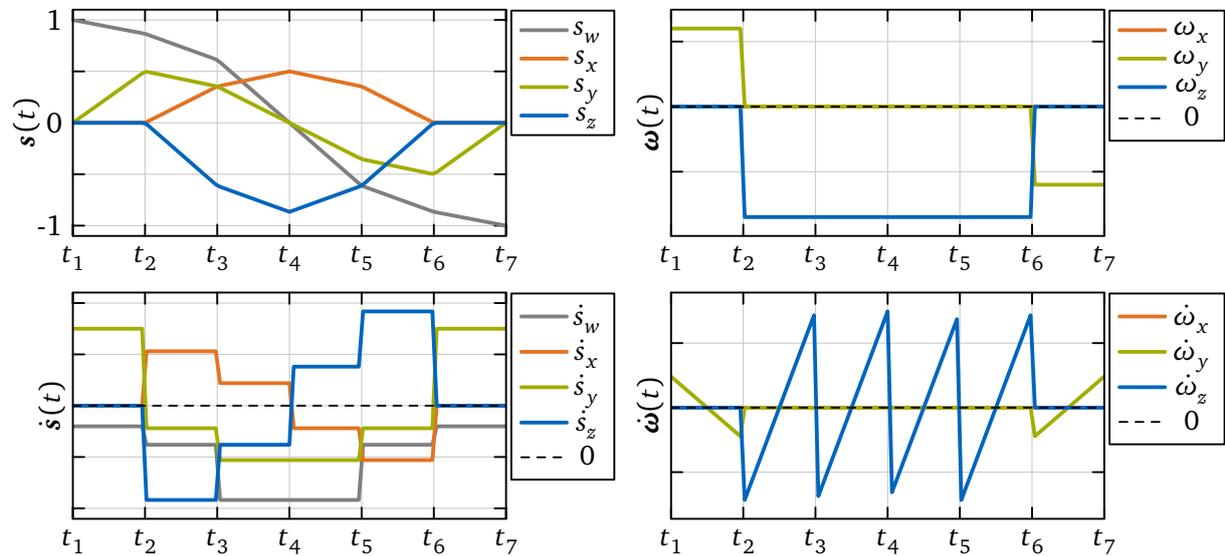


Figure B.1: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using a quaternion spline consisting of six consecutive LERP segments (uniform segmentation). Each LERP segment connects two keyframes. The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η). For computing ω and $\dot{\omega}$, we assume $s^{-1} \approx \bar{s}$ although $s(t)$ is *not* a unit quaternion (only at $t = t_i$).

B.3.2 Normalized Linear Interpolation (NLERP)

In order to obtain a unit quaternion, a straight forward extension of LERP is to normalize the result after interpolation. This method is known as *Normalized Linear Interpolation (NLERP)* and is given by

$$s(\eta) = \text{NLERP}(s_1, s_2, \eta) := \text{norm}(\text{LERP}(s_1, s_2, \eta)) \in \mathbb{H}^1 \quad \text{with } \eta \in [0, 1]. \quad (\text{B.34})$$

Compared to other interpolation methods NLERP is still very fast. However, the normalization makes it difficult to find an analytic formulation of the i -th derivative. For this reason, derivatives of NLERP are computed numerically in *Broccoli*. Same as for LERP, a quaternion spline chaining multiple interconnected NLERP segments is C^0 -continuous at most, see Figure B.2.

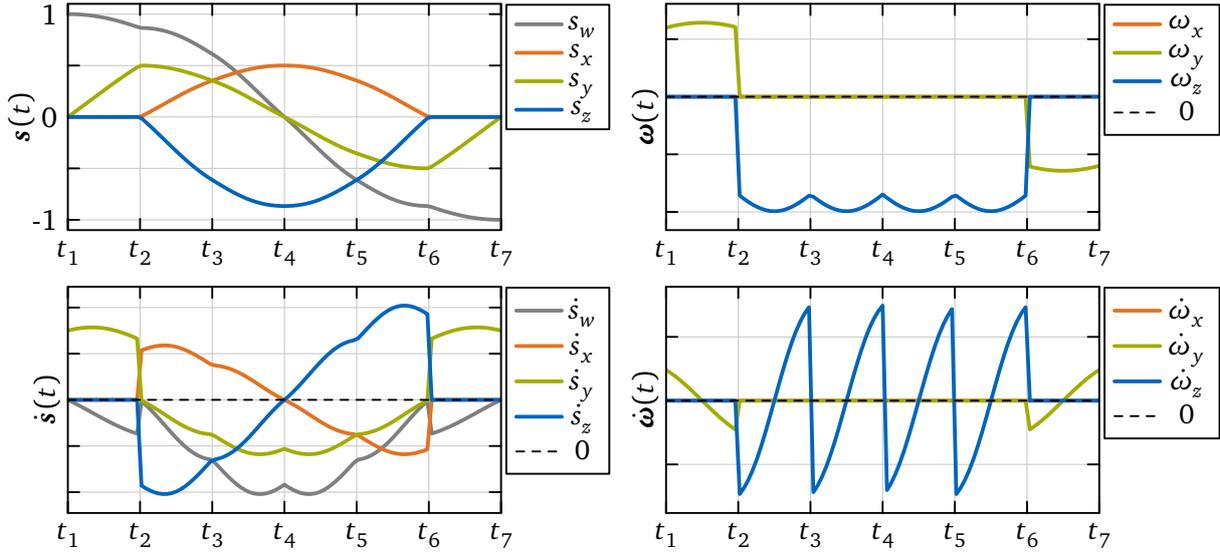


Figure B.2: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using a quaternion spline consisting of six consecutive NLERP segments (uniform segmentation). Each NLERP segment connects two keyframes. The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η). Note that the angular velocity ω is *not* constant between keyframes.

B.3.3 Spherical Linear Interpolation (SLERP)

A more elegant interpolation method is *Spherical Linear Interpolation (SLERP)* alias *great circle arc in-betweening* originally introduced by SHOEMAKE in [378]. It follows the same path on the unit sphere as LERP and NLERP, but simultaneously guarantees constant angular velocity and a normalized output quaternion [123]. For $s_1, s_2 \in \mathbb{H}^1$ and $\eta \in [0, 1]$ it is given by

$$\begin{aligned}
 s(\eta) = \text{SLERP}(s_1, s_2, \eta) &:= s_1 \otimes (s_1^{-1} \otimes s_2)^\eta \in \mathbb{H}^1, && \text{(original def. by SHOEMAKE [378])} \\
 &= (s_1 \otimes s_2^{-1})^{1-\eta} \otimes s_2 \in \mathbb{H}^1, && \text{(equiv. shown by DAM et al. [123, p. 42f])} \\
 &= (s_2 \otimes s_1^{-1})^\eta \otimes s_1 \in \mathbb{H}^1, && \text{(equiv. shown by DAM et al. [123, p. 42f])} \\
 &= s_2 \otimes (s_2^{-1} \otimes s_1)^{1-\eta} \in \mathbb{H}^1. && \text{(equiv. shown by DAM et al. [123, p. 42f])}
 \end{aligned} \tag{B.35}$$

Alternatively, it can be computed through [378]

$$s(\eta) = \text{SLERP}(s_1, s_2, \eta) := \begin{cases} \frac{\sin((1-\eta)\vartheta)}{\sin(\vartheta)} s_1 + \frac{\sin(\eta\vartheta)}{\sin(\vartheta)} s_2 & \text{if } 1 - |s_1 \odot s_2| \geq \varepsilon, \\ \text{NLERP}(s_1, s_2, \eta) & \text{else} \end{cases} \tag{B.36}$$

with

$$\cos(\vartheta) := s_1 \odot s_2 = \cos\left(\frac{\varphi}{2}\right) \text{ from Equation B.30.} \tag{B.37}$$

Typically, Equation B.35 is used for analysis while Equation B.36 represents an efficient implementation. Note that the second case in Equation B.36 represents a fallback to avoid division by (almost) zero which occurs if s_1 and s_2 describe (almost) the same orientation (small rotations with $\sin(\vartheta) \rightarrow 0$). A major advantage of SLERP is that it provides rather simple analytic derivatives [123, p. 45f]

$$\frac{\partial^i s(\eta)}{\partial \eta^i} = \text{SLERP}(s_1, s_2, \eta) \otimes \ln(s_1^{-1} \otimes s_2)^i \quad \text{for } i \geq 1, \tag{B.38}$$

where intermediate results can be efficiently recycled for evaluating the curve and its derivatives at the same time. As with LERP and NLERP, a quaternion spline chaining multiple interconnected SLERP segments is C^0 -continuous at most, see Figure B.3.

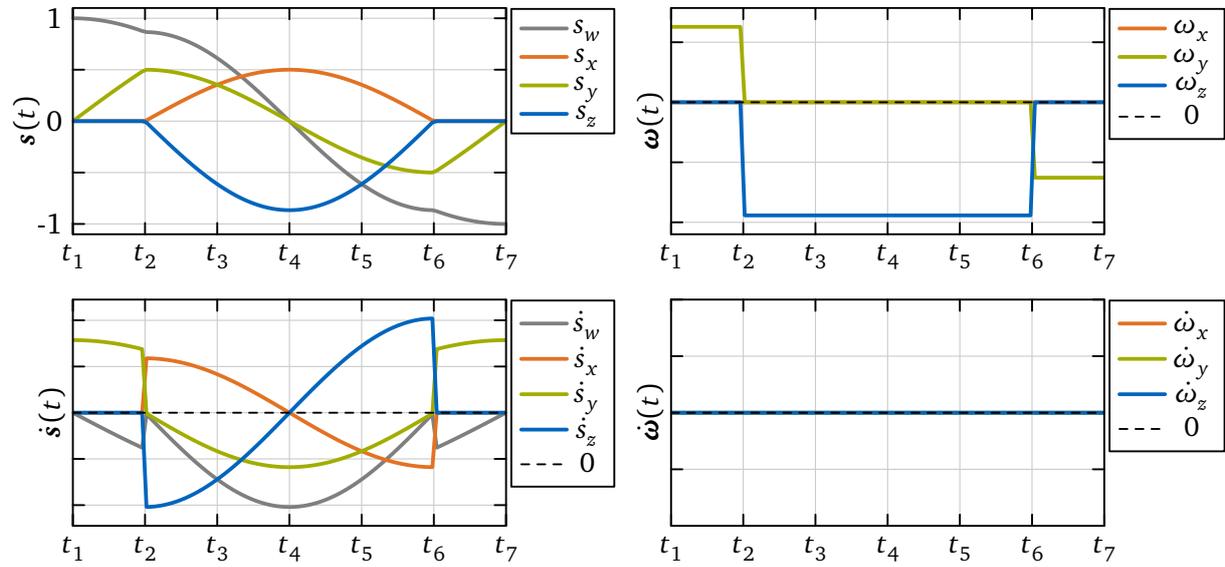


Figure B.3: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using a quaternion spline consisting of six consecutive SLERP segments (uniform segmentation). Each SLERP segment connects two keyframes. The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η). Note that the angular velocity ω is *constant* between keyframes but may jump at t_i .

Angular Velocity In order to obtain an analytic expression for the angular velocity ω , we use the third row of Equation B.35 together with the rule $\partial s_3^\eta / \partial \eta = s_3^\eta \otimes \ln(s_3)$ for the auxiliary (constant) unit quaternion $s_3 := s_2 \otimes s_1^{-1} \in \mathbb{H}^1$ [123, p. 23] to obtain

$$s(\eta) = s_3^\eta \otimes s_1 \quad \text{and} \quad \dot{s}(\eta) = \frac{\partial s(\eta)}{\partial \eta} \frac{d\eta}{dt} = \frac{1}{a} s_3^\eta \otimes \ln(s_3) \otimes s_1, \quad (\text{B.39})$$

where we assumed a linear mapping between the interpolation parameter η and the time t given by $t := a\eta + b$ such that $\dot{\eta} = 1/a$. By inserting $s(\eta)$ and $\dot{s}(\eta)$ into Equation B.22 (second form) and using the abbreviation $s_4(\eta) := s_3^\eta$, we obtain

$$\omega = -2 \text{vec}(s \otimes \dot{s}) = -\frac{2}{a} \text{vec}\left(s_4 \otimes \underbrace{s_1 \otimes \bar{s}_1}_{\mathbf{1}_{\mathbb{H}}} \otimes \underbrace{\overline{\ln(s_3)}}_{-\ln(s_3)} \otimes \bar{s}_4\right). \quad (\text{B.40})$$

By exploiting the relationships $s_1 \otimes \bar{s}_1 = \mathbf{1}_{\mathbb{H}}$ (since $s_1 \in \mathbb{H}^1$) and $\overline{\ln(s_3)} = -\ln(s_3)$ (since $s_3 \in \mathbb{H}^1$ such that $\text{real}(\ln(s_3)) = 0$, cf. Equation B.17), we obtain

$$\omega = \frac{2}{a} \text{vec}(s_4 \otimes \ln(s_3) \otimes \bar{s}_4). \quad (\text{B.41})$$

Since the angular velocity is constant [123], this relation has to hold for all $\eta \in [0, 1]$, thus, also for the special case $\eta = 0$ such that $s_4(\eta = 0) = s_3^0 = \mathbf{1}_{\mathbb{H}}$ and consequently

$$\omega = \frac{2}{a} \text{vec}(\ln(s_3)) = \frac{2}{a} \text{vec}(\ln(s_2 \otimes s_1^{-1})). \quad (\text{B.42})$$

Minimum Rotation It has to be highlighted that SLERP only produces a minimum rotation alias “shortest path”, if $\mathbf{s}_1 \odot \mathbf{s}_2 \geq 0$ holds, cf. Appendix B.2. Indeed, various implementations of SLERP (e.g. the one of *Eigen* [182]) use the dot product to check if the sign of the second quaternion has to be flipped and automatically return the minimum rotation. Unfortunately, flipping the sign depending on an if-else statement can lead to discontinuities in more advanced interpolation schemes which build on top of SLERP such as QBézier and SQUAD (see below). Thus, we use a custom implementation of SLERP strictly following Equation B.36, i.e., without branching to obtain the minimum rotation (see method `quaternionSLERP` in the module *geometry* of *Broccoli*).

B.3.4 Quaternion BÉZIER (QBézier) Curve

In order to achieve C^1 -continuity for a quaternion spline, SHOEMAKE introduced in [378] the so-called *Quaternion BÉZIER (QBézier)* curve as an equivalent to (cubic) BÉZIER curves known from \mathbb{R}^n . In contrast to the aforementioned interpolation methods, a QBézier curve is specified by four quaternions $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4 \in \mathbb{H}^1$ where \mathbf{s}_1 and \mathbf{s}_4 specify the orientation at the start and end of the curve and \mathbf{s}_2 and \mathbf{s}_3 represent interior control-quaternions which are not passed through in general. For evaluating a QBézier curve, the DE CASTELJAU algorithm [155, p. 43ff] is formulated for \mathbb{H}^1 (replacing linear interpolation with SLERP). In particular, we compute

$$\begin{aligned} \mathbf{s}_{A,1}(\eta) &:= \text{SLERP}(\mathbf{s}_1, \mathbf{s}_2, \eta), & \mathbf{s}_{A,2}(\eta) &:= \text{SLERP}(\mathbf{s}_2, \mathbf{s}_3, \eta), & \mathbf{s}_{A,3}(\eta) &:= \text{SLERP}(\mathbf{s}_3, \mathbf{s}_4, \eta), \\ \mathbf{s}_{B,1}(\eta) &:= \text{SLERP}(\mathbf{s}_{A,1}(\eta), \mathbf{s}_{A,2}(\eta), \eta), & \mathbf{s}_{B,2}(\eta) &:= \text{SLERP}(\mathbf{s}_{A,2}(\eta), \mathbf{s}_{A,3}(\eta), \eta), & & \\ \mathbf{s}(\eta) &= \text{QBézier}(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \eta) := \text{SLERP}(\mathbf{s}_{B,1}(\eta), \mathbf{s}_{B,2}(\eta), \eta) \in \mathbb{H}^1, \end{aligned} \quad (\text{B.43})$$

with $\eta \in [0, 1]$. The first, second, and third row of Equation B.43 represent the first, second, and third iteration of the DE CASTELJAU algorithm. Since a QBézier curve represents a chain of six SLERP operations, deriving an analytic formulation of the i -th derivative would be possible but very tedious. Thus, in *Broccoli*, the numeric derivative is used for QBézier curves instead.

C^1 -Continuity Without further action, a quaternion spline consisting of consecutive QBézier curves will be only C^0 -continuous (if \mathbf{s}_1 and \mathbf{s}_4 are set to the corresponding keyframe quaternions). However, it is possible to place the interior control-quaternions \mathbf{s}_2 and \mathbf{s}_3 in a way such that the quaternion spline becomes C^1 -continuous. This is achieved by enforcing equality of the first-order derivatives at the interconnection between two spline segments, see SHOEMAKE [378] (original formulation) and KIM et al. [250] (proper derivation) for details.

A C^1 -continuous QBézier spline interpolating n keyframe quaternions $\{\mathbf{s}_i\}$ with $i = 1, \dots, n$ can be generated as follows. For each of the $n - 1$ pairs of input keyframe quaternions $(\mathbf{s}_i, \mathbf{s}_{i+1})$, we define a corresponding QBézier curve given by $\text{QBézier}(\mathbf{s}_i, \mathbf{s}_{i,a}, \mathbf{s}_{i,b}, \mathbf{s}_{i+1}, \eta)$. Note that the first and last quaternion of the QBézier curve are already defined by the keyframe quaternions \mathbf{s}_i and \mathbf{s}_{i+1} , thus, it is only left to compute the interior control-quaternions $\mathbf{s}_{i,a}$ and $\mathbf{s}_{i,b}$. In a first run, we compute $\mathbf{s}_{i,a}$ for each of the $n - 1$ segments by

$$\mathbf{s}_{i,a} := \begin{cases} \mathbf{s}_1 & \text{if } i = 1, \quad (\text{enforce zero velocity at start}) \\ \text{norm}(2(\mathbf{s}_{i-1} \odot \mathbf{s}_i) \mathbf{s}_i - \mathbf{s}_{i-1} + \mathbf{s}_{i+1}) & \text{else.} \end{cases} \quad (\text{B.44})$$

In a second run, we compute $\mathbf{s}_{i,b}$ for each segment by

$$\mathbf{s}_{i,b} := \begin{cases} \exp\left(\frac{h_i}{h_{i+1}} \ln(\mathbf{s}_{i+1,a} \otimes \mathbf{s}_{i+1}^{-1})\right)^{-1} \otimes \mathbf{s}_{i+1} & \text{if } i < n - 1, \\ \mathbf{s}_n & \text{else.} \quad (\text{enforce zero velocity at end}) \end{cases} \quad (\text{B.45})$$

Here, h_i and h_{i+1} denote the proportions of the segment i and $i + 1$ within the spline (alias “duration”). For a uniform segmentation $h_i = h_{i+1}$, the first line in Equation B.45 simplifies to

$$s_{i,b} = (s_{i+1,a} \otimes s_{i+1}^{-1})^{-1} \otimes s_{i+1} = \text{SLERP}(s_{i+1}, s_{i+1,a}, -1) \quad \text{if } i < n - 1, \quad (\text{B.46})$$

which shows that the second control-quaternion $s_{i,b}$ is obtained by “mirroring” (extrapolation by SLERP with $\eta = -1$) the first control-quaternion of the following segment $s_{i+1,a}$ at the joint keyframe s_{i+1} of both segments (same “tangents”).

If the interior control-quaternions are chosen according to the Equations B.44 and B.45, then the resulting quaternion spline passes through all keyframe quaternions, is C^1 -continuous, and has zero angular velocity at the start and end of the spline, see Figure B.4.

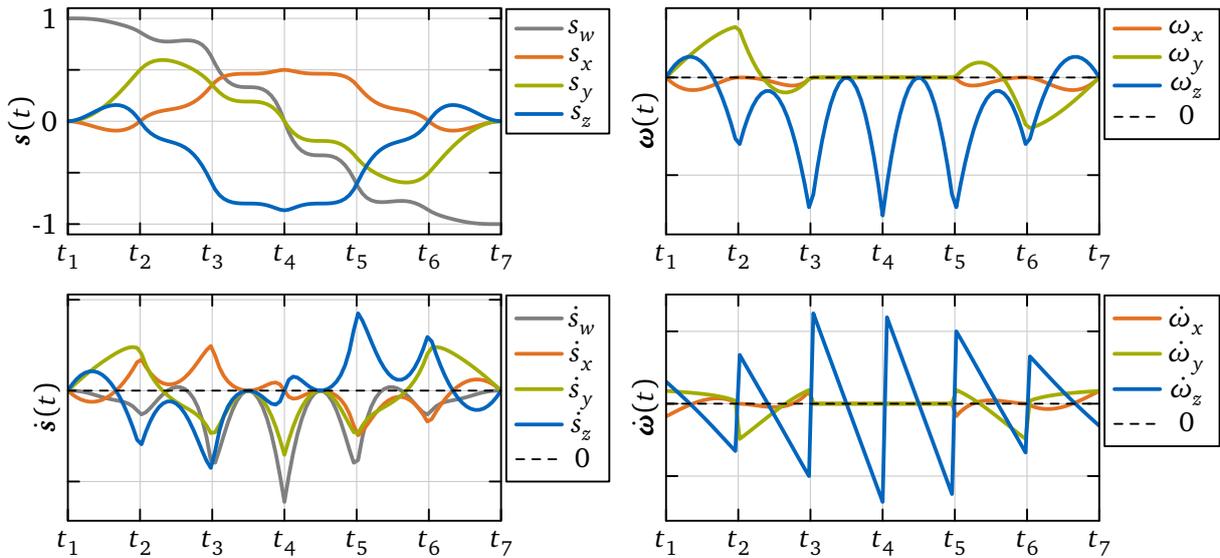


Figure B.4: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using a quaternion spline consisting of six consecutive QBézier segments (uniform segmentation). Each QBézier segment connects two keyframes where the interior control-quaternions are chosen to establish C^1 -continuity. The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η).

B.3.5 Spherical Quadrangle (SQUAD) Curve

Based on the *quadrangle* curve of BÖHM [88, p. 209f] (a cubic lying on a hyperbolic paraboloid), SHOEMAKE furthermore introduced the so-called *Spherical Quadrangle (SQUAD)* curve. Unfortunately, the original source is not available anymore, however, one can use the excellent works of DAM et al. [123] (revisited introduction and analysis of SQUAD) and KIM et al. [250] (focus on derivatives) as references instead. Same as for QBézier curves, a SQUAD curve is parameterized by four quaternions $s_1, s_2, s_3, s_4 \in \mathbb{H}^1$. Again, s_1 and s_4 specify the orientation at the start and end of the curve while s_2 and s_3 represent interior control-quaternions which are not passed through in general. The evaluation of a SQUAD curve involves three SLERP operations:

$$\begin{aligned} s_{A,1}(\eta) &:= \text{SLERP}(s_1, s_4, \eta), & s_{A,2}(\eta) &:= \text{SLERP}(s_2, s_3, \eta), \\ s(\eta) &= \text{SQUAD}(s_1, s_2, s_3, s_4, \eta) := \text{SLERP}(s_{A,1}(\eta), s_{A,2}(\eta), 2\eta(1-\eta)) \in \mathbb{H}^1, \end{aligned} \quad (\text{B.47})$$

with $\eta \in [0, 1]$. Same as for QBézier, deriving an analytic formulation of the i -th derivative of SQUAD is complex such that the numeric derivative is used in *Broccoli* instead.

\mathcal{C}^1 -**Continuity** Again, the interior control-quaternions can be used to achieve \mathcal{C}^1 -continuity for a quaternion spline consisting of SQUAD segments, see DAM et al. [123] for details. In particular, a \mathcal{C}^1 -continuous SQUAD spline interpolating n keyframe quaternions $\{s_i\}$ with $i = 1, \dots, n$ is generated by defining a SQUAD curve given by $\text{SQUAD}(s_i, s_{i,a}, s_{i,b}, s_{i+1}, \eta)$ for each of the $n - 1$ pairs of input keyframe quaternions (s_i, s_{i+1}) . The first and last quaternion of the SQUAD curve are defined by the keyframe quaternions s_i and s_{i+1} , such that we only need to compute the interior control-quaternions $s_{i,a}$ and $s_{i,b}$. For each of the $n - 1$ segments we compute

$$s_{i,a} := \begin{cases} s_1 & \text{if } i = 1, \\ s_i \otimes \exp\left(-\frac{h_{i-1} \ln(s_i^{-1} \otimes s_{i+1}) + h_i \ln(s_i^{-1} \otimes s_{i-1})}{2(h_{i-1} + h_i)}\right) & \text{else} \end{cases} \quad (\text{B.48})$$

and

$$s_{i,b} := \begin{cases} s_{i+1} \otimes \exp\left(-\frac{h_i \ln(s_{i+1}^{-1} \otimes s_{i+2}) + h_{i+1} \ln(s_{i+1}^{-1} \otimes s_i)}{2(h_i + h_{i+1})}\right) & \text{if } i < n - 1, \\ s_n & \text{else.} \end{cases} \quad (\text{B.49})$$

Here, h_{i-1} , h_i and h_{i+1} denote the proportions of the segment $i - 1$, i , and $i + 1$ within the spline (alias “duration”). The resulting quaternion spline passes through all keyframe quaternions and is \mathcal{C}^1 -continuous, however, it does *not* have zero angular velocity at the start and end of the spline, see Figure B.5. Compared to a QBézier spline, a SQUAD spline is much less “bumpy” with lower accelerations but also non-zero velocity at the start and end. Moreover, SQUAD is more efficient since it only involves three instead of six chained SLERP operations.

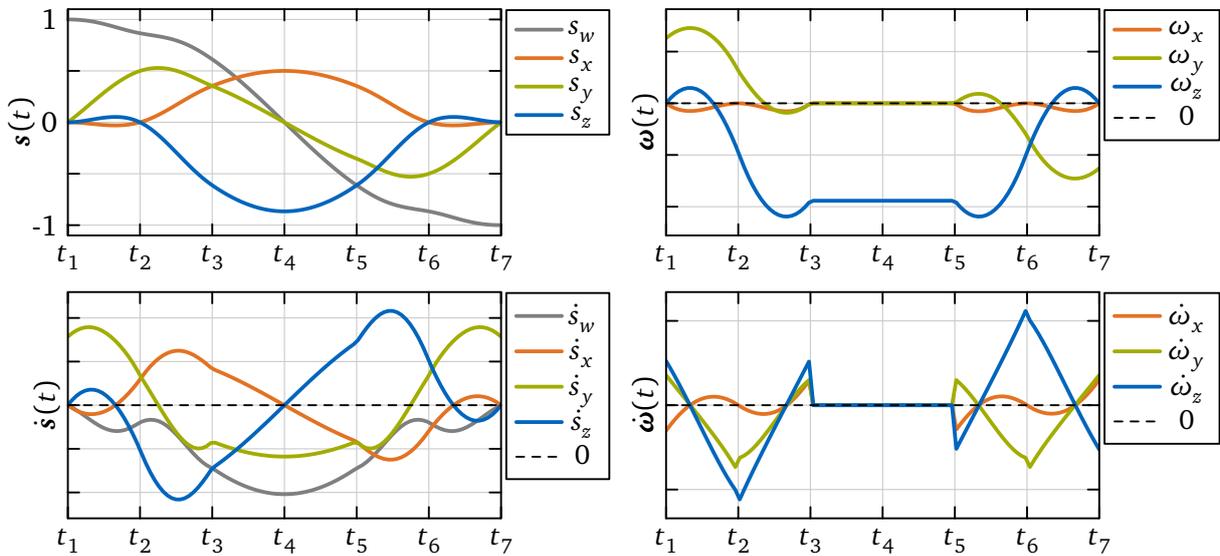


Figure B.5: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using a quaternion spline consisting of six consecutive SQUAD segments (uniform segmentation). Each SQUAD segment connects two keyframes where the interior control-quaternions are chosen to establish \mathcal{C}^1 -continuity. The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η).

B.3.6 Quaternion B-Spline (QBSpline) Curve

The interpolation methods presented so far generate quaternion splines which either provide simple analytic derivatives but are only \mathcal{C}^0 -continuous (SLERP), or are \mathcal{C}^1 -continuous but have

complex derivatives (QBézier and SQUAD). However, in robotics we are typically interested in a quaternion spline which is continuous up to a high order (e. g. C^2 for continuous joint torques) and provides simple analytic derivatives (e. g. to efficiently evaluate angular velocity and acceleration). An interpolation method which fulfills these requirements has been introduced by KIM et al. in [248]. In particular, they formulate a k -th order *Basis Spline (B-Spline)* [129, p. 87ff] curve for quaternions – in the following called *Quaternion B-Spline (QBSpline)* curve – and transfer important properties such as continuity and local control from Euclidean space to \mathbb{H}^1 . Most importantly, a k -th order QBSpline curve is C^{k-2} -continuous, (optionally) interpolates the very first and last keyframe quaternion, and provides analytic derivatives which can be evaluated with reasonable effort.

B-Spline Curve in Euclidean Space As a preparation to formulating the QBSpline curve in \mathbb{H}^1 , we briefly recall the fundamentals of a B-spline curve in Euclidean space \mathbb{R}^x (with $x \in \mathbb{N}_1$). For details, the book of DE BOOR [129] is recommended.

As a first step, one has to select the desired degree $p \geq 0$ respectively order $k = p + 1 > 0$ of the B-spline curve which is equivalent to the degree and order of a polynomial spline with monomial basis (affects smoothness). Furthermore, we assume a given set of $m \geq k$ control-points $\{\mathbf{r}_1, \dots, \mathbf{r}_m\}$ with $\mathbf{r}_i \in \mathbb{R}^x$. Finally, the user has to declare a sequence of $n = m + k$ non-decreasing knots $\boldsymbol{\tau} := \{\tau_1, \dots, \tau_n\}$ with $\tau_i \in \mathbb{R}$ and $\tau_i \leq \tau_{i+1}$, which specify the segmentation of the B-spline curve. The resulting k -th order B-spline curve $\mathbf{r}(\eta)$ is then given by

$$\mathbf{r}(\eta) := \sum_{i=1}^m B_{i,k,\boldsymbol{\tau}}(\eta) \mathbf{r}_i \quad \text{with} \quad \eta \in [\tau_k, \tau_{m+1}[, \quad (\text{B.50})$$

which represents a weighted sum over $B_{i,j,\boldsymbol{\tau}}(\eta) \in \mathbb{R}$ as the i -th normalized B-spline of order j for the knot sequence $\boldsymbol{\tau}$ defined by the recurrence relation

$$\begin{aligned} B_{i,j=1,\boldsymbol{\tau}}(\eta) &:= \begin{cases} 1 & \text{if } \tau_i \leq \eta < \tau_{i+1}, \\ 0 & \text{else} \end{cases}, \\ B_{i,j>1,\boldsymbol{\tau}}(\eta) &:= \left(\frac{\eta - \tau_i}{\tau_{i+j-1} - \tau_i} \right) B_{i,j-1,\boldsymbol{\tau}}(\eta) + \left(\frac{\tau_{i+j} - \eta}{\tau_{i+j} - \tau_{i+1}} \right) B_{i+1,j-1,\boldsymbol{\tau}}(\eta), \end{aligned} \quad (\text{B.51})$$

with $i = 1, \dots, m$ and $j = 1, \dots, k$. Since the control-points \mathbf{r}_i are constant, the d -th order derivative of the B-spline curve $\mathbf{r}(\eta)$ is fully specified by the d -th order derivative of $B_{i,j,\boldsymbol{\tau}}(\eta)$ given by [129, p. 115ff]

$$\begin{aligned} \frac{\partial^d B_{i,j \leq d,\boldsymbol{\tau}}(\eta)}{\partial \eta^d} &= 0, \\ \frac{\partial^d B_{i,j > d,\boldsymbol{\tau}}(\eta)}{\partial \eta^d} &= \left(\frac{j-1}{\tau_{i+j-1} - \tau_i} \right) \frac{\partial^{d-1} B_{i,j-1,\boldsymbol{\tau}}(\eta)}{\partial \eta^{d-1}} - \left(\frac{j-1}{\tau_{i+j} - \tau_{i+1}} \right) \frac{\partial^{d-1} B_{i+1,j-1,\boldsymbol{\tau}}(\eta)}{\partial \eta^{d-1}}. \end{aligned} \quad (\text{B.52})$$

Knot Sequence The knot sequence $\boldsymbol{\tau} := \{\tau_1, \dots, \tau_n\}$ can be chosen arbitrarily as long as $\tau_i \leq \tau_{i+1}$ holds (continuity of the B-spline curve may degrade for $\tau_i = \tau_{i+1}$). Within the context of this thesis, we focus on a special type of knot sequences:

- *clamped*: the first and last k knots are “clamped”, i. e., $\tau_1 = \dots = \tau_k$ and $\tau_{n-k+1} = \dots = \tau_n$, such that the B-spline curve $\mathbf{r}(\eta)$ interpolates the first and last control-point \mathbf{r}_1 and \mathbf{r}_m ,
- *uniform*: the remaining “interior” knots $\{\tau_{k+1}, \dots, \tau_{n-k}\}$ are distributed equally within the interval $] \tau_k, \tau_{n-k+1}[$ to obtain a uniform segmentation,
- *normalized*: the knot sequence is scaled to the range $[0, 1]$, i. e., $\tau_1 = 0$ and $\tau_n = 1$, which yields best numerical stability and simplifies the range of the interpolation parameter in Equation B.50 to $\eta \in [0, 1[$ (similar to other interpolation methods presented so far).

By using a clamped, uniform, and normalized knot sequence, the user only has to specify the desired order k and m control-points $\{\mathbf{r}_i\}$. Note that the count of control-points m has to fulfill $m \geq k$. For this reason, one might choose to replicate the first and last control-point.

Cumulative B-Spline Basis For the purpose of formulating the QBSpline curve, KIM et al. first introduced the so-called *cumulative* B-spline given by [248]

$$\tilde{B}_{i,k,\tau}(\eta) := \sum_{c=i}^m B_{c,k,\tau}(\eta) \in \mathbb{R} \quad \text{with } B_{c,k,\tau}(\eta) \text{ from Equation B.51 and } i = 1, \dots, m. \quad (\text{B.53})$$

Written as a vector, the cumulative B-spline basis has the form

$$[\tilde{B}_{1,k,\tau}(\eta), \dots, \tilde{B}_{m,k,\tau}(\eta)]^T = \left[\underbrace{1, \dots, 1}_{f_1\text{-elem.}}, \underbrace{*, \dots, *}_{f_2\text{-elem.}}, \underbrace{0, \dots, 0}_{f_3\text{-elem.}} \right]^T \in \mathbb{R}^m \quad (\text{B.54})$$

where $*$ represent placeholders for real-valued scalars within the interval $[0, 1]$ and $f_1, f_2,$ and f_3 are the element counts of the corresponding blocks of 1's, $*$'s, and 0's respectively. The d -th derivative of the cumulative B-spline is simply given by

$$\frac{\partial^d \tilde{B}_{i,k,\tau}(\eta)}{\partial \eta^d} = \sum_{c=i}^m \frac{\partial^d B_{c,k,\tau}(\eta)}{\partial \eta^d} \quad \text{with } \frac{\partial^d B_{c,k,\tau}(\eta)}{\partial \eta^d} \text{ from Equation B.52 and } i = 1, \dots, m \quad (\text{B.55})$$

which has the form

$$\left[\frac{\partial^d \tilde{B}_{1,k,\tau}(\eta)}{\partial \eta^d}, \dots, \frac{\partial^d \tilde{B}_{m,k,\tau}(\eta)}{\partial \eta^d} \right]^T = \left[\underbrace{0, \dots, 0}_{f_1\text{-elem.}}, \underbrace{*, \dots, *}_{f_2\text{-elem.}}, \underbrace{0, \dots, 0}_{f_3\text{-elem.}} \right]^T \in \mathbb{R}^m \quad (\text{B.56})$$

with $*$ representing placeholders for unbounded real-valued scalars.

Quaternion B-Spline (QBSpline) Curve Given a set of $m \geq k$ control-quaternions $\{\mathbf{s}_1, \dots, \mathbf{s}_m\}$ with $\mathbf{s}_i \in \mathbb{H}^1$ and a sequence of $n = m + k$ non-decreasing knots $\tau := \{\tau_1, \dots, \tau_n\}$ with $\tau_i \in \mathbb{R}$ and $\tau_i \leq \tau_{i+1}$, the k -th order QBSpline curve $\mathbf{s}(\eta)$ is given by [248]

$$\mathbf{s}(\eta) = \text{QBSpline}(k, \{\mathbf{s}_i\}, \tau, \eta) := \mathbf{s}_1^{\tilde{B}_{1,k,\tau}(\eta)} \otimes \prod_{i=2}^m \underbrace{\exp(\ln(\mathbf{s}_{i-1}^{-1} \otimes \mathbf{s}_i) \tilde{B}_{i,k,\tau}(\eta))}_{(\mathbf{s}_{i-1}^{-1} \otimes \mathbf{s}_i)^{\tilde{B}_{i,k,\tau}(\eta)}} \in \mathbb{H}^1 \quad (\text{B.57})$$

with $\eta \in [\tau_k, \tau_{m+1}[$. Note that quaternions are chained in \prod through the HAMILTON product. If we define the virtual control-quaternion $\mathbf{s}_0 := \mathbf{1}_{\mathbb{H}}$ (\mathbf{s}_i for $i = 0$) and further introduce the *constant* auxiliary quaternion $\Omega_i := \ln(\mathbf{s}_{i-1}^{-1} \otimes \mathbf{s}_i) \in \mathbb{H}$, the definition of the QBSpline curve can be rewritten in the shorter form

$$\mathbf{s}(\eta) = \prod_{i=1}^m \exp(\Omega_i \tilde{B}_{i,k,\tau}(\eta)) \quad \text{with } \eta \in [\tau_k, \tau_{m+1}[. \quad (\text{B.58})$$

Comparing Equation B.58 with Equation B.50 shows the similarity between a B-spline curve in Euclidean space and \mathbb{H}^1 : weighted addition in Euclidean space becomes weighted multiplication in \mathbb{H}^1 . Indeed, a QBSpline curve can be seen as a blended sequence of SLERP operations.

Evaluating the entire product in Equation B.58 would be expensive for large counts of control-quaternions (alias “long” curves). Fortunately, similar to its Euclidean equivalent, a QBSpline curve has local control, i. e., the “value” of the curve depends only on a finite set of close-by control-quaternions (specified by the order k of the curve). This becomes clear from the

special structure of the cumulative B-spline basis (cf. Equation B.54), which can be exploited to formulate the QBSpline curve in a much more efficient way:

$$\mathbf{s}(\eta) = \mathbf{s}_{f_1} \otimes \prod_{i=f_1+1}^{f_1+f_2} \exp(\Omega_i \tilde{B}_{i,k,\tau}(\eta)) = \mathbf{s}_{f_1} \otimes \prod_{i=1}^{f_2} \Gamma_{i+f_1}(\eta) \quad \text{with } \Gamma_i(\eta) := \exp(\Omega_i \tilde{B}_{i,k,\tau}(\eta)). \quad (\text{B.59})$$

In this new form, the product involves only f_2 factors where f_2 is limited by the order k of the curve. Thus, the computational cost of evaluating a QBSpline curve is (almost) independent of the count of control-quaternions. Note that the integers f_1 and f_2 depend on η and can be efficiently determined in parallel to evaluating the cumulative B-spline basis.⁹⁴

We can find the d -th order derivative of a QBSpline curve by applying the general LEIBNIZ rule (generalized form for more than two factors)

$$\frac{\partial^d \mathbf{s}(\eta)}{\partial \eta^d} = \mathbf{s}_{f_1} \otimes \sum_{\mathcal{T}_L} \left[\binom{d}{L_1, L_2, \dots, L_{f_2}} \cdot \prod_{i=1}^{f_2} \frac{\partial^{L_i} \Gamma_{i+f_1}(\eta)}{\partial \eta^{L_i}} \right] \quad (\text{B.60})$$

where the multinomial coefficient is given by

$$\binom{d}{L_1, L_2, \dots, L_{f_2}} = \frac{d!}{L_1! L_2! \dots L_{f_2}!} \quad (\text{B.61})$$

and \mathcal{T}_L is the set of generalized LEIBNIZ tuples, i. e., the set of all f_2 -tuples (L_1, \dots, L_{f_2}) of non-negative integers which satisfy $L_1 + \dots + L_{f_2} = d$. The d -th order derivative of the auxiliary term $\Gamma_i(\eta)$ is obtained by applying the formula of FAÀ DI BRUNO

$$\frac{\partial^d \Gamma_i(\eta)}{\partial \eta^d} = \sum_{\mathcal{T}_F} \left[\left(\frac{d!}{F_1! F_2! \dots F_d!} \right) \cdot \left(\Gamma_i(\eta) \otimes \prod_{j=1}^{F_1+\dots+F_d} \Omega_i \right) \cdot \prod_{j=1}^d \left(\frac{1}{j!} \frac{\partial^j \tilde{B}_{i,k,\tau}(\eta)}{\partial \eta^j} \right)^{F_j} \right] \quad (\text{B.62})$$

where \mathcal{T}_F is the set of FAÀ DI BRUNO tuples, i. e., the set of all d -tuples (F_1, \dots, F_d) of non-negative integers which satisfy $F_1 + 2F_2 + \dots + dF_d = d$. Note that the derivatives of the cumulative B-spline $\tilde{B}_{i,k,\tau}$ have already been given in Equation B.55.

Indeed, computing the general LEIBNIZ tuples \mathcal{T}_L and FAÀ DI BRUNO tuples \mathcal{T}_F for every evaluation of the derivative of $\mathbf{s}(\eta)$ would be very expensive. However, one can simply implement a lookup table for these tuples (up to a certain maximum order of derivative d) similar to a typical implementation of the factorial.⁹⁵ Consequently, although Equation B.60 looks rather complex, the d -th order derivative of a QBSpline curve can be implemented efficiently. Note that the implementation of the QBSpline curve in *Broccoli* is quite efficient while still covering the most general case. In contrast, a specialization for a certain type (e. g. a special order k) would allow further optimizations.

Spline and Interpolation In contrast to the other interpolation methods mentioned so far, the QBSpline curve allows a custom count of control-quaternions and thus, already describes a spline by itself. If we choose a clamped, uniform, and normalized knot sequence τ , then the generated k -th order QBSpline curve will be C^{k-2} -continuous and interpolates the very first and last keyframe quaternion, see Figure B.6. Similar to a B-spline in Euclidean space, it will get close to the interior keyframe quaternions, but it will not interpolate them (in general). From B-splines in Euclidean space it is known [129], that one can compute a set of control-points such

⁹⁴See the method `CumulativeBSplineBasisSample::evaluate` from the module curve of *Broccoli* for details.

⁹⁵See the methods `math::faaDiBrunoTuples`, `math::generalLeibnizTuples`, and `math::factorial` from the module `core` of *Broccoli*.

that the curve interpolates a given set of keyframes. This typically involves the solution of a LSE. In [249], KIM et al. propose an equivalent method to compute the control-quaternions of a 4-th order QBSpline curve for interpolating a given set of keyframe quaternions. Unfortunately, the linear relations from Euclidean space become non-linear in \mathbb{H}^1 such that an iterative scheme is required. Furthermore, the relative rotation between two consecutive keyframe quaternions has an upper bound in order to guarantee convergence. Since the interpolation algorithm presented in [249] is not used within the WPG module of LOLA, it has not been implemented in *Broccoli* (yet). However, it should be considered for future investigations.

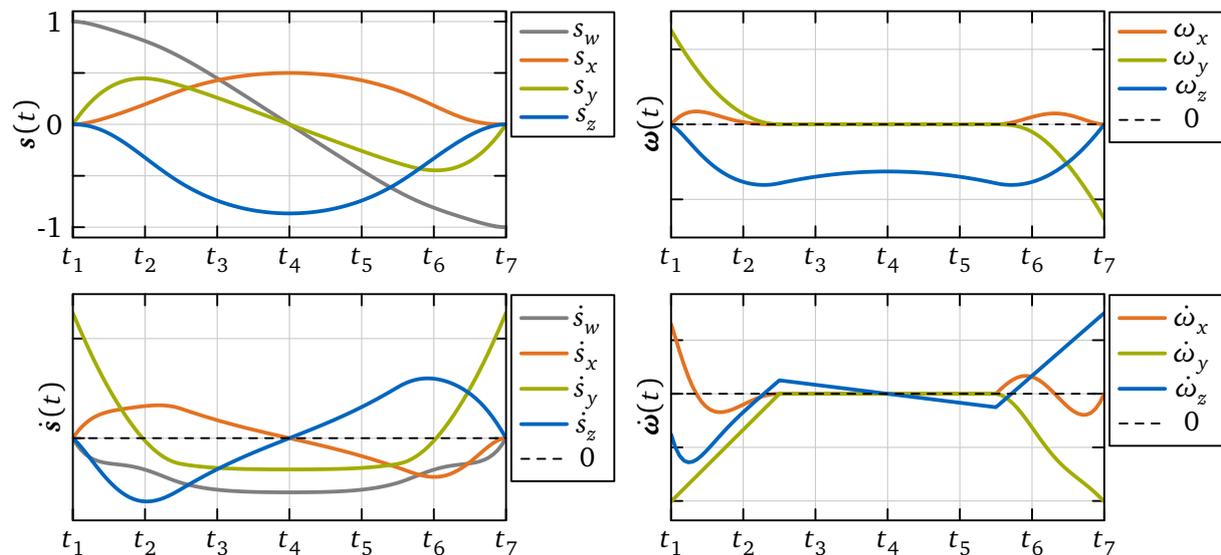


Figure B.6: Interpolation of keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 ($m = 7$) using a quaternion spline consisting of a single 4-th order ($k = 4$) QBSpline curve with a clamped, uniform, and normalized knot sequence $\tau = \{0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1, 1\}$ ($n = m + k = 11$). The plots show the components of s , \dot{s} , ω , and $\dot{\omega}$ over the spline interpolation parameter t alias “time” (linear mapping between t and η). The curve is C^2 -continuous and interpolates the very first and last keyframe quaternion s_1 and s_7 but *not* the interior keyframe quaternions s_2 to s_6 .

B.3.7 Comparison

A comparison of the CoSy motions resulting from the aforementioned interpolation methods is shown in Figure B.7. In addition, Table B.1 summarizes the main properties of the presented methods. For comparing the computational cost of evaluating a corresponding quaternion spline, the runtime time of evaluating a single sample of the given example trajectory (shown in Figure B.7) is given. The runtimes are averaged over 1,000 full trajectory evaluations (each with $\Delta t = 0.001$). Apart from the computation of the current orientation $s(t)$, also evaluating the first- and second-order derivatives $\dot{s}(t)$ and $\ddot{s}(t)$ (triggered through the same function call to allow reuse of intermediate results) are measured. The runtime analysis was performed on a single core of an *AMD Ryzen 7 1700X@3.4 GHz* CPU running *Ubuntu 18.04 64bit*.

Indeed, there are numerous methods for interpolating spatial rotation with quaternions which have not been mentioned here. The presented methods introduce only a small but popular subset of algorithms. Within the WPG module of LOLA, mainly the SLERP and QBSpline (of order $k = 4$) curves are used.

Table B.1: Overview of the main properties of the presented quaternion interpolation methods. The given timing information describes the average execution time of evaluating a single sample ($s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$) of a corresponding quaternion spline using the *Broccoli* implementation. Note that the restriction to a special type of QBSpline (e. g. fixed order k and count of control-quaternions m) would allow further source-code optimizations.

Property	LERP	NLERP	SLERP	QBézier	SQUAD	QBSpline
complexity (implementation)	○	○	●	●●	●●	●●●
continuity of corresp. spline	C^0	C^0	C^0	$\leq C^1$	$\leq C^1$	$\leq C^{k-2}$
simple analytic derivatives	yes	no	yes	no	no	yes
runtime / μs ($s(t)$)	0.11	0.13	0.21	0.70	0.39	2.63
runtime / μs ($s(t), \dot{s}(t)$)	0.17	0.45	0.40	2.31	1.27	6.63
runtime / μs ($s(t), \dot{s}(t), \ddot{s}(t)$)	0.23	0.91	0.45	4.38	2.55	12.10

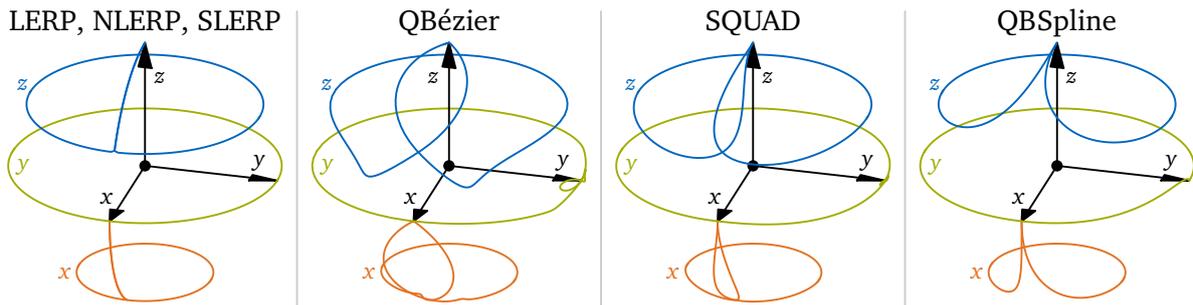


Figure B.7: Resulting CoSy motion for interpolating the keyframe sequence $\{(s_i, t_i)\}$ from Equation B.31 using the presented quaternion interpolation methods. From left to right: LERP, NLERP, and SLERP (same path but different traversal speed – C^0 -continuous); QBézier and SQUAD (C^1 -continuous); 4-th order QBSpline with clamped, uniform, and normalized knot sequence (C^2 -continuous). The colored lines visualize the path of the tip of the x-axis (orange), the y-axis (green), and the z-axis (blue).

B.3.8 Advanced Speed Control

Additional to continuity within a quaternion spline, one is typically also interested in a zero velocity and acceleration at the start and end. This ensures a smooth acceleration and deceleration at the beginning and end of a motion (e. g. a footstep). For this purpose, the *Broccoli* implementation introduces the concept of a *quaternion trajectory* which combines a quaternion spline $s(\eta) \in \mathbb{H}^1$ (“shape”) with a so-called *parameter spline* $\eta(t) \in \mathbb{R}$ (“timing”). The evaluation of a quaternion trajectory simply represents the chained operation $s(\eta(t))$. For obtaining the d -th order time derivative, we apply the formula of FAÀ DI BRUNO (generalized chain rule)

$$\frac{d^d s(\eta(t))}{dt^d} = \sum_{\mathcal{T}_F} \left[\left(\frac{d!}{F_1! F_2! \dots F_d!} \right) \cdot \left(\frac{\partial^x s(\eta)}{\partial \eta^x} \right) \cdot \prod_{j=1}^d \left(\frac{1}{j!} \frac{d^j \eta(t)}{dt^j} \right)^{F_j} \right] \text{ with } x := F_1 + \dots + F_d \quad (\text{B.63})$$

where \mathcal{T}_F is the set of FAÀ DI BRUNO tuples, i. e., the set of all d -tuples (F_1, \dots, F_d) of non-negative integers which satisfy $F_1 + 2F_2 + \dots + dF_d = d$ (again obtained from lookup table).

Through an appropriate choice of $\eta(t)$, it is possible to control the traversal speed of the quaternion spline such that the velocity and acceleration at the start and end zero-out (e. g. by using a quintic polynomial with zero velocity and acceleration at its boundaries). Note that it is also possible to generate a C^2 -continuous SLERP-, QBézier-, or SQUAD-based trajectory if $\eta(t)$ is formulated as a chain of piecewise quintic polynomials with zero velocity and acceleration at the interconnection of segments (requires a full stop at each keyframe).

Appendix C

Swept Sphere Volumes (SSVs)

Computing the shortest connection between two 3D objects is a common mathematical problem. In the field of robotics, such distance evaluations are in particular relevant for avoiding collisions. Note that one has to differentiate between testing for intersection and evaluating the minimum distance. While the former is typically used for collision *detection* (e. g. within a physics simulation), the latter is suitable for collision *avoidance* (e. g. by formulating a corresponding repelling cost gradient). Depending on the geometric representation of the object's shape, the calculations can be fast but coarse (e. g. using bounding boxes) or accurate but slow (e. g. using full CAD models). For getting “sufficient” accuracy within a reasonable amount of time, one may approximate the object's shape by geometric primitives for which numerous solutions have been proposed. Such shape models reach from simple boxes, spheres, and cylinders [328] to complex compounds such as *Sphere-Torus-Patches Bounding Volumes* [78, 148]. A typical common property of these shape models is convexity.

The locomotion system of LOLA makes use of *Swept Sphere Volumes (SSVs)* which are constructed by sweeping the center of a sphere over a certain base geometry, see Figure C.1. Originally, LARSEN et al. introduced SSVs for a rectangular base [267]. As one of the first adopters, SUGIURA et al. used point- and line-SSVs in [398] to realize self-collision avoidance on a humanoid robot. This motivated SCHWIENBACHER et al. to introduce SSVs also for LOLA [371]. Indeed, SCHWIENBACHER is the main author of the original SSV implementation of LOLA which he described in detail in [372, p. 89ff]. Recently, this implementation has been replaced by a modern SSV library developed by POSCHER, which is documented in his bachelor's thesis [30]. Same as with the original implementation by SCHWIENBACHER, the new library provides efficient algorithms for computing the distance between SSVs with a point, line, or triangle as base geometry. Notable improvements are the correct handling of (rare) special cases and using *Eigen* as backend for linear algebra. Moreover, it could be shown that the new library is slightly faster than the previous implementation [30] – at least on modern hardware.

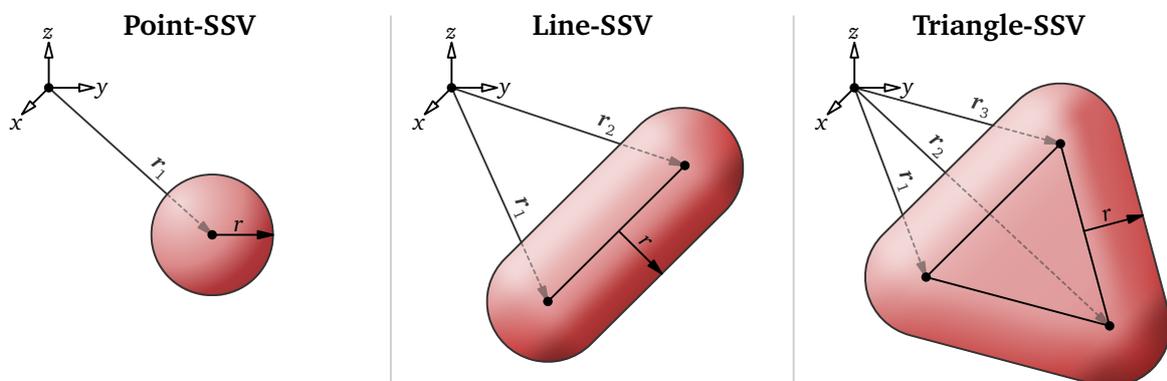


Figure C.1: SSV Elements constructed by sweeping the center of a sphere over the corresponding base geometry. From left to right: point-SSV, line-SSV, and triangle-SSV element. An SSV element is fully defined by its vertices $\{r_i\}$ (described in local CoSy of SSV segment) and its radius $r > 0$. Illustration based on [372, p. 93f].

Evaluating the minimum distance between two *SSV Elements* (point / line / triangle) boils down to computing the shortest connection between the corresponding base geometries. While this is trivial in the case of two point-SSVs, many special cases have to be considered for more complex pairings such as two triangle-SSVs. Implementation details for all possible element combinations are given in [372, p. 94ff], however, [30, p. 20ff] should be preferred since it contains some fixes and extensions. Finally, the distance between two SSV elements is simply given by the length of the vector connecting the closest points on the respective base geometries subtracted by the corresponding element radii. As a consequence, the minimum distance may become negative which indicates penetration. For the purpose of acceleration, various auxiliary variables can be pre-computed for each SSV element (e. g. edges, normals, etc.).

In order to describe objects with complex shape, SSV elements can be grouped to a rigid compound forming a so-called *SSV Segment*, see Figure C.2 left. For computing the minimum distance between two SSV segments, each element of the first segment has to be tested against all elements of the second segment. In many cases, one is only interested in the exact distance if it is below a certain threshold (e. g. for skipping obstacles which are far apart). For this reason, each SSV segment additionally maintains its own axis-aligned bounding box. The box allows a rapid computation of a lower and upper bound for the distance between two segments. In combination with the aforementioned threshold, this can be used to avoid unnecessary SSV evaluations. On an even higher level, SSV segments are organized within an *SSV Scene*, see Figure C.2 right. The scene allows to specify an explicit list of segment pairs to evaluate, which drastically reduces overall runtime in case only certain combinations are of interest.

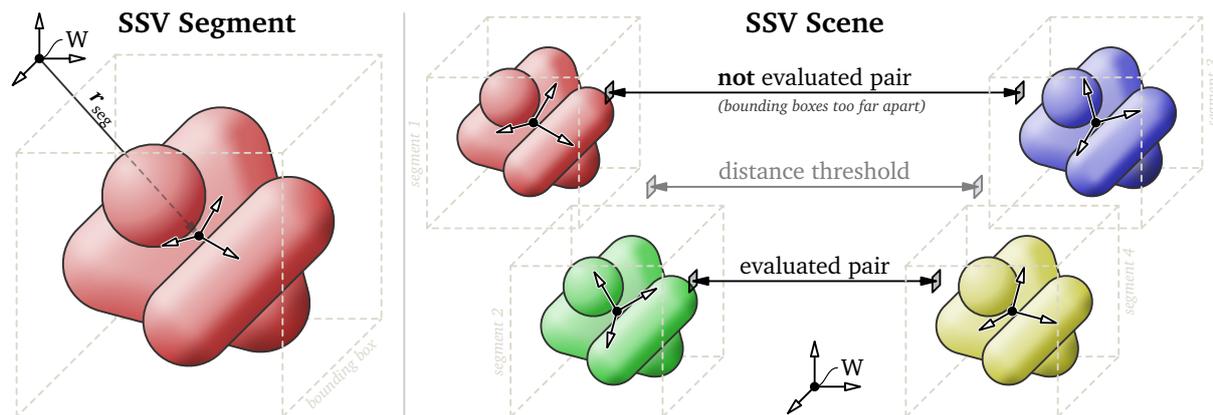


Figure C.2: Construction of an *SSV Segment* as rigid compound of SSV elements (left) and organization of SSV segments within an *SSV Scene* (right). For each segment, the proposed library automatically maintains an axis-aligned bounding box which is used for acceleration in case a distance threshold is specified. Instead of testing a segment against all other segments within the scene, the user can specify a list of segment pairs to evaluate.

The source code of POSCHER has been refactored and integrated into the open-source library *Broccoli* [15] (see the module *geometry*) by the author of this dissertation. Moreover, the author has made several extensions to the work of POSCHER, most notably

- acceleration by axis-aligned bounding boxes as explained above and
- (optional) parallel evaluation of SSV scenes by a dedicated thread pool with automatic load balancing (estimates evaluation costs based on the SSV segment complexity).

Note that the source code published in *Broccoli* is fully documented, extensively tested through numerous unit tests checking all possible element pairings and configurations, and comes with an example application to encourage other roboticists in using the library. The current implementation is designed for scenes of moderate complexity (element count). A useful future extension would be to (optionally) organize SSV segments in a tree structure (such as a *Bounding Volume Hierarchy (BVH)* as proposed by LARSEN et al. [267]) to handle also very large scenes.

Appendix D

Step Parameters

The *Step Parameters* represent the first six components of a QPWT (cf. Table 5.2) and describe the relative transform between the *previous* stance foot TCP frame (${}^W\mathbf{r}_{SF,pre}$, ${}^W\mathbf{A}_{SF,pre}$) linked to the *Begin Pose* of a *Transition* and the *next* stance foot TCP frame (${}^W\mathbf{r}_{SF,next}$, ${}^W\mathbf{A}_{SF,next}$) linked to the *End Pose* of a *Transition*. The specification used within this thesis builds on top of the step parameter formulation for a “standard circular path” by BUSCHMANN [100, p. 59ff].⁹⁶ To allow a specification of the full 6D pose of the next stance foot, we extend the step parameters of BUSCHMANN by a step length in vertical direction (along ${}^W\mathbf{e}_z$) and a horizontal foot rotation (around local x - and y -axis). In particular, foot orientations are described using CARDAN angles alias yaw-pitch-roll or z - y - x EULER angles – a special form of TAIT-BRYAN angles:

$${}^W\mathbf{A}_{SF,pre} := \mathbf{A}_z(\alpha_{zp})\mathbf{A}_y(\alpha_{yp})\mathbf{A}_x(\alpha_{xp}), \quad {}^W\mathbf{A}_{SF,next} := \mathbf{A}_z(\alpha_{zn})\mathbf{A}_y(\alpha_{yn})\mathbf{A}_x(\alpha_{xn}) \quad (\text{D.1})$$

where $\mathbf{A}_{x|y|z}$ denote the elementary rotation matrices around the x -, y -, and z -axis as defined in Equation A.2 and $\alpha_{x|y|z,p|n}$ are the corresponding EULER angles (using $p := pre$ and $n := next$ for brevity). As a preparation, we further define the auxiliary CoSys

W_{zp} ...world frame rotated around ${}^W\mathbf{e}_z$ to align with *previous* stance foot: ${}^W\mathbf{A}_{W_{zp}} := \mathbf{A}_z(\alpha_{zp})$

W_{zn} ...world frame rotated around ${}^W\mathbf{e}_z$ to align with *next* stance foot: ${}^W\mathbf{A}_{W_{zn}} := \mathbf{A}_z(\alpha_{zn})$

Finally, the step parameters $l_{x|y|z}$ and $\varphi_{x|y|z}$ are defined as (cf. Figure 4.6 in [100, p. 60])

l_x ...step length in sagittal plane (equivalent to L_x in [100, p. 60])

represents (signed) “arc length” in W , W_{zp} , and W_{zn} FoR (relative to previous stance)

l_y ...step length in lateral plane (equivalent to L_y in [100, p. 60])

represents lateral shift in W_{zn} FoR (relative to previous stance)

l_z ...step length in vertical direction (extension of [100, p. 60])

represents vertical shift in W , W_{zp} , and W_{zn} FoR (relative to previous stance)

φ_x ...step angle around (positive) “local” x -axis (extension of [100, p. 60])

$\varphi_x := \alpha_{xn}$ (independent of previous stance)

φ_y ...step angle around (positive) “local” y -axis (extension of [100, p. 60])

$\varphi_y := \alpha_{yn}$ (independent of previous stance)

φ_z ...step angle around (positive) vertical axis (equivalent to φ_{step} in [100, p. 60])

$\varphi_z := \alpha_{zn} - \alpha_{zp}$ (relative to previous stance)

To summarize, the next stance foot’s position ${}^W\mathbf{r}_{SF,next}$ and orientation ${}^W\mathbf{A}_{SF,next}$ depend on the position of the previous stance foot ${}^W\mathbf{r}_{SF,pre}$, the rotation of the previous stance foot around

⁹⁶Note that we only consider the “basic” calculation from Equations 4.1 and 4.2 [100, p. 61] (rotation around foot TCP) but *not* the modification from Equation 4.3 [100, p. 61] (rotation around foot centroid). Although the latter would lead to a “more natural looking rotation about the center of the foot” [100, p. 61], it would make step parameter specification by the user (e. g. from input files) less intuitive and more error-prone. This holds true especially for partial contact situations, where the centroid of the contact area is different from the one during full support. Thus, we stick to the TCP as reference point for each foot. Note that this choice only has an effect on *teleoperated walking* and *fixed sequence walking (input file)* (cf. Section 5.2) since for all other actions, the WPG computes the 6D foot poses directly and may use the step parameters only as exchange format (without loss of information).

the vertical axis α_{zp} , and the step parameters $l_{x|y|z}$ and $\varphi_{x|y|z}$ but *not* on the rotation of the previous stance foot around the horizontal axes α_{xp} and α_{yp} . Making the next stance foot pose independent of α_{xp} and α_{yp} simplifies the specification of contact sequences for uneven and inclined terrain (e. g. for stepping on ramps).

Next Stance from Step Parameters This paragraph describes how the next stance foot pose ${}^W\mathbf{r}_{\text{SF,next}}$ and ${}^W\mathbf{A}_{\text{SF,next}}$ is computed from ${}^W\mathbf{r}_{\text{SF,pre}}$, α_{zp} , $l_{x|y|z}$, and $\varphi_{x|y|z}$. The position of the next stance foot TCP frame described in the world FoR is obtained from the vector chain

$${}^W\mathbf{r}_{\text{SF,next}} = {}^W\mathbf{r}_{\text{SF,pre}} + {}^W\mathbf{A}_{W_{zp}} W_{zp} \mathbf{r}_{\text{SF,pre-SF,next}} \quad (\text{D.2})$$

with $W_{zp} \mathbf{r}_{\text{SF,pre-SF,next}}$ as the relative vector from the previous to the next stance TCP described in the auxiliary FoR W_{zp} (with ${}^W\mathbf{A}_{W_{zp}} = \mathbf{A}_z(\alpha_{zp})$). The relative vector is given by

$$W_{zp} \mathbf{r}_{\text{SF,pre-SF,next}} = W_{zp} \mathbf{r}_{\text{SF,next}} - W_{zp} \mathbf{r}_{\text{SF,pre}} = \begin{bmatrix} \sin \varphi_z \left(\frac{l_x}{\varphi_z} - l_y - w \right) \\ \frac{l_x}{\varphi_z} + w - \cos \varphi_z \left(\frac{l_x}{\varphi_z} - l_y - w \right) \\ l_z \end{bmatrix}. \quad (\text{D.3})$$

Here w denotes the (signed) half of the lateral foot separation (cf. idle pose in Section 4.5.2), which is given by

$$w := \begin{cases} +0.1375 \text{ m} & \text{if the previous stance is the } \textit{right} \text{ foot,} \\ -0.1375 \text{ m} & \text{else} \end{cases}. \quad (\text{D.4})$$

See also Figure 4.6 in [100, p. 60] for a visualization of these geometrical considerations. Same as in [100, p. 61], we use a TAYLOR series expansion for the case $|\varphi_z| < 0.1$ rad:

$$W_{zp} \mathbf{r}_{\text{SF,pre-SF,next}} \approx \begin{bmatrix} l_x - (l_y + w) \varphi_z - \frac{1}{6} l_x \varphi_z^2 \\ 2w + l_y + \frac{1}{2} l_x \varphi_z - (l_y + w) \varphi_z^2 \\ l_z \end{bmatrix} \quad (\text{D.5})$$

The orientation of the next stance foot TCP frame described in the world FoR is given by

$${}^W\mathbf{A}_{\text{SF,next}} = \begin{bmatrix} c\alpha_{yn} c\alpha_{zn} & s\alpha_{xn} s\alpha_{yn} c\alpha_{zn} - c\alpha_{xn} s\alpha_{zn} & c\alpha_{xn} s\alpha_{yn} c\alpha_{zn} + s\alpha_{xn} s\alpha_{zn} \\ c\alpha_{yn} s\alpha_{zn} & s\alpha_{xn} s\alpha_{yn} s\alpha_{zn} + c\alpha_{xn} c\alpha_{zn} & c\alpha_{xn} s\alpha_{yn} s\alpha_{zn} - s\alpha_{xn} c\alpha_{zn} \\ -s\alpha_{yn} & s\alpha_{xn} c\alpha_{yn} & c\alpha_{xn} c\alpha_{yn} \end{bmatrix} \quad (\text{D.6})$$

where we used c and s as abbreviations for \cos and \sin , respectively.

Step Parameters from Next Stance In order to automatically generate a QPWT, e. g. as additional output of the contact planner for autonomous locomotion (cf. Section 5.5), it is necessary to compute the step parameters $l_{x|y|z}$ and $\varphi_{x|y|z}$ from the previous and next stance foot pose, i. e., ${}^W\mathbf{r}_{\text{SF,pre}}$, ${}^W\mathbf{A}_{\text{SF,pre}}$, ${}^W\mathbf{r}_{\text{SF,next}}$, and ${}^W\mathbf{A}_{\text{SF,next}}$. First, we compute α_{xp} , α_{yp} , α_{zp} and α_{xn} , α_{yn} , α_{zn} from the rotation matrices ${}^W\mathbf{A}_{\text{SF,pre}}$ and ${}^W\mathbf{A}_{\text{SF,next}}$ with [122, p. 43]

$$\alpha_{yi} = \text{atan2}\left(-{}^W A_{\text{SF},i,31}, \sqrt{{}^W A_{\text{SF},i,11}^2 + {}^W A_{\text{SF},i,21}^2}\right), \quad (\text{D.7})$$

$$\alpha_{xi} = \text{atan2}\left(\frac{{}^W A_{\text{SF},i,32}}{\cos \alpha_{yi}}, \frac{{}^W A_{\text{SF},i,33}}{\cos \alpha_{yi}}\right), \quad (\text{D.8})$$

$$\alpha_{zi} = \text{atan2}\left(\frac{{}^W A_{\text{SF},i,21}}{\cos \alpha_{yi}}, \frac{{}^W A_{\text{SF},i,11}}{\cos \alpha_{yi}}\right) \quad (\text{D.9})$$

for $i \in \{p = \text{pre}, n = \text{next}\}$. Note that a singularity, i. e., $\alpha_{yi} \in \{-\frac{\pi}{2}, \frac{\pi}{2}\}$ and thus, $\cos \alpha_{yi} = 0$ would mean that the corresponding foot points down- or upwards which is not reasonable for biped walking anyways. We obtain the rotational step parameters through

$$\varphi_x = \alpha_{xn}, \quad \varphi_y = \alpha_{yn}, \quad \varphi_z = \alpha_{zn} - \alpha_{zp}. \quad (\text{D.10})$$

For deriving the translational step parameters, we first compute

$$w_{zp} \mathbf{r}_{\text{SF,pre-SF,next}} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = w \mathbf{A}_{W_{zp}}^T (w \mathbf{r}_{\text{SF,next}} - w \mathbf{r}_{\text{SF,pre}}) = \mathbf{A}_z^T(\alpha_{zp}) (w \mathbf{r}_{\text{SF,next}} - w \mathbf{r}_{\text{SF,pre}}) \quad (\text{D.11})$$

where we introduced r_x , r_y , and r_z as abbreviations for the components of the relative vector. From Equation D.3, we directly obtain $l_z = r_z$. Moreover, we get

$$\begin{bmatrix} \frac{\sin \varphi_z}{\varphi_z} & -\sin \varphi_z \\ \frac{1}{\varphi_z} - \frac{\cos \varphi_z}{\varphi_z} & \cos \varphi_z \end{bmatrix} \begin{bmatrix} l_x \\ l_y \end{bmatrix} = \underbrace{\begin{bmatrix} r_x + \sin \varphi_z w \\ r_y - w(1 + \cos \varphi_z) \end{bmatrix}}_{=: \mathbf{b}_1} \quad (\text{D.12})$$

which we can easily solve for l_x and l_y by

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \frac{\varphi_z}{\sin \varphi_z} \begin{bmatrix} \cos \varphi_z & \sin \varphi_z \\ \frac{\cos \varphi_z}{\varphi_z} - \frac{1}{\varphi_z} & \frac{\sin \varphi_z}{\varphi_z} \end{bmatrix} \mathbf{b}_1 = \frac{1}{\sin \varphi_z} \begin{bmatrix} \varphi_z \cos \varphi_z & \varphi_z \sin \varphi_z \\ \cos \varphi_z - 1 & \sin \varphi_z \end{bmatrix} \mathbf{b}_1. \quad (\text{D.13})$$

To guarantee that the solution exists, we constrain $|\varphi_z| \in [0.1 \text{ rad}, \pi[$. Note that hitting the second boundary $|\varphi_z| = \pi$ would mean that the feet point in opposite directions which does not make sense for regular walking patterns and is not kinematically feasible for the robot anyways. Same as before, we formulate a special solution for the case $|\varphi_z| < 0.1 \text{ rad}$. In particular, we recall Equation D.5 and find

$$\begin{bmatrix} 1 - \frac{1}{6} \varphi_z^2 & -\varphi_z \\ \frac{1}{2} \varphi_z & 1 - \varphi_z^2 \end{bmatrix} \begin{bmatrix} l_x \\ l_y \end{bmatrix} \approx \underbrace{\begin{bmatrix} r_x + \varphi_z w \\ r_y - 2w + w \varphi_z^2 \end{bmatrix}}_{=: \mathbf{b}_2} \quad (\text{D.14})$$

which we can easily solve for l_x and l_y by

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} \approx \frac{1}{\left(1 - \frac{2}{3} \varphi_z^2 + \frac{1}{6} \varphi_z^4\right)} \begin{bmatrix} 1 - \varphi_z^2 & \varphi_z \\ -\frac{1}{2} \varphi_z & 1 - \frac{1}{6} \varphi_z^2 \end{bmatrix} \mathbf{b}_2. \quad (\text{D.15})$$

Since $|\varphi_z| < 0.1 \text{ rad}$, the denominator $1 - \frac{2}{3} \varphi_z^2 + \frac{1}{6} \varphi_z^4$ lies within the interval $[0.99335, 1]$, thus, the solution is guaranteed to exist.

Appendix E

Simplified Leg Kinematics

In order to reason about kinematic feasibility, the vertical RMT planner uses the simplified leg kinematics model presented in Figure 4.7. In particular, we are interested in the minimum and maximum vertical RMT position such that the limits of the (joint-space) angles $\varphi_{\text{hf}} \in [-86.3^\circ, 39^\circ]$, $\varphi_{\text{kf}} \in [0.6^\circ, 114.5^\circ]$, and $\varphi_{\text{sf}} \in [-43^\circ, 38.8^\circ]$ (highlighted in orange in Figure 4.7) are respected. The presented intervals have been derived from the physical limits specified in Table H.1 to which a safety margin of 5° has been added.

Since the kinematic model is planar, we project all 3D quantities to the sagittal plane. For this purpose, we introduce the auxiliary CoSys ‘‘S’’ which is generated by rotating the world frame around ${}_{\text{W}}\mathbf{e}_z$ such that the x -axis of the upper body frame UB lies in the x - z -plane of S. Within this auxiliary FoR, we assemble the kinematic chain from the position of the TCP frame of the currently investigated foot ${}_{\text{S}}\mathbf{r}_f$ to the position of the RMT ${}_{\text{S}}\mathbf{r}_t$:

$${}_{\text{S}}\mathbf{r}_{t,x}(\varphi_{\text{kf}}, \varphi_{\text{sf}}) = \underbrace{{}_{\text{S}}\mathbf{r}_{f,x} + l_1 s\varphi_f - l_2 c\varphi_f - l_3 c\alpha + l_4 s\alpha + l_5 s\beta - l_6 s\gamma + l_7 s\varphi_{\text{UB}}}_{=: {}_{\text{S}}\mathbf{r}_{\text{sf},x}}, \quad (\text{E.1})$$

$${}_{\text{S}}\mathbf{r}_{t,z}(\varphi_{\text{kf}}, \varphi_{\text{sf}}) = {}_{\text{S}}\mathbf{r}_{f,z} + l_1 c\varphi_f + l_2 s\varphi_f + l_3 s\alpha + l_4 c\alpha + l_5 c\beta + l_6 c\gamma + l_7 c\varphi_{\text{UB}}, \quad (\text{E.2})$$

where $\alpha := \varphi_f - \varphi_{\text{zf}}$, $\beta := \alpha - \varphi_{\text{sf}}$, and $\gamma := \varphi_{\text{kf}} - \beta$ denote auxiliary angles which help to shorten expressions. Moreover, we used c and s as abbreviations for \cos and \sin , respectively. Note that

- ${}_{\text{S}}\mathbf{r}_{t,x}$ is known from the first cycle of the horizontal RMT planner (cf. Section 6.14.2),
- ${}_{\text{S}}\mathbf{r}_{f,x|z}$, φ_f , and φ_{zf} are known from the foot/toe motion planner (cf. Sections 6.7 and 6.8),
- φ_{UB} is known from the upper body orientation planner (cf. Section 6.6), and
- the lengths $\{l_i\}$ are parameterized according to Table 4.4.

The three angles φ_{hf} , φ_{kf} , and φ_{sf} represent the input parameters for computing ${}_{\text{S}}\mathbf{r}_{t,z}$ which, however, are not independent of each other. In particular, we find

$$\varphi_{\text{hf}}(\varphi_{\text{kf}}, \varphi_{\text{sf}}) = \beta - \varphi_{\text{kf}} - \varphi_{\text{UB}} = \varphi_f - \varphi_{\text{zf}} - \varphi_{\text{sf}} - \varphi_{\text{kf}} - \varphi_{\text{UB}} \quad (\text{E.3})$$

which is used to check violation of the corresponding joint limits (although φ_{hf} does not explicitly occur in Equations E.1 and E.2). Using Equation E.1, we can express φ_{kf} as a function of φ_{sf} :

$$\varphi_{\text{kf}}(\varphi_{\text{sf}}) = \beta(\varphi_{\text{sf}}) + \text{asin}(\chi(\varphi_{\text{sf}})) \quad \text{with} \quad \chi(\varphi_{\text{sf}}) := \frac{{}_{\text{S}}\mathbf{r}_{\text{sf},x} - {}_{\text{S}}\mathbf{r}_{t,x}}{l_6} + \frac{l_5}{l_6} s\beta(\varphi_{\text{sf}}) + \frac{l_7}{l_6} s\varphi_{\text{UB}} \quad (\text{E.4})$$

and ${}_{\text{S}}\mathbf{r}_{\text{sf},x}$ from Equation E.1 (describes the x -position of the ankle flexion joint). In the special case $|\chi(\varphi_{\text{sf}})| > 1$, there exists no φ_{kf} such that we can reach the horizontal RMT position ${}_{\text{S}}\mathbf{r}_{t,x}$. By inserting Equation E.4 into Equation E.2 we obtain ${}_{\text{S}}\mathbf{r}_{t,z}(\varphi_{\text{sf}})$ such that we only have to sample the angle φ_{sf} in order to find the vertical RMT boundaries as described in Section 6.14.1.

Appendix F

Dynamics of the Five-Mass Model

In order to reason about dynamic feasibility, the horizontal RMT planner approximates the multi-body dynamics of the robot through the five-mass model presented in Figure 4.7. This section is dedicated to the derivation and analysis of the EoM of this model. The most important quantities of the five-mass model are

- $m_t(t)$, ${}_{\text{UB}}\Theta_t^t = \text{const.}$ (cf. Equation 4.4), $\mathbf{r}_t(t)$, and $\mathbf{s}_{\text{UB}}(t)$ as the mass, mass moment of inertia tensor, position, and orientation of the virtual torso segment,
- $m_f = \text{const.}$ and $\mathbf{r}_f(t)$ as the mass and position of the foot $f \in \{\text{RF}, \text{LF}\}$,
- $\mathbf{W}_{f,\text{cont}}^{\text{W}} = [\mathbf{F}_{f,\text{cont}}^{\text{T}}, \mathbf{T}_{f,\text{cont}}^{\text{T}}]^{\text{T}}$ as the combined contact wrench of the right and left foot acting on the robot (cf. Equations 4.5 and 4.6),
- $m_h(t)$, $\mathbf{r}_h(t)$, and $\mathbf{W}_{h,\text{ext}}^h(t) = [\mathbf{F}_{h,\text{ext}}^{\text{T}}, \mathbf{T}_{h,\text{ext}}^{\text{T}}]^{\text{T}}$ as the mass, position, and external (multi-contact) wrench of the hand $h \in \{\text{RH}, \text{LH}\}$, and
- $\mathbf{r}_{\text{CoM}}(t)$ as the position of the CoM.

All masses are subject to the gravitational acceleration vector ${}_{\text{W}}\mathbf{g} = [0, 0, -g]^{\text{T}}$. As already explained in Section 4.5.2, the total mass m of the robot is constant (cf. Equation 4.3) such that there is no mass transport over the system's boundaries. However, the masses $m_t(t)$ and $m_h(t)$ depend on the task-space selection factors $\xi_h(t)$ (cf. Equation 4.2), therefore, we also have to consider their n -th order time derivatives (with $n > 0$)

$$\begin{aligned} m_h(t) &:= \xi_h(t) M_h m, & \dot{m}_h(t) &= \dot{\xi}_h(t) M_h m, & m_h^{(n)}(t) &= \xi_h^{(n)}(t) M_h m, \\ m_t(t) &:= m - \sum_e m_e(t), & \dot{m}_t(t) &= -\sum_h \dot{m}_h(t), & m_t^{(n)}(t) &= -\sum_h m_h^{(n)}(t). \end{aligned} \quad (\text{F.1})$$

where $e \in \{\text{RF}, \text{LF}, \text{RH}, \text{LH}\}$. Within the context of the horizontal RMT planner, all quantities except for the horizontal RMT position ${}_{\text{W}}\mathbf{r}_{t,x|y}(t)$, the foot-ground contact wrench $\mathbf{W}_{f,\text{cont}}^{\text{W}}$, and the position of the CoM $\mathbf{r}_{\text{CoM}}(t)$ are known. For time-dependent quantities, we have access to analytic time derivatives of arbitrary order (all previously planned trajectories are of polynomial or QBSpline type). Note that this also holds for the upper body orientation $\mathbf{s}_{\text{UB}}(t)$, i. e., we can easily derive the angular velocity $\boldsymbol{\omega}_{\text{UB}}$ (cf. Equation B.22) and the angular acceleration $\dot{\boldsymbol{\omega}}_{\text{UB}}$ (cf. Equation B.23) from the corresponding quaternion representations $\dot{\mathbf{s}}_{\text{UB}}(t)$ and $\ddot{\mathbf{s}}_{\text{UB}}(t)$. In the following, all time derivatives of FoR-dependent quantities are defined to be absolute, i. e., with respect to an inertial FoR such as the world frame W.

Change of Linear Momentum Since there is no mass transport over the system's boundaries, the change of the five-mass model's total linear momentum ${}_{\text{W}}\mathbf{p} \in \mathbb{R}^3$ observed in the inertial world FoR (left hand subscripts omitted in the following for brevity) is given by (*Principle of Linear Momentum*, see [179, p. 185ff])

$$\frac{d\mathbf{p}}{dt} = \sum_i \mathbf{F}_i \quad \text{with} \quad \mathbf{p} = m_t \dot{\mathbf{r}}_t + \sum_e m_e \dot{\mathbf{r}}_e \quad \text{and} \quad \sum_i \mathbf{F}_i = m \mathbf{g} + \mathbf{F}_{f,\text{cont}} + \sum_h \xi_h \mathbf{F}_{h,\text{ext}}. \quad (\text{F.2})$$

Note that we multiplied the external (multi-contact) force $\mathbf{F}_{h,\text{ext}}$ with the corresponding task-space selection factor ξ_h . This is motivated by the fact that \mathbf{r}_h , i. e., the point of action of $\mathbf{F}_{h,\text{ext}}$ and $\mathbf{T}_{h,\text{ext}}$, is only known if the corresponding hand resides in the task-space ($\xi_h = 1$). Thus, in case the hand is assigned to the null-space ($\xi_h = 0$), we avoid erroneous contributions to the EoM (\mathbf{r}_h occurs in contribution to the angular momentum). However, the task-space selection factor planner (cf. Section 6.11) and the external wrench planner (cf. Section 6.13) use a timing parametrization such that ξ_h is (almost always) equal to one whenever the corresponding external (multi-contact) wrench is non-zero. From Equation F.2 we obtain

$$\mathbf{F}_{f,\text{cont}} = m_t \ddot{\mathbf{r}}_t + \dot{m}_t \dot{\mathbf{r}}_t + \mathbf{C}_1 \quad \text{with} \quad \mathbf{C}_1 := \sum_e (m_e \ddot{\mathbf{r}}_e + \dot{m}_e \dot{\mathbf{r}}_e) - \sum_h (\xi_h \mathbf{F}_{h,\text{ext}}) - m \mathbf{g} \quad (\text{F.3})$$

where $\mathbf{C}_1 \in \mathbb{R}^3$ represents an auxiliary variable consisting entirely of known quantities.

Change of Angular Momentum Since there is no mass transport over the system's boundaries, the change of the five-mass model's total angular momentum ${}^W L^W \in \mathbb{R}^3$ observed and measured in the inertial world FoR W (left hand subscript; omitted in the following for brevity) using the origin of the world FoR W (right hand superscript) as reference point is given by (*Principle of Angular Momentum*, see [179, p. 185ff])

$$\begin{aligned} \frac{dL^W}{dt} &= \sum_i \mathbf{T}_i \quad \text{with} \quad L^W = m_t \mathbf{r}_t \times \dot{\mathbf{r}}_t + \boldsymbol{\Theta}_t^t \boldsymbol{\omega}_{\text{UB}} + \sum_e m_e \mathbf{r}_e \times \dot{\mathbf{r}}_e, \\ \sum_i \mathbf{T}_i &= m_t \mathbf{r}_t \times \mathbf{g} + \sum_e (m_e \mathbf{r}_e \times \mathbf{g}) + \mathbf{T}_{f,\text{cont}} + \sum_h \xi_h (\mathbf{r}_h \times \mathbf{F}_{h,\text{ext}} + \mathbf{T}_{h,\text{ext}}). \end{aligned} \quad (\text{F.4})$$

Similar to before, this can be reformulated to

$$\mathbf{T}_{f,\text{cont}} = m_t \mathbf{r}_t \times (\ddot{\mathbf{r}}_t - \mathbf{g}) + \dot{m}_t \mathbf{r}_t \times \dot{\mathbf{r}}_t + \mathbf{C}_2 \quad (\text{F.5})$$

where $\mathbf{C}_2 \in \mathbb{R}^3$ denotes an auxiliary variable consisting entirely of known quantities:

$$\mathbf{C}_2 := \frac{d(\boldsymbol{\Theta}_t^t \boldsymbol{\omega}_{\text{UB}})}{dt} + \sum_e (m_e \mathbf{r}_e \times (\ddot{\mathbf{r}}_e - \mathbf{g}) + \dot{m}_e \mathbf{r}_e \times \dot{\mathbf{r}}_e) - \sum_h \xi_h (\mathbf{r}_h \times \mathbf{F}_{h,\text{ext}} + \mathbf{T}_{h,\text{ext}}) \quad (\text{F.6})$$

with the change of angular momentum originating from the mass moment of inertia tensor of the virtual torso given by (see [179, p. 191])

$$\frac{d(\boldsymbol{\Theta}_t^t \boldsymbol{\omega}_{\text{UB}})}{dt} = \boldsymbol{\omega}_{\text{UB}} \times (\boldsymbol{\Theta}_t^t \boldsymbol{\omega}_{\text{UB}}) + \boldsymbol{\Theta}_t^t \dot{\boldsymbol{\omega}}_{\text{UB}}. \quad (\text{F.7})$$

Center of Mass (CoM) Dynamics The position of the CoM \mathbf{r}_{CoM} , its (absolute) velocity $\dot{\mathbf{r}}_{\text{CoM}}$, and its n -th order (absolute) time derivative $\mathbf{r}_{\text{CoM}}^{(n)}$ are given by

$$\begin{aligned} \mathbf{r}_{\text{CoM}} &:= \frac{1}{m} \left[m_t \mathbf{r}_t + \sum_f m_f \mathbf{r}_f + \sum_h m_h \mathbf{r}_h \right], \\ \dot{\mathbf{r}}_{\text{CoM}} &= \frac{1}{m} \left[\dot{m}_t \mathbf{r}_t + m_t \dot{\mathbf{r}}_t + \sum_f m_f \dot{\mathbf{r}}_f + \sum_h (\dot{m}_h \mathbf{r}_h + m_h \dot{\mathbf{r}}_h) \right], \\ \mathbf{r}_{\text{CoM}}^{(n)} &= \frac{1}{m} \left[\sum_{k=0}^n \binom{n}{k} m_t^{(k)} \mathbf{r}_t^{(n-k)} + \sum_f m_f \mathbf{r}_f^{(n)} + \sum_h \sum_{k=0}^n \binom{n}{k} m_h^{(k)} \mathbf{r}_h^{(n-k)} \right]. \end{aligned} \quad (\text{F.8})$$

Reduced Model Torso (RMT) Dynamics The position of the RMT \mathbf{r}_t , its (absolute) velocity $\dot{\mathbf{r}}_t$, and its n -th order (absolute) time derivative $\mathbf{r}_t^{(n)}$ are given by

$$\begin{aligned} \mathbf{r}_t &= \frac{1}{m_t} \left[m \mathbf{r}_{\text{CoM}} - \sum_f m_f \mathbf{r}_f - \sum_h m_h \mathbf{r}_h \right], \\ \dot{\mathbf{r}}_t &= \frac{1}{m_t} \left[m \dot{\mathbf{r}}_{\text{CoM}} - \dot{m}_t \mathbf{r}_t - \sum_f m_f \dot{\mathbf{r}}_f - \sum_h (\dot{m}_h \mathbf{r}_h + m_h \dot{\mathbf{r}}_h) \right], \\ \mathbf{r}_t^{(n)} &= \frac{1}{m_t} \left[m \mathbf{r}_{\text{CoM}}^{(n)} - \sum_{k=1}^n \binom{n}{k} m_t^{(k)} \mathbf{r}_t^{(n-k)} - \sum_f m_f \mathbf{r}_f^{(n)} - \sum_h \sum_{k=0}^n \binom{n}{k} m_h^{(k)} \mathbf{r}_h^{(n-k)} \right]. \end{aligned} \quad (\text{F.9})$$

Note that due to the time dependency of $m_t(t)$, the n -th order derivative $\mathbf{r}_t^{(n)}$ depends on all lower-order derivatives $\mathbf{r}_t^{(d)}$ with $d < n$.

Static Case If we consider all time derivatives to be zero, the contact wrench of the foot-ground interface simplifies to

$$\mathbf{F}_{f,\text{cont}} = \mathbf{F}_{\text{ZMP}} = - \sum_h (\xi_h \mathbf{F}_{h,\text{ext}}) - m \mathbf{g}, \quad (\text{F.10})$$

$$\mathbf{T}_{f,\text{cont}} = \mathbf{T}_{\text{ZMP}} + \mathbf{r}_{\text{ZMP}} \times \mathbf{F}_{\text{ZMP}} = -m \mathbf{r}_{\text{CoM}} \times \mathbf{g} - \sum_h \xi_h (\mathbf{r}_h \times \mathbf{F}_{h,\text{ext}} + \mathbf{T}_{h,\text{ext}}). \quad (\text{F.11})$$

This can be reformulated to

$$m_t \mathbf{r}_t \times \mathbf{g} = \underbrace{\left(m \mathbf{r}_{\text{ZMP}} - \sum_e m_e \mathbf{r}_e \right) \times \mathbf{g} + \sum_h \xi_h [(\mathbf{r}_{\text{ZMP}} - \mathbf{r}_h) \times \mathbf{F}_{h,\text{ext}} - \mathbf{T}_{h,\text{ext}}]}_{=: \mathbf{C}_3} - \mathbf{T}_{\text{ZMP}} \quad (\text{F.12})$$

where $\mathbf{C}_3 \in \mathbb{R}^3$ denotes an auxiliary variable consisting entirely of known quantities. For the individual components along the x -, y -, and z -axis of the world frame W , we obtain

$$m_t \begin{bmatrix} W^r_{t,x} \\ W^r_{t,y} \\ W^r_{t,z} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \begin{bmatrix} W^{\mathbf{C}_{3,x}} \\ W^{\mathbf{C}_{3,y}} \\ W^{\mathbf{C}_{3,z}} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ W^{\mathbf{T}_{\text{ZMP},z}} \end{bmatrix} \Rightarrow \begin{bmatrix} -W^r_{t,y} m_t g \\ W^r_{t,x} m_t g \\ 0 \end{bmatrix} = \begin{bmatrix} W^{\mathbf{C}_{3,x}} \\ W^{\mathbf{C}_{3,y}} \\ W^{\mathbf{C}_{3,z}} - W^{\mathbf{T}_{\text{ZMP},z}} \end{bmatrix} \quad (\text{F.13})$$

which allows us to explicitly compute $W^r_{t,x}$, $W^r_{t,y}$, and $W^{\mathbf{T}_{\text{ZMP},z}}$ for the static case through

$$W^r_{t,x} = \frac{W^{\mathbf{C}_{3,y}}}{m_t g} \quad \text{and} \quad W^r_{t,y} = -\frac{W^{\mathbf{C}_{3,x}}}{m_t g} \quad \text{and} \quad W^{\mathbf{T}_{\text{ZMP},z}} = W^{\mathbf{C}_{3,z}}. \quad (\text{F.14})$$

Static Case Without External Contact Wrenches If we additionally assume that there are no external (multi-contact) wrenches acting on the robot, i. e., if we insert $\mathbf{F}_{h,\text{ext}} = \mathbf{T}_{h,\text{ext}} = \mathbf{0}$ into Equations F.10 and F.11, we find

$$\mathbf{T}_{\text{ZMP}} = m (\mathbf{r}_{\text{ZMP}} - \mathbf{r}_{\text{CoM}}) \times \mathbf{g} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ W^{\mathbf{T}_{\text{ZMP},z}} \end{bmatrix} = m \begin{bmatrix} W^{\mathbf{r}_{\text{ZMP},x}} - W^{\mathbf{r}_{\text{CoM},x}} \\ W^{\mathbf{r}_{\text{ZMP},y}} - W^{\mathbf{r}_{\text{CoM},y}} \\ W^{\mathbf{r}_{\text{ZMP},z}} - W^{\mathbf{r}_{\text{CoM},z}} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \quad (\text{F.15})$$

and thus

$$W^{\mathbf{r}_{\text{CoM},x}} = W^{\mathbf{r}_{\text{ZMP},x}} \quad \text{and} \quad W^{\mathbf{r}_{\text{CoM},y}} = W^{\mathbf{r}_{\text{ZMP},y}} \quad \text{and} \quad W^{\mathbf{T}_{\text{ZMP},z}} = 0 \quad (\text{F.16})$$

which shows that the horizontal positions of the CoM and the ZMP coincide in this case.

Appendix G

Cubic and Quintic Spline Interpolation and Collocation

This appendix has already been published in [2].

This appendix integrates the article *Fast Approximation of Over-Determined Second-Order Linear Boundary Value Problems by Cubic and Quintic Spline Collocation* [2] by SEIWALD and RIXEN which was published on June 25, 2020 in the journal *MDPI Robotics*. The article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>). While the algorithms described in the article are very generic and not restricted to the field of robotics, they represent core elements of the motion generator presented in Chapter 6. Since the article contains essential contributions of the author which were made within the context of the work on this dissertation, it was decided to integrate the article in the form of an appendix into this document. The following represents an almost exact replication of [2] where only slight modifications were made to highlight the connections to the contents of this thesis. Moreover, text passages carrying information already provided in other parts of this document have been shortened. However, the main contents (equations, algorithms, findings, etc.) are identical to [2].

Abstract We present an efficient and generic algorithm for approximating second-order linear BVPs through spline collocation. In contrast to the majority of other approaches, our algorithm is designed for over-determined problems. These typically occur in control theory, where a system, e. g. a robot, should be transferred from a certain initial state to a desired target state while respecting characteristic system dynamics. Our method uses polynomials of maximum degree three/five as base functions and generates a cubic/quintic spline, which is C^2/C^4 -continuous and satisfies the underlying ODE at user-defined collocation sites. Moreover, the approximation is forced to fulfill an over-determined set of two-point BCs, which are specified by the given control problem. The algorithm is suitable for time-critical applications, where accuracy only plays a secondary role. For consistent BCs, we experimentally validate convergence towards the analytic solution, while for inconsistent BCs our algorithm is still able to find a “reasonable” approximation. However, to avoid divergence, collocation sites have to be appropriately chosen. The proposed scheme is evaluated experimentally through comparison with the analytical solution of a simple test system. Furthermore, a fully documented C++ implementation with unit tests as example applications is provided.

G.1 Introduction

In almost every field of natural science and engineering we face differential equations, which are typically used for modeling dynamical systems. Especially in engineering, the more specific case of BVPs is very prominent. In other words, one often searches for the behavior of the investigated system between some kind of fixed, i. e., known, spatial or temporal boundaries. One can try to derive the analytical solution to the problem, however, for complex systems this

can be difficult or even impossible. In such cases, it can be sufficient to approximate the solution for which a variety of techniques exists.

Methods based on finite differences approximate the derivatives by difference quotients to obtain a system of equations which depends only on the primal function. This allows a solution to be found by formulating a linear system of (algebraic) equations representing the transformed system at certain grid points. In contrast, shooting methods aim at the iterative solution of an equivalent *Initial Value Problem (IVP)*, which is typically easier to handle than the original BVP. While for single shooting the IVP is evaluated over the complete time interval, multiple shooting considers a partitioned time domain. Another technique is given by the finite element approach, which is mainly used for *Partial Differential Equations (PDEs)* as they occur, for example, in structural-, thermo-, and fluid dynamics. Typically finite element methods are based on a weak formulation of the residual and on splitting the considered domain into elements on which the local base functions for approximating the solution are defined. Although designed for PDEs, they can be applied to the simpler case of ODEs, which are the focus of this contribution (see for example LEE et al. [273]). A variety of other techniques exist. However, these three groups appear to be used the most in the field of engineering.

In the following, we approximate the solution through *spline collocation* using PP trial functions, which is a well-known technique to solve BVPs, see, for example, AHLBERG et al. [53, p.52], DE BOOR and SWARTZ [127], and AHLBERG and ITO [54]. Over the past decades, various algorithms emerged, which can be classified into two types. The first type, also known as *smoothest spline collocation* (or just *spline collocation* as in CHRISTARA and NG [116]), aims at matching the differential equation at one collocation site per spline segment, which is typically a knot or the mid-point of the segment, while simultaneously forcing the spline to have maximum smoothness, i. e., highest possible continuity [116]. The second type, in [116] called *Gaussian collocation*, removes the constraints on higher order continuity and instead uses more collocation sites, which are typically chosen to be the Gaussian points of each segment. This class, which is also called *orthogonal spline collocation* (see BIALECKI and FAIRWEATHER [80]), originates from DE BOOR and SWARTZ [127] and aims at maximizing the order of convergence. In order to also provide a competitive convergence for the first type of methods, special variants for quadratic and quintic spline collocation have been proposed by HOUSTIS et al. in [212] and by IRODOU-ELLINA and HOUSTIS in [220], respectively. Optimal methods for quadratic and cubic splines on non-uniform grids, i. e., for an inhomogeneous segmentation of the spline, have been presented by CHRISTARA and NG in [116].

Note that in addition to general purpose codes, such as *Colsys* (Fortran) for non-linear mixed-order systems of multi-point boundary value ODEs introduced by ASCHER in [63], highly specialized algorithms, which aim at obtaining the best possible approximation for certain use cases, have also been published. Recent examples are methods for integro-differential equations (e. g. ZHANG et al. [454]) or fractional differential equations (e. g. AKRAM and TARIQ [57]), which occur in certain material models exhibiting memory effects. Along with ODEs, various kinds of (multi-variable) PDEs have been investigated. See BIALECKI and FAIRWEATHER [80] for a survey on corresponding Gaussian collocation methods. Note that, for PDEs, the time domain is typically discretized using finite differences, e. g., by the Crank-Nicholson approach as used in [80], or the second-order backward difference used by ZHANG et al. in [454], while spline collocation is used for approximating the spacial variables. Lastly, methods for *Differential Algebraic Equations (DAEs)* have also been developed, e. g., the *Colsys* extension *Coldae* by ASCHER and SPITERI [65] for semi-explicit DAEs of index 2 and fully implicit DAEs of index 1.

In our opinion, despite the variety of techniques, there is still a lack of simple methods prioritizing execution time over approximation quality, which is essential for time critical control applications. The aim of this contribution is to provide an algorithm satisfying these needs while focusing on the special case of second-order linear ODEs, which are very common for dynamic systems. Moreover, we focus on over-determined BVPs, i. e., where more BCs are given

than necessary. This may at first seem to be a restriction of our algorithm since it needs more information than other implementations, however, it allows us to also consider inconsistent BVPs, i. e., the case where no exact solution to the problem exists.

Relations to this Thesis It might appear strange to search for an approximation of something that actually does not exist. However, we face exactly this situation during motion generation for our humanoid robot LOLA. In particular, we use a simplified model of the robot’s multi-body dynamics, cf. Figure 4.7 right, to plan the (horizontal) motion of the RMT over a certain time horizon. The planned RMT motion resembles the dynamics of the model, which can be formulated as second-order linear ODE (cf. Section 6.14.2), and is constrained to certain values on position-, velocity-, and acceleration-level at the boundaries of the planning horizon. This leads to an over-determined BVP of the type investigated in this contribution. The BCs at the beginning reflect the current state of the robot while the BCs at the end represent the target state, e. g. the static idle pose at the end of the sequence. For a seamless motion of the robot it is crucial to guarantee the satisfaction of the BCs, i. e., a perfect match of the boundary states. In contrast, it is sufficient to approximate the dynamics of the underlying ODE since it is derived from a simplified model which is an approximation in itself. This is the key idea behind the formulation of an over-determined BVP, which may thus not have a proper “real” solution. To the author’s knowledge, all comparable algorithms assume that a solution of the BVP exists, while most of them also require it to be unique and “sufficiently” smooth. In the following, we do not further restrict ourselves to the special application of motion generation for LOLA. Since the proposed algorithm is generic, we derive it in the most general way since it may be useful also for different applications in robotics.

Additional Remarks Our method can be seen as *smoothest spline collocation*, i. e., it belongs to the “first” type as classified above. We do not apply special techniques to increase the order of convergence, but instead adhere to its basic form. This leads to a much simpler derivation and implementation. In addition, it makes the algorithm faster by sacrificing approximation quality. This complies with the needs of our target application as explained in the previous paragraph. We emphasize that our focus lies on simplicity, robustness, and efficiency. Thus we will not search for a mathematical formulation of the convergence order. Instead, the algorithm is evaluated mainly through experiments, where runtime performance is our primary concern.

As stated before, there exist numerous methods for solving linear BVPs or spline interpolation/collocation in general. For most approaches, solving a large-sparse or small-dense LSE represents the main workload. To overcome this bottleneck, parallelized algorithms have been developed, which typically exploit the special structure of the involved matrices, e. g. the scheme proposed by WRIGHT in [445] for staircase matrix structures. Although we aim at efficiency, we do not consider explicit parallelization as acceleration technique. This is because our algorithm is designed for embedded systems which typically feature only few physical CPU cores running also other time-critical tasks. Moreover, the use of general purpose GPUs, e. g. through *Cuda* or *OpenCL*, is often not feasible since the CPU-GPU interface lacks capabilities for hard real-time requirements. Finally, dedicated to our target application, we only consider (comparatively) small problems with runtimes ≈ 1 ms. For such systems the performance boost obtained through parallelization is likely to be canceled out by the synchronization overhead. Nevertheless, our algorithm may also be used for large scale problems. In this case an “off the shelf” parallel solver for dense LSEs may be used, cf. MAGOULÉS et al. [297, p. 105ff]. However, one should keep in mind that by using an iterative scheme execution time is not deterministic anymore, and, even worse, the solver might not converge. As an alternative, there exists an efficient way for the parallel solution of decoupled, multi-dimensional BVPs, which takes advantage of intermediate results, see Appendix G.3 for details.

G.2 Materials and Methods

In the following, two versions of our algorithm are presented: one using a cubic spline and the other using a quintic spline for approximating the BVP. As the derivations and resulting algorithms are similar, we show the connecting links by presenting both methods in parallel. The version based on cubic splines is naturally simpler, although, using quintic splines leads to a smoother approximation. Indeed, it is C^4 - instead of C^2 -continuous, which can be preferable for some use cases. Moreover, quintic splines allow us to directly preset first and second-order derivatives at both boundaries, which otherwise requires the introduction of virtual control-points and in turn can lead to poor results, as discussed in Appendix G.4. Although deriving the proposed collocation algorithm for quintic splines is more advanced than its cubic counterpart, overall performance is superior, because the same approximation quality can be obtained with less collocation sites and hence with less computational effort as shown in Appendix G.4. However, this requires the underlying ODE to be sufficiently smooth. Note that, in contrast to our intention, most other investigations choose quintic splines for approximating fourth-order ODEs, e. g., in [57, 220, 419, 454] (similar to choosing cubic splines for second-order ODEs).

We highlight that the proposed method is not only inspired by, but is also heavily based on the interpolation and collocation algorithms presented by MUND et al. in [312] and BUSCHMANN et al. in [98] (based itself on RUSSELL and SHAMPINE [362]), respectively. The main contribution of this article is the combination, extension, and runtime optimization of those methods. Moreover, we provide a detailed and self-contained derivation together with a fully documented open-source C++ reference implementation. Having a background in mechanical engineering, the author's intention is to present a simple and self-contained derivation of the proposed algorithm, which can be easily understood, implemented, and extended by readers also lacking a dedicated mathematical background.

G.2.1 Problem Statement

Consider the second-order linear time-variant ODE

$$\alpha(t)\ddot{F}(t) + \beta(t)\dot{F}(t) + \gamma(t)F(t) = \tau(t) \quad \text{for } t \in [t_0, t_n] \quad (\text{G.1})$$

where $\dot{F}(t)$ and $\ddot{F}(t)$ denote the first and second derivative of the unknown function $F(t)$ with respect to time t , i. e.,

$$\dot{F}(t) = \frac{dF(t)}{dt} \quad \text{and} \quad \ddot{F}(t) = \frac{d^2F(t)}{dt^2}. \quad (\text{G.2})$$

Note that t does not have to represent time. However, this synonym is used in the following due to the typical appearance of Equation G.1 in dynamical systems. The coefficients α , β , γ , and the right-hand side τ are arbitrary, in general nonlinear, but known functions of t . Let the system from Equation G.1 be constrained by the BCs

$$\begin{aligned} F(t_0) &= F_0, & \dot{F}(t_0) &= \dot{F}_0, & \ddot{F}(t_0) &= \ddot{F}_0, \\ F(t_n) &= F_n, & \dot{F}(t_n) &= \dot{F}_n, & \ddot{F}(t_n) &= \ddot{F}_n, \end{aligned} \quad (\text{G.3})$$

where t_0 and t_n define the considered time interval $t \in [t_0, t_n]$ and $F_0, \dot{F}_0, \ddot{F}_0, F_n, \dot{F}_n, \ddot{F}_n$ are user-defined constants. Then the system from Equation G.1 together with the BCs from Equation G.3 represent the second-order linear two-point BVP for which an approximation is to be found. Note that Equation G.3 considers Dirichlet, Neumann, and second-order BCs independently of each other. In contrast, various other algorithms assume Robin BCs, i. e., a linear combination

of Dirichlet and Neumann BCs, which is not equivalent to our approach. Due to Equation G.3, the BVP is over-determined and the existence of a solution $F(t)$ depends on the consistency of the BCs with the ODE.

In the following, we investigate the approximation of $F(t)$ through spline collocation, i. e., we generate a spline $y(t)$ which satisfies the underlying ODE from Equation G.1 at a user-defined set of distinct collocation sites $\{t_k\}$, numbered in increasing order, which lie within the considered interval (t_0, t_n) , i. e.,

$$\alpha(t_k)\ddot{y}(t_k) + \beta(t_k)\dot{y}(t_k) + \gamma(t_k)y(t_k) = \tau(t_k) \quad \text{for } t_0 < t_k < t_{k+1} < t_n. \quad (\text{G.4})$$

Moreover, $y(t)$ is forced to fulfill the BCs specified in Equation G.3 at t_0 and t_n , i. e.,

$$\begin{aligned} y(t_0) &= y_0 = F_0, & \dot{y}(t_0) &= \dot{y}_0 = \dot{F}_0, & \ddot{y}(t_0) &= \ddot{y}_0 = \ddot{F}_0, \\ y(t_n) &= y_n = F_n, & \dot{y}(t_n) &= \dot{y}_n = \dot{F}_n, & \ddot{y}(t_n) &= \ddot{y}_n = \ddot{F}_n. \end{aligned} \quad (\text{G.5})$$

Here we use $y(t)$ to denote the approximating spline while the exact solution is represented by $F(t)$. For clarity, we also use different denominations for $\{t_k\}$ and $\{y_k\}$ by using the terms *collocation sites* and *collocation points*, respectively. While $\{t_k\}$ are user-defined parameters, $\{y_k\}$ describe the solution to be found.

Since the proposed collocation algorithm is strongly related to the interpolation of cubic and quintic splines, which may not be common to some readers, spline interpolation is recapitulated in Appendix G.2.3. Then, the proposed collocation method is derived in Appendix G.2.7. Moreover, we reuse core elements of the interpolation algorithm during collocation, thus, we cannot omit its derivation.

G.2.2 Spline Parametrization

Before diving into the derivation of algorithms, one first has to decide which spline representation to use. In the literature, formulations such as B-Splines are common, since they feature inherent continuity and local control, which typically leads to banded systems [64]. In general, B-Splines do not pass through their control-points, which seems to make interpolation difficult at first sight, however, efficient algorithms for interpolation and collocation exist, see, for example, DE BOOR [129, p. 171ff, 243ff]. In [64], ASCHER et al. have shown that B-Splines might not be as stable and efficient as other representations, namely monomial and Hermite type bases, especially when it comes to implementation. In particular, monomial bases have been recommended due to their superior condition, and thus lower roundoff errors. For all three forms, B-Spline, Hermite type, and monomial, the core operation during Gaussian collocation is typically the solution of an *Almost Block Diagonal (ABD)* LSE [80, 129]. A generic solver for these type of systems is *Solveblok* [128] by DE BOOR and WEISS, while the special structure occurring for monomial bases is exploited by *Abdpack* introduced by MAJAESS et al. in [298] which features increased speed and lower memory consumption. Unfortunately, for smoothest spline collocation as presented in the following, the corresponding collocation matrix is dense, thus, we cannot apply these algorithms. However, when compared to Gaussian collocation, the count of collocation sites and thus the dimension of the corresponding LSE is much smaller, which can lead to comparable performance.

Despite the popularity of B-Splines, we use the PP form [129, p. 69], which describes the spline through the coefficients of interconnected, but independently defined, polynomial segments. We use a special type of monomial bases, namely the canonical form of the polynomials, which may not be as efficient as the choice in [64], however, it makes our algorithm much simpler. By using this formulation, continuity between the spline segments needs to be explicitly established. The evaluation of the resulting spline, however, boils down to the evaluation of a

single polynomial belonging to the corresponding segment, which is in general much quicker than evaluating the equivalent B-Spline form⁹⁷. This is essential for time-critical applications, where the resulting spline has to be evaluated as quickly as possible. Note that we are free to construct the spline in B-Spline formulation and convert it to the corresponding PP form in a post-processing step, see [129, p. 101]. However, this introduces an additional (expensive) step which we try to avoid since, in our case, not only the evaluation but also the construction of the spline is time critical.

Let the spline $y(t)$ be defined as

$$y(t) = s_i(\eta_i(t)) \quad \text{for } t_0 \leq t_i \leq t < t_{i+1} \leq t_n \quad \text{with } i = 0, \dots, n-1, \quad (\text{G.6})$$

where s_i represents the i -th of the $n > 1$ spline segments parameterized by the normalized interpolation parameter η_i . We call $t \in [t_0, t_n]$ and $\eta_i \in [0, 1]$ the global and local interpolation parameters, respectively, for which we choose the linear mapping

$$\eta_i(t) = \frac{1}{h_i} t - \frac{t_i}{h_i} = (t - t_i) g_i \quad (\text{G.7})$$

with $h_i > 0$ as the duration of the i -th segment $h_i = t_{i+1} - t_i$ and its reciprocal $g_i = 1/h_i$. The partitioning of the spline into n segments is visualized in Figure G.1. In the following, we predominantly derive expressions in local segment space, i. e., with respect to η_i , since this makes the notation clearer, especially in Appendix G.2.7. Note that, in contrast to some other approaches, we do not require homogeneous partitioning, thus, the segmentation can be chosen arbitrarily as long as spline knots do not coincide. However, in Appendix G.2.3, we show that for best numerical stability uniform partitioning should be used.

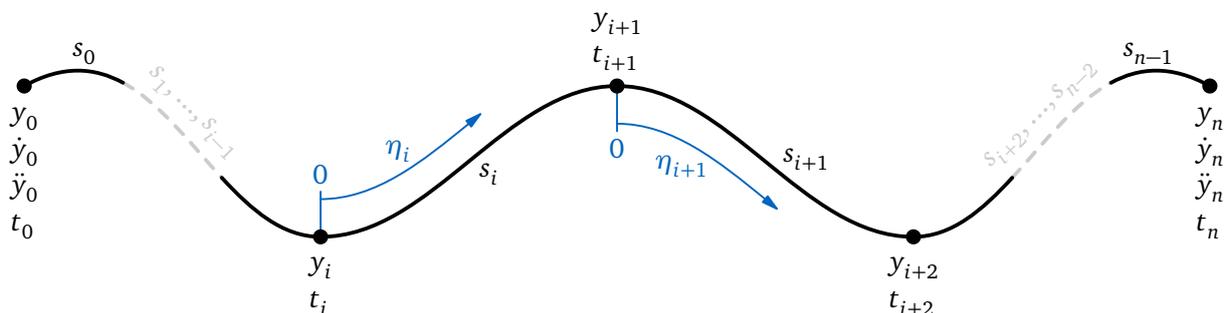


Figure G.1: Segmentation and parametrization of the investigated spline $y(t)$. The spline consists of n interconnected segments, which share the interior knots with their neighbors. Each segment is described through the local interpolation parameter $\eta_i \in [0, 1]$.

In the case of cubic splines (left subscript C), each segment ${}_C s_i$ represents a polynomial of degree three,

$${}_C s_i(\eta_i) = {}_C a_i \eta_i^3 + {}_C b_i \eta_i^2 + {}_C c_i \eta_i + {}_C d_i \quad \text{with } i = 0, \dots, n-1 \quad (\text{G.8})$$

where ${}_C a_i$, ${}_C b_i$, ${}_C c_i$, and ${}_C d_i$ are its constant coefficients. The first two derivatives of ${}_C s_i(\eta_i(t))$

⁹⁷The evaluation of B-Splines of degree p with the well-known DE BOOR's algorithm [126] takes $\mathcal{O}(p^2) + \mathcal{O}(p)$ operations [89]. There are optimized versions of it as proposed by LEE in [271] and BÖHM in [89], however, these are numerically less stable [272]. In contrast, evaluating a polynomial of degree p with the method of HORNER and GILBERT [208, p. 308ff] takes only $2p$, i. e., $\mathcal{O}(p)$, operations [197, p. 94].

with respect to t are obtained by applying the chain rule:

$$c\dot{s}_i(\eta_i) = \left(\frac{d c s_i}{dt} \right) = \left(\frac{\partial c s_i}{\partial \eta_i} \right) \left(\frac{d \eta_i}{dt} \right) = \frac{3}{h_i} c a_i \eta_i^2 + \frac{2}{h_i} c b_i \eta_i + \frac{1}{h_i} c c_i, \quad (\text{G.9})$$

$$c\ddot{s}_i(\eta_i) = \left(\frac{d^2 c s_i}{dt^2} \right) = \left(\frac{\partial^2 c s_i}{\partial \eta_i^2} \right) \left(\frac{d \eta_i}{dt} \right)^2 + \underbrace{\left(\frac{\partial c s_i}{\partial \eta_i} \right) \left(\frac{d^2 \eta_i}{dt^2} \right)}_{=0} = \frac{6}{h_i^2} c a_i \eta_i + \frac{2}{h_i^2} c b_i. \quad (\text{G.10})$$

For quintic splines (left subscript Q), each segment ${}_Q s_i$ represents a polynomial of degree five,

$${}_Q s_i(\eta_i) = {}_Q a_i \eta_i^5 + {}_Q b_i \eta_i^4 + {}_Q c_i \eta_i^3 + {}_Q d_i \eta_i^2 + {}_Q e_i \eta_i + {}_Q f_i \quad \text{with } i = 0, \dots, n-1 \quad (\text{G.11})$$

where ${}_Q a_i$, ${}_Q b_i$, ${}_Q c_i$, ${}_Q d_i$, ${}_Q e_i$, and ${}_Q f_i$ are its constant coefficients. As in the cubic case, we obtain the first four derivatives of ${}_Q s_i(\eta_i(t))$ with respect to t through the chain rule:

$${}_Q \dot{s}_i(\eta_i) = \left(\frac{d {}_Q s_i}{dt} \right) = \frac{5}{h_i} {}_Q a_i \eta_i^4 + \frac{4}{h_i} {}_Q b_i \eta_i^3 + \frac{3}{h_i} {}_Q c_i \eta_i^2 + \frac{2}{h_i} {}_Q d_i \eta_i + \frac{1}{h_i} {}_Q e_i, \quad (\text{G.12})$$

$${}_Q \ddot{s}_i(\eta_i) = \left(\frac{d^2 {}_Q s_i}{dt^2} \right) = \frac{20}{h_i^2} {}_Q a_i \eta_i^3 + \frac{12}{h_i^2} {}_Q b_i \eta_i^2 + \frac{6}{h_i^2} {}_Q c_i \eta_i + \frac{2}{h_i^2} {}_Q d_i, \quad (\text{G.13})$$

$${}_Q s_i^{(3)}(\eta_i) = \left(\frac{d^3 {}_Q s_i}{dt^3} \right) = \frac{60}{h_i^3} {}_Q a_i \eta_i^2 + \frac{24}{h_i^3} {}_Q b_i \eta_i + \frac{6}{h_i^3} {}_Q c_i, \quad (\text{G.14})$$

$${}_Q s_i^{(4)}(\eta_i) = \left(\frac{d^4 {}_Q s_i}{dt^4} \right) = \frac{120}{h_i^4} {}_Q a_i \eta_i + \frac{24}{h_i^4} {}_Q b_i. \quad (\text{G.15})$$

G.2.3 Spline Interpolation: Preliminaries

In the following, we recall how a given set of $n + 1$ data points $\{(t_i, y_i)\}$ with $i = 0, \dots, n$ can be interpolated with a C^2 or C^4 smooth cubic or quintic spline, respectively. The derivation explicitly uses the PP form of the spline and leads to the same algorithm as presented by MUND et al. in [312], except for slight modifications in notation. Note that [312] only deals with quintic splines. However, the method for cubic segments presented in this thesis is a simplified version of the same scheme. Moreover, the derivation is investigated in more detail than it is in [312]. Readers not interested in these details are referred to Algorithm G.1 which summarizes the results of this section. In contrast to [312], we only consider the case of predefined first- and second-order derivatives at the boundaries of the quintic spline, i. e., we assume \dot{y}_0 , \ddot{y}_0 , \dot{y}_n , and \ddot{y}_n to be given (as indicated in Figure G.1). For the cubic counterpart, we lose two degrees of freedom, allowing us to predefine only two constraints out of $\{\dot{y}_0, \ddot{y}_0, \dot{y}_n, \ddot{y}_n\}$. For the remainder of this section, we restrict ourselves to the case of predefined second-order time derivatives \ddot{y}_0 and \ddot{y}_n as this allows an efficient algorithm similar to the one presented for quintic splines. Note that this choice includes the common case of natural cubic splines, i. e., $\ddot{y}_0 = \ddot{y}_n = 0$. For cubic splines, we postpone the enforcement of the remaining boundary conditions, i. e., \dot{y}_0 and \dot{y}_n , to the end of Appendix G.2.7. In the following, we only consider distinct and ascending interpolation sites, i. e., $t_0 \leq t_i < t_{i+1} \leq t_n$ for $i = 0, \dots, n-1$.

$${}_c\Lambda_i = \begin{cases} -h_0 \ddot{y}_0 - h_{n-1} \ddot{y}_n & \text{for } n = 2, & (\equiv -{}_cL_1 {}_cX_0 - {}_cU_{n-1} {}_cX_n) \\ -h_0 \ddot{y}_0 & \text{for } n > 2 \wedge i = 1, & (\equiv -{}_cL_1 {}_cX_0) \\ -h_{n-1} \ddot{y}_n & \text{for } n > 2 \wedge i = n - 1, & (\equiv -{}_cU_{n-1} {}_cX_n) \\ 0 & \text{else} \end{cases}. \quad (\text{G.25})$$

Herein ${}_cL_i$, ${}_cD_i$, and ${}_cU_i \in \mathbb{R}$ represent the lower diagonal, diagonal, and upper diagonal elements of ${}_c\mathbf{A}$, respectively. Note that ${}_c\mathbf{A}$ only depends on the choice of $\{t_i\}$, thus, it can be reused in case we need to perform further interpolations with the same segmentation $\{t_i\}$. While ${}_c\mathbf{A}$ represents the interpolation sites $\{t_i\}$, the interpolation points $\{y_i\}$ are encoded in ${}_c\mathbf{B}$. The additional term ${}_c\Lambda_i$ incorporates the BCs such that we can write the system in the neat form of Equation G.21. From the solution ${}_c\mathbf{X}$ we can directly obtain the unknowns \ddot{y}_i which in turn can be used together with the data points $\{(t_i, y_i)\}$ and BCs to compute the segment coefficients given in Equation G.18 such that the spline $y(t)$ is fully defined. Note that ${}_c\mathbf{A}$ is symmetric, i. e., ${}_c\mathbf{A} = {}_c\mathbf{A}^T$, however, we do not make use of this property.

Although one can compute ${}_c\mathbf{X}$ from Equation G.20 using an arbitrary solver for linear systems of equations, there is a more efficient way for doing so: since ${}_c\mathbf{A}$ is tridiagonal, we can solve Equation G.20 with the THOMAS algorithm [352, p. 93ff], [221, p. 55ff]. Derived from an LU decomposition of ${}_c\mathbf{A}$, one performs a recursive forward elimination

$${}_cH_i := \begin{cases} \frac{{}_cU_1}{{}_cD_1} & \text{for } i = 1, \\ \frac{{}_cU_i}{{}_cD_i - {}_cL_i {}_cH_{i-1}} & \text{for } i = 2, \dots, n - 2 \end{cases}, \quad (\text{G.26})$$

$${}_cP_i := \begin{cases} \frac{{}_cB_1}{{}_cD_1} & \text{for } i = 1, \\ \frac{{}_cB_i - {}_cL_i {}_cP_{i-1}}{{}_cD_i - {}_cL_i {}_cH_{i-1}} & \text{for } i = 2, \dots, n - 1 \end{cases} \quad (\text{G.27})$$

followed by a backward substitution

$${}_cX_i = \begin{cases} {}_cP_{n-1} & \text{for } i = n - 1, \\ {}_cP_i - {}_cH_i {}_cX_{i+1} & \text{for } i = n - 2, n - 3, \dots, 1 \end{cases}. \quad (\text{G.28})$$

Computing ${}_c\mathbf{X}$ out of ${}_c\mathbf{A}$ and ${}_c\mathbf{B}$ thus boils down to $5n - 9$ operations⁹⁸ in total [221, p. 57]. Since ${}_c\mathbf{A}$ is symmetric and positive definite, one may think of using an algorithm based on CHOLESKY factorization instead of LU decomposition, as this has proven to be approximately twice as efficient where applicable. However, this rule of thumb seems to be no longer valid for the special case of tridiagonal matrices: the CHOLESKY factorization $T = L D_L^{-1} L^T$ in MEURANT [305], where the computation of the lower diagonal matrix L exploits the special structure and the diagonal matrix D_L is used to avoid evaluating expensive square roots, leads to $7n - 10$ operations in total.

Numerical Stability The THOMAS algorithm is guaranteed to be numerically stable if ${}_c\mathbf{A}$ is diagonally dominant, i. e., if

$$|{}_cD_i| = |2(h_{i-1} + h_i)| > |{}_cL_i| + |{}_cU_i| = |h_{i-1}| + |h_i| \quad \text{for } i = 1, \dots, n - 1 \quad (\text{G.29})$$

⁹⁸Note that in contrast to ISAACSON and KELLER [221] where $A \in \mathbb{R}^{n \times n}$ in our case ${}_c\mathbf{A} \in \mathbb{R}^{(n-1) \times (n-1)}$, thus, n changes to $n - 1$ for computing the count of operations.

holds [352, p. 94], whereas for $i = 1$ and $i = n-1$ we use ${}_C L_1 = 0$ and ${}_C U_{n-1} = 0$, respectively. As is easily verified with Equation G.29, this holds true for any choice of $h_{i-1} > 0$ and $h_i > 0$ (i. e., for distinct and ascending interpolation sites), thus, the presented method is always stable.

G.2.5 Quintic Spline Interpolation: Derivation

As previously mentioned, the following is a detailed version of the derivation given by MUND et al. in [312]. Just as in the cubic case, our task is to pass through the given data points $\{(t_i, y_i)\}$, thus we enforce the interpolation constraints

$$\begin{aligned} {}_Q s_i(\eta_i) \Big|_{\eta_i=0} &\stackrel{!}{=} y_i & \text{for } i = 0, \dots, n-1, & \quad (n \text{ equations}) \\ {}_Q s_i(\eta_i) \Big|_{\eta_i=1} &\stackrel{!}{=} y_{i+1} & \text{for } i = 0, \dots, n-1. & \quad (n \text{ equations}) \end{aligned} \quad (\text{G.30})$$

We use the additional degrees of freedom to enforce not only C^2 -, but instead C^4 - continuity with

$$\begin{aligned} {}_Q \dot{s}_i(\eta_i) \Big|_{\eta_i=1} &\stackrel{!}{=} {}_Q \dot{s}_{i+1}(\eta_{i+1}) \Big|_{\eta_{i+1}=0} = \dot{y}_{i+1} & \text{for } i = 0, \dots, n-2, & \quad (n-1 \text{ equations}) \\ {}_Q \ddot{s}_i(\eta_i) \Big|_{\eta_i=1} &\stackrel{!}{=} {}_Q \ddot{s}_{i+1}(\eta_{i+1}) \Big|_{\eta_{i+1}=0} = \ddot{y}_{i+1} & \text{for } i = 0, \dots, n-2, & \quad (n-1 \text{ equations}) \\ {}_Q s_i^{(3)}(\eta_i) \Big|_{\eta_i=1} &\stackrel{!}{=} {}_Q s_{i+1}^{(3)}(\eta_{i+1}) \Big|_{\eta_{i+1}=0} = y_{i+1}^{(3)} & \text{for } i = 0, \dots, n-2, & \quad (n-1 \text{ equations}) \\ {}_Q s_i^{(4)}(\eta_i) \Big|_{\eta_i=1} &\stackrel{!}{=} {}_Q s_{i+1}^{(4)}(\eta_{i+1}) \Big|_{\eta_{i+1}=0} = y_{i+1}^{(4)} & \text{for } i = 0, \dots, n-2. & \quad (n-1 \text{ equations}) \end{aligned} \quad (\text{G.31})$$

Inserting Equations G.11 to G.13 into Equation G.30 and into the first two rows of Equation G.31 allows us to reformulate the spline coefficients to

$$\begin{aligned} {}_Q a_i &= -6 y_i + 6 y_{i+1} - 3 \dot{y}_i h_i - 3 \dot{y}_{i+1} h_i - \frac{1}{2} \ddot{y}_i h_i^2 + \frac{1}{2} \ddot{y}_{i+1} h_i^2, \\ {}_Q b_i &= +15 y_i - 15 y_{i+1} + 8 \dot{y}_i h_i + 7 \dot{y}_{i+1} h_i + \frac{3}{2} \ddot{y}_i h_i^2 - \ddot{y}_{i+1} h_i^2, \\ {}_Q c_i &= -10 y_i + 10 y_{i+1} - 6 \dot{y}_i h_i - 4 \dot{y}_{i+1} h_i - \frac{3}{2} \ddot{y}_i h_i^2 + \frac{1}{2} \ddot{y}_{i+1} h_i^2, \\ {}_Q d_i &= +\frac{1}{2} \ddot{y}_i h_i^2, \\ {}_Q e_i &= +\dot{y}_i h_i, \\ {}_Q f_i &= +y_i, \end{aligned} \quad (\text{G.32})$$

where \dot{y}_i and \ddot{y}_i for $i = 1, \dots, n-1$ are the $2(n-1)$ unknowns which we still have to determine using the remaining $2(n-1)$ equations given by the last two rows of Equation G.31. In particular, we expand the fourth row of Equation G.31 with Equation G.15 and insert ${}_Q a_i$, ${}_Q b_i$, and ${}_Q b_{i+1}$ from Equation G.32 to obtain

$$\begin{aligned} -56 g_{i-1}^3 \dot{y}_{i-1} - 8 g_{i-1}^2 \ddot{y}_{i-1} - 64 \dot{y}_i (g_{i-1}^3 + g_i^3) + 12 \ddot{y}_i (g_{i-1}^2 - g_i^2) - 56 g_i^3 \dot{y}_{i+1} + 8 g_i^2 \ddot{y}_{i+1} &= \\ = -120 g_i^4 (y_{i+1} - y_i) - 120 g_{i-1}^4 (y_i - y_{i-1}). & \end{aligned} \quad (\text{G.33})$$

In the same manner, we expand the third row of Equation G.31 with Equation G.14 and insert ${}_Q a_i$, ${}_Q b_i$, ${}_Q c_i$ and ${}_Q c_{i+1}$ from Equation G.32 which leads to

$$\begin{aligned} -8 g_{i-1}^2 \dot{y}_{i-1} - g_{i-1} \ddot{y}_{i-1} - 12 \dot{y}_i (g_{i-1}^2 - g_i^2) + 3 \ddot{y}_i (g_{i-1} + g_i) + 8 g_i^2 \dot{y}_{i+1} - g_i \ddot{y}_{i+1} &= \\ = 20 g_i^3 (y_{i+1} - y_i) - 20 g_{i-1}^3 (y_i - y_{i-1}). & \end{aligned} \quad (\text{G.34})$$

case, the solution ${}_Q\mathbf{X}$ represents not only \dot{y}_i , but also y_i , which is now additionally required to compute the segment coefficients from Equation G.32.

Since ${}_Q\mathbf{A}$ is block-tridiagonal, we can again solve Equation G.35 efficiently with the generalization of the THOMAS algorithm to block-tridiagonal matrices [221, p. 58ff]. Based on an LU decomposition of ${}_Q\mathbf{A}$, we first run a recursive forward elimination

$${}_QH_i := \begin{cases} {}_Q\mathbf{D}_1^{-1} {}_QU_1 & \text{for } i = 1, \\ ({}_Q\mathbf{D}_i - {}_QL_i {}_QH_{i-1})^{-1} {}_QU_i & \text{for } i = 2, \dots, n-2 \end{cases}, \quad (\text{G.42})$$

$${}_QP_i := \begin{cases} {}_Q\mathbf{D}_1^{-1} {}_QB_1 & \text{for } i = 1, \\ ({}_Q\mathbf{D}_i - {}_QL_i {}_QH_{i-1})^{-1} ({}_QB_i - {}_QL_i {}_QP_{i-1}) & \text{for } i = 2, \dots, n-1 \end{cases} \quad (\text{G.43})$$

followed by a backward substitution

$${}_Q\mathbf{X}_i = \begin{cases} {}_QP_{n-1} & \text{for } i = n-1, \\ {}_QP_i - {}_QH_i {}_Q\mathbf{X}_{i+1} & \text{for } i = n-2, n-3, \dots, 1 \end{cases}. \quad (\text{G.44})$$

Computing ${}_Q\mathbf{X}$ out of ${}_Q\mathbf{A}$ and ${}_Q\mathbf{B}$ requires at a maximum $36n - 60$ operations⁹⁹ in total [221, p. 60]. For this upper bound, explicit computation of the inverse of ${}_Q\mathbf{D}_1$ and $({}_Q\mathbf{D}_i - {}_QL_i {}_QH_{i-1})$ by Gaussian elimination is assumed, which in practice should be avoided by solving a 2×2 LSE instead [221]. Thus, a corresponding implementation can be expected to require even less operations.

Numerical Stability The presented scheme for solving block-tridiagonal systems is a special form of block-Gaussian elimination without pivoting and is guaranteed to be numerically stable if ${}_Q\mathbf{A}$ is block-diagonally dominant, i. e., if

$$\Gamma_i := \|\|{}_Q\mathbf{D}_i^{-1}\| (\|\|{}_QL_i\| + \|\|{}_QU_i\|\|) \leq 1 \quad \text{for } i = 1, \dots, n-1 \quad (\text{G.45})$$

holds for an arbitrary matrix norm $\|\cdot\|$ (see VARAH [425]). Note that for $i = 1$ and $i = n-1$ we set ${}_QL_1 = \mathbf{0}$ and ${}_QU_{n-1} = \mathbf{0}$, respectively. In order to verify Equation G.45 for ${}_Q\mathbf{D}_i$, ${}_QL_i$, and ${}_QU_i$ as specified in Equations G.37 and G.38, we assume a constant ratio $\omega = g_i/g_{i-1} = g_{i+1}/g_i$ to obtain

$${}_Q\mathbf{D}_i = g_i^3 \underbrace{\begin{bmatrix} 64\left(\frac{1}{\omega^3} + 1\right) & 12\kappa\left(\frac{1}{\omega^2} - 1\right) \\ 12\kappa\left(\frac{1}{\omega^2} - 1\right) & 3\kappa^2\left(\frac{1}{\omega} + 1\right) \end{bmatrix}}_{=: {}_Q\hat{\mathbf{D}}_i(\omega)}, \quad {}_QL_i = g_i^3 \underbrace{\begin{bmatrix} 56\frac{1}{\omega^3} & -8\kappa\frac{1}{\omega^3} \\ 8\kappa\frac{1}{\omega^2} & -\kappa^2\frac{1}{\omega^2} \end{bmatrix}}_{=: {}_Q\hat{\mathbf{L}}_i(\omega)}, \quad {}_QU_i = g_i^3 \underbrace{\begin{bmatrix} 56 & 8\kappa\omega \\ -8\kappa & -\kappa^2\omega \end{bmatrix}}_{=: {}_Q\hat{\mathbf{U}}_i(\omega)}$$

and further

$$\Gamma_i(\omega) = \|\|{}_Q\mathbf{D}_i^{-1}(\omega)\| (\|\|{}_QL_i(\omega)\| + \|\|{}_QU_i(\omega)\|\|) = \|\|{}_Q\hat{\mathbf{D}}_i^{-1}(\omega)\| (\|\|{}_Q\hat{\mathbf{L}}_i(\omega)\| + \|\|{}_Q\hat{\mathbf{U}}_i(\omega)\|\|) \quad (\text{G.46})$$

which shows that Γ_i does not depend on g_i for a constant ratio ω . It can be easily verified through numerical evaluation of Equation G.46 that a homogeneous partitioning of the spline, i. e., $\omega = 1$, results in the lowest value for Γ_i . Using the spectral matrix norm and the special choice of κ given in Equation G.41 leads to

$$\Gamma_i(\omega = 1) \approx 1.24 \not\leq 1 \quad \text{for } i = 2, \dots, n-2. \quad (\text{G.47})$$

⁹⁹Again, ISAACSON and KELLER [221] use $A \in \mathbb{R}^{2n \times 2n}$ while in our case ${}_Q\mathbf{A} \in \mathbb{R}^{2(n-1) \times 2(n-1)}$ holds, thus, n changes to $n-1$ for computing the count of operations.

Unfortunately, the condition from Equation G.45 is violated, even for the ideal case of a homogeneously partitioned spline. However, Equation G.45 represents a sufficient but not necessary condition for numerical stability, thus, we can still use Γ_i as a measure for the pivotal growth (see VARAH [425]) which should be minimized. In doing so, we can conclude that for best numerical stability one should use a “reasonable” ratio ω which is close to 1. Note that the special choice of κ in Equation G.41 has been suggested by MUND et al. in [312] to minimize Γ_i and thus optimize numerical stability. This intention becomes clear when observing that

$$\|_{\mathcal{Q}}\hat{\mathbf{D}}_i^{-1}(\omega = 1)\|_2 = \sqrt{\lambda_{\max}(\mathcal{Q}\hat{\mathbf{D}}_i^{-T}(\omega = 1)\mathcal{Q}\hat{\mathbf{D}}_i^{-1}(\omega = 1))} = \begin{cases} \frac{1}{6\kappa^2} > \frac{1}{128} & \text{for } \kappa^2 < \frac{64}{3}, \\ \frac{1}{128} & \text{for } \kappa^2 \geq \frac{64}{3} \end{cases},$$

where $\lambda_{\max}(\dots)$ denotes the maximum eigenvalue of a given matrix. Since both $\|_{\mathcal{Q}}\hat{\mathbf{L}}_i(\omega)\|_2$ and $\|_{\mathcal{Q}}\hat{\mathbf{U}}_i(\omega)\|_2$, increase with growing $|\kappa|$, the optimum from Equation G.41 is chosen. This finding can be easily verified through numerical investigation. Note that numerical stability only depends on $|\kappa|$, thus, one could also choose the negative form $\kappa = -\sqrt{64/3}$.

G.2.6 Algorithm for Cubic/Quintic Spline Interpolation

Since the presented derivation is rather lengthy, the key steps for interpolating cubic/quintic splines following the proposed method are summarized in Algorithm G.1.

Algorithm G.1: Cubic/Quintic Spline Interpolation. See the class `PolynomialSpline` of the module `curve` of `Broccoli` for a reference implementation.

Input: Interpolation parameters consisting of

- $n + 1$ distinct and ascending interpolation sites $\{t_i\}$,
- their corresponding data values $\{y_i\}$,
- BCs $\{\ddot{y}_0, \ddot{y}_n\}$ (cubic) or $\{\dot{y}_0, \ddot{y}_0, \dot{y}_n, \ddot{y}_n\}$ (quintic) at t_0 and t_n

where $i = 0, \dots, n$ and $n > 1$.

Output: Cubic/quintic spline combining n interconnected segments in PP form, which

- interpolates the given data values at all spline knots $\{t_0, \dots, t_n\}$,
- is \mathcal{C}^2 (cubic) or \mathcal{C}^4 (quintic) continuous at interior spline knots $\{t_1, \dots, t_{n-1}\}$,
- satisfies the BCs.

begin

Cubic: Setup ${}_cL_i$, ${}_cD_i$, ${}_cU_i$, and ${}_cB_i$ from (G.22), (G.23), (G.24), and (G.25).

Quintic: Setup ${}_qL_i$, ${}_qD_i$, ${}_qU_i$, and ${}_qB_i$ from (G.37), (G.38), (G.39), and (G.40).

Cubic: Compute ${}_cX_i$ using the scheme given in (G.26), (G.27), and (G.28).

Quintic: Compute ${}_qX_i$ using the scheme given in (G.42), (G.43), and (G.44).

Cubic: Extract \dot{y}_i from ${}_cX_i$ and compute the segment coefficients ${}_c a_i$, ${}_c b_i$, ${}_c c_i$, and ${}_c d_i$ using (G.18).

Quintic: Extract \dot{y}_i , \ddot{y}_i from ${}_qX_i$ and compute the segment coefficients ${}_q a_i$, ${}_q b_i$, ${}_q c_i$, ${}_q d_i$, ${}_q e_i$, and ${}_q f_i$ using (G.32).

end

For details on convergence order and approximation error of quintic spline interpolation, the interested reader is referred to [312] where these issues have been experimentally investigated for various examples.

G.2.7 Spline Collocation: Derivation

The following is based on the collocation algorithm presented by BUSCHMANN in [98, 100]. However, we extend the method from cubic to quintic splines. Moreover, we do not use natural splines, but instead integrate the BCs directly into the scheme. Lastly, in contrast to [98, 100], we do not need to modify the right-hand side of Equation G.1, thus, leading to a “true” collocation of the ODE for all collocation sites, which are chosen to be the interior spline knots. As runtime performance is of the highest priority for our application, we choose smoothest spline collocation. This minimizes the count of (expensive) collocation sites, thus, reduces the count of equations to solve, and instead uses the available degrees of freedom to force C^2 (cubic spline) or C^4 (quintic spline) continuity. Moreover, in our application, $y(t)$ is used as input for controlling the motion of a robot, thus, a smooth $y(t)$ is equivalent to small changes in joint accelerations, i. e., motor jerks, which in turn improves overall stability during locomotion.

As stated in Appendix G.2.1, we require the approximation $y(t)$ to fulfill the underlying ODE at certain collocation sites $\{t_k\}$, see Equation G.4, while simultaneously satisfying the BCs as specified in Equation G.5. Note that we use the index k instead of i to highlight that our new task consists in collocating the ODE at the interior knots, i. e., $k = 1, \dots, n-1$ rather than the previously investigated interpolation at all knots, i. e., $i = 0, \dots, n$. Furthermore, it should be pointed out that although Equation G.4 holds, this does not imply that $y(t_k) = F(t_k)$, $\dot{y}(t_k) = \dot{F}(t_k)$, or $\ddot{y}(t_k) = \ddot{F}(t_k)$. In other words, $y(t)$ will not coincide with the real solution $F(t)$ at the collocation sites $\{t_k\}$. However, it will behave similarly at these spots (meaning that they will satisfy the same Equation G.1), which is illustrated in Figure G.2.

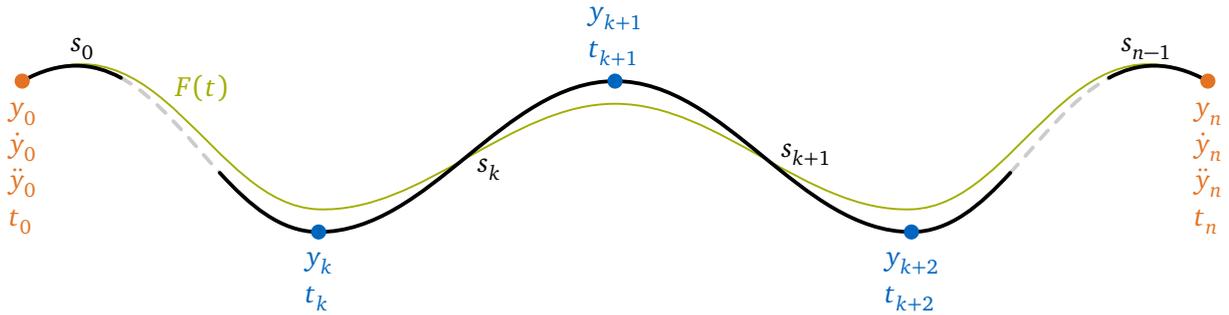


Figure G.2: The computed spline $y(t)$ (black) approximating the real solution $F(t)$ (green). The approximation satisfies the underlying ODE at the specified collocation sites $\{t_k\}$ (blue), and fulfills the BCs at t_0 and t_n (orange), but does not necessarily coincide at the collocation points.

As first step, we introduce the auxiliary variables λ , ξ , and \mathbf{r} which are defined as

$$\begin{aligned} {}_c\lambda &:= [y_1, \dots, y_{n-1}]^T, & {}_c\xi &:= [\ddot{y}_1, \dots, \ddot{y}_{n-1}]^T, & {}_c\mathbf{r} &:= [y_0, \ddot{y}_0, y_n, \ddot{y}_n]^T, \\ {}_q\lambda &:= [y_1, \dots, y_{n-1}]^T, & {}_q\xi &:= [\dot{y}_1, \ddot{y}_1, \dots, \dot{y}_{n-1}, \ddot{y}_{n-1}]^T, & {}_q\mathbf{r} &:= [y_0, \dot{y}_0, \ddot{y}_0, y_n, \dot{y}_n, \ddot{y}_n]^T, \end{aligned} \quad (\text{G.48})$$

where ${}_c\lambda$, ${}_c\xi$, ${}_c\mathbf{r}$ and ${}_q\lambda$, ${}_q\xi$, ${}_q\mathbf{r}$ are the corresponding counterparts for the case of cubic and quintic splines, respectively. While λ represents the (yet unknown) collocation points $\{y_k\}$, ξ contains their corresponding first (and second) order time derivatives, which can be seen as “internal” unknowns, as they will be implicitly defined through an embedded spline interpolation. Lastly, \mathbf{r} depicts the BCs, where we lack \dot{y}_0 and \dot{y}_n in the case of cubic splines as has been previously explained. From Equations G.18, G.32 and G.48 we observe that the spline segments s_i are linear with respect to λ , ξ , and \mathbf{r} , i. e.,

$$s_i(\eta_i) = \underbrace{\left(\frac{\partial s_i(\eta_i)}{\partial \lambda} \right)}_{\text{known}} \lambda + \underbrace{\left(\frac{\partial s_i(\eta_i)}{\partial \xi} \right)}_{\text{known}} \xi + \underbrace{\left(\frac{\partial s_i(\eta_i)}{\partial \mathbf{r}} \right)}_{\text{known}} \mathbf{r} \quad \text{for } i = 0, \dots, n-1 \quad (\text{G.49})$$

holds. The gradients are fully defined by the spline partitioning $\{t_i\}$, which is assumed to be known. Thus, the construction of the spline $y(t)$ is equivalent to the search for a corresponding λ and ξ . Note that to obtain Equation G.49, we used Equations G.18 and G.32, which in turn were derived from fulfilling the interpolation condition together with enforcing continuity of the second time derivative (cubic spline), or first and second time derivative (quintic spline) at the interior knots. In order to accomplish full C^2 - and C^4 -continuity, we further make use of $\mathbf{A}\mathbf{X} = \mathbf{B}$ from Equations G.20 and G.35, which represents continuity of the first time derivative (cubic spline) or third and fourth time derivative (quintic spline), respectively. In particular, we observe from Equations G.24, G.25, G.39 and G.40, that \mathbf{B} is linear with respect to λ and \mathbf{r} , thus

$$\mathbf{A}\mathbf{X} = \mathbf{B} = \underbrace{\left(\frac{\partial \mathbf{B}}{\partial \lambda}\right)}_{\text{known}} \lambda + \underbrace{\left(\frac{\partial \mathbf{B}}{\partial \mathbf{r}}\right)}_{\text{known}} \mathbf{r} \quad (\text{G.50})$$

holds, where the gradients again depend only on the known partitioning $\{t_i\}$. We further observe that, according to the definitions in Equations G.22, G.37 and G.48, we can write the mapping

$$\mathbf{X} = \mathbf{S} \xi \quad \text{with } \mathbf{c}\mathbf{S} := \mathbf{1} \quad \text{and} \quad \mathbf{Q}\mathbf{S} := \text{diag}(\mathbf{Q}\mathbf{S}_1, \dots, \mathbf{Q}\mathbf{S}_{n-1}) \quad \text{where } \mathbf{Q}\mathbf{S}_i := \text{diag}\left(-1, \frac{1}{\kappa g_i}\right) \quad (\text{G.51})$$

with $\mathbf{1}$ being the identity matrix of appropriate size. Since \mathbf{A} and \mathbf{S} do not depend on the yet unknown λ or ξ and are assumed to be non-singular, it is clear from Equation G.50 that not only \mathbf{B} , but also \mathbf{X} and thus ξ are linear with respect to λ and \mathbf{r} . Hence, one can write

$$\mathbf{X} = \left(\frac{\partial \mathbf{X}}{\partial \lambda}\right) \lambda + \left(\frac{\partial \mathbf{X}}{\partial \mathbf{r}}\right) \mathbf{r} \quad \text{and} \quad \xi = \left(\frac{\partial \xi}{\partial \lambda}\right) \lambda + \left(\frac{\partial \xi}{\partial \mathbf{r}}\right) \mathbf{r}. \quad (\text{G.52})$$

Note that \mathbf{A} and $\mathbf{Q}\mathbf{S}$ only depend on the known $\{t_i\}$, which allows us to safely differentiate Equation G.50 with respect to λ and \mathbf{r} to obtain

$$\mathbf{A} \underbrace{\left(\frac{\partial \mathbf{X}}{\partial \lambda}\right)}_{\mathbf{S} \left(\frac{\partial \xi}{\partial \lambda}\right)} = \underbrace{\left(\frac{\partial \mathbf{B}}{\partial \lambda}\right)}_{\text{known}} \quad \text{and} \quad \mathbf{A} \underbrace{\left(\frac{\partial \mathbf{X}}{\partial \mathbf{r}}\right)}_{\mathbf{S} \left(\frac{\partial \xi}{\partial \mathbf{r}}\right)} = \underbrace{\left(\frac{\partial \mathbf{B}}{\partial \mathbf{r}}\right)}_{\text{known}}. \quad (\text{G.53})$$

which we can use to compute the yet unknown gradients in Equation G.52. Note that this can be done very efficiently due to the (block-)tridiagonal form of \mathbf{A} . Since \mathbf{S} is diagonal, this property also holds for the product $\mathbf{A}\mathbf{S}$. However, for best numerical stability, one should solve for the gradients of \mathbf{X} first and use the mapping from Equation G.51 to obtain the gradients of ξ afterwards, which is of negligible cost since \mathbf{S} , and thus also \mathbf{S}^{-1} , is diagonal. The right-hand sides necessary to solve Equation G.53 only depend on $\{t_i\}$ and are derived in Appendix G.6. Lastly, we insert ξ from Equation G.52 into Equation G.49 and obtain

$$s_i(\eta_i) = \left[\left(\frac{\partial s_i(\eta_i)}{\partial \lambda}\right) + \left(\frac{\partial s_i(\eta_i)}{\partial \xi}\right) \left(\frac{\partial \xi}{\partial \lambda}\right) \right] \lambda + \left[\left(\frac{\partial s_i(\eta_i)}{\partial \mathbf{r}}\right) + \left(\frac{\partial s_i(\eta_i)}{\partial \xi}\right) \left(\frac{\partial \xi}{\partial \mathbf{r}}\right) \right] \mathbf{r} \quad (\text{G.54})$$

or equivalently

$$s_i(\eta_i) = \nabla_{\lambda} s_i(\eta_i) \lambda + \nabla_{\mathbf{r}} s_i(\eta_i) \mathbf{r} \quad (\text{G.55})$$

with the known spline gradients

$$\nabla_{\lambda} s_i(\eta_i) := \left[\left(\frac{\partial s_i(\eta_i)}{\partial \lambda}\right) + \left(\frac{\partial s_i(\eta_i)}{\partial \xi}\right) \left(\frac{\partial \xi}{\partial \lambda}\right) \right], \quad \nabla_{\mathbf{r}} s_i(\eta_i) := \left[\left(\frac{\partial s_i(\eta_i)}{\partial \mathbf{r}}\right) + \left(\frac{\partial s_i(\eta_i)}{\partial \xi}\right) \left(\frac{\partial \xi}{\partial \mathbf{r}}\right) \right]. \quad (\text{G.56})$$

We can obtain the corresponding expressions for the first and second time derivatives by following the exact same scheme. In particular, we get

$$\dot{s}_i(\eta_i) = \nabla_{\lambda} \dot{s}_i(\eta_i) \boldsymbol{\lambda} + \nabla_{\mathbf{r}} \dot{s}_i(\eta_i) \mathbf{r}, \quad \ddot{s}_i(\eta_i) = \nabla_{\lambda} \ddot{s}_i(\eta_i) \boldsymbol{\lambda} + \nabla_{\mathbf{r}} \ddot{s}_i(\eta_i) \mathbf{r} \quad (\text{G.57})$$

with

$$\nabla_{\lambda} \dot{s}_i(\eta_i) := \left[\left(\frac{\partial \dot{s}_i(\eta_i)}{\partial \boldsymbol{\lambda}} \right) + \left(\frac{\partial \dot{s}_i(\eta_i)}{\partial \boldsymbol{\xi}} \right) \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\lambda}} \right) \right], \quad \nabla_{\mathbf{r}} \dot{s}_i(\eta_i) := \left[\left(\frac{\partial \dot{s}_i(\eta_i)}{\partial \mathbf{r}} \right) + \left(\frac{\partial \dot{s}_i(\eta_i)}{\partial \boldsymbol{\xi}} \right) \left(\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{r}} \right) \right], \quad (\text{G.58})$$

$$\nabla_{\lambda} \ddot{s}_i(\eta_i) := \left[\left(\frac{\partial \ddot{s}_i(\eta_i)}{\partial \boldsymbol{\lambda}} \right) + \left(\frac{\partial \ddot{s}_i(\eta_i)}{\partial \boldsymbol{\xi}} \right) \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\lambda}} \right) \right], \quad \nabla_{\mathbf{r}} \ddot{s}_i(\eta_i) := \left[\left(\frac{\partial \ddot{s}_i(\eta_i)}{\partial \mathbf{r}} \right) + \left(\frac{\partial \ddot{s}_i(\eta_i)}{\partial \boldsymbol{\xi}} \right) \left(\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{r}} \right) \right]. \quad (\text{G.59})$$

Although computing the spline gradients can be done very efficiently, their mathematical representation is rather lengthy. A formulation of the gradients, which is ready for implementation, is given in Appendix G.6. Lastly, we fulfill the dynamics of the ODE by inserting Equations G.55 and G.57 into Equation G.4, which leads to

$$\alpha(t_k) \ddot{s}_k(\eta_k = 0) + \beta(t_k) \dot{s}_k(\eta_k = 0) + \gamma(t_k) s_k(\eta_k = 0) = \tau(t_k) \quad (\text{G.60})$$

for $t_0 < t_k < t_n$ and $t_k < t_{k+1}$ with $k = 1, \dots, n-1$. This can be formulated as LSE

$$\mathbf{A}_{\text{coll}} \boldsymbol{\lambda} = \mathbf{B}_{\text{coll}} \quad (\text{G.61})$$

with

$$\mathbf{A}_{\text{coll}} = \begin{bmatrix} \mathbf{A}_{\text{coll},1} \\ \vdots \\ \mathbf{A}_{\text{coll},k} \\ \vdots \\ \mathbf{A}_{\text{coll},n-1} \end{bmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}, \quad \mathbf{B}_{\text{coll}} = \begin{bmatrix} \mathbf{B}_{\text{coll},1} \\ \vdots \\ \mathbf{B}_{\text{coll},k} \\ \vdots \\ \mathbf{B}_{\text{coll},n-1} \end{bmatrix} \in \mathbb{R}^{(n-1)} \quad (\text{G.62})$$

where the k -th row of \mathbf{A}_{coll} and \mathbf{B}_{coll} corresponds to the collocation site t_k and is given by

$$\mathbf{A}_{\text{coll},k} = \alpha(t_k) \nabla_{\lambda} \ddot{s}_k(0) + \beta(t_k) \nabla_{\lambda} \dot{s}_k(0) + \gamma(t_k) \nabla_{\lambda} s_k(0) \in \mathbb{R}^{1 \times (n-1)}, \quad (\text{G.63})$$

$$\mathbf{B}_{\text{coll},k} = \tau(t_k) - [\alpha(t_k) \nabla_{\mathbf{r}} \ddot{s}_k(0) + \beta(t_k) \nabla_{\mathbf{r}} \dot{s}_k(0) + \gamma(t_k) \nabla_{\mathbf{r}} s_k(0)] \mathbf{r} \in \mathbb{R}. \quad (\text{G.64})$$

Note that we choose a partitioning of the spline such that the collocation sites coincide with the starting (“left”) knot of each segment, i. e., $\eta_k = 0$, (see Figure G.2). This allows us to skip the computation of certain spline gradients¹⁰⁰, which simplifies the implementation and improves the overall performance. Solving Equation G.61 for $\boldsymbol{\lambda}$ represents the key operation (i. e., the bottleneck for large n) of the proposed collocation method, since \mathbf{A}_{coll} is in general dense while all other operations are either simple explicit expressions or linear systems in (block-)tridiagonal form, which can be solved efficiently. This justifies our strategy to minimize the count of collocation points and instead force high order continuity.

As soon as $\boldsymbol{\lambda}$ has been obtained, we can compute $\boldsymbol{\xi}$ directly from Equation G.52, since the gradients of $\boldsymbol{\xi}$ with respect to $\boldsymbol{\lambda}$ and \mathbf{r} are already available as by-products of computing $\boldsymbol{\lambda}$, see Equation G.53. From $\boldsymbol{\lambda}$, $\boldsymbol{\xi}$, and \mathbf{r} the segment coefficients can finally be computed using Equation G.18 or Equation G.32.

¹⁰⁰Since the underlying ODE is of order two, only gradients of the last three coefficients, i. e., ${}_c b_i$, ${}_c c_i$, ${}_c d_i$ (cubic) or ${}_q d_i$, ${}_q e_i$, ${}_q f_i$ (quintic), have to be computed. For details see Appendix G.6.

G.2.8 Satisfying First Order Boundary Conditions for Cubic Splines

The presented method for cubic spline collocation respects y_0 , \dot{y}_0 , y_n , and \dot{y}_n as BCs. However, our task was to fulfill all BCs given in Equation G.5, which seems at first to be only possible with quintic spline collocation. Also satisfying the first-order BCs, i. e., \dot{y}_0 and \dot{y}_n , can be achieved with moderate effort. For that purpose we insert two auxiliary knots, the so-called virtual control-points $t_{\text{virt},1}$ and $t_{\text{virt},2}$, which give us the necessary degrees of freedom to introduce additional constraints. The term “virtual” highlights that these points are not used as collocation sites. Both, $t_{\text{virt},1}$ and $t_{\text{virt},2}$, have to lie within the specified start- and endtime and must not coincide with the collocation sites. This way, the spline remains properly partitioned. For simplicity, we place the virtual control-points at the centers of the (originally) first and last segments, i. e.,

$$t_{\text{virt},1} := \frac{t_{\text{orig},0} + t_{\text{orig},1}}{2} \quad \text{and} \quad t_{\text{virt},2} := \frac{t_{\text{orig},n-1} + t_{\text{orig},n}}{2}. \quad (\text{G.65})$$

Obviously, inserting two knots leads to a different segmentation of the spline, i. e., $n \rightarrow n + 2$, however, the boundaries and collocation sites remain unchanged. If we adapt the indexing such that $t_1 := t_{\text{virt},1}$ and $t_{n-1} := t_{\text{virt},2}$, all findings derived so far are also valid for this case. The only difference is that we do not force $y(t)$ to fulfill the underlying ODE at t_1 and t_{n-1} anymore. Instead, we satisfy the BCs \dot{y}_0 and \dot{y}_n by replacing the first and last row of \mathbf{A}_{coll} and \mathbf{B}_{coll} with

$$\begin{aligned} \mathbf{A}_{\text{coll},1} &= \nabla_{\lambda} \dot{s}_0(\eta_0 = 0), & \mathbf{B}_{\text{coll},1} &= \dot{y}_0 - \nabla_r \dot{s}_0(\eta_0 = 0) \mathbf{r}, \\ \mathbf{A}_{\text{coll},n-1} &= \nabla_{\lambda} \dot{s}_{n-1}(\eta_{n-1} = 1), & \mathbf{B}_{\text{coll},n-1} &= \dot{y}_n - \nabla_r \dot{s}_{n-1}(\eta_{n-1} = 1) \mathbf{r}. \end{aligned} \quad (\text{G.66})$$

In this way, the resulting cubic spline fulfills all BCs given in Equation G.5, satisfies the underlying ODE at the specified collocation sites, and is \mathcal{C}^2 -continuous at the interior spline knots (which includes $t_{\text{virt},1}$ and $t_{\text{virt},2}$).

Note that in Equation G.66, we evaluate the last spline segment at $\eta_{n-1} = 1$, thus, skipping certain spline gradients is not a possibility anymore. However, the additional computational cost is negligible when compared to solving Equation G.61.

G.2.9 Algorithm for Cubic/Quintic Spline Collocation

We summarize our findings in Algorithm G.2. Note that, for the case of cubic splines, the modification necessary to satisfy first-order BCs is already integrated.

G.3 Implementation

Algorithms G.1 and G.2 provide detailed descriptions of the presented interpolation and collocation methods which can be used as a reference during implementation. However, the author of this thesis also published a fully documented C++ implementation as part¹⁰¹ of *Broccoli*. Aside from basic matrix and vector operations, the *Broccoli* implementation uses *Eigen* to solve dense linear systems of equations such as $\mathbf{A}_{\text{coll}}^{-1} \mathbf{B}_{\text{coll}}$ in Equation G.61, but also ${}_{\mathcal{Q}}\mathbf{D}_1^{-1}(\cdot)$ and $({}_{\mathcal{Q}}\mathbf{D}_i - {}_{\mathcal{Q}}\mathbf{L}_i {}_{\mathcal{Q}}\mathbf{H}_{i-1})^{-1}(\cdot)$ in Equations G.42 and G.43. In particular, we use a Householder rank-revealing QR decomposition with column-pivoting (in particular `ColPivHouseholderQR` from *Eigen*) since it provides the best trade-off between speed, accuracy, and robustness for our use

¹⁰¹See the class `PolynomialSpline` of the module `curve` of *Broccoli* for cubic/quintic spline interpolation and the classes `Cubic|QuinticSplineCollocator` of the module `ode` of *Broccoli* for cubic/quintic spline collocation.

Algorithm G.2: Cubic/Quintic Spline Collocation. See the classes `Cubic|QuinticSplineCollocator` of the module `ode` of *Broccoli* for a reference implementation.

Input: Collocation parameters consisting of

- starttime t_0 and endtime t_n with $t_0 < t_n$ defining the boundaries of the spline,
- $n - 1$ distinct and ascending collocation sites $\{t_k\}$ with $t_0 < t_k < t_n$,
- coefficients $\alpha(t_k)$, $\beta(t_k)$, $\gamma(t_k)$, and right-hand side $\tau(t_k)$ of the underlying ODE at the collocation sites,
- BCs $\{y_0, \dot{y}_0, \ddot{y}_0, y_n, \dot{y}_n, \ddot{y}_n\}$ at t_0 and t_n

where $k = 1, \dots, n - 1$ and $n > 1$.

Output: Cubic/quintic spline, consisting of $n = n_{\text{orig}} + 2$ (cubic) or $n = n_{\text{orig}}$ (quintic) interconnected segments in PP form, which

- satisfies the underlying ODE from Equation G.1 at the given collocation sites,
- is C^2 (cubic) or C^4 (quintic) continuous at interior spline knots,
- satisfies the BCs.

begin

Cubic: Compute virtual control-points using (G.65) and remap indices $n \rightarrow n + 2$.

Cubic: Setup ${}_C L_i$, ${}_C D_i$, and ${}_C U_i$ from (G.22) and (G.23).

Setup $(\partial {}_C B_i / \partial {}_C \lambda)$ and $(\partial {}_C B_i / \partial {}_C r)$ from (G.71).

Quintic: Setup ${}_Q L_i$, ${}_Q D_i$, and ${}_Q U_i$ from (G.37) and (G.38).

Setup $(\partial {}_Q B_i / \partial {}_Q \lambda)$ and $(\partial {}_Q B_i / \partial {}_Q r)$ from (G.72) and (G.73).

Solve (G.53) for $(\partial X / \partial \lambda)$ and $(\partial X / \partial r)$ through the recursive scheme given in (G.26), (G.27), and (G.28) (cubic) or (G.42), (G.43), and (G.44) (quintic) while using $(\partial B / \partial \lambda)$, and $(\partial B / \partial r)$ instead of B as right-hand sides.

Note that the recursive scheme allows block-wise operations on the right-hand side, i. e., computing all columns of $(\partial X / \partial \lambda)$ and $(\partial X / \partial r)$ in parallel. For quintic splines this can be done very efficiently through LU decomposition of $({}_Q D_i - {}_Q L_i {}_Q H_{i-1})$.

Compute $(\partial \xi / \partial \lambda)$ and $(\partial \xi / \partial r)$ from $S^{-1}(\partial X / \partial \lambda)$ and $S^{-1}(\partial X / \partial r)$.

Compute the spline gradients $\nabla_{\lambda} s_k(0)$, $\nabla_{\lambda} \dot{s}_k(0)$, $\nabla_{\lambda} \ddot{s}_k(0)$, $\nabla_r s_k(0)$, $\nabla_r \dot{s}_k(0)$, and $\nabla_r \ddot{s}_k(0)$ from (G.56), (G.58), (G.59), and Appendix G.6 for all collocation sites. In the cubic case, additionally compute the spline gradients $\nabla_{\lambda} \dot{s}_0(0)$, $\nabla_r \dot{s}_0(0)$, $\nabla_{\lambda} \dot{s}_{n-1}(1)$, and $\nabla_r \dot{s}_{n-1}(1)$.

Assemble A_{coll} and B_{coll} using (G.62), (G.63), (G.64) (and (G.66) in cubic case) and solve (G.61) for λ .

Compute ξ from (G.52).

Cubic: Extract y_i from ${}_C \lambda$ and \ddot{y}_i from ${}_C \xi$ and compute the segment coefficients ${}_C a_i$, ${}_C b_i$, ${}_C c_i$, and ${}_C d_i$ using (G.18).

Quintic: Extract y_i from ${}_Q \lambda$ and \dot{y}_i , \ddot{y}_i from ${}_Q \xi$ and compute the segment coefficients ${}_Q a_i$, ${}_Q b_i$, ${}_Q c_i$, ${}_Q d_i$, ${}_Q e_i$, and ${}_Q f_i$ using (G.32).

end

case. Note that for large scale systems, solving $A_{\text{coll}}^{-1} B_{\text{coll}}$ turns out to be the bottleneck. Thus, in this case one might choose a parallel solver (for example `PartialPivLU` from *Eigen* with *OpenMP* [329] enabled) for Equation G.61 instead.

The basic structure of the source code related to the classes `CubicSplineCollocator` and `QuinticSplineCollocator` is illustrated in Figure G.3 left. Both classes are derived from the

abstract base class `SplineCollocatorBase`, which declares the interface for data in-/output and the main processing operators. In order to trigger Algorithm G.2 for a given input dataset, a simple call of `process()` is sufficient. For a more fine-grained control, `process()` is split up into publicly available subroutines (names starting with “substep_”). Note that providing public access to the subroutines not only allows detailed runtime measurements by the user, but also enables an efficient parallel solution for decoupled, multi-dimensional BVPs, see Figure G.3 right. For this, we exploit that large parts of Algorithm G.2 only depend on the spline segmentation, i. e., the collocation sites, (green box). In contrast, the remaining subroutines require explicit information about the investigated BVP (blue box). If all dimensions of the BVP, e. g. the x - and y -component of LOLA’s RMT motion, share the same spline segmentation, subroutines covered by the green box in Figure G.3 have to be called only once. Subsequently, the used instance of `Cubic|QuinticSplineCollocator` (holding intermediate results) is copied such that the remaining subroutines, covered by the blue box in Figure G.3, can be run in parallel. Note that this includes solving $A_{\text{coll}}^{-1} B_{\text{coll}}$ (`substep_solveCollocationLSE()`), which is the bottleneck for large-scale systems.

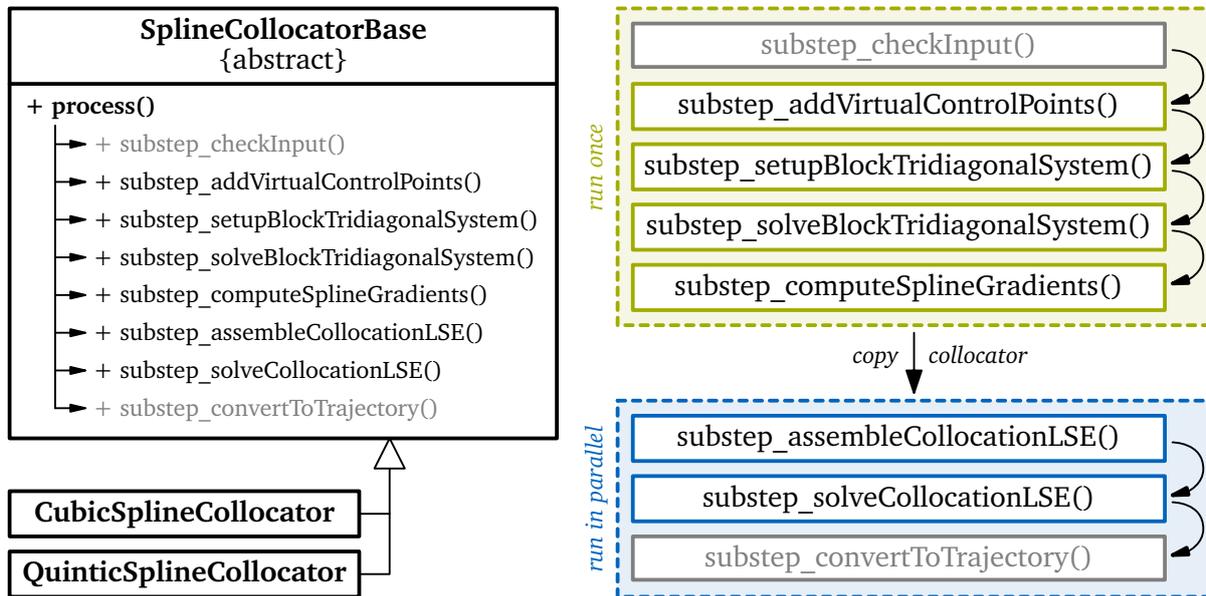


Figure G.3: Design of the classes `Cubic|QuinticSplineCollocator` in *Broccoli*. Left: class inheritance and segmentation of `process()` into subroutines. Right: proposed strategy for efficient parallelization in the case of a decoupled, multi-dimensional BVP. Note that the first and last subroutine (highlighted in gray) are optional and perform a validity check of the given input parameters and convert the final result into a corresponding `Curve::Trajectory` data structure for convenient evaluation of the generated polynomial spline, respectively.

G.4 Results

In order to evaluate the proposed collocation method and its variants, a simple mass-spring-damper system is considered, see Figure G.4 left. For the sake of simplicity, we do not consider external excitation, i. e., $\tau(t) := 0$. The corresponding ODE (see Equation G.1) describing the system dynamics simplifies to

$$\alpha \ddot{F}(t) + \beta \dot{F}(t) + \gamma F(t) = 0 \quad (\text{G.67})$$

where α , β , and γ are constants representing the mass and (linear) damping/stiffness coefficients of the system, respectively.

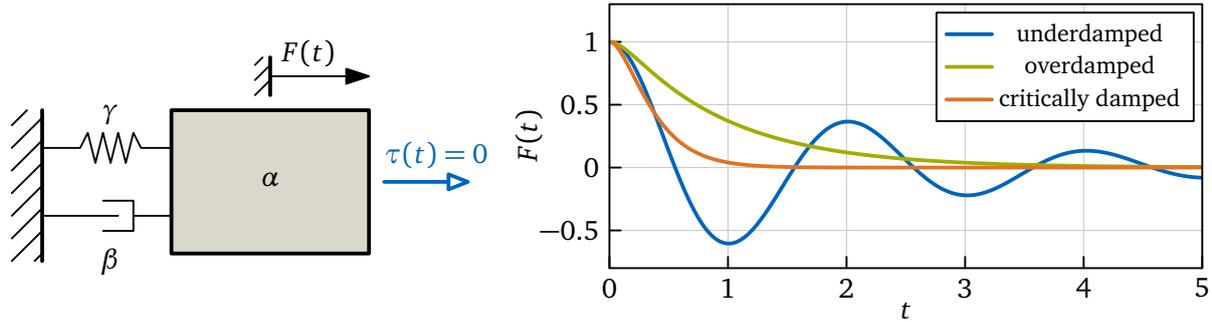


Figure G.4: Left: mass-spring-damper system used for validation. Right: analytical solution for the underdamped case ($\alpha = 1$, $\beta = 1$, $\gamma = 10$), the overdamped case ($\alpha = 1$, $\beta = 10$, $\gamma = 10$) and the critically damped case ($\alpha = 1$, $\beta = 10$, $\gamma = 25$). The solution is plotted for the initial conditions $F_0 = 1$ and $\dot{F}_0 = 0$.

Using textbook mathematics, we can find the analytical solution given by

$$F(t) = \begin{cases} e^{\sigma_1 t} \left[F_0 \cos(\sigma_2 t) + \left(\frac{\dot{F}_0 - F_0 \sigma_1}{\sigma_2} \right) \sin(\sigma_2 t) \right] & \text{for } \frac{\beta^2}{4\alpha^2} - \frac{\gamma}{\alpha} < 0, \\ & \text{(underdamped)} \\ \left(\frac{\dot{F}_0 - F_0(\sigma_1 - \sigma_2)}{2\sigma_2} \right) e^{(\sigma_1 + \sigma_2)t} + \left(\frac{F_0(\sigma_1 + \sigma_2) - \dot{F}_0}{2\sigma_2} \right) e^{(\sigma_1 - \sigma_2)t} & \text{for } \frac{\beta^2}{4\alpha^2} - \frac{\gamma}{\alpha} > 0, \\ & \text{(overdamped)} \\ e^{\sigma_1 t} [F_0 + (\dot{F}_0 - F_0 \sigma_1) t] & \text{for } \frac{\beta^2}{4\alpha^2} - \frac{\gamma}{\alpha} = 0 \\ & \text{(critically damped)} \end{cases}$$

where we used the initial conditions $F(t_0 = 0) = F_0$ and $\dot{F}(t_0 = 0) = \dot{F}_0$, and the abbreviations

$$\sigma_1 := -\frac{\beta}{2\alpha} \quad \text{and} \quad \sigma_2 := \sqrt{\left| \frac{\beta^2}{4\alpha^2} - \frac{\gamma}{\alpha} \right|}. \quad (\text{G.68})$$

The characteristic shape of each branch of the analytical solution is visualized in Figure G.4 (right) for the parametrization $\alpha = 1$, $\beta = 1$, $\gamma = 10$ in the underdamped case, $\alpha = 1$, $\beta = 10$, $\gamma = 10$ in the overdamped case, and $\alpha = 1$, $\beta = 10$, $\gamma = 25$ in the critically damped case. For the remainder of this section we will adhere to this parametrization. Moreover, we assume the initial conditions to be given with $F_0 = 1$, $\dot{F}_0 = 0$. Note that by choosing $\sigma_1 < 0$, we obtain asymptotically stable behavior. As we never constrained the underlying ODE to be stable, our algorithm can also be used to approximate instable systems. Since tests with $\beta = -1$ showed results comparable to the underdamped case (with $\beta = 1$), we omit an explicit discussion of this case for brevity. Although Equation G.67 describes a very simple system, the proposed algorithm has also been applied to the (much more complex) horizontal RMT planner of LOLA, see Section 6.14.2. However, the properties and characteristics of the proposed method can be better investigated and explained by means of a less complex test system.

Convergence for Consistent and Inconsistent Boundary Conditions As already mentioned, we focus on over-determined BVPs. Thus, we consider two cases for evaluating our algorithm. For the first analysis, we use the initial conditions $y_0 = F_0 = 1$, $\dot{y}_0 = \dot{F}_0 = 0$ and correspondingly $\ddot{y}_0 = \ddot{F}_0 = -\gamma/\alpha$, see Equation G.67, together with the analytical solution to compute the BCs $y_n = F_n$, $\dot{y}_n = \dot{F}_n$, and $\ddot{y}_n = \ddot{F}_n$ at $t_n = 5$. In this way, the BCs are guaranteed to be *consistent* because they belong to the same analytic solution $F(t)$. In the second case, we keep the initial BCs unchanged, but force $y_n = \dot{y}_n = \ddot{y}_n = 0$ for $t_n = 5$ which deviates from the previous solution

F_n , \dot{F}_n , and \ddot{F}_n , especially in the underdamped case, as can be clearly seen in Figure G.4 right. Thus, the second analysis handles *inconsistent* BCs.

In the following, we only focus on the underdamped and overdamped case, since the critically damped case can be seen as a special form of these with $\sigma_2 \rightarrow 0$. By evaluating both cases, we aim at covering oscillating and non-oscillating dynamics. Moreover, we run tests for different counts of collocation sites¹⁰² $\nu := |\{t_k\}|$, where we use a uniform segmentation of the spline, i. e., $h_i = h_{i+1} \forall i$. Note that an inhomogeneous partitioning $h_i \neq h_{i+1}$ is evaluated per default in the unit tests provided with *Broccoli*. The approximation of the BVP by spline collocation with consistent BCs and $\nu = 1, \dots, 100$ is depicted in Figure G.5. Note that for cubic spline collocation without virtual control-points, the BCs \dot{y}_0 and \dot{y}_n are violated (see Appendix G.2.7).

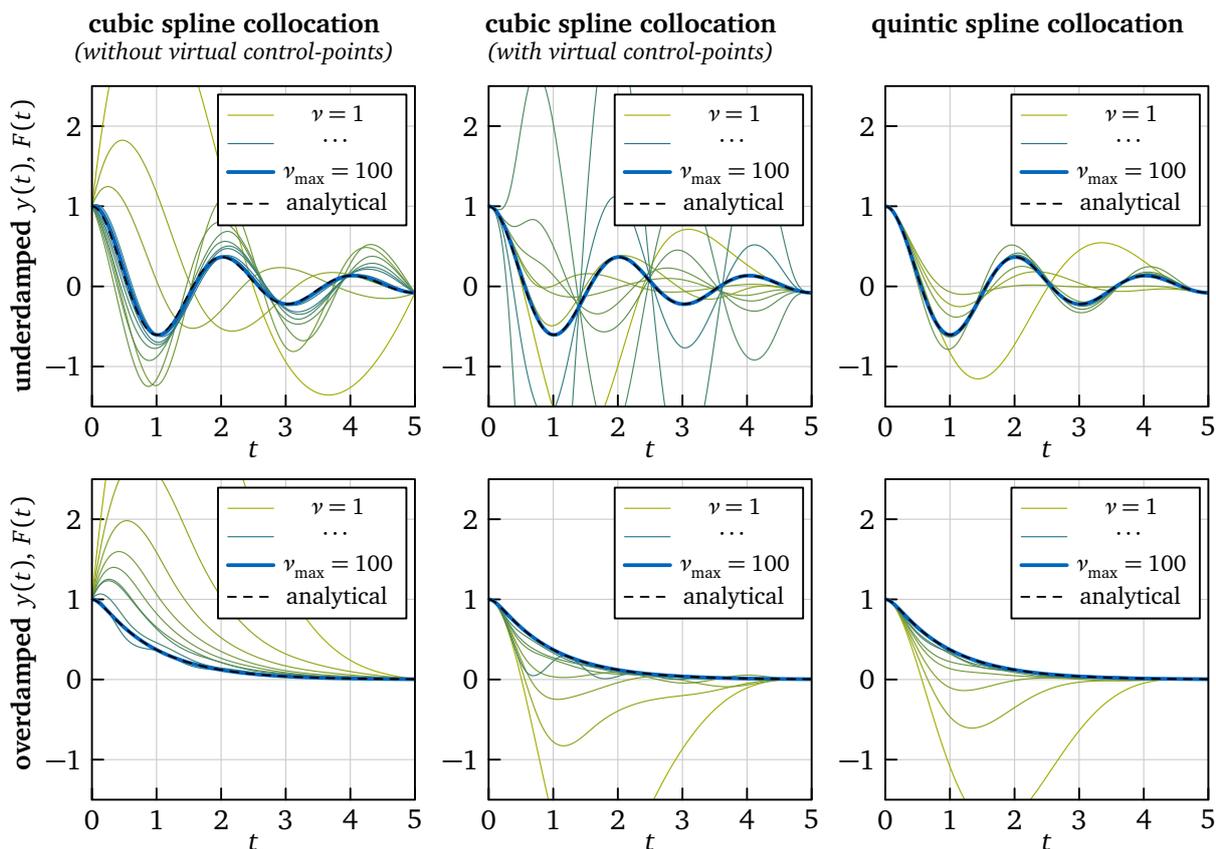


Figure G.5: Consistent BCs: convergence of the approximation $y(t)$ (blue and green) towards the analytical solution $F(t)$ (black, dashed) for $\nu = 1, \dots, 9, 30, 50, 70, 100$. The top row belongs to the underdamped case while the bottom row represents the overdamped case. From left to right: approximation with cubic spline *without* virtual control-points (left), cubic spline *with* virtual control-points (center) and quintic spline (right). The corresponding best approximation ν_{\max} is drawn in bold blue.

For the case of consistent BCs, we observe that the approximation $y(t)$ indeed converges for increasing ν to the analytical solution $F(t)$. From a qualitative point of view, the convergence order using a quintic spline (right column of Figure G.5) is clearly higher than the corresponding cubic counterparts (left and center columns of Figure G.5). In order to compare convergence using a quantitative measure, we use the *Root Mean Square (RMS)* of the approximation error

¹⁰²For cubic and quintic spline collocation *without* virtual control-points $\nu = n - 2$ holds. In contrast, $\nu = n - 4$ holds for cubic spline collocation *with* virtual control-points.

$e(t)$ and of the residual $R(t)$, defined as

$$\text{RMS}(e) := \sqrt{\frac{1}{t_n - t_0} \int_{t_0}^{t_n} [e(t)]^2 dt} \quad \text{with} \quad e(t) := F(t) - y(t), \quad (\text{G.69})$$

$$\text{RMS}(R) := \sqrt{\frac{1}{t_n - t_0} \int_{t_0}^{t_n} [R(t)]^2 dt} \quad \text{with} \quad R(t) := \alpha \ddot{y}(t) + \beta \dot{y}(t) + \gamma y(t) \underbrace{- \tau}_{=0}. \quad (\text{G.70})$$

For numerical evaluation of Equation G.69 and Equation G.70, we discretize the embedded integral using a time step size of $\Delta t = 0.01$ for which we obtain the results presented in Figure G.6 left. Since we stop the test series at $\nu_{\max} = 100$, we find the optimum for all variants to be at $\nu_{\text{opt}} = \nu_{\max}$. However, due to convergence, we expect the theoretical optimum to be at $\nu_{\text{opt}} \rightarrow \infty$. We observe that also from a quantitative point of view, quintic spline collocation clearly outperforms the cubic variants for the same ν . Furthermore, forcing first-order BCs through virtual control-points of the cubic spline seems to have a negative influence, especially for increasing ν . The influence of virtual control-points on the residual is visualized in Figure G.7 where peaks in $R(t)$ occur at these spots. Note that without virtual control-points, the first-order boundary conditions at $t_0 = 0$ and $t_n = 5$ are missed. Thus, in contrast to the collocation sites, the residual does not drop to zero at the boundaries. By comparing the left and right plot of Figure G.7, we observe that $R(t)$ behaves similarly for consistent and inconsistent BCs.

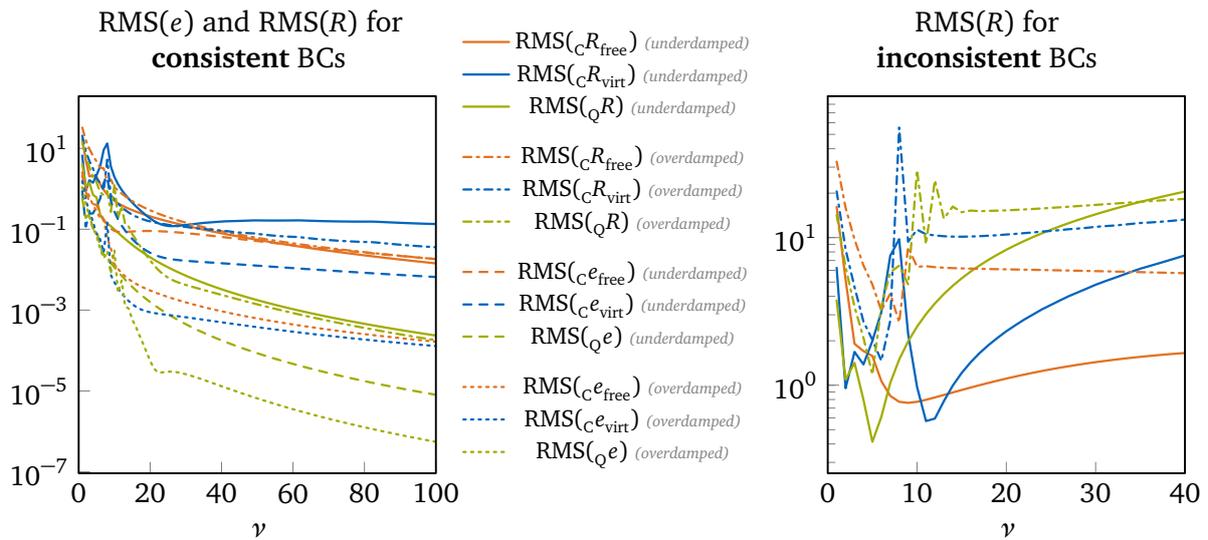


Figure G.6: Root Mean Square (RMS) of error $e(t)$ and residual $R(t)$ as defined in Equations G.69 and G.70. The left subscript differs between cubic C and quintic Q spline collocation. Moreover, for cubic spline collocation, we identify the variants without virtual control-points (i. e., *free* first-order boundaries \dot{y}_0, \dot{y}_n) and with virtual control-points by the right subscript *free* and *virt*, respectively. The left plot belongs to the case of consistent BCs while the right one was obtained using inconsistent BCs. Note that for inconsistent BCs an analytic solution does not exist, thus, the error $e(t)$ is not defined and we consider only the residual $R(t)$.

For consistent BCs, we can state that, at least for the investigated test system, the proposed algorithm leads to an approximation which converges to the real solution where the “speed” of convergence depends on the chosen variant of Algorithm G.2. For inconsistent BCs, however, our analysis draws a different picture: while we can find an optimal count of collocation sites ν_{opt} for each variant and test case, the approximation diverges for $\nu \rightarrow \infty$, see Figure G.6 right and Figure G.8. Note that this was expected since we are attempting to find an approximation of a solution which does not actually exist. For small ν , we still find a “reasonable” $y(t)$ where the spline is still able to smooth out the wrong BCs. However, as we refine the segmentation of the spline, it gets harder to compensate the error, which in turn leads to undesired oscillations of $y(t)$. Note that although we call this behavior *divergence*, our collocation algorithm still satisfies

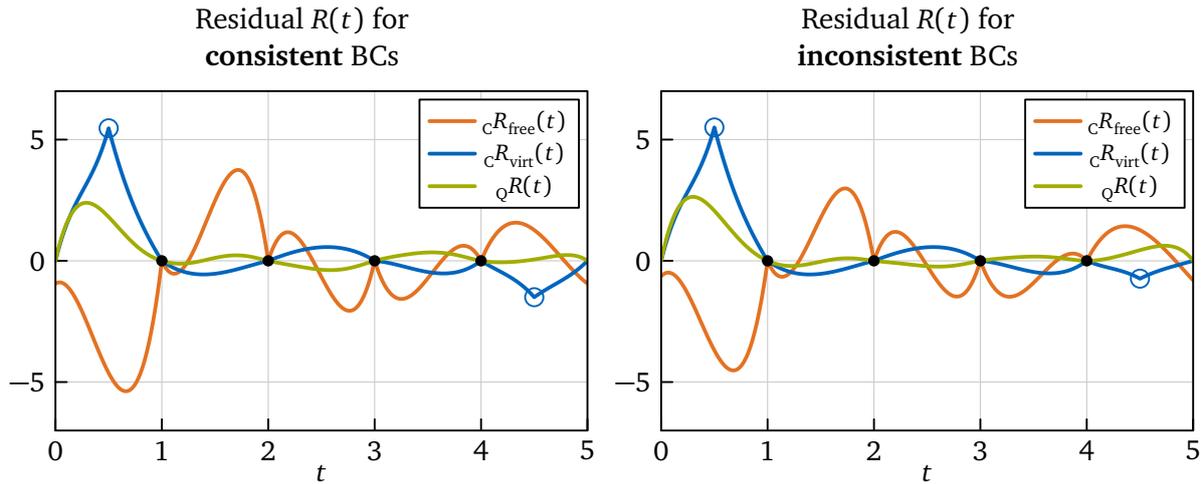


Figure G.7: Residual $R(t)$ as defined in Equation G.70 for consistent (left) and inconsistent (right) BCs in the underdamped case. For best presentation, the count of collocation sites is chosen as $\nu = 4$ such that the collocation sites (black dots) are given by $\{t_k\} = \{1, 2, 3, 4\}$. For cubic spline collocation, the left subscript C and for the quintic counterpart Q is used. Moreover, the right subscripts free and virt are used to differentiate between the variants without and with virtual control-points, respectively. The virtual control-points are highlighted with circles.

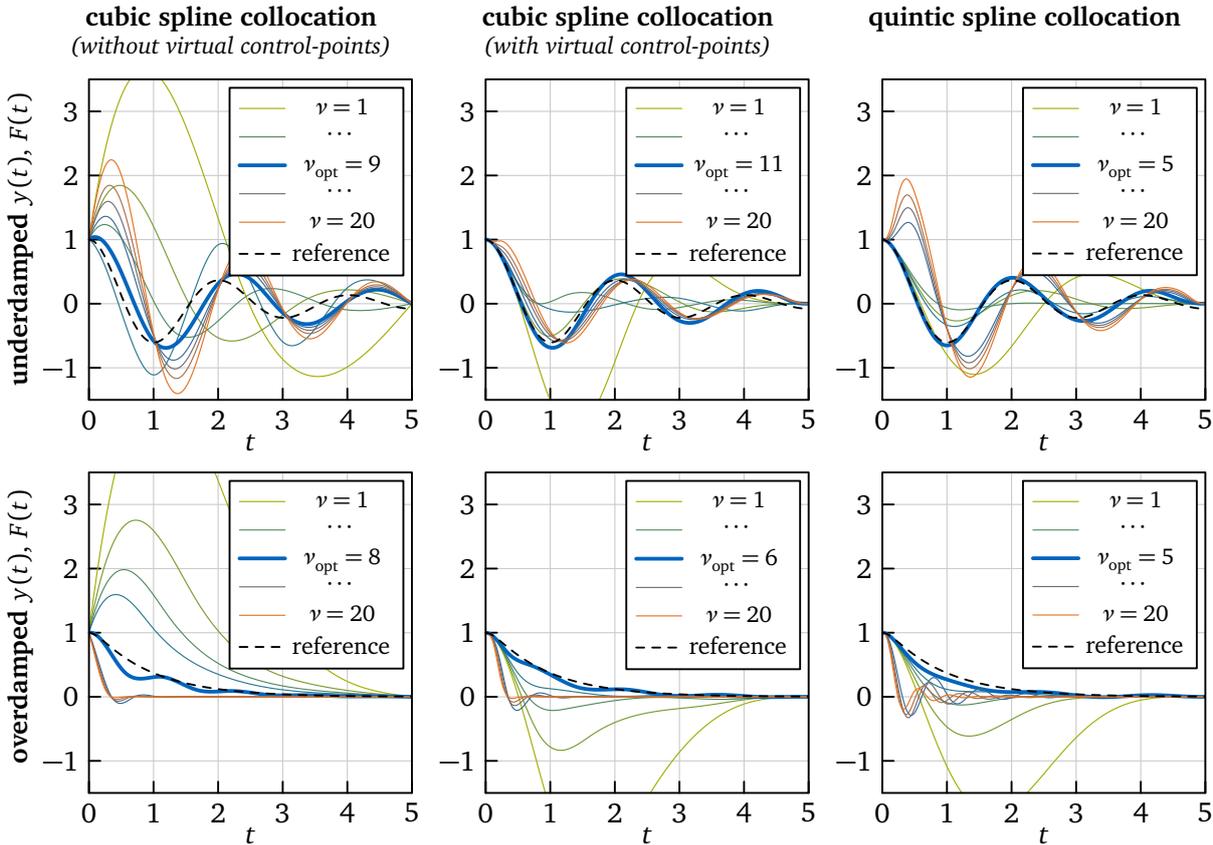


Figure G.8: Inconsistent BCs: approximation $y(t)$ (blue, green, and orange) and reference system $F(t)$ (black, dashed) for $\nu = 1, \dots, 4, \nu_{\text{opt}}, 13, 15, 17, 20$. The top row belongs to the underdamped case while the bottom row represents the overdamped case. From left to right: approximation with cubic spline (no virtual control-points, left), cubic spline (with virtual control-points, center) and quintic spline (right). The corresponding best approximation ν_{opt} is drawn in bold blue. Diverging approximations for $\nu > \nu_{\text{opt}}$ are colored orange.

all constraints that we defined: using a given partitioning, it fulfills the BCs and satisfies the underlying ODE at the given collocation sites. Thus, the divergence is not a fault of the proposed

algorithm, but rather is due to the non-existence of the solution $F(t)$. Moreover, the sensitivity to undesired oscillations depends heavily on the underlying BVP: for our target application of planning horizontal RMT motion (cf. Section 6.14.2), we did not observe any oscillations up to $\nu = 10^4$. The critical value for ν is expected to be even higher. However, we could not determine it due to memory limitations of our test system.

In order to avoid undesired oscillations, an optimal ν has to be chosen, which seems to be difficult at the first sight, since in practice we do not know the analytical solution, and thus cannot evaluate $e(t)$. However, one can use the residual $R(t)$ as measure for the approximation error and use this to formulate a governing optimization for finding ν_{opt} . Moreover, one can also use a non-uniform segmentation to give specific sections more weight. Such adaptive techniques for automatic mesh refinement have also been developed for other collocation methods, see, for example, CHRISTARA and NG [117]. However, for our application it is sufficient to choose h_i once (fixed), thus, we withhold this idea for future investigations.

Runtime Analysis In the following, we present measurements, which have been made by using the implementation given in *Broccoli* with the version *primo* (v1.0.0). We used an *AMD Ryzen 7 1700X 8x (16x) @3.4 GHz* CPU with 32 GiB DDR4 RAM @2.133 GHz as hardware backend and *Ubuntu 18.04.2 LTS 64bit* (Linux kernel 4.15.0-51) together with *Clang* (version 6.0.0-1) on optimization level 3 as software basis. Although our algorithm is sequential, we run different test cases on four physical cores of the CPU in parallel. For all tests, SMT of the CPU was disabled. For runtime evaluation, we take 1,000 samples for every code section in Algorithm G.2 and choose the minimum execution time as reference to minimize the risk of wrong measurements due to high system load and context switching effects.

In Figure G.9 (left), we present runtime measurements for all three variants of our algorithm. We restrict our analysis to the case of an underdamped parametrization and consistent BCs since we expect comparable results for the other test cases¹⁰³. We observe that quintic spline collocation is more expensive than the cubic counterparts for the same ν , which complies with theory since the (block-)tridiagonal system for quintic splines is twice the size of the tridiagonal system for cubic splines. However, for increasing ν , the gap becomes smaller since solving the collocation equations, where A_{coll} is of the same dimension for cubic and quintic splines, becomes the bottleneck, see Figure G.10. Note that there is only a small difference in runtime between cubic spline collocation with and without virtual control-points. This also complies with theory, since the only difference is two additional spline knots, which increases the dimension of A_{coll} by two. Lastly, the small ripples in Figure G.9 (left) are due to vectorization and SIMD optimizations handled by *Eigen* and the compiler, which give slightly better performance if the dimension of arrays in memory are a multiple of two.

Comparing runtimes for the same ν might not be a meaningful basis for choosing a method. Instead, one is typically interested in getting the best approximation in the shortest time. For this purpose, the RMS of the residual $R(t)$ is plotted over runtime in Figure G.9 (right). As can be seen, quintic spline collocation significantly outperforms both other variants. Moreover, bad convergence of cubic spline collocation with virtual control-points is also visible in this comparison. In addition to measuring total runtime, we also performed a detailed analysis on the relative cost of each code section of Algorithm G.2 during quintic spline collocation. Figure G.10 demonstrates that in the vicinity of $\nu \approx 160$, evaluating the block-tridiagonal systems to enforce BCs and continuity, and solving $A_{\text{coll}} \lambda = B_{\text{coll}}$ for actual collocation share approximately the same portion of total runtime. With increasing ν , solving for λ becomes more relevant since the corresponding system is dense while the block-tridiagonal LSE can be solved efficiently using the recursive scheme discussed in Appendix G.2.3.

¹⁰³Since our algorithm has a deterministic runtime which only depends on the chosen variant (cubic/quintic, with/without virtual control-points) and ν , there is no reason why it should be faster or slower with another parametrization.

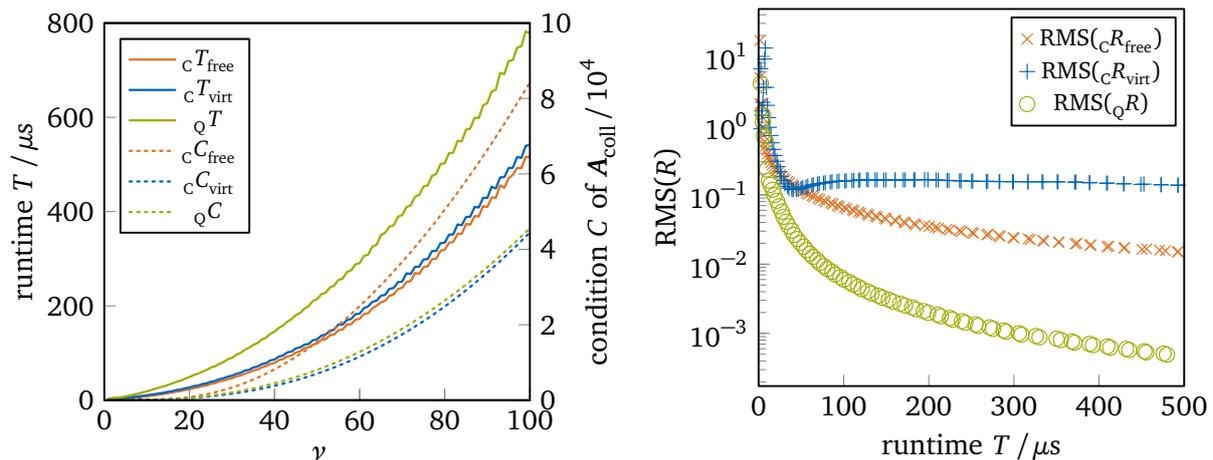


Figure G.9: Left: (minimum) runtime T and condition C of A_{coll} for running Algorithm G.2 over count of collocation sites ν . Right: RMS of residual $R(t)$ as defined in Equation G.70 over (minimum) runtime T . The left subscript C and Q belong to the cubic or quintic spline version of the algorithm, respectively. Moreover, the right subscript indicates if cubic spline collocation was performed without (free) or with (virt) virtual control-points. All measurements were performed using the underdamped parametrization and consistent BCs.

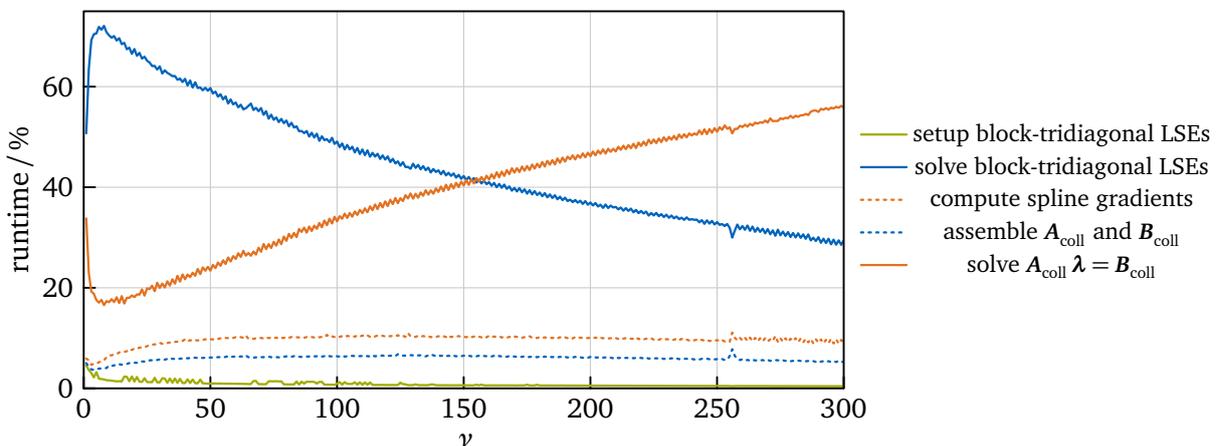


Figure G.10: Runtime (percentile) of relevant steps of Algorithm G.2 relative to total runtime for quintic spline collocation. Code sections with negligible execution time are not plotted and also not accounted for total runtime. The measurements were obtained by using an extended time horizon $t_n = 50$ and the parametrization $\alpha = 1$, $\beta = 0.1$, $\gamma = 10$ (underdamped) to allow a better representation of high counts of collocation sites of up to $\nu = 300$.

Lastly, we want to point out that the condition of A_{coll} gets worse for an increasing count of collocation sites ν , see Figure G.9 left. Thus, there might be an upper limit of ν for the proposed algorithm.

G.5 Discussion

As shown in the previous section, the presented algorithm performs well if the BCs are consistent and fully known. Even in the case of inconsistent BCs, we still obtain a reasonable approximation as long as we carefully pick the collocation sites. However, if we exceed the optimum, undesired oscillations may occur, which is an indicator for putting too much emphasis on satisfying the underlying ODE while simultaneously trying to compensate the “broken” BCs. In order to automatically determine an optimal partitioning of the spline, a higher level optimization may be applied, which may be the focus of further investigations.

Obviously, if the investigated BVP is well-posed, i. e., if it is not over-determined, other techniques should be preferred over our approach. However, for applications where the enforcement of certain BCs is more important than approximating the underlying dynamics, the proposed method seems to be a valid approach. Moreover, we want to emphasize that no variable recursion or iteration is involved. This makes execution time predictable, which is especially relevant for real-time applications such as in our use case.

Comparing different variants of our algorithm showed that collocation using a quintic spline is in general superior to using the somewhat simpler cubic splines. Note that although its derivation is more involved, the final implementation is of approximately the same complexity, since virtual control points have to be introduced in the cubic case. At this point, we also want to emphasize that, based on the results of our study, we do not recommend to use cubic spline collocation with virtual control points. Although the full set of BCs is satisfied, the additional knots seem to significantly downgrade convergence. However, other choices of $t_{\text{virt},1}$ and $t_{\text{virt},2}$ may lead to different results.

Within this contribution, we only considered second-order BVPs. An extension to ODEs of higher order seems to be straightforward, since only the collocation equations, see Equations G.63 and G.64 have to be extended by the corresponding gradients while the overall dimension of \mathbf{A}_{coll} and \mathbf{B}_{coll} stays the same. Moreover, our approach may be applied to nonlinear systems as well, by using the common approach of linearization and embedding the scheme into a NEWTON iteration.

Lastly, we want to highlight that our focus lies on runtime efficiency. However, for embedded systems especially, memory consumption may also be a limiting factor. Although we expect our algorithm to have similar requirements when compared to other techniques, we have not looked into this issue so far.

G.6 Attachment: Spline Gradients

In the following, explicit expressions for the spline gradients used in Equations G.56, G.58 and G.59 are given. Note that the gradients differ depending on the type of the underlying spline (cubic/quintic).

Cubic Spline Gradients From Equations G.8 to G.10, we obtain for each ${}_C\boldsymbol{\rho} \in \{{}_C\boldsymbol{\lambda}, {}_C\boldsymbol{\xi}, {}_C\mathbf{r}\}$ and each segment ${}_C s_i$ with $i = 0, \dots, n-1$

$$\begin{aligned} \left(\frac{\partial {}_C s_i(\eta_i)}{\partial {}_C \boldsymbol{\rho}} \right) &= \left(\frac{\partial {}_C a_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i^3 + \left(\frac{\partial {}_C b_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i^2 + \left(\frac{\partial {}_C c_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i + \left(\frac{\partial {}_C d_i}{\partial {}_C \boldsymbol{\rho}} \right), \\ \left(\frac{\partial {}_C \dot{s}_i(\eta_i)}{\partial {}_C \boldsymbol{\rho}} \right) &= \frac{3}{h_i} \left(\frac{\partial {}_C a_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i^2 + \frac{2}{h_i} \left(\frac{\partial {}_C b_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i + \frac{1}{h_i} \left(\frac{\partial {}_C c_i}{\partial {}_C \boldsymbol{\rho}} \right), \\ \left(\frac{\partial {}_C \ddot{s}_i(\eta_i)}{\partial {}_C \boldsymbol{\rho}} \right) &= \frac{6}{h_i^2} \left(\frac{\partial {}_C a_i}{\partial {}_C \boldsymbol{\rho}} \right) \eta_i + \frac{2}{h_i^2} \left(\frac{\partial {}_C b_i}{\partial {}_C \boldsymbol{\rho}} \right). \end{aligned}$$

Moreover, by using the indices $u = 1, \dots, n-1$, $v = 1, \dots, n-1$, and $w = 1, \dots, 4$ to specify the elements of

$$\begin{aligned} {}_C \boldsymbol{\lambda} &= [{}_C \lambda_1, \dots, {}_C \lambda_u, \dots, {}_C \lambda_{n-1}]^T &= [y_1, \dots, y_u, \dots, y_{n-1}]^T \\ {}_C \boldsymbol{\xi} &= [{}_C \xi_1, \dots, {}_C \xi_v, \dots, {}_C \xi_{n-1}]^T &= [\ddot{y}_1, \dots, \ddot{y}_v, \dots, \ddot{y}_{n-1}]^T \\ {}_C \mathbf{r} &= [{}_C r_1, \dots, {}_C r_w, \dots, {}_C r_4]^T &= [y_0, \ddot{y}_0, y_n, \ddot{y}_n]^T \end{aligned}$$

we obtain

$$\begin{aligned} \left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}}\right) &= \mathbf{0}, \quad \left(\frac{\partial_{\mathcal{C}} b_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}}\right) = \mathbf{0}, \quad \left(\frac{\partial_{\mathcal{C}} c_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}_u}\right) = \begin{cases} -1 & \text{for } u = i, \\ 1 & \text{for } u = i + 1, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} d_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}_u}\right) = \begin{cases} 1 & \text{for } u = i, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{\xi}_v}\right) &= \begin{cases} -\frac{h_i^2}{6} & \text{for } v = i, \\ \frac{h_i^2}{6} & \text{for } v = i + 1, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} c_i}{\partial_{\mathcal{C}} \boldsymbol{\xi}_v}\right) = \begin{cases} -\frac{h_i^2}{3} & \text{for } v = i, \\ -\frac{h_i^2}{6} & \text{for } v = i + 1, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial_{\mathcal{C}} b_i}{\partial_{\mathcal{C}} \boldsymbol{\xi}_v}\right) &= \begin{cases} \frac{h_i^2}{2} & \text{for } v = i, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} d_i}{\partial_{\mathcal{C}} \boldsymbol{\xi}}\right) = \mathbf{0}, \\ \left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{r}_w}\right) &= \begin{cases} -\frac{h_i^2}{6} & \text{for } i = 0 \wedge w = 2, \\ \frac{h_i^2}{6} & \text{for } i = n - 1 \wedge w = 4, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} c_i}{\partial_{\mathcal{C}} \boldsymbol{r}_w}\right) = \begin{cases} -1 & \text{for } i = 0 \wedge w = 1, \\ -\frac{h_i^2}{3} & \text{for } i = 0 \wedge w = 2, \\ 1 & \text{for } i = n - 1 \wedge w = 3, \\ -\frac{h_i^2}{6} & \text{for } i = n - 1 \wedge w = 4, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial_{\mathcal{C}} b_i}{\partial_{\mathcal{C}} \boldsymbol{r}_w}\right) &= \begin{cases} \frac{h_i^2}{2} & \text{for } i = 0 \wedge w = 2, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} d_i}{\partial_{\mathcal{C}} \boldsymbol{r}_w}\right) = \begin{cases} 1 & \text{for } i = 0 \wedge w = 1, \\ 0 & \text{else} \end{cases}, \end{aligned}$$

which can easily be derived from Equation G.18 and the definition of $\mathcal{C}\boldsymbol{\lambda}$, $\mathcal{C}\boldsymbol{\xi}$, and $\mathcal{C}\boldsymbol{r}$. Note that for $\eta_i = 0$, the computation of

$$\left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}}\right), \left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{\xi}}\right), \text{ and } \left(\frac{\partial_{\mathcal{C}} a_i}{\partial_{\mathcal{C}} \boldsymbol{r}}\right)$$

can be skipped entirely since up to the second time derivative of the gradients of $s_i(\eta_i = 0)$ these terms are multiplied with zero and have no effect anyway. In order to solve Equation G.53, we have to further compute the corresponding right-hand sides which are given by

$$\left(\frac{\partial_{\mathcal{C}} B_i}{\partial_{\mathcal{C}} \boldsymbol{\lambda}_u}\right) = \begin{cases} \frac{6}{h_{i-1}} & \text{for } u = i - 1, \\ -\frac{6}{h_i} - \frac{6}{h_{i-1}} & \text{for } u = i, \\ \frac{6}{h_i} & \text{for } u = i + 1, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{C}} B_i}{\partial_{\mathcal{C}} \boldsymbol{r}_w}\right) = \begin{cases} \frac{6}{h_{i-1}} & \text{for } i = 1 \wedge w = 1, \\ -h_{i-1} & \text{for } i = 1 \wedge w = 2, \\ \frac{6}{h_i} & \text{for } i = n - 1 \wedge w = 3, \\ -h_i & \text{for } i = n - 1 \wedge w = 4, \\ 0 & \text{else} \end{cases}, \quad (\text{G.71})$$

for $i = 1, \dots, n - 1$, where we used Equations G.24 and G.25 and the definition of $\mathcal{C}\boldsymbol{\lambda}$ and $\mathcal{C}\boldsymbol{r}$.

Quintic Spline Gradients From Equations G.11 to G.13, we obtain for each $\mathcal{Q}\boldsymbol{\rho} \in \{\mathcal{Q}\boldsymbol{\lambda}, \mathcal{Q}\boldsymbol{\xi}, \mathcal{Q}\boldsymbol{r}\}$ and each segment $\mathcal{Q}s_i$ with $i = 0, \dots, n - 1$

$$\begin{aligned} \left(\frac{\partial_{\mathcal{Q}} s_i(\eta_i)}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) &= \left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^5 + \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^4 + \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^3 + \left(\frac{\partial_{\mathcal{Q}} d_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^2 + \left(\frac{\partial_{\mathcal{Q}} e_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i + \left(\frac{\partial_{\mathcal{Q}} f_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right), \\ \left(\frac{\partial_{\mathcal{Q}} \dot{s}_i(\eta_i)}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) &= \frac{5}{h_i} \left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^4 + \frac{4}{h_i} \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^3 + \frac{3}{h_i} \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^2 + \frac{2}{h_i} \left(\frac{\partial_{\mathcal{Q}} d_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i + \frac{1}{h_i} \left(\frac{\partial_{\mathcal{Q}} e_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right), \\ \left(\frac{\partial_{\mathcal{Q}} \ddot{s}_i(\eta_i)}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) &= \frac{20}{h_i^2} \left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^3 + \frac{12}{h_i^2} \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i^2 + \frac{6}{h_i^2} \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right) \eta_i + \frac{2}{h_i^2} \left(\frac{\partial_{\mathcal{Q}} d_i}{\partial_{\mathcal{Q}} \boldsymbol{\rho}}\right). \end{aligned}$$

Moreover, by using the indices $u = 1, \dots, n-1$, $v = 1, \dots, 2(n-1)$, and $w = 1, \dots, 4$ to specify the elements of

$$\begin{aligned} {}_Q\boldsymbol{\lambda} &= [{}_Q\lambda_1, \dots, {}_Q\lambda_u, \dots, {}_Q\lambda_{n-1}]^T = [y_1, \dots, y_u, \dots, y_{n-1}]^T \\ {}_Q\boldsymbol{\xi} &= [{}_Q\xi_1, \dots, {}_Q\xi_v, \dots, {}_Q\xi_{2(n-1)}]^T = [\dot{y}_1, \ddot{y}_1, \dots, \dot{y}_{(v+1)/2}, \ddot{y}_{v/2}, \dots, \dot{y}_{n-1}, \ddot{y}_{n-1}]^T \\ {}_Q\boldsymbol{r} &= [{}_Qr_1, \dots, {}_Qr_w, \dots, {}_Qr_6]^T = [y_0, \dot{y}_0, \ddot{y}_0, y_n, \dot{y}_n, \ddot{y}_n]^T \end{aligned}$$

we obtain

$$\begin{aligned} \left(\frac{\partial {}_Q a_i}{\partial {}_Q \lambda_u} \right) &= \begin{cases} -6 & \text{for } u = i, \\ 6 & \text{for } u = i + 1, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q d_i}{\partial {}_Q \boldsymbol{\lambda}} \right) &= \mathbf{0}, \\ \left(\frac{\partial {}_Q b_i}{\partial {}_Q \lambda_u} \right) &= \begin{cases} 15 & \text{for } u = i, \\ -15 & \text{for } u = i + 1, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q e_i}{\partial {}_Q \boldsymbol{\lambda}} \right) &= \mathbf{0}, \\ \left(\frac{\partial {}_Q c_i}{\partial {}_Q \lambda_u} \right) &= \begin{cases} -10 & \text{for } u = i, \\ 10 & \text{for } u = i + 1, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q f_i}{\partial {}_Q \lambda_u} \right) &= \begin{cases} 1 & \text{for } u = i, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial {}_Q a_i}{\partial {}_Q \xi_v} \right) &= \begin{cases} -3h_i & \text{for } v = 2i - 1, \\ -\frac{h_i^2}{2} & \text{for } v = 2i, \\ -3h_i & \text{for } v = 2i + 1, \\ \frac{h_i^2}{2} & \text{for } v = 2i + 2, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q d_i}{\partial {}_Q \xi_v} \right) &= \begin{cases} \frac{h_i^2}{2} & \text{for } v = 2i, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial {}_Q b_i}{\partial {}_Q \xi_v} \right) &= \begin{cases} 8h_i & \text{for } v = 2i - 1, \\ \frac{3h_i^2}{2} & \text{for } v = 2i, \\ 7h_i & \text{for } v = 2i + 1, \\ -h_i^2 & \text{for } v = 2i + 2, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q e_i}{\partial {}_Q \xi_v} \right) &= \begin{cases} h_i & \text{for } v = 2i - 1, \\ 0 & \text{else} \end{cases}, \\ \left(\frac{\partial {}_Q c_i}{\partial {}_Q \xi_v} \right) &= \begin{cases} -6h_i & \text{for } v = 2i - 1, \\ -\frac{3h_i^2}{2} & \text{for } v = 2i, \\ -4h_i & \text{for } v = 2i + 1, \\ \frac{h_i^2}{2} & \text{for } v = 2i + 2, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q f_i}{\partial {}_Q \boldsymbol{\xi}} \right) &= \mathbf{0}, \\ \left(\frac{\partial {}_Q a_i}{\partial {}_Q r_w} \right) &= \begin{cases} -6 & \text{for } i = 0 \wedge w = 1, \\ -3h_i & \text{for } i = 0 \wedge w = 2, \\ -\frac{h_i^2}{2} & \text{for } i = 0 \wedge w = 3, \\ 6 & \text{for } i = n - 1 \wedge w = 4, \\ -3h_i & \text{for } i = n - 1 \wedge w = 5, \\ \frac{h_i^2}{2} & \text{for } i = n - 1 \wedge w = 6, \\ 0 & \text{else} \end{cases}, & \left(\frac{\partial {}_Q d_i}{\partial {}_Q r_w} \right) &= \begin{cases} \frac{h_i^2}{2} & \text{for } i = 0 \wedge w = 3, \\ 0 & \text{else} \end{cases}, \end{aligned}$$

$$\left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} r_w}\right) = \begin{cases} 15 & \text{for } i = 0 \wedge w = 1, \\ 8h_i & \text{for } i = 0 \wedge w = 2, \\ \frac{3h_i^2}{2} & \text{for } i = 0 \wedge w = 3, \\ -15 & \text{for } i = n-1 \wedge w = 4, \\ 7h_i & \text{for } i = n-1 \wedge w = 5, \\ -h_i^2 & \text{for } i = n-1 \wedge w = 6, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{Q}} e_i}{\partial_{\mathcal{Q}} r_w}\right) = \begin{cases} h_i & \text{for } i = 0 \wedge w = 2, \\ 0 & \text{else} \end{cases},$$

$$\left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} r_w}\right) = \begin{cases} -10 & \text{for } i = 0 \wedge w = 1, \\ -6h_i & \text{for } i = 0 \wedge w = 2, \\ -\frac{3h_i^2}{2} & \text{for } i = 0 \wedge w = 3, \\ 10 & \text{for } i = n-1 \wedge w = 4, \\ -4h_i & \text{for } i = n-1 \wedge w = 5, \\ \frac{h_i^2}{2} & \text{for } i = n-1 \wedge w = 6, \\ 0 & \text{else} \end{cases}, \quad \left(\frac{\partial_{\mathcal{Q}} f_i}{\partial_{\mathcal{Q}} r_w}\right) = \begin{cases} 1 & \text{for } i = 0 \wedge w = 1, \\ 0 & \text{else} \end{cases},$$

which can easily be derived from Equation G.32 and the definition of $_{\mathcal{Q}}\lambda$, $_{\mathcal{Q}}\xi$, and $_{\mathcal{Q}}r$. Note that for $\eta_i = 0$ the computation of

$$\left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} \lambda}\right), \left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} \xi}\right), \left(\frac{\partial_{\mathcal{Q}} a_i}{\partial_{\mathcal{Q}} r}\right), \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} \lambda}\right), \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} \xi}\right), \left(\frac{\partial_{\mathcal{Q}} b_i}{\partial_{\mathcal{Q}} r}\right), \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} \lambda}\right), \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} \xi}\right), \left(\frac{\partial_{\mathcal{Q}} c_i}{\partial_{\mathcal{Q}} r}\right),$$

can be skipped entirely since up to the second time derivative of the gradients of $s_i(\eta_i = 0)$ these terms are multiplied with zero and have no effect anyway. In order to solve Equation G.53, we further have to compute the corresponding right-hand sides, which are given by

$$\left(\frac{\partial_{\mathcal{Q}} \mathbf{B}_i}{\partial_{\mathcal{Q}} \lambda_u}\right) = \begin{cases} \begin{bmatrix} 120 g_{i-1}^4 \\ 20 \kappa g_{i-1}^3 g_i \end{bmatrix} & \text{for } u = i-1, \quad \begin{bmatrix} -120 g_i^4 \\ 20 \kappa g_i^4 \end{bmatrix} & \text{for } u = i+1, \\ \begin{bmatrix} 120(g_i^4 - g_{i-1}^4) \\ -20 \kappa g_i(g_i^3 + g_{i-1}^3) \end{bmatrix} & \text{for } u = i, \quad \mathbf{0} & \text{else} \end{cases}, \quad (\text{G.72})$$

$$\left(\frac{\partial_{\mathcal{Q}} \mathbf{B}_i}{\partial_{\mathcal{Q}} r_w}\right) = \begin{cases} \begin{bmatrix} 120 g_{i-1}^4 \\ 20 \kappa g_{i-1}^3 g_i \end{bmatrix} & \text{for } i = 1 \wedge w = 1, \quad \begin{bmatrix} -120 g_i^4 \\ 20 \kappa g_i^4 \end{bmatrix} & \text{for } i = n-1 \wedge w = 4, \\ \begin{bmatrix} 56 g_{i-1}^3 \\ 8 \kappa g_{i-1}^2 g_i \end{bmatrix} & \text{for } i = 1 \wedge w = 2, \quad \begin{bmatrix} 56 g_i^3 \\ -8 \kappa g_i^3 \end{bmatrix} & \text{for } i = n-1 \wedge w = 5, \\ \begin{bmatrix} 8 g_{i-1}^2 \\ \kappa g_{i-1} g_i \end{bmatrix} & \text{for } i = 1 \wedge w = 3, \quad \begin{bmatrix} -8 g_i^2 \\ \kappa g_i^2 \end{bmatrix} & \text{for } i = n-1 \wedge w = 6, \\ \mathbf{0} & \text{else} \end{cases} \quad (\text{G.73})$$

for $i = 1, \dots, n-1$, where we used Equations G.39 and G.40 and the definition of $_{\mathcal{Q}}\lambda$ and $_{\mathcal{Q}}r$.

Appendix H

Hardware Details

This section gives a brief summary of the main mechanical and electrical specifications of the humanoid robot LOLA v1.1, i. e., the hardware configuration *after* the upgrade of the upper body.

H.1 Mechanical Specifications

Table H.1: Mechanical parameters of the segments of LOLA v1.1. From left to right: corresponding DoF with mechanical joint limits (rotation around positive z -axis of segment frame); segment (seg) codename; parent (par) codename; mass m ; rotation matrix from segment frame to parent frame ${}_{\text{par}}A_{\text{seg}}$ (with $c := \cos(10^\circ)$ and $s := \sin(10^\circ)$); position of segment frame origin described in parent frame ${}_{\text{par}}r_{\text{seg}}$; CoM of segment described in segment frame ${}_{\text{seg}}r_{\text{CoM}}$; mass moment of inertia tensor of segment with respect to origin of segment frame described in segment frame ${}_{\text{seg}}\Theta_{\text{seg}}^{\text{seg}}$.

DoF	seg	par	m / kg	${}_{\text{par}}A_{\text{seg}}$	${}_{\text{par}}r_{\text{seg}}$ / m	${}_{\text{seg}}r_{\text{CoM}}$ / m	${}_{\text{seg}}\Theta_{\text{seg}}^{\text{seg}}$ / kgm^2
q_1, \dots, q_6 (see Footnote 52)	torso	–	14.836	–	–	$\begin{bmatrix} +6.485e-2 \\ +2.268e-1 \\ -2.652e-3 \end{bmatrix}$	$\begin{bmatrix} +1.030e+0 & -2.659e-1 & +4.404e-3 \\ -2.659e-1 & +2.729e-1 & +9.269e-3 \\ +4.404e-3 & +9.269e-3 & +1.100e+0 \end{bmatrix}$
q_7 [$-19.4^\circ, 19.4^\circ$]	br	torso	1.530	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ -2.000e-3 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.509e-2 \\ -4.169e-6 \\ -4.730e-2 \end{bmatrix}$	$\begin{bmatrix} +5.405e-3 & +1.541e-6 & +1.135e-3 \\ +1.541e-6 & +7.053e-3 & +1.384e-6 \\ +1.135e-3 & +1.384e-6 & +2.994e-3 \end{bmatrix}$
q_8 [$-15.1^\circ, 14.7^\circ$]	ba	br	6.080	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +5.800e-2 \\ +0.000e+0 \\ -5.100e-2 \end{bmatrix}$	$\begin{bmatrix} +1.420e-2 \\ -9.972e-6 \\ -5.154e-2 \end{bmatrix}$	$\begin{bmatrix} +7.532e-2 & +2.277e-5 & +5.529e-3 \\ +2.277e-5 & +3.866e-2 & +7.155e-5 \\ +5.529e-3 & +7.155e-5 & +5.288e-2 \end{bmatrix}$
q_9 [$-26.1^\circ, 24.7^\circ$]	hrr	ba	1.840	$\begin{bmatrix} 0 & -s & c \\ 0 & c & s \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1.423e-1 \\ -1.230e-1 \\ -5.000e-2 \end{bmatrix}$	$\begin{bmatrix} -7.551e-2 \\ -2.764e-5 \\ +4.554e-2 \end{bmatrix}$	$\begin{bmatrix} +1.051e-2 & -1.011e-5 & +5.909e-4 \\ -1.011e-5 & +2.968e-2 & +3.878e-7 \\ +5.909e-4 & +3.878e-7 & +2.078e-2 \end{bmatrix}$
q_{10} [$-34.5^\circ, 40.4^\circ$]	har	hrr	2.813	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +9.694e-6 \\ +8.072e-3 \\ -9.212e-3 \end{bmatrix}$	$\begin{bmatrix} +8.421e-3 & -2.941e-6 & +5.015e-7 \\ -2.941e-6 & +7.415e-3 & +3.105e-5 \\ +5.015e-7 & +3.105e-5 & +5.116e-3 \end{bmatrix}$
q_{11} [$-92^\circ, 44.9^\circ$]	hfr	har	6.498	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -5.270e-3 \\ +1.957e-1 \\ -4.418e-3 \end{bmatrix}$	$\begin{bmatrix} +3.437e-1 & +5.185e-3 & +2.749e-5 \\ +5.185e-3 & +2.847e-2 & -1.089e-3 \\ +2.749e-5 & -1.089e-3 & +3.423e-1 \end{bmatrix}$
q_{12} [$-4.5^\circ, 120^\circ$]	kfr	hfr	3.309	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +4.400e-1 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -1.259e-3 \\ +1.336e-1 \\ +3.192e-5 \end{bmatrix}$	$\begin{bmatrix} +1.152e-1 & +3.931e-3 & +6.464e-6 \\ +3.931e-3 & +6.247e-3 & -4.682e-5 \\ +6.464e-6 & -4.682e-5 & +1.145e-1 \end{bmatrix}$
q_{13} [$-23.5^\circ, 23.9^\circ$]	sar	kfr	0.141	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +4.300e-1 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -8.113e-7 \\ +5.777e-3 \\ -6.145e-3 \end{bmatrix}$	$\begin{bmatrix} +1.617e-4 & -3.625e-9 & -1.099e-8 \\ -3.625e-9 & +9.129e-5 & +0.000e+0 \\ -1.099e-8 & +0.000e+0 & +7.697e-5 \end{bmatrix}$
q_{14} [$-49^\circ, 43.8^\circ$]	sfr	sar	1.777	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.375e-3 \\ +5.846e-2 \\ -3.099e-3 \end{bmatrix}$	$\begin{bmatrix} +1.221e-2 & +6.201e-5 & -3.424e-4 \\ +6.201e-5 & +1.039e-2 & +4.707e-6 \\ -3.424e-4 & +4.707e-6 & +1.460e-2 \end{bmatrix}$
q_{15} [$-49.5^\circ, 14.3^\circ$]	zfr	sfr	0.862	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} +1.301e-1 \\ +6.050e-2 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.218e-2 \\ +1.236e-2 \\ -1.483e-2 \end{bmatrix}$	$\begin{bmatrix} +2.466e-3 & -2.749e-4 & +1.421e-4 \\ -2.749e-4 & +2.608e-3 & +3.023e-4 \\ +1.421e-4 & +3.023e-4 & +1.091e-3 \end{bmatrix}$
q_{16} [$-24.3^\circ, 25.7^\circ$]	hrl	ba	1.840	$\begin{bmatrix} 0 & -s & c \\ 0 & -c & -s \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1.423e-1 \\ +1.230e-1 \\ -5.000e-2 \end{bmatrix}$	$\begin{bmatrix} +7.551e-2 \\ +2.764e-5 \\ +4.554e-2 \end{bmatrix}$	$\begin{bmatrix} +1.051e-2 & -1.011e-5 & -5.909e-4 \\ -1.011e-5 & +2.968e-2 & -3.878e-7 \\ -5.909e-4 & -3.878e-7 & +2.078e-2 \end{bmatrix}$
q_{17} [$-31.3^\circ, 38.1^\circ$]	hal	hrl	2.813	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -1.059e-5 \\ +8.072e-3 \\ +9.212e-3 \end{bmatrix}$	$\begin{bmatrix} +8.421e-3 & +3.085e-6 & +5.015e-7 \\ +3.085e-6 & +7.415e-3 & -3.116e-5 \\ +5.015e-7 & -3.116e-5 & +5.116e-3 \end{bmatrix}$
q_{18} [$-44^\circ, 91.3^\circ$]	hfl	hal	6.412	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +5.503e-3 \\ +1.950e-1 \\ -4.558e-3 \end{bmatrix}$	$\begin{bmatrix} +3.376e-1 & -5.452e-3 & -4.350e-5 \\ -5.452e-3 & +2.813e-2 & -1.034e-3 \\ -4.350e-5 & -1.034e-3 & +3.363e-1 \end{bmatrix}$
q_{19} [$-119.5^\circ, 4.4^\circ$]	kfl	hfl	3.309	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +4.400e-1 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +2.574e-3 \\ +1.335e-1 \\ +1.646e-4 \end{bmatrix}$	$\begin{bmatrix} +1.149e-1 & -5.552e-3 & -3.654e-5 \\ -5.552e-3 & +6.775e-3 & -1.967e-4 \\ +1.149e-1 & -5.552e-3 & -3.654e-5 \end{bmatrix}$

continued on next page...

Table H.1: Continuation (see previous page).

DoF	seg	par	m / kg	$\text{par } A_{\text{seg}}$	$\text{par } r_{\text{seg}} / \text{m}$	$\text{seg } r_{\text{CoM}} / \text{m}$	$\text{seg } \Theta_{\text{seg}}^{\text{seg}} / \text{kgm}^2$
q_{20} [-23°, 24.5°]	sal	kfl	0.141	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +4.300e-1 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -8.113e-7 \\ -5.777e-3 \\ +6.145e-3 \end{bmatrix}$	$\begin{bmatrix} +1.617e-4 & +3.625e-9 & +1.099e-8 \\ +3.625e-9 & +9.129e-5 & +0.000e+0 \\ +1.099e-8 & +0.000e+0 & +7.697e-5 \end{bmatrix}$
q_{21} [-44°, 48°]	sfl	sal	1.777	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -1.362e-3 \\ +5.846e-2 \\ -2.883e-3 \end{bmatrix}$	$\begin{bmatrix} +1.221e-2 & -6.204e-5 & +3.411e-4 \\ -6.204e-5 & +1.039e-2 & -2.225e-5 \\ +3.411e-4 & -2.225e-5 & +1.460e-2 \end{bmatrix}$
q_{22} [-15.8°, 48.8°]	zfl	sfl	0.873	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1.301e-1 \\ +6.050e-2 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} -1.229e-2 \\ +1.259e-2 \\ +2.485e-3 \end{bmatrix}$	$\begin{bmatrix} +2.481e-3 & +2.812e-4 & +1.436e-4 \\ +2.812e-4 & +2.629e-3 & -2.999e-4 \\ +1.436e-4 & -2.999e-4 & +1.109e-3 \end{bmatrix}$
q_{23} [-∞, ∞]	afr	torso	1.046	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} +1.080e-1 \\ +2.870e-1 \\ +1.590e-1 \end{bmatrix}$	$\begin{bmatrix} -1.795e-2 \\ -3.691e-4 \\ -4.267e-2 \end{bmatrix}$	$\begin{bmatrix} +3.917e-3 & -3.251e-5 & -1.218e-3 \\ -3.251e-5 & +4.700e-3 & -2.528e-5 \\ -1.218e-3 & -2.528e-5 & +1.557e-3 \end{bmatrix}$
q_{24} [-180°, 0°]	aar	afr	1.278	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ -6.400e-2 \end{bmatrix}$	$\begin{bmatrix} +3.185e-4 \\ +1.211e-1 \\ +1.353e-3 \end{bmatrix}$	$\begin{bmatrix} +2.512e-2 & -3.884e-5 & +9.593e-7 \\ -3.884e-5 & +1.582e-3 & +6.484e-6 \\ +9.593e-7 & +6.484e-6 & +2.486e-2 \end{bmatrix}$
q_{25} [-∞, ∞]	arr	aar	1.345	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.540e-2 \\ -3.702e-3 \\ -3.169e-1 \end{bmatrix}$	$\begin{bmatrix} +1.451e-1 & +9.875e-5 & +7.322e-3 \\ +9.875e-5 & +1.459e-1 & -1.597e-3 \\ +7.322e-3 & -1.597e-3 & +1.944e-3 \end{bmatrix}$
q_{26} [-158°, 24°]	efr	arr	1.305	$\begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ -3.830e-1 \end{bmatrix}$	$\begin{bmatrix} -1.962e-4 \\ +2.182e-1 \\ -3.542e-2 \end{bmatrix}$	$\begin{bmatrix} +9.251e-2 & +3.504e-5 & -4.370e-6 \\ +3.504e-5 & +3.602e-3 & +1.402e-2 \\ -4.370e-6 & +1.402e-2 & +8.973e-2 \end{bmatrix}$
q_{27} [-∞, ∞]	afl	torso	1.046	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} +1.080e-1 \\ +2.870e-1 \\ -1.590e-1 \end{bmatrix}$	$\begin{bmatrix} +1.795e-2 \\ -3.491e-4 \\ -4.259e-2 \end{bmatrix}$	$\begin{bmatrix} +3.907e-3 & +3.141e-5 & +1.210e-3 \\ +3.141e-5 & +4.690e-3 & -2.280e-5 \\ +1.210e-3 & -2.280e-5 & +1.557e-3 \end{bmatrix}$
q_{28} [-180°, 0°]	aal	afl	1.278	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ -6.400e-2 \end{bmatrix}$	$\begin{bmatrix} +4.121e-4 \\ +1.211e-1 \\ -1.368e-3 \end{bmatrix}$	$\begin{bmatrix} +2.512e-2 & -6.053e-5 & +9.593e-7 \\ -6.053e-5 & +1.582e-3 & -1.191e-5 \\ +9.593e-7 & -1.191e-5 & +2.486e-2 \end{bmatrix}$
q_{29} [-∞, ∞]	arl	aal	1.345	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.540e-2 \\ +3.667e-3 \\ -3.170e-1 \end{bmatrix}$	$\begin{bmatrix} +1.452e-1 & -9.555e-5 & +7.331e-3 \\ -9.555e-5 & +1.460e-1 & +1.587e-3 \\ +7.331e-3 & +1.587e-3 & +1.944e-3 \end{bmatrix}$
q_{30} [-24°, 158°]	efl	arl	1.305	$\begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ -3.830e-1 \end{bmatrix}$	$\begin{bmatrix} +1.935e-4 \\ +2.182e-1 \\ -3.542e-2 \end{bmatrix}$	$\begin{bmatrix} +9.251e-2 & -3.504e-5 & +4.064e-6 \\ -3.504e-5 & +3.602e-3 & +1.402e-2 \\ +4.064e-6 & +1.402e-2 & +8.973e-2 \end{bmatrix}$
q_{31} [-113°, 113°]	vp	torso	0.242	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} +1.550e-1 \\ +5.155e-1 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +1.175e-4 \\ +1.140e-4 \\ -3.463e-2 \end{bmatrix}$	$\begin{bmatrix} +3.937e-4 & -1.262e-5 & +7.440e-6 \\ -1.262e-5 & +4.438e-4 & +1.262e-5 \\ +7.440e-6 & +1.262e-5 & +1.448e-4 \end{bmatrix}$
q_{32} [-51°, 11°]	vt	vp	0.270	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} +0.000e+0 \\ +0.000e+0 \\ +0.000e+0 \end{bmatrix}$	$\begin{bmatrix} +9.498e-3 \\ +3.322e-2 \\ -4.455e-4 \end{bmatrix}$	$\begin{bmatrix} +6.124e-4 & -1.218e-4 & -5.040e-6 \\ -1.218e-4 & +2.266e-4 & -8.505e-7 \\ -5.040e-6 & -8.505e-7 & +5.125e-4 \end{bmatrix}$

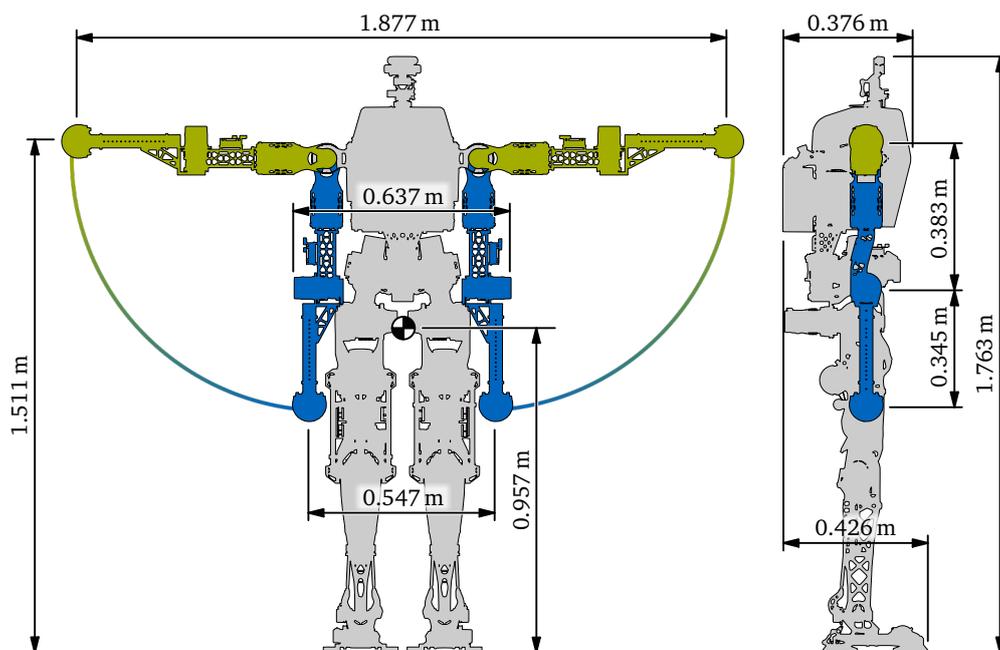


Figure H.1: Main dimensions of LOLA v1.1. Note that the height of the CoM depends on the actual pose of the robot and is given exemplary for the depicted configuration with hanging arms (blue).

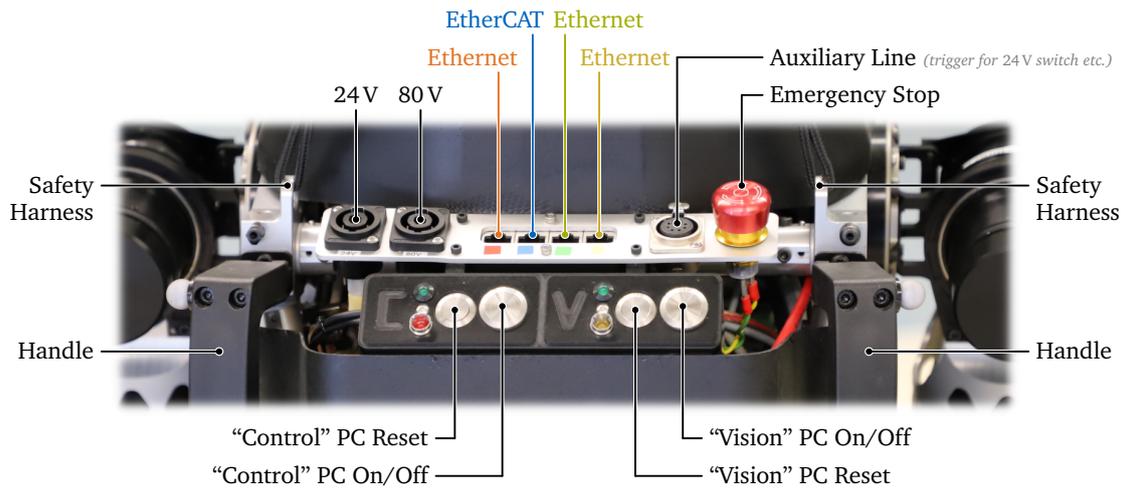


Figure H.3: Main connection terminal of LOLA v1.1 providing an interface for the external power supply (80 V and 24 V) and communication with the laboratory infrastructure (high-level signals and monitoring/logging). The emergency stop interrupts the 80 V input line feeding the motors (except vp and vt).

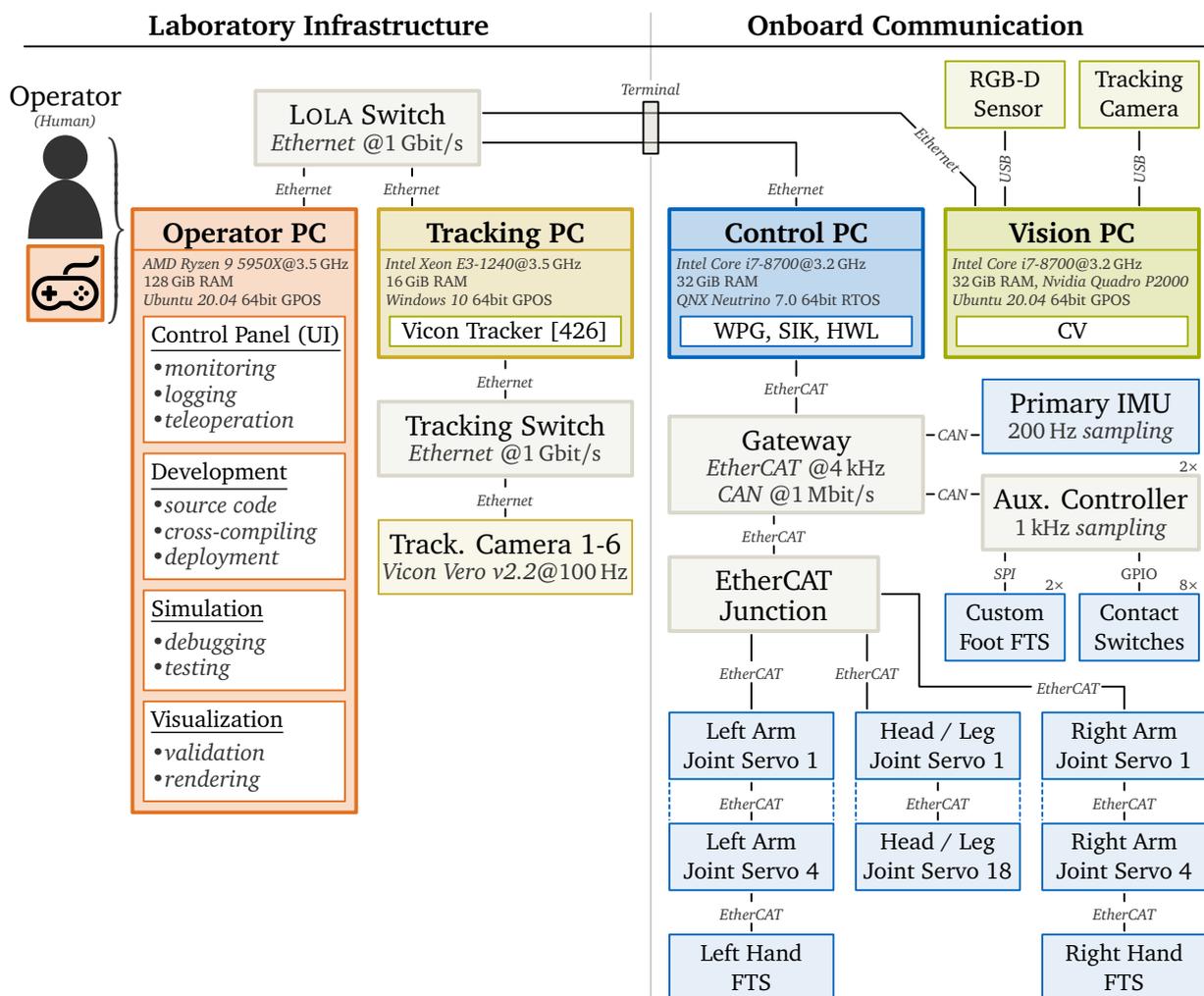


Figure H.4: Communication system of LOLA v1.1 and its connection to the “external” laboratory infrastructure. The entire locomotion system of the robot, i. e., CV, WPG, SIK, and HWL, is executed onboard. The communication with the laboratory infrastructure is used only for receiving high-level signals (e. g. start/stop, walking speed, direction, goal, etc.) and data transmission for monitoring and logging. The data of the external motion capture (“tracking”) system is only used for post-experimental (offline) analysis and is *not* used by the locomotion system of LOLA.

H.3 Calibration

For kinematic calibration, MARKUS SCHWIENBACHER designed a special jig to bring the robot into a well-defined pose, see [291, p. 124f]. Once the robot is attached to this fixture, the joint angles measured by the link-side absolute encoders are saved, which represent constant offsets to the “zero” position of each DoF. The offsets are used after startup to get the robot’s current joint-space configuration. This makes “homing”, i. e., moving each DoF into one of its limits, unnecessary. An exception are the DoF v_p and v_t , which do not feature an absolute encoder. Since self-collisions are not possible for these two DoF, an automatic homing is triggered at every startup which neither represents a dangerous maneuver nor requires much time.

Due to the revision of the upper body, the original calibration jig did not fit the hardware anymore. Instead of modifying the existing fixture, it was decided to create a new system, which

- simplifies the calibration process (previously the attachment of the robot was tedious),
- allows joint calibration with LOLA in *horizontal* position (the robot’s weight fixes it to the jig – tension belts are used to secure it against detaching from the frame),
- allows IMU calibration with LOLA in *vertical* position (calibrating the main IMU in a horizontal pose is problematic due to the deformation of the wire rope isolators), and
- simultaneously represents a robust fixture for transport (e. g. inside a wooden box).

The new fixture was designed in large parts by the student assistants DANIEL PÖLZLEITNER and REINHOLD POSCHER. It basically consists of interconnected (standardized) aluminum profiles, see Figure H.5. The actual interface to the robot is realized by milled structures integrating high-precision bushings. These represent counterparts to calibration bolts permanently attached to the robot. For highest accuracy, the holes for the bushings and all surfaces meant to get in contact with the robot were manufactured *after* the fixture was assembled (all-in-one). This required processing on a large-scale CNC milling machine through an external contractor.

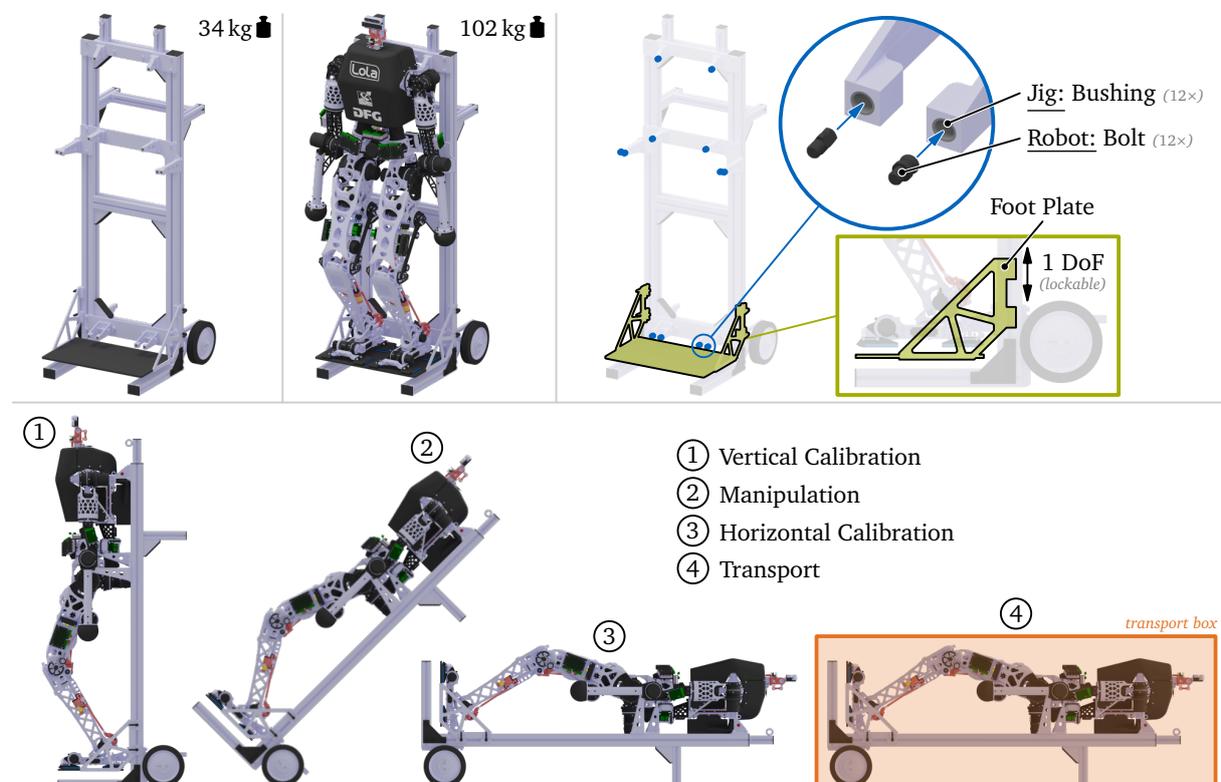


Figure H.5: New calibration jig of LOLA v1.1 used for kinematic calibration and as fixture for transportation. The wheels allow handling by a single person. The movable foot plate is meant for additional support during transport.

Co-authored Publications

I.1 Scientific Publications

- [1] **Seiwald, P.**, Wu, S.-C., Sygulla, F., Berninger, T. F. C., Staufenberg, N.-S., Sattler, M. F., Neuburger, N., Rixen, D., and Tombari, F. “LOLA v1.1 – An Upgrade in Hardware and Software Design for Dynamic Multi-Contact Locomotion”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Munich, Germany, July 2021, pp. 9–16. DOI: 10.1109/HUMANOIDS47582.2021.9555790.
- [2] **Seiwald, P.** and Rixen, D. “Fast Approximation of Over-Determined Second-Order Linear Boundary Value Problems by Cubic and Quintic Spline Collocation”. In: *Robotics* 9.2 (June 2020). DOI: 10.3390/robotics9020048.
- [3] **Seiwald, P.**, Sygulla, F., Staufenberg, N.-S., and Rixen, D. “Quintic Spline Collocation for Real-Time Biped Walking-Pattern Generation with variable Torso Height”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Toronto, Canada, Oct. 2019, pp. 56–63. DOI: 10.1109/Humanoids43949.2019.9035076.
- [4] **Seiwald, P.**, Leyerer, A., Sygulla, F., and Rixen, D. “Parameter Optimization of a Reduced Model for Multi-Contact Locomotion of Humanoid Robots”. In: *Proceedings in Applied Mathematics and Mechanics* 18.1 (Dec. 2018), pp. 1–2. DOI: 10.1002/pamm.201800138.
- [5] Berninger, T. F. C., **Seiwald, P.**, Sygulla, F., and Rixen, D. J. “Evaluating the Mechanical Redesign of a Biped Walking Robot Using Experimental Modal Analysis”. In: *Topics in Modal Analysis & Testing, Volume 8*. Ed. by Dilworth, B. J. and Mains, M. Springer International Publishing, 2022, pp. 45–52. DOI: 10.1007/978-3-030-75996-4_6.
- [6] Wahrmann, D., Hildebrandt, A.-C., Bates, T., Wittmann, R., Sygulla, F., **Seiwald, P.**, and Rixen, D. “Vision-Based 3D Modeling of Unknown Dynamic Environments for Real-Time Humanoid Navigation”. In: *International Journal of Humanoid Robotics* 16.1 (2019). DOI: 10.1142/S0219843619500026.
- [7] Gienger, M., Ruiken, D., Bates, T., Regaieg, M., Meißner, M., Kober, J., **Seiwald, P.**, and Hildebrandt, A.-C. “Human-Robot Cooperative Object Manipulation with Contact Changes”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, Oct. 2018, pp. 1354–1360. DOI: 10.1109/IROS.2018.8594140.
- [8] Hildebrandt, A.-C., Ritt, K., Wahrmann, D., Wittmann, R., Sygulla, F., **Seiwald, P.**, Rixen, D., and Buschmann, T. “Torso height optimization for bipedal locomotion”. In: *International Journal of Advanced Robotic Systems* 15.5 (Oct. 2018), pp. 1–11. DOI: 10.1177/1729881418804442.
- [9] Hildebrandt, A.-C., Schwerd, S., Wittmann, R., Wahrmann, D., Sygulla, F., **Seiwald, P.**, Rixen, D., and Buschmann, T. “Kinematic Optimization for Bipedal Robots: A Framework for Real-Time Collision Avoidance”. In: *Autonomous Robots* (Aug. 2018). DOI: 10.1007/s10514-018-9789-3.

- [10] Sygulla, F., Wittmann, R., **Seiwald, P.**, Berninger, T., Hildebrandt, A.-C., Wahrmann, D., and Rixen, D. “An EtherCAT-Based Real-Time Control System Architecture for Humanoid Robots”. In: *IEEE International Conference on Automation Science and Engineering (CASE)*. Munich, Germany, Aug. 2018, pp. 483–490. DOI: 10.1109/COASE.2018.8560532.
- [11] Wahrmann, D., Wu, Y., Sygulla, F., Hildebrandt, A.-C., Wittmann, R., **Seiwald, P.**, and Rixen, D. “Time-variable, event-based walking control for biped robots”. In: *International Journal of Advanced Robotic Systems* 15.2 (Apr. 2018). DOI: 10.1177/1729881418768918.
- [12] Sygulla, F., Wittmann, R., **Seiwald, P.**, Hildebrandt, A.-C., Wahrmann, D., and Rixen, D. “Hybrid Position/Force Control for Biped Robot Stabilization with Integrated Center of Mass Dynamics”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Birmingham, UK, Nov. 2017, pp. 742–748. DOI: 10.1109/HUMANOIDS.2017.8246955.
- [13] Hildebrandt, A.-C., Klischat, M., Wahrmann, D., Wittmann, R., Sygulla, F., **Seiwald, P.**, Rixen, D., and Buschmann, T. “Real-Time Path Planning in Unknown Environments for Bipedal Robots”. In: *IEEE Robotics and Automation Letters* 2.4 (Oct. 2017), pp. 1856–1863. DOI: 10.1109/LRA.2017.2712650.
- [14] Wahrmann, D., Knopp, T., Wittmann, R., Hildebrandt, A.-C., Sygulla, F., **Seiwald, P.**, Rixen, D., and Buschmann, T. “Modifying the Estimated Ground Height to Mitigate Error Effects on Bipedal Robot Walking”. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. Munich, Germany, July 2017, pp. 1471–1476. DOI: 10.1109/AIM.2017.8014226.

I.2 Published Software

Each of the following entries redirects to a “static” (permanently archived at the TUM, see the field DOI) and “dynamic” (code repository, see the field URL) version of the dataset.

- [15] **Seiwald, P.** and Sygulla, F. *broccoli: Beautiful Robot C++ Code Library*. Release v3.1.0. 2022. DOI: 10.14459/2022mp1686390.001. URL: <https://gitlab.lrz.de/AM/broccoli> (visited on 03/02/2022).
- [16] **Seiwald, P.** *am2b-vision-interface: Interface between a Computer Vision System and the Walking Pattern Generation System of the Humanoid Robot LOLA*. Release v2.5.0. 2022. DOI: 10.14459/2021mp1686394.001. URL: <https://gitlab.lrz.de/AM/lola/am2b-vision-interface> (visited on 03/02/2022).

I.3 Published Videos

Each of the following entries redirects to a “static” (permanently archived at the TUM, see the field DOI) and “dynamic” (*YouTube* link, see the field URL) version of the dataset.

- [17] **Seiwald, P.** and Wu, S.-C. *Humanoid Robot LOLA - Vision Guided Autonomous Multi-Contact Locomotion*. Chair of Applied Mechanics, Technical University of Munich. Jan. 31, 2022. DOI: 10.14459/2022mp1686386. URL: <https://youtu.be/ovG2Rz9-1p8> (visited on 02/01/2022).

- [18] **Seiwald, P.** *Humanoid Robot LOLA - Walking Pattern Generation for Autonomous Multi-Contact Locomotion*. Chair of Applied Mechanics, Technical University of Munich. Jan. 31, 2022. DOI: 10.14459/2022mp1686396. URL: https://youtu.be/mGlsc_revMc (visited on 02/01/2022).
- [19] **Seiwald, P.**, Wu, S.-C., Sygulla, F., Berninger, T. F. C., Staufenberg, N.-S., Sattler, M. F., Neuburger, N., Rixen, D., and Tombari, F. *LOLA v1.1 - An Upgrade in Hardware and Software Design for Dynamic Multi-Contact Locomotion*. Chair of Applied Mechanics, Technical University of Munich. July 20, 2021. DOI: 10.14459/2022mp1686398. URL: <https://youtu.be/T0CiZQbd9H0> (visited on 02/01/2022).
- [20] **Seiwald, P.** and Sygulla, F. *Humanoid Robot LOLA - Dynamic Multi-Contact Locomotion*. Chair of Applied Mechanics, Technical University of Munich. Mar. 16, 2021. DOI: 10.14459/2021mp1686400. URL: <https://youtu.be/gUNZ0AmLiWU> (visited on 02/01/2022).
- [21] **Seiwald, P.** and Sygulla, F. *Humanoid Robot LOLA v1.1 - Validation of Hardware Upgrade - Initial Tests*. Chair of Applied Mechanics, Technical University of Munich. Dec. 18, 2020. DOI: 10.14459/2020mp1686402. URL: <https://youtu.be/JCYmq6uOEEc> (visited on 02/01/2022).
- [22] **Seiwald, P.** et al. *Humanoid Robot LOLA v1.1 - Hardware Upgrade for Multi-Contact Locomotion*. Chair of Applied Mechanics, Technical University of Munich. Oct. 26, 2020. DOI: 10.14459/2020mp1686404. URL: <https://youtu.be/mpDqMFppT68> (visited on 02/01/2022).
- [23] **Seiwald, P.**, Sygulla, F., Staufenberg, N.-S., and Rixen, D. *Smooth Real-Time Walking-Pattern Generation for Humanoid Robot LOLA*. Chair of Applied Mechanics, Technical University of Munich. July 3, 2019. DOI: 10.14459/2019mp1686406. URL: https://youtu.be/piQm_oTYXic (visited on 02/01/2022).
- [24] **Seiwald, P.**, Sygulla, F., et al. *The Humanoid Robot Lola walks Outside*. Chair of Applied Mechanics, Technical University of Munich. May 24, 2019. DOI: 10.14459/2019mp1686407. URL: <https://youtu.be/cNkQT2SUegE> (visited on 02/01/2022).
- [25] Buschmann, T., Wahrmann, D., Hildebrandt, A.-C., Wittmann, R., **Seiwald, P.**, et al. *Overview Humanoid Robot LOLA 2018*. Chair of Applied Mechanics, Technical University of Munich. Apr. 15, 2019. DOI: 10.14459/2019mp1686408. URL: <https://youtu.be/EctlCoMPyS4> (visited on 02/01/2022).

I.4 Press Reports (Indirect Publications)

The following press reports summarize core achievements of the multi-contact project and bring them closer to the general public. These reports have been created on the basis of direct and indirect (conference presentation) communication with the author of this thesis.

- [26] Ackerman, E. *IEEE Spectrum: Bipedal Robots Are Learning To Move With Arms as Well as Legs*. Apr. 1, 2021. URL: <https://spectrum.ieee.org/bipedal-robot-learning-to-move-arms-legs> (visited on 05/09/2022).
- [27] Marsiske, H.-A. *Humanoids: Wie Roboter Arme und Hände zum Laufen nutzen*. July 20, 2021. URL: <https://www.heise.de/news/Humanoids-Wie-Roboter-Arme-und-Haende-zum-Laufen-nutzen-6143268.html> (visited on 08/16/2022).

Appendix J

Supervised Student Theses

Some results of this dissertation have been obtained in collaboration with students assistants. This appendix explicitly lists student theses which have been conducted under the supervision of the author of this dissertation. In addition to a citation in the corresponding chapter, a brief summary is given in the following to highlight the links. Note that some works cannot be assigned to a specific chapter of this document, thus, they have not been cited yet. The following list is sorted by descending relevance for this dissertation.

Kinematic Structure Optimization In [28], NEUBURGER developed a generic framework for optimizing kinematic structures. One use case was the optimization of LOLA’s new arm topology, see Section 3.3. This dissertation uses the computed optimum topology and certain concepts from workspace evaluation to perform the high-resolution analysis shown in Figure 3.6.

- [28] Neuburger, N. “Kinematic Structure Optimization for Humanoid Robots”. Master’s thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2019. URL: <https://mediatum.ub.tum.de/1523789>.

Parametrization of the Five-Mass Model In [29], LEYERER performed a parameter optimization to find the ideal mass distribution of the five-mass model. This dissertation uses the optimum foot and hand masses (as proportions of the total mass) computed by LEYERER. See Section 4.5.2 for details and [4] for a summary of [29].

- [29] Leyerer, A. “Optimierung und Validierung Reduzierter Dynamikmodelle für Humanoide Roboter”. Term paper. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2017.

SSV Library In [30], POSCHER realized a modern implementation of the SSV library originally developed by SCHWIENBACHER et al. [371]. The work of POSCHER has been refactored, extended, and integrated into *Broccoli* [15] by the author of this dissertation. The SSV library is used for efficient collision and proximity tests. See Appendix C for details.

- [30] Poscher, R. “Effiziente Abstandsberechnungen mit Swept-Sphere Volumen für Echtzeit Kollisionsvermeidung in der Robotik”. Bachelor’s thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2020. URL: <https://mediatum.ub.tum.de/1580089>.

Multi-Level Contact Planning The real-time contact planner presented in Chapter 5 is realized through a hierarchical graph search on multiple levels of detail. An early feasibility study on using a multi-level graph search for contact planning has been conducted in [31] by HÖRMANN. The work of HÖRMANN is based on another pre-study by MOHAMED REGAIEG as part of an internship which was supervised jointly by the author of this dissertation and MICHAEL GIENGER. Subsequent investigations by GIENGER et al. also led to the publication [7].

- [31] Hörmann, B. “Multi-Level Contact Planning for Humanoid Robots”. Bachelor’s thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2019.

Contact Surface Evaluation and Hand Motion In preparation of developing a WPG capable of multi-contact locomotion, two pre-studies have been conducted. In the first study, RODRÍGUEZ [32] developed methods to evaluate a surface (represented by a triangle mesh) for potential hand contacts. The second study by RÓDER [33] investigated optimal hand trajectories to establish hand support during locomotion. Both, [32] and [33], represent early feasibility studies which did not make it into the final WPG of LOLA, however, they helped in its development.

- [32] Rodríguez, I. “Search and Evaluation of Optimal Support Areas for Humanoid Robots”. Interdisciplinary project. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2017.
- [33] Roder, S. “Planung optimaler Armtrajektorien für Humanoide Roboter”. Master’s thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2017.

Simulation of Vision Sensor In [34], BECKERT developed a simulation of the *Asus Xtion Pro Live* sensor previously used by LOLA. It takes a 3D model of the scene as input and computes the corresponding RGB-D image as it would be delivered by the camera in a real experiment. For a realistic emulation, the software mimics the characteristic noise of the sensor by custom *OpenGL* shaders. Due to the lack of time, the work of BECKERT has not been used in this dissertation. However, it may be used in future to improve the synthesis of the environment model (see Section 7.4) in particular through consideration of view occlusion.

- [34] Beckert, D. “Simulation of a Computer Vision System for Humanoid Robots”. Interdisciplinary project. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2018.

SLAM for LOLA The work of DIEMER [35] represents an early pre-study investigating the application of existing SLAM algorithms on LOLA. The main objective was a feasibility analysis for improving the localization of the robot by compensating drift due to slippage. This task is now taken over by the new CV system developed by WU et al. (see Section 4.4).

- [35] Diemer, A. “Evaluation of SLAM Algorithms on the Robot LOLA”. Interdisciplinary project. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2018.

Autonomous Safety Frame Independent of the multi-contact revision of the robot, the author of this dissertation also supervised the upgrade of LOLA’s safety frame for autonomous operation (safety harness automatically following the robot). The new hardware of the lab equipment was designed, realized, and initially tested by SCHLEIBINGER [36]. Subsequently, CHBALY [37] and ERJIAGE [38] worked on the software for low-level and high-level control, respectively.

- [36] Schleibinger, M. “Development and Realization of an Autonomous Safety Frame for Humanoid Robots”. Term paper. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2017.
- [37] Chbaly, K. “Autonomous Safety Frame for Humanoid Robots: Low-Level Control and Initial Operation”. Interdisciplinary project. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2019.
- [38] Erjiage, G. “Autonomous Safety Frame for Humanoid Robots: High-level Control and Human Machine Interface”. Interdisciplinary project. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2019.

Appendix K

Bibliography

- [39] Ackerman, E. *IEEE Spectrum: Boston Dynamics' AlphaDog Quadruped Robot Prototype on Video*. Sept. 30, 2011. URL: <https://spectrum.ieee.org/boston-dynamics-alphadog-prototype-on-video> (visited on 03/02/2022).
- [40] Ackerman, E. *IEEE Spectrum: Boston Dynamics' Cheetah Robot Now Faster than Fastest Human*. Sept. 5, 2012. URL: <https://spectrum.ieee.org/boston-dynamics-cheetah-robot-now-faster-than-fastest-human> (visited on 03/02/2022).
- [41] Ackerman, E. *IEEE Spectrum: Whoa: Boston Dynamics Announces New WildCat Quadruped Robot*. Oct. 4, 2013. URL: <https://spectrum.ieee.org/whoa-boston-dynamics-announces-new-wildcat-quadruped> (visited on 03/02/2022).
- [42] Ackerman, E. *IEEE Spectrum: Agility Robotics Introduces Cassie, a Dynamic and Talented Robot Delivery Ostrich*. Feb. 9, 2017. URL: <https://spectrum.ieee.org/agility-robotics-introduces-cassie-a-dynamic-and-talented-robot-delivery-ostrich> (visited on 03/02/2022).
- [43] Ackerman, E. *IEEE Spectrum: Toyota Gets Back Into Humanoid Robots With New T-HR3*. Nov. 22, 2017. URL: <https://spectrum.ieee.org/toyota-gets-back-into-humanoid-robots-with-new-thr3> (visited on 03/02/2022).
- [44] Ackerman, E. *IEEE Spectrum: ANYbotics Introduces Sleek New ANYmal C Quadruped*. Aug. 22, 2019. URL: <https://spectrum.ieee.org/anybotics-introduces-sleek-new-anymal-c-quadruped> (visited on 03/02/2022).
- [45] Ackerman, E. *IEEE Spectrum: Agility Robotics Unveils Upgraded Digit Walking Robot*. Oct. 14, 2019. URL: <https://spectrum.ieee.org/agility-robotics-digit-v2-biped-robot> (visited on 03/02/2022).
- [46] Ackerman, E. *IEEE Spectrum: How Boston Dynamics Taught Its Robots to Dance*. Jan. 7, 2021. URL: <https://spectrum.ieee.org/how-boston-dynamics-taught-its-robots-to-dance> (visited on 03/15/2022).
- [47] Ackerman, E. *IEEE Spectrum: Unitree's Go1 Robot Dog Looks Pretty Great, Costs Just USD \$2700*. June 9, 2021. URL: <https://spectrum.ieee.org/unitrees-go1-robot-dog-looks-pretty-great-costs-just-usd-2700> (visited on 03/02/2022).
- [48] Acontis Technologies. *EtherCAT Master Stack Software Development Kit*. URL: <https://www.acontis.com/de/ethercat-master.html> (visited on 08/18/2022).
- [49] Adamy, J. *Nichtlineare Systeme und Regelungen*. 2nd ed. Berlin Heidelberg: Springer, 2014. DOI: 10.1007/978-3-642-45013-6.
- [50] Advantech Co., Ltd. *AIMB-276 - 8th/9th Gen Intel Core i7/i5/i3 (Coffee Lake) Mini-ITX*. URL: https://www.advantech.com/products/68ccaea2-9ff5-4f85-97f2-3d11244b0a08/aimb-276/mod_30c35540-bf71-499b-b37e-b024e9f33357 (visited on 08/05/2022).
- [51] Agility Robotics. *IEEE Robots: Cassie*. 2016. URL: <https://robots.ieee.org/robots/cassie> (visited on 03/02/2022).

- [52] Agility Robotics. *IEEE Robots: Digit*. 2019. URL: <https://robots.ieee.org/robots/digit> (visited on 03/02/2022).
- [53] Ahlberg, J. H., Nilson, E. N., and Walsh, J. L. *The Theory of Splines and Their Applications*. Mathematics in Science and Engineering. New York, London: Academic Press, 1967. ISBN: 978-0-080-95545-2.
- [54] Ahlberg, J. and Ito, T. “A Collocation Method for Two-Point Boundary Value Problems”. In: *Mathematics of Computation* 29.131 (1975), pp. 761–776. DOI: 10.1090/S0025-5718-1975-0375785-7.
- [55] AIST and Kawada Industries. *IEEE Robots: HRP-2*. 2002. URL: <https://robots.ieee.org/robots/hrp2> (visited on 03/02/2022).
- [56] AIST. *IEEE Robots: HRP-5P*. 2018. URL: <https://robots.ieee.org/robots/hrp5p> (visited on 03/02/2022).
- [57] Akram, G. and Tariq, H. “Quintic spline collocation method for fractional boundary value problems”. In: *Journal of the Association of Arab Universities for Basic and Applied Sciences* 23.1 (2017), pp. 57–65. DOI: 10.1016/j.jaubas.2016.03.003.
- [58] Albers, A., Brudniok, S., Ottnad, J., Sauter, C., and Sedchaicharn, K. “Upper Body of a new Humanoid Robot - the Design of ARMAR III”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Genova, Italy, Dec. 2006, pp. 308–313. DOI: 10.1109/ICHR.2006.321289.
- [59] An, S. and Lee, D. “Prioritized Inverse Kinematics with Multiple Task Definitions”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, USA, May 2015, pp. 1423–1430. DOI: 10.1109/ICRA.2015.7139376.
- [60] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., and Yoon, W.-K. “RT-Middleware: Distributed Component Middleware for RT (Robot Technology)”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, AB, Canada, Aug. 2005, pp. 3933–3938. DOI: 10.1109/IROS.2005.1545521.
- [61] Asano, Y., Kozuki, T., Ookubo, S., Kawasaki, K., Shirai, T., Kimura, K., Okada, K., and Inaba, M. “A Sensor-driver Integrated Muscle Module with High-Tension Measurability and Flexibility for Tendon-Driven Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, Sept. 2015, pp. 5960–5965. DOI: 10.1109/IROS.2015.7354225.
- [62] Asano, Y. et al. “Human Mimetic Musculoskeletal Humanoid Kengoro toward Real World Physically Interactive Actions”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 876–883. DOI: 10.1109/HUMANOIDS.2016.7803376.
- [63] Ascher, U. “Solving Boundary-Value Problems with a Spline-Collocation Code”. In: *Journal of Comp. Physics* 34.3 (1980), pp. 401–413. DOI: 10.1016/0021-9991(80)90097-2.
- [64] Ascher, U., Pruess, S., and Russell, R. D. “On Spline Basis Selection for Solving Differential Equations”. In: *SIAM Journal on Numerical Analysis* 20.1 (1983), pp. 121–142. DOI: 10.1137/0720009.
- [65] Ascher, U. and Spiteri, R. “Collocation Software for Boundary Value Differential-Algebraic Equations”. In: *SIAM Journal on Scientific Computing* 15 (Dec. 1995). DOI: 10.1137/0915056.
- [66] Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., and Dillmann, R. “ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Genova, Italy, Dec. 2006, pp. 169–175. DOI: 10.1109/ICHR.2006.321380.

- [67] Asfour, T., Schill, J., Peters, H., Klas, C., Bücker, J., Sander, C., Schulz, S., Kargov, A., Werner, T., and Bartenbach, V. “ARMAR-4: A 63 DOF Torque Controlled Humanoid Robot”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Atlanta, USA, Oct. 2013, pp. 390–396. DOI: 10.1109/HUMANOIDS.2013.7030004.
- [68] Asfour, T., Waechter, M., Kaul, L., Rader, S., Weiner, P., Ottenhaus, S., Grimm, R., Zhou, Y., Grotz, M., and Paus, F. “ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real-World Scenarios”. In: *IEEE Robotics Automation Magazine* 26.4 (2019), pp. 108–121. DOI: 10.1109/MRA.2019.2941246.
- [69] ASUS. *Multimedia - Xtion PRO LIVE*. Apr. 4, 2015. URL: https://web.archive.org/web/20150404000121/http://www.asus.com/Multimedia/Xtion_PRO_LIVE/ (visited on 05/23/2022).
- [70] Atkeson, C. G. et al. “No Falls, No Resets: Reliable Humanoid Behavior in the DARPA Robotics Challenge”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea, Nov. 2015, pp. 623–630. DOI: 10.1109/HUMANOIDS.2015.7363436.
- [71] Audren, H. and Kheddar, A. “3-D Robust Stability Polyhedron in Multicontact”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 388–403. DOI: 10.1109/TRO.2017.2786683.
- [72] Baerentzen, J. A. and Aanaes, H. “Signed Distance Computation Using the Angle Weighted Pseudonormal”. In: *IEEE Transactions on Visualization and Computer Graphics* 11.3 (2005), pp. 243–253. DOI: 10.1109/TVCG.2005.49.
- [73] Bagheri, M., Ajoudani, A., Lee, J., Caldwell, D. G., and Tsagarakis, N. G. “Kinematic Analysis and Design Considerations for Optimal Base Frame Arrangement of Humanoid Shoulders”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, Washington, May 2015, pp. 2710–2715. DOI: 10.1109/ICRA.2015.7139566.
- [74] Barabanov, M. “A Linux-based Real-Time Operating System”. Master’s thesis. Socorro, New Mexico, USA: New Mexico Institute of Mining and Technology, 1997.
- [75] Berninger, T. F. C., Sygulla, F., Fuderer, S., and Rixen, D. J. “Experimental Analysis of Structural Vibration Problems of a Biped Walking Robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, May 2020. DOI: 10.1109/ICRA40945.2020.9197282.
- [76] Besl, P. J. and McKay, N. D. “A Method for Registration of 3-D Shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.
- [77] Beckhoff Automation. *Ethernet for Control Automation Technology (EtherCAT)*. URL: <https://www.ethercat.org> (visited on 03/11/2022).
- [78] Benallegue, M., Escande, A., Miossec, S., and Kheddar, A. “Fast C1 Proximity Queries Using Support Mapping of Sphere-Torus-Patches Bounding Volumes”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 483–488. DOI: 10.1109/ROBOT.2009.5152722.
- [79] Bessonnet, G., Chessé, S., and Sardain, P. “Optimal Gait Synthesis of a Seven-Link Planar Biped”. In: *The International Journal of Robotics Research* 23.10-11 (2004), pp. 1059–1073. DOI: 10.1177/027836490404047393.
- [80] Bialecki, B. and Fairweather, G. “Orthogonal spline collocation methods for partial differential equations”. In: *Journal of Computational and Applied Mathematics* 128.1 (2001), pp. 55–82. DOI: 10.1016/S0377-0427(00)00509-4.

- [81] Bjelonic, M., Grandia, R., Harley, O., Galliard, C., Zimmermann, S., and Hutter, M. “Whole-Body MPC and Online Gait Sequence Generation for Wheeled-Legged Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic, Sept. 2021, pp. 8388–8395. DOI: 10.1109/IROS51168.2021.9636371.
- [82] Bjelonic, M. et al. *Complex motion decomposition: combining offline motion libraries with online MPC*. Robotic Systems Lab, ETH Zürich. Oct. 4, 2021. URL: <https://youtu.be/39rRhTqcQc0> (visited on 03/04/2022).
- [83] BlackBerry. *QNX Neutrino RTOS*. URL: <https://blackberry.qnx.com/en/products/foundation-software/qnx-rtos> (visited on 03/17/2022).
- [84] Bledt, G., Powell, M. J., Katz, B., Di Carlo, J., Wensing, P. M., and Kim, S. “MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, Oct. 2018, pp. 2245–2252. DOI: 10.1109/IROS.2018.8593885.
- [85] Blender Foundation. *Home of the Blender project – Free and Open 3D Creation Software*. Release v2.93.7. URL: <https://www.blender.org> (visited on 07/07/2022).
- [86] Blickhan, R. “The spring-mass model for running and hopping”. In: *Journal of Biomechanics* 22.11 (1989), pp. 1217–1227. DOI: 10.1016/0021-9290(89)90224-8.
- [87] Bloesch, M., Omari, S., Hutter, M., and Siegwart, R. “Robust Visual Inertial Odometry Using a Direct EKF-Based Approach”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, Sept. 2015, pp. 298–304. DOI: 10.1109/IROS.2015.7353389.
- [88] Böhm, W. “On Cubics: A Survey”. In: *Computer Graphics and Image Processing* 19.3 (1982), pp. 201–226. DOI: 10.1016/0146-664X(82)90009-0.
- [89] Böhm, W. “Efficient Evaluation of Splines”. In: *Computing* 33.2 (1984), pp. 171–177. DOI: 10.1007/BF02240188.
- [90] Boston Dynamics. *Legacy Robots*. URL: <https://www.bostondynamics.com/legacy> (visited on 03/02/2022).
- [91] Boston Dynamics. *IEEE Robots: Atlas*. 2016. URL: <https://robots.ieee.org/robots/atlas2016> (visited on 03/02/2022).
- [92] Boston Dynamics. *IEEE Robots: Spot*. 2016. URL: <https://robots.ieee.org/robots/spotmini> (visited on 03/02/2022).
- [93] Bourke, P. *Polygonising a scalar field (Marching Cubes)*. May 1994. URL: <http://paulbourke.net/geometry/polygonise/> (visited on 07/20/2022).
- [94] Bouyarmane, K., Caron, S., Escande, A., and Kheddar, A. “Multi-contact Motion Planning and Control”. In: *Humanoid Robotics: A Reference*. Ed. by Goswami, A. and Vadakkepat, P. Dordrecht: Springer Netherlands, 2019, pp. 1763–1804. DOI: 10.1007/978-94-007-6046-2_32.
- [95] Bretl, T. and Lall, S. “Testing Static Equilibrium for Legged Robots”. In: *IEEE Transactions on Robotics* 24.4 (Aug. 2008), pp. 794–807. DOI: 10.1109/TRO.2008.2001360.
- [96] Brossette, S., Vaillant, J., Keith, F., Escande, A., and Kheddar, A. “Point-Cloud Multi-Contact Planning for Humanoids: Preliminary Results”. In: *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. Nov. 2013, pp. 19–24. DOI: 10.1109/RAM.2013.6758553.
- [97] Buschmann, T., Lohmeier, S., Ulbrich, H., and Pfeiffer, F. “Optimization Based Gait Pattern Generation for a Biped Robot”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Dec. 2005, pp. 98–103. DOI: 10.1109/ICHR.2005.1573552.

- [98] Buschmann, T., Lohmeier, S., Bachmayer, M., Ulbrich, H., and Pfeiffer, F. “A Collocation Method for Real-Time Walking Pattern Generation”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Pittsburgh, USA, Nov. 2007, pp. 1–6. DOI: 10.1109/ICHR.2007.4813841.
- [99] Buschmann, T., Lohmeier, S., Schwienbacher, M., Favot, V., Ulbrich, H., von Hundelshausen, F., Rohe, G., and Wuensche, H.-J. “Walking in Unknown Environments – a Step Towards More Autonomy”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nashville, TN, USA, Dec. 2010, pp. 237–244. DOI: 10.1109/ICHR.2010.5686338.
- [100] Buschmann, T. “Simulation and Control of Biped Walking Robots”. Dissertation. Germany: Technical University of Munich, 2010. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20101201-997204-1-6>.
- [101] Buschmann, T., Favot, V., Lohmeier, S., Schwienbacher, M., and Ulbrich, H. “Experiments in Fast Biped Walking”. In: *IEEE International Conference on Mechatronics*. Istanbul, Turkey, Apr. 2011, pp. 863–868. DOI: 10.1109/ICMECH.2011.5971235.
- [102] Buschmann, T., Ewald, A., Ulbrich, H., and Büschges, A. “Event-Based Walking Control – From Neurobiology to Biped Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura-Algarve, Portugal, Oct. 2012, pp. 1793–1800. DOI: 10.1109/IROS.2012.6385783.
- [103] Buschmann, T., Wittmann, R., Schwienbacher, M., and Ulbrich, H. “A Method for Real-Time Kineto-Dynamic Trajectory Generation”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Osaka, Japan, Nov. 2012, pp. 190–197. DOI: 10.1109/HUMANOIDS.2012.6651519.
- [104] Buschmann, T. “Dynamics and Control of Redundant Robots”. Habilitation. Germany: Technical University of Munich, 2014. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20150605-1254716-1-0>.
- [105] Butteltmann, M. and Lohmann, B. “Optimization with Genetic Algorithms and an Application for Model Reduction”. In: *at – Automatisierungstechnik* 52.4 (2004), pp. 151–163. DOI: doi:10.1524/auto.52.4.151.29416.
- [106] Cafolla, D. and Ceccarelli, M. “An Experimental Validation of a Novel Humanoid Torso”. In: *Robotics and Autonomous Systems* 91 (2017), pp. 299–313. DOI: 10.1016/j.robot.2017.02.005.
- [107] CAN in Automation (CiA). *Controller Area Network (CAN)*. URL: <https://www.can-cia.org> (visited on 03/11/2022).
- [108] Caron, S., Pham, Q.-C., and Nakamura, Y. “Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics”. In: *Robotics: Science and System*. July 2015. DOI: 10.15607/RSS.2015.XI.028.
- [109] Caron, S. and Kheddar, A. “Multi-contact Walking Pattern Generation based on Model Preview Control of 3D COM Accelerations”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 550–557. DOI: 10.1109/HUMANOIDS.2016.7803329.
- [110] Caron, S., Pham, Q.-C., and Nakamura, Y. “ZMP Support Areas for Multicontact Mobility Under Frictional Constraints”. In: *IEEE Transactions on Robotics* 33.1 (Feb. 2017), pp. 67–80. DOI: 10.1109/TRO.2016.2623338.
- [111] Carpentier, J., Tonneau, S., Naveau, M., Stasse, O., and Mansard, N. “A Versatile and Efficient Pattern Generator for Generalized Legged Locomotion”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, May 2016, pp. 3555–3561. DOI: 10.1109/ICRA.2016.7487538.

- [112] Catmull, E. and Clark, J. “Recursively generated B-spline surfaces on arbitrary topological meshes”. In: *Computer-Aided Design* 10.6 (1978), pp. 350–355. DOI: 10.1016/0010-4485(78)90110-0.
- [113] Chair of Applied Mechanics, Technical University of Munich. *IEEE Robots: Lola*. 2010. URL: <https://robots.ieee.org/robots/lola> (visited on 08/05/2022).
- [114] Chestnutt, J. “Navigation Planning for Legged Robots”. Dissertation. Pennsylvania, USA: Carnegie Mellon University, 2007.
- [115] Chou, J. C. K. “Quaternion Kinematic and Dynamic Differential Equations”. In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 53–64. DOI: 10.1109/70.127239.
- [116] Christara, C. C. and Ng, K. S. “Optimal Quadratic and Cubic Spline Collocation on Nonuniform Partitions”. In: *Computing* 76.3 (Nov. 2005), pp. 227–257. DOI: 10.1007/s00607-005-0140-4.
- [117] Christara, C. C. and Ng, K. S. “Adaptive Techniques for Spline Collocation”. In: *Computing* 76.3 (Jan. 2006), pp. 259–277. DOI: 10.1007/s00607-005-0141-3.
- [118] Chung, S.-Y. and Khatib, O. “Contact-Consistent Elastic Strips for Multi-Contact Locomotion Planning of Humanoid Robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA, May 2015, pp. 6289–6294. DOI: 10.1109/ICRA.2015.7140082.
- [119] Collins, S. H., Wisse, M., and Ruina, A. “A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees”. In: *The International Journal of Robotics Research* 20.7 (2001), pp. 607–615. DOI: 10.1177/02783640122067561.
- [120] Collins, S., Ruina, A., Tedrake, R., and Wisse, M. “Efficient Bipedal Robots Based on Passive-Dynamic Walkers”. In: *Science* 307.5712 (2005), pp. 1082–1085. DOI: 10.1126/science.1107799.
- [121] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. 3rd ed. Cambridge, Massachusetts: The MIT Press, 2009. ISBN: 978-0-262-03384-8.
- [122] Craig, J. J. *Introduction to Robotics: Mechanics and Control*. 3rd ed. Pearson, 2005. ISBN: 9780201543612.
- [123] Dam, E. B., Koch, M., and Lillholm, M. *Quaternions, Interpolation and Animation*. Tech. rep. DIKU-TR-98/5. Department of Computer Science, University of Copenhagen, 1998. URL: <http://web.mit.edu/2.998/www/QuaternionReport1.pdf> (visited on 08/28/2022).
- [124] Defense Advanced Research Projects Agency (DARPA). *DARPA Robotics Challenge (DRC)*. May 7, 2015. URL: https://web.archive.org/web/20150507154516/http://www.darpa.mil/Our_Work/TTO/Programs/DARPA_Robotics_Challenge.aspx (visited on 03/02/2022).
- [125] Dassault Systèmes. *Catia – Design Engineering*. Version 5-6 Release 2015. URL: <https://www.3ds.com/products-services/catia/> (visited on 08/03/2022).
- [126] de Boor, C. “On calculating with B-splines”. In: *Journal of Approximation Theory* 6.1 (1972), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9.
- [127] de Boor, C. and Swartz, B. “Collocation at Gaussian Points”. In: *SIAM Journal on Numerical Analysis* 10.4 (Sept. 1973), pp. 582–606. DOI: 10.1137/0710052.
- [128] de Boor, C. and Weiss, R. “SOLVEBLOK: A Package for Solving Almost Block Diagonal Linear Systems”. In: *ACM Transactions on Mathematical Software* 6.1 (Mar. 1980), pp. 80–87. DOI: 10.1145/355873.355880.

- [129] de Boor, C. *A Practical Guide to Splines*. New York: Springer, 2001. ISBN: 978-0387953663.
- [130] Deits, R. and Tedrake, R. “Footstep Planning on Uneven Terrain with Mixed-Integer Convex Optimization”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain, Nov. 2014, pp. 279–286. DOI: 10.1109/HUMANOIDS.2014.7041373.
- [131] Denk, J. and Schmidt, G. “Synthesis of Walking Primitive Databases for Biped Robots in 3D-Environments”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Taipei, Taiwan, Sept. 2003, pp. 1343–1349. DOI: 10.1109/ROBOT.2003.1241778.
- [132] Department of Defense, United States of America. *Design Criteria Standard Human Engineering (MIL-STD-1472G)*. 2012. ISBN: 9781478264071.
- [133] DFKI Robotics Innovation Center. *IEEE Robots: Charlie*. 2012. URL: <https://robots.ieee.org/robots/istruct> (visited on 03/04/2022).
- [134] Diftler, M. et al. “Robonaut 2 – The First Humanoid Robot in Space”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 2178–2183. DOI: 10.1109/ICRA.2011.5979830.
- [135] Dijkstra, E. W. “A Note on Two Problems in Connexion with Graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/BF01386390.
- [136] DLR. *IEEE Robots: Rollin’ Justin*. 2008. URL: <https://robots.ieee.org/robots/justin> (visited on 03/03/2022).
- [137] DLR. *IEEE Robots: Toro*. 2013. URL: <https://robots.ieee.org/robots/toro> (visited on 03/02/2022).
- [138] Domingos, P. “A Few Useful Things to Know about Machine Learning”. In: *Communications of the ACM* 55.10 (Oct. 2012), pp. 78–87. DOI: 10.1145/2347736.2347755.
- [139] Ebdndt, R. and Drechsler, R. “Weighted A* search – unifying view and application”. In: *Artificial Intelligence* 173.14 (2009), pp. 1310–1342. DOI: 10.1016/j.artint.2009.06.004.
- [140] Elmo Motion Control Ltd. *Servo Motor Drives – Elmo Motion Control Technology & Systems*. URL: <https://www.elmomc.com/servo-drives/> (visited on 07/05/2022).
- [141] Engelsberger, J., Ott, C., Roa, M. A., Albu-Schäffer, A., and Hirzinger, G. “Bipedal walking control based on Capture Point dynamics”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, Sept. 2011, pp. 4420–4427. DOI: 10.1109/IROS.2011.6094435.
- [142] Engelsberger, J., Ott, C., and Albu-Schäffer, A. “Three-dimensional bipedal walking control using Divergent Component of Motion”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, Nov. 2013, pp. 2600–2607. DOI: 10.1109/IROS.2013.6696723.
- [143] Engelsberger, J. et al. “Overview of the torque-controlled humanoid robot TORO”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain, Nov. 2014, pp. 916–923. DOI: 10.1109/HUMANOIDS.2014.7041473.
- [144] Engelsberger, J., Ott, C., and Albu-Schäffer, A. “Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion”. In: *IEEE Transactions on Robotics* 31.2 (Apr. 2015), pp. 355–368. DOI: 10.1109/TRO.2015.2405592.
- [145] Engelsberger, J., Mesesan, G., and Ott, C. “Smooth trajectory generation and push-recovery based on Divergent Component of Motion”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada, Nov. 2017, pp. 4560–4567. DOI: 10.1109/IROS.2017.8206324.

- [146] Enidine. *Compact Wire Rope - Miniature Vibration Isolators*. URL: <https://www.enidine.com/en-US/Products/CWRMain/> (visited on 07/28/2022).
- [147] Escande, A., Kheddar, A., and Miossec, S. "Planning support contact-points for humanoid robots and experiments on HRP-2". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Beijing, China, Oct. 2006, pp. 2974–2979. DOI: 10.1109/IROS.2006.282154.
- [148] Escande, A., Miossec, S., and Kheddar, A. "Continuous gradient proximity distance for humanoids free-collision optimized-postures". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Pittsburgh, PA, USA, Nov. 2007, pp. 188–195. DOI: 10.1109/ICHR.2007.4813867.
- [149] Escande, A., Kheddar, A., Miossec, S., and Garsault, S. "Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2". In: *11th International Symposium on Experimental Robotics*. Athens, Greece, July 2008, pp. 293–302. DOI: 10.1007/978-3-642-00196-3_35.
- [150] Escande, A. and Kheddar, A. "Contact Planning for Acyclic Motion with Task Constraints and Experiment on HRP-2 Humanoid". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, Oct. 2009, pp. 416–417. DOI: 10.1109/IROS.2009.5353971.
- [151] Escande, A., Kheddar, A., and Miossec, S. "Planning contact points for humanoid robots". In: *Robotics and Autonomous Systems* 61.5 (May 2013), pp. 428–442. DOI: 10.1016/j.robot.2013.01.008.
- [152] ESD Electronics GmbH. *CAN-EtherCAT Gateway*. URL: <https://esd.eu/en/products/can-ethercat> (visited on 07/28/2022).
- [153] Ewald, A. "Improving the Versatility of Humanoid Walking Machines". Dissertation. Germany: Technical University of Munich, 2014. ISBN: 978-3-8439-1953-1.
- [154] Fankhauser, P., Bjelonic, M., Bellicoso, C. D., Miki, T., and Hutter, M. "Robust Rough-Terrain Locomotion with a Quadrupedal Robot". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD, Australia, May 2018, pp. 1–8. DOI: 10.1109/ICRA.2018.8460731.
- [155] Farin, G. *Curves and Surfaces for CAD: A Practical Guide*. 5th ed. Morgan Kaufmann Publishers, 2002. ISBN: 1-55860-737-4.
- [156] Favot, V., Buschmann, T., Schvienbacher, M., Ewald, A., and Ulbrich, H. "The Sensor-Controller Network of the Humanoid Robot LOLA". In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Osaka, Japan, Nov. 2012, pp. 805–810. DOI: 10.1109/HUMANOIDS.2012.6651612.
- [157] Favot, V. "Hierarchical Joint Control of Humanoid Robots". Dissertation. Germany: Technical University of Munich, 2016. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20161206-1294180-1-9>.
- [158] Fayyad-Kazan, H., Perneel, L., and Timmerman, M. "Linux PREEMPT-RT vs. Commercial RTOSs: How Big is the Performance Gap?" In: *GSTF Journal on Computing* 3.1 (2013).
- [159] Fredman, M. L. and Tarjan, R. E. "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms". In: *Journal of the Association for Computing Machinery* 34.3 (July 1987), pp. 596–615. DOI: 10.1145/28869.28874.
- [160] Freeman, P. and Hart, D. "A Science of Design for Software-Intensive Systems". In: *Communications of the ACM* 47.8 (Aug. 2004), pp. 19–21. DOI: 10.1145/1012037.1012054.

- [161] Fuchs, M. et al. “Rollin’ Justin - Design considerations and realization of a mobile platform for a humanoid upper body”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 4131–4137. DOI: 10.1109/ROBOT.2009.5152464.
- [162] Fujita, M. “AIBO: Toward the Era of Digital Creatures”. In: *The International Journal of Robotics Research* 20.10 (2001), pp. 781–794. DOI: 10.1177/02783640122068092.
- [163] Fujita, M., Kuroki, Y., Ishida, T., and Doi, T. T. “A Small Humanoid Robot SDR-4X for Entertainment Applications”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. Kobe, Japan, July 2003, pp. 938–943. DOI: 10.1109/AIM.2003.1225468.
- [164] Fukuda, K. and Prodon, A. “Double Description Method Revisited”. In: *Combinatorics and Computer Science*. Ed. by Deza, M., Euler, R., and Manoussakis, I. Berlin, Heidelberg: Springer, 1996, pp. 91–111. DOI: 10.1007/3-540-61576-8_77.
- [165] Gailly, J.-l., Adler, M., et al. *zlib*. URL: <http://zlib.net/> (visited on 08/18/2022).
- [166] Gamkrelidze, R. V. “Discovery of the Maximum Principle”. In: *Journal of Dynamical and Control Systems* 5.4 (1999), pp. 437–451. DOI: 10.1023/A:1021783020548.
- [167] Geigle, C. et al. *cpptoml*. Release v0.1.1. 2018. URL: <https://github.com/skystribe/cptoml> (visited on 08/26/2022).
- [168] Geyer, H. and Saranli, U. “Gait Based on the Spring-Loaded Inverted Pendulum”. In: *Humanoid Robotics: A Reference*. Ed. by Goswami, A. and Vadakkepat, P. Dordrecht: Springer Netherlands, 2018, pp. 1–25. DOI: 10.1007/978-94-007-7194-9_43-1.
- [169] Gienger, M., Löffler, K., and Pfeiffer, F. “Towards the Design of a Biped Jogging Robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. Seoul, Korea, May 2001, pp. 4140–4145. DOI: 10.1109/ROBOT.2001.933265.
- [170] Gienger, M. “Entwurf und Realisierung einer zweibeinigen Laufmaschine”. Dissertation. Germany: Technical University of Munich, 2004.
- [171] Giraud-Esclasse, K., Fernbach, P., Buondonno, G., Mastalli, C., and Stasse, O. “Motion Planning with Multi-Contact and Visual Servoing on Humanoid Robots”. In: *IEEE/SICE International Symposium on System Integration (SII)*. Honolulu, HI, USA, Jan. 2020, pp. 156–163. DOI: 10.1109/SII46433.2020.9026291.
- [172] Goldbeck, C., Kaul, L., Vahrenkamp, N., Worgotter, F., Asfour, T., and Braun, J.-M. “Two Ways of Walking: Contrasting a Reflexive Neuro-Controller and a LIP-Based ZMP-Controller on the Humanoid Robot ARMAR-4”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexiko, Nov. 2016, pp. 966–972. DOI: 10.1109/HUMANOIDS.2016.7803389.
- [173] Google. *GoogleTest – Google’s C++ test framework*. URL: <https://github.com/google/googletest> (visited on 02/17/2023).
- [174] Goswami, A. “Foot rotation indicator (FRI) point: A new gait planning tool to evaluate postural stability of biped robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Detroit, MI, USA, May 1999, pp. 47–52. DOI: 10.1109/ROBOT.1999.769929.
- [175] Goswami, A. and Kallem, V. “Rate of change of angular momentum and balance maintenance of biped robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA, Apr. 2004, pp. 3785–3790. DOI: 10.1109/ROBOT.2004.1308858.

- [176] Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., and Maisonnier, B. “Mechatronic design of NAO humanoid”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 769–774. DOI: 10.1109/ROBOT.2009.5152516.
- [177] Grebenstein, M. et al. “The DLR Hand Arm System”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 3175–3182. DOI: 10.1109/ICRA.2011.5980371.
- [178] Griffin, R. J., Wiedebach, G., McCrory, S., Bertrand, S., Lee, I., and Pratt, J. “Footstep Planning for Autonomous Walking Over Rough Terrain”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Toronto, ON, Canada, Oct. 2019, pp. 9–16. DOI: 10.1109/Humanoids43949.2019.9035046.
- [179] Gross, D., Hauger, W., Schröder, J., Wall, W. A., and Govindjee, S. *Engineering Mechanics 3 – Dynamics*. 2nd ed. Berlin Heidelberg: Springer, 2014. DOI: 10.1007/978-3-642-53712-7.
- [180] Gross, D., Hauger, W., Schröder, J., Wall, W. A., and Bonet, J. *Engineering Mechanics 2 – Mechanics of Materials*. 2nd ed. Berlin Heidelberg: Springer, 2018. DOI: 10.1007/978-3-662-56272-7.
- [181] Grotz, M., Habra, T., Ronsse, R., and Asfour, T. “Autonomous View Selection and Gaze Stabilization for Humanoid Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada, Sept. 2017, pp. 1427–1434. DOI: 10.1109/IROS.2017.8205944.
- [182] Guennebaud, G., Jacob, B., et al. *Eigen*. Release v3.4.0. 2021. URL: <https://eigen.tuxfamily.org> (visited on 03/02/2022).
- [183] Guizzo, E. and Ackerman, E. *IEEE Spectrum: Boston Dynamics Officially Unveils Its Wheel-Leg Robot: “Best of Both Worlds”*. Feb. 27, 2017. URL: <https://spectrum.ieee.org/boston-dynamics-handle-robot> (visited on 03/03/2022).
- [184] Guizzo, E. *IEEE Spectrum: Boston Dynamics’ Spot Robot Dog Goes on Sale*. Sept. 24, 2019. URL: <https://spectrum.ieee.org/boston-dynamics-spot-robot-dog-goes-on-sale> (visited on 03/02/2022).
- [185] Hall, B. C. *Lie Groups, Lie Algebras, and Representations*. Ed. by Axler, S. and Ribet, K. 2nd ed. Springer, 2015. DOI: 10.1007/978-3-319-13467-3.
- [186] Hamano, J., Pearce, S., and Torvalds, L. *git – free and open source distributed version control system*. URL: <https://git-scm.com/> (visited on 02/17/2023).
- [187] Hamilton, S. W. R. “On quaternions; or on a new system of imaginaries in algebra”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 25 (1844), pp. 10–13. DOI: 10.1080/14786444408644923.
- [188] Hansen, N. “The CMA Evolution Strategy: A Comparing Review”. In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Ed. by Lozano, J. A., Larrañaga, P., Inza, I., and Bengoetxea, E. Berlin, Heidelberg: Springer, 2006, pp. 75–102. DOI: 10.1007/3-540-32494-1_4.
- [189] Harmonic Drive LLC. *Harmonic Drive High Precision Gear*. URL: <https://www.harmonicdrive.net> (visited on 06/29/2022).
- [190] Hart, P. E., Nilsson, N. J., and Raphael, B. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (July 1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.
- [191] Hartenberg, R. S. and Denavit, J. *Kinematic Synthesis of Linkages*. Ed. by Drake, R. M. and Kline, S. J. New York: McGraw-Hill, 1964.

- [192] Hauser, K., Bretl, T., and Latombe, J.-C. “Non-Gaited Humanoid Locomotion Planning”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Tsukuba, Japan, Dec. 2005, pp. 7–12. DOI: 10.1109/ICHR.2005.1573537.
- [193] Henze, B., Werner, A., Roa, M. A., Garofalo, G., Engelsberger, J., and Ott, C. “Control applications of TORO – A Torque controlled humanoid robot”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain, Nov. 2014, p. 841. DOI: 10.1109/HUMANOIDS.2014.7041461.
- [194] Heo, J.-W., Lee, I.-H., and Oh, J.-H. “Development of Humanoid Robots in HUBO Laboratory, KAIST”. In: *Journal of the Robotics Society of Japan* 30.4 (2012), pp. 367–371. DOI: 10.7210/jrsj.30.367.
- [195] Hereid, A., Cousineau, E. A., Hubicki, C. M., and Ames, A. D. “3D Dynamic Walking with Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, May 2016, pp. 1447–1454. DOI: 10.1109/ICRA.2016.7487279.
- [196] Herron, R., Cuzzi, J., and Hugg, J. *Mass Distribution of the Human Body using Biostereometrics*. Tech. rep. June 1976. URL: https://archive.org/details/DTIC_ADA029402 (visited on 06/17/2022).
- [197] Higham, N. J. *Accuracy and Stability of Numerical Algorithms*. 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002. ISBN: 0898715210.
- [198] Hildebrandt, A.-C., Wittmann, R., Wahrmann, D., Ewald, A., and Buschmann, T. “Real-Time 3D Collision Avoidance for Biped Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, USA, Sept. 2014, pp. 4184–4190. DOI: 10.1109/IROS.2014.6943152.
- [199] Hildebrandt, A.-C., Wahrmann, D., Wittmann, R., Rixen, D., and Buschmann, T. “Real-Time Pattern Generation Among Obstacles for Biped Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, Sept. 2015, pp. 2780–2786. DOI: 10.1109/IROS.2015.7353759.
- [200] Hildebrandt, A.-C., Demmeler, M., Wittmann, R., Wahrmann, D., Sygulla, F., Rixen, D., and Buschmann, T. “Real-Time Predictive Kinematic Evaluation and Optimization for Biped Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, Oct. 2016, pp. 5789–5796. DOI: 10.1109/IROS.2016.7759852.
- [201] Hildebrandt, A.-C. “Autonomous Robots in Unknown and Dynamic Scenarios. Biped Navigation, Real-Time Motion Generation and Collision Avoidance”. Dissertation. Germany: Technical University of Munich, 2018. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20181019-1441767-1-4>.
- [202] Hildebrandt, A.-C., Wittmann, R., Sygulla, F., Wahrmann, D., Rixen, D., and Buschmann, T. “Versatile and robust bipedal walking in unknown environments: real-time collision avoidance and disturbance rejection”. In: *Autonomous Robots* (Feb. 2019). DOI: 10.1007/s10514-019-09838-3.
- [203] Hill, S. C., Jelemensky, J., and Heene, M. R. “Queued serial peripheral interface for use in a data processing system”. US patent 4816996. Mar. 28, 1989.
- [204] Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., Fujiwara, K., and Morisawa, M. “A Universal Stability Criterion of the Foot Contact of Legged Robots - Adios ZMP”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Orlando, FL, USA, May 2006, pp. 1976–1983. DOI: 10.1109/ROBOT.2006.1641995.

- [205] Hirzinger, G., Sporer, N., Albu-Schaffer, A., Hahnle, M., Krenn, R., Pascucci, A., and Schedl, M. “DLR’s torque-controlled light weight robot III – are we reaching the technological limits now?” In: *IEEE International Conference on Robotics and Automation (ICRA)*. Washington, DC, USA, May 2002, pp. 1710–1716. DOI: 10.1109/ROBOT.2002.1014788.
- [206] Hoffman, B., Martin, K., King, B., Cole, D., Neundorf, A., and Stimpson, C. *CMake – Cross-Platform Make*. URL: <https://cmake.org/> (visited on 02/17/2023).
- [207] Honda. *IEEE Robots: Asimo*. 2000. URL: <https://robots.ieee.org/robots/asimo> (visited on 03/02/2022).
- [208] Horner, W. G. and Gilbert, D. “A new method of solving numerical equations of all orders, by continuous approximation”. In: *Philosophical Transactions of the Royal Society of London*. London, 1819, pp. 308–335.
- [209] Hornung, A. and Bennewitz, M. “Adaptive Level-of-Detail Planning for Efficient Humanoid Navigation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Saint Paul, MN, USA, May 2012, pp. 997–1002. DOI: 10.1109/ICRA.2012.6224898.
- [210] Hornung, A., Dornbush, A., Likhachev, M., and Bennewitz, M. “Anytime Search-Based Footstep Planning with Suboptimality Bounds”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Osaka, Japan, Nov. 2012, pp. 674–679. DOI: 10.1109/HUMANOIDS.2012.6651592.
- [211] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3 (2013), pp. 189–206. DOI: 10.1007/s10514-012-9321-0.
- [212] Houstis, E. N., Christara, C. C., and Rice, J. R. “Quadratic-Spline Collocation Methods for Two-Point Boundary Value Problems”. In: *International Journal for Numerical Methods in Engineering* 26.4 (1988), pp. 935–952. DOI: 10.1002/nme.1620260412.
- [213] Hubicki, C., Grimes, J., Jones, M., Renjewski, D., Spröwitz, A., Abate, A., and Hurst, J. “ATRIAS: Design and validation of a tether-free 3D-capable spring-mass bipedal robot”. In: *The International Journal of Robotics Research* 35.12 (2016), pp. 1497–1521. DOI: 10.1177/0278364916648388.
- [214] Hutter, M. and Gehring, C. *proNeu Documentation - Derivation of Analytical Kinematics & Dynamics*. Tech. rep. Autonomous Systems Lab, ETH Zürich, 2012. URL: https://bitbucket.org/leggedrobotics/proneu/src/master/documentation/manual/proNeu_documentation.pdf (visited on 03/15/2022).
- [215] Hutter, M., Gehring, C., Bloesch, M., Hoepflinger, M. A., Remy, C. D., and Siegwart, R. “STARLETH. A compliant quadrupedal robot for fast, efficient, and versatile locomotion”. In: *International Conference on Climbing and Walking Robot (CLAWAR)*. Zürich: Autonomous Systems Lab, ETH Zürich, 2012. DOI: 10.3929/ethz-a-010034688.
- [216] Hutter, M. et al. “ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, Oct. 2016, pp. 38–44. DOI: 10.1109/IROS.2016.7758092.
- [217] iMAR Navigation. *iVRU-FC/iVRU-FQ: Inertial Measurement System with integrated GNSS and Odometer Interface*. URL: <https://www.imar-navigation.de/en/products/by-product-names/item/ivru-fc-ivru-fq-inertial-measurement-system-with-integrated-gnss-and-odometer-interface> (visited on 05/23/2022).
- [218] Intel RealSense. *Depth Camera D435*. URL: <https://www.intelrealsense.com/depth-camera-d435> (visited on 03/14/2022).

- [219] Intel RealSense. *Tracking Camera T265*. URL: <https://www.intelrealsense.com/tracking-camera-t265> (visited on 03/14/2022).
- [220] Irodoutou-Ellina, M. and Houstis, E. N. “An $\mathcal{O}(h^6)$ Quintic Spline Collocation Method for Fourth Order Two-Point Boundary Value Problems”. In: *BIT Numerical Mathematics* 28.2 (June 1988), pp. 288–301. DOI: 10.1007/BF01934092.
- [221] Isaacson, E. and Keller, H. B. *Analysis of Numerical Methods*. New York: Dover Publications, Inc., 1966. ISBN: 0-486-68029-0.
- [222] Josefsson, S. *The Base16, Base32, and Base64 Data Encodings*. Tech. rep. RFC 4648. Request for Comments, RFC Editor, 2006. DOI: 10.17487/RFC4648.
- [223] Jung, T., Lim, J., Bae, H., Lee, K. K., Joe, H.-M., and Oh, J.-H. “Development of the Humanoid Disaster Response Platform DRC-HUBO+”. In: *IEEE Transactions on Robotics* 34.1 (2018), pp. 1–17. DOI: 10.1109/TRO.2017.2776287.
- [224] Kaiser, P., Vahrenkamp, N., Schültje, F., Borràs, J., and Asfour, T. “Extraction of Whole-Body Affordances for Loco-Manipulation Tasks”. In: *International Journal of Humanoid Robotics* 12.3 (2015). DOI: 10.1142/S0219843615500310.
- [225] Kaiser, P., Aksoy, E. E., Grotz, M., and Asfour, T. “Towards a Hierarchy of Loco-Manipulation Affordances”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, Oct. 2016, pp. 2839–2846. DOI: 10.1109/IROS.2016.7759440.
- [226] Kaiser, P., Kanoulas, D., Grotz, M., Muratore, L., Rocchi, A., Hoffman, E. M., Tsagarakis, N. G., and Asfour, T. “An Affordance-Based Pilot Interface for High-Level Control of Humanoid Robots in Supervised Autonomy”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 621–628. DOI: 10.1109/HUMANOIDS.2016.7803339.
- [227] Kaiser, P., Mandery, C., Boltres, A., and Asfour, T. “Affordance-Based Multi-Contact Whole-Body Pose Sequence Planning for Humanoid Robots in Unknown Environments”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, May 2018, pp. 3114–3121. DOI: 10.1109/ICRA.2018.8461087.
- [228] KAIST. *IEEE Robots: Hubo 2*. 2009. URL: <https://robots.ieee.org/robots/hubo> (visited on 03/02/2022).
- [229] KAIST and Rainbow Robotics. *IEEE Robots: DRC-Hubo+*. 2015. URL: <https://robots.ieee.org/robots/drchubo> (visited on 03/04/2022).
- [230] Kajita, S. and Tani, K. “Study of Dynamic Biped Locomotion on Rugged Terrain – Derivation and Application of the Linear Inverted Pendulum Mode”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Sacramento, California, Apr. 1991, pp. 1405–1411. DOI: 10.1109/ROBOT.1991.131811.
- [231] Kajita, S. and Tani, K. “Experimental Study of Biped Dynamic Walking in the Linear Inverted Pendulum Mode”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3. Nagoya, Japan, May 1995, pp. 2885–2891. DOI: 10.1109/ROBOT.1995.525693.
- [232] Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. “The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Maui, Hawaii, USA, Oct. 2001, pp. 239–246. DOI: 10.1109/IROS.2001.973365.
- [233] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. “Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Taipei, Taiwan, Sept. 2003, pp. 1620–1626. DOI: 10.1109/ROBOT.2003.1241826.

- [234] Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., and Yokoi, K. “Biped Walking Stabilization Based on Linear Inverted Pendulum Tracking”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, Oct. 2010, pp. 4489–4496. DOI: 10.1109/IROS.2010.5651082.
- [235] Kálmán, R. E. “A New Approach to Linear Filtering and Prediction Problems”. In: *ASME Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. DOI: 10.1115/1.3662552.
- [236] Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., and Isozumi, T. “Humanoid Robot HRP-2”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA, Apr. 2004, pp. 1083–1090. DOI: 10.1109/ROBOT.2004.1307969.
- [237] Kaneko, K., Harada, K., Kanehiro, F., Miyamori, G., and Akachi, K. “Humanoid Robot HRP-3”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nice, France, Sept. 2008, pp. 2471–2478. DOI: 10.1109/IROS.2008.4650604.
- [238] Kaneko, K., Kanehiro, F., Morisawa, M., Miura, K., Nakaoka, S., and Kajita, S. “Cybernetic Human HRP-4C”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Paris, France, Dec. 2009, pp. 7–14. DOI: 10.1109/ICHR.2009.5379537.
- [239] Kaneko, K., Kanehiro, F., Morisawa, M., Akachi, K., Miyamori, G., Hayashi, A., and Kanehira, N. “Humanoid Robot HRP-4 - Humanoid Robotics Platform with Lightweight and Slim Body”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, Sept. 2011, pp. 4400–4407. DOI: 10.1109/IROS.2011.6094465.
- [240] Kaneko, K., Morisawa, M., Kajita, S., Nakaoka, S., Sakaguchi, T., Cisneros, R., and Kanehiro, F. “Humanoid Robot HRP-2Kai – Improvement of HRP-2 Towards Disaster Response Tasks”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea, Nov. 2015, pp. 132–139. DOI: 10.1109/HUMANOIDS.2015.7363526.
- [241] Kaneko, K., Kaminaga, H., Sakaguchi, T., Kajita, S., Morisawa, M., Kumagai, I., and Kanehiro, F. “Humanoid Robot HRP-5P: An Electrically Actuated Humanoid Robot With High-Power and Wide-Range Joints”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1431–1438. DOI: 10.1109/LRA.2019.2896465.
- [242] Kapoor, C., Cetin, M., and Tesar, D. “Performance Based Redundancy Resolution With Multiple Criteria”. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Atlanta, Georgia, USA, Sept. 1998. DOI: 10.1115/DETC98/MECH-5864.
- [243] Karkowski, P. and Bennewitz, M. “Real-Time Footstep Planning Using a Geometric Approach”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, May 2016, pp. 1782–1787. DOI: 10.1109/ICRA.2016.7487323.
- [244] Karkowski, P., Oßwald, S., and Bennewitz, M. “Real-Time Footstep Planning in 3D Environments”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 69–74. DOI: 10.1109/HUMANOIDS.2016.7803256.
- [245] Kato, I. “[Development of the Biped Robot WABOT-1]”. In: *[Biomechanisms Japan]* 2 (1973). Translated from Japanese using <https://translate.google.com>, pp. 173–174. DOI: 10.3951/biomechanisms.2.173.
- [246] Kennedy, J. and Eberhart, R. “Particle Swarm Optimization”. In: *International Conference on Neural Networks (ICNN)*. Perth, WA, Australia, Nov. 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.
- [247] Kim, J.-O. and Khosla, P. K. “Dexterity Measures for Design and Control of Manipulators”. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS)*. Osaka, Japan, Nov. 1991, pp. 758–763. DOI: 10.1109/IROS.1991.174572.

- [248] Kim, M.-J., Kim, M.-S., and Shin, S. Y. “A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives”. In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 369–376. DOI: 10.1145/218380.218486.
- [249] Kim, M.-J., Kim, M.-S., and Shin, S. Y. “A C^2 -continuous B-spline Quaternion Curve Interpolating a Given Sequence of Solid Orientations”. In: *Proceedings Computer Animation'95*. Geneva, Switzerland, Apr. 1995, pp. 72–81. DOI: 10.1109/CA.1995.393545.
- [250] Kim, M.-J., Kim, M.-S., and Shin, S. Y. “A Compact Differential Formula for the First Derivative of a Unit Quaternion Curve”. In: *The Journal of Visualization and Computer Animation* 7.1 (1996), pp. 43–57. DOI: 10.1002/(SICI)1099-1778(199601)7:1<43::AID-VIS136>3.0.CO;2-T.
- [251] Kimura, K. and Murase, Y. “The Industrialization of Humanoid Robot HOAP Series”. In: *Journal of the Robotics Society of Japan* 22.1 (2004). Translated from Japanese using <https://translate.google.com>, pp. 10–12. DOI: 10.7210/jrsj.22.10.
- [252] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.
- [253] KIT. *IEEE Robots: Armar*. 2017. URL: <https://robots.ieee.org/robots/armar> (visited on 03/03/2022).
- [254] Klopčar, N. and Lenarčič, J. “Kinematic Model for Determination of Human Arm Reachable Workspace”. In: *Meccanica* 40 (2005), pp. 203–219. DOI: 10.1007/s11012-005-3067-0.
- [255] Kojima, K. et al. “Development of Life-sized High-Power Humanoid Robot JAXON for Real-World Use”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Seoul, Korea, Nov. 2015, pp. 838–843. DOI: 10.1109/HUMANOIDS.2015.7363459.
- [256] Kucuk, S. and Bingul, Z. “Robot Workspace Optimization Based on a Novel Local and Global Performance Indices”. In: *IEEE International Symposium on Industrial Electronics (ISIE)*. Dubrovnik, Croatia, June 2005, pp. 1593–1598. DOI: 10.1109/ISIE.2005.1529170.
- [257] Kühn, D., Schilling, M., Stark, T., Zenzes, M., and Kirchner, F. “System Design and Testing of the Hominid Robot Charlie”. In: *Journal of Field Robotics* 34.4 (July 2016), pp. 666–703. DOI: 10.1002/rob.21662.
- [258] Kühn, D., Dettmann, A., and Kirchner, F. “Analysis of Using an Active Artificial Spine in a Quadruped Robot”. In: *International Conference on Control, Automation and Robotics (ICCAR)*. Auckland, New Zealand, Apr. 2018, pp. 37–42. DOI: 10.1109/ICCAR.2018.8384641.
- [259] Kuffner, J. J. and LaValle, S. M. “RRT-Connect: An Efficient Approach to Single-Query Path Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA, USA, Apr. 2000, pp. 995–1001. DOI: 10.1109/ROBOT.2000.844730.
- [260] Kuindersma, S. *The Art and Engineering Behind a Humanoid Dance Routine*. Within workshop *Can we build Baymax?* at IEEE-RAS International Conference on Humanoid Robots (Humanoids). Munich, Germany, July 2021.
- [261] Kumagai, I., Morisawa, M., Nakaoka, S., and Kanehiro, F. “Efficient Locomotion Planning for a Humanoid Robot with Whole-Body Collision Avoidance Guided by Footsteps and Centroidal Sway Motion”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Beijing, China, Nov. 2018, pp. 251–256. DOI: 10.1109/HUMANOIDS.2018.8624927.

- [262] Kumagai, I., Morisawa, M., Benallegue, M., and Kanehiro, F. “Bipedal Locomotion Planning for a Humanoid Robot Supported by Arm Contacts Based on Geometrical Feasibility”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Toronto, ON, Canada, Oct. 2019, pp. 132–139. DOI: 10.1109/Humanoids43949.2019.9035072.
- [263] Kumar, A. and Waldron, K. J. “The Workspaces of a Mechanical Manipulator”. In: *Journal of Mechanical Design* 103.3 (July 1981), pp. 665–672. DOI: 10.1115/1.3254968.
- [264] Lack, J., Powell, M. J., and Ames, A. D. “Planar Multi-Contact Bipedal Walking Using Hybrid Zero Dynamics”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, May 2014, pp. 2582–2588. DOI: 10.1109/ICRA.2014.6907229.
- [265] Lantinga, S. et al. *Simple DirectMedia Layer (SDL)*. URL: <https://www.libsdl.org/> (visited on 08/26/2022).
- [266] Larkin, D. H., Sen, S., and Tarjan, R. E. “A Back-to-Basics Empirical Study of Priority Queues”. In: *2014 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, pp. 61–72. DOI: 10.1137/1.9781611973198.7.
- [267] Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. “Fast Distance Queries with Rectangular Swept Sphere Volumes”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA, USA, Apr. 2000, pp. 3719–3726. DOI: 10.1109/ROBOT.2000.845311.
- [268] Lasguignes, T., Maroger, I., Fallon, M., Ramezani, M., Marchionni, L., Stasse, O., Mansard, N., and Watier, B. “ICP Localization and Walking Experiments on a TALOS Humanoid Robot”. In: *International Conference on Advanced Robotics (ICAR)*. Ljubljana, Slovenia, Dec. 2021, pp. 800–805. DOI: 10.1109/ICAR53236.2021.9659474.
- [269] LaValle, S. M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Iowa, USA: Computer Science Dept., Iowa State University, Oct. 1998. URL: <http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf> (visited on 03/21/2022).
- [270] LaValle, S. M. and Kuffner, J. J. “Randomized Kinodynamic Planning”. In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400. DOI: 10.1177/02783640122067453.
- [271] Lee, E. T. Y. “A Simplified B-Spline Computation Routine”. In: *Computing* 29.4 (1982), pp. 365–371. DOI: 10.1007/BF02246763.
- [272] Lee, E. T. Y. “Comments on Some B-Spline Algorithms”. In: *Computing* 36.3 (1986), pp. 229–238. DOI: 10.1007/BF02240069.
- [273] Lee, J.-F., Lee, R., and Cangellaris, A. “Time-Domain Finite-Element Methods”. In: *IEEE Transactions on Antennas and Propagation* 45.3 (Mar. 1997), pp. 430–442. DOI: 10.1109/8.558658.
- [274] Lee, S.-H. and Goswami, A. “Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Rome, Italy, Apr. 2007, pp. 4667–4672. DOI: 10.1109/ROBOT.2007.364198.
- [275] Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. “Learning quadrupedal locomotion over challenging terrain”. In: *Science Robotics* 5.47 (2020), eabc5986. DOI: 10.1126/scirobotics.abc5986.
- [276] Lenarčič, J., Stanič, U. J., and Oblak, P. “Some Kinematic Considerations for the Design of Robot Manipulators”. In: *Robotics and Computer-Integrated Manufacturing* 5.2 (1989), pp. 235–241. DOI: 10.1016/0736-5845(89)90069-0.

- [277] Lenarčič, J. and Umek, A. “Simple Model of Human Arm Reachable Workspace”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 24.8 (Aug. 1994), pp. 1239–1246. DOI: 10.1109/21.299704.
- [278] Lenarčič, J. and Klopčar, N. “Positional kinematics of humanoid arms”. In: *Robotica* 24.1 (Oct. 2006), pp. 105–112. DOI: 10.1017/S0263574705001906.
- [279] Lengagne, S., Mathieu, P., Kheddar, A., and Yoshida, E. “Generation of Dynamic Multi-Contact Motions: 2D case studies”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Nashville, TN, USA, Dec. 2010, pp. 14–20. DOI: 10.1109/ICHR.2010.5686836.
- [280] Lengagne, S., Vaillant, J., Yoshida, E., and Kheddar, A. “Generation of whole-body optimal dynamic multi-contact motions”. In: *The International Journal of Robotics Research* 32.9–10 (2013), pp. 1104–1119. DOI: 10.1177/0278364913478990.
- [281] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. “Keyframe-based visual-inertial odometry using nonlinear optimization”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334. DOI: 10.1177/0278364914554813.
- [282] Li, S.-F. and Cheng, C.-Y. “Particle swarm optimization with fitness adjustment parameters”. In: *Computers & Industrial Engineering* 113 (2017), pp. 831–841. DOI: 10.1016/j.cie.2017.06.006.
- [283] Liégeois, A. “Automatic Supervisory Control of the Configuration and Behaviour of Multibody Mechanisms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 7.12 (Dec. 1977), pp. 868–871. DOI: 10.1109/TSMC.1977.4309644.
- [284] Likhachev, M., Gordon, G., and Thrun, S. “ARA*: Anytime A* with Provable Bounds on Sub-Optimality”. In: *Neural Information Processing Systems (NeurIPS)*. Dec. 2003, pp. 767–774.
- [285] Likhachev, M. and Stentz, A. “R* Search”. In: *AAAI 23rd National Conference on Artificial Intelligence*. Chicago, Illinois, July 2008, pp. 344–350.
- [286] Lin, Y.-C. and Berenson, D. “Using Previous Experience for Humanoid Navigation Planning”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 794–801. DOI: 10.1109/HUMANOIDS.2016.7803364.
- [287] Lin, Y.-C., Righetti, L., and Berenson, D. “Robust Humanoid Contact Planning With Learned Zero- and One-Step Capturability Prediction”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2451–2458. DOI: 10.1109/LRA.2020.2972825.
- [288] Löffler, K., Gienger, M., Pfeiffer, F., and Ulbrich, H. “Sensors and Control Concept of a Biped Robot”. In: *IEEE Transactions on Industrial Electronics* 51.5 (Oct. 2004), pp. 972–980. DOI: 10.1109/TIE.2004.834948.
- [289] Löffler, K. “Dynamik und Regelung einer zweibeinigen Laufmaschine”. Dissertation. Germany: Technical University of Munich, 2005.
- [290] Lohmeier, S., Löffler, K., Gienger, M., Ulbrich, H., and Pfeiffer, F. “Computer System and Control of Biped “Johnnie””. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 4. New Orleans, LA, USA, Apr. 2004, pp. 4222–4227. DOI: 10.1109/ROBOT.2004.1308939.
- [291] Lohmeier, S. “Design and Realization of a Humanoid Robot for Fast and Autonomous Bipedal Locomotion”. Dissertation. Germany: Technical University of Munich, 2010. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20101126-980754-1-4>.
- [292] Lorch, O. “Beiträge zur visuellen Führung zweibeiniger Laufroboter in einem strukturierten Szenario”. Dissertation. Germany: Technical University of Munich, 2003.

- [293] Lorensen, W. E. and Cline, H. E. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *Computer Graphics* 21.4 (July 1987), pp. 163–169.
- [294] Lunze, J. *Regelungstechnik 1*. 11th ed. Berlin Heidelberg: Springer, 2016. DOI: 10.1007/978-3-662-52678-1.
- [295] Luo, R. C., Lee, K. C., and Spalanzani, A. “Humanoid Robot Walking Pattern Generation Based on Five-Mass with Angular Momentum Model”. In: *IEEE International Symposium on Industrial Electronics (ISIE)*. June 2016, pp. 375–380. DOI: 10.1109/ISIE.2016.7744919.
- [296] Ma, O. and Angeles, J. “Optimum Architecture Design of Platform Manipulators”. In: *Fifth International Conference on Advanced Robotics – Robots in Unstructured Environments*. Pisa, Italy, June 1991, pp. 1130–1135. DOI: 10.1109/ICAR.1991.240404.
- [297] Magoulés, F., Roux, F.-X., and Houzeaux, G. *Parallel Scientific Computing*. John Wiley & Sons, Ltd, 2015. DOI: 10.1002/9781118761687.
- [298] Majaess, F., Keast, P., and Fairweather, G. “Packages for solving almost block diagonal linear systems arising in spline collocation at Gaussian points with monomial basis functions”. In: *Scientific Software Systems*. Ed. by Mason, J. C. and Cox, M. G. Dordrecht: Springer Netherlands, 1990, pp. 47–58. DOI: 10.1007/978-94-009-0841-3_3.
- [299] Mansard, N., Khatib, O., and Kheddar, A. “A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks”. In: *IEEE Transactions on Robotics* 25.3 (June 2009), pp. 670–685. DOI: 10.1109/TRO.2009.2020345.
- [300] Mason, S., Rotella, N., Schaal, S., and Righetti, L. “An MPC Walking Framework With External Contact Forces”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, May 2018, pp. 1785–1790. DOI: 10.1109/ICRA.2018.8461236.
- [301] Maus, H.-M., Lipfert, S., Gross, M., Rummel, J., and Seyfarth, A. “Upright human gait did not provide a major mechanical challenge for our ancestors”. In: *Nature Communications* 1.70 (Sept. 2010). DOI: 10.1038/ncomms1073.
- [302] Maximo, M. R. O. A., Ribeiro, C. H. C., and Afonso, R. J. M. “Mixed-Integer Programming for Automatic Walking Step Duration”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, Oct. 2016, pp. 5399–5404. DOI: 10.1109/IROS.2016.7759794.
- [303] Mayne, D. “A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems”. In: *International Journal of Control* 3.1 (1966), pp. 85–95. DOI: 10.1080/00207176608921369.
- [304] McGeer, T. “Passive Dynamic Walking”. In: *The International Journal of Robotics Research* 9.2 (1990), pp. 62–82. DOI: 10.1177/027836499000900206.
- [305] Meurant, G. “A Review on the Inverse of Symmetric Tridiagonal and Block Tridiagonal Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 13.3 (1992), pp. 707–728. DOI: 10.1137/0613045.
- [306] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. “Discrete Differential-Geometry Operators for Triangulated 2-Manifolds”. In: *Visualization and Mathematics III*. Ed. by Hege, H.-C. and Polthier, K. Berlin, Heidelberg: Springer, 2003, pp. 35–57. DOI: 10.1007/978-3-662-05105-4_2.
- [307] Mittendorf, P. and Cheng, G. “Humanoid Multimodal Tactile-Sensing Modules”. In: *IEEE Transactions on Robotics* 27.3 (2011), pp. 401–410. DOI: 10.1109/TRO.2011.2106330.

- [308] Mizuuchi, I., Nakanishi, Y., Sodeyama, Y., Namiki, Y., Nishino, T., Muramatsu, N., Urata, J., Hongo, K., Yoshikai, T., and Inaba, M. “An Advanced Musculoskeletal Humanoid Kojiro”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Pittsburgh, PA, USA, Nov. 2007, pp. 294–299. DOI: 10.1109/ICHR.2007.4813883.
- [309] Möller, T. and Trumbore, B. “Fast, Minimum Storage Ray-Triangle Intersection”. In: *Journal of Graphics Tools* 2.1 (1997), pp. 21–28. DOI: 10.1080/10867651.1997.10487468.
- [310] Morais, J. P., Georgiev, S., and Sprößig, W. *Real Quaternionic Calculus Handbook*. 1st ed. Birkhäuser Basel, 2014. DOI: 10.1007/978-3-0348-0622-0.
- [311] Motzkin, T. S., Raiffa, H., Thompson, G. L., and Thrall, R. M. “The Double Description Method”. In: *Contributions to the Theory of Games*. Ed. by Kuhn, H. W. and Tucker, A. W. Vol. 2. Princeton University Press, 1953, pp. 51–73.
- [312] Mund, E. H., Hallet, P., and Hennart, J. P. “An algorithm for the interpolation of functions using quintic splines”. In: *Journal of Computational and Applied Mathematics* 1.4 (1975), pp. 279–288. DOI: 10.1016/0771-050X(75)90020-0.
- [313] Murooka, M., Kumagai, I., Morisawa, M., Kanehiro, F., and Kheddar, A. “Humanoid Loco-Manipulation Planning Based on Graph Search and Reachability Maps”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1840–1847. DOI: 10.1109/LRA.2021.3060728.
- [314] Murooka, M., Chappellet, K., Tanguy, A., Benallegue, M., Kumagai, I., Morisawa, M., Kanehiro, F., and Kheddar, A. “Humanoid Loco-Manipulations Pattern Generation and Stabilization Control”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5597–5604. DOI: 10.1109/LRA.2021.3077858.
- [315] Murphy, M. P., Saunders, A., Moreira, C., Rizzi, A. A., and Raibert, M. “The LittleDog robot”. In: *The International Journal of Robotics Research* 30.2 (2011), pp. 145–149. DOI: 10.1177/0278364910387457.
- [316] Nagarajan, U. and Yamane, K. “Automatic Task-specific Model Reduction for Humanoid Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, Nov. 2013, pp. 2578–2585. DOI: 10.1109/IROS.2013.6696720.
- [317] Nagasaka, K., Inoue, H., and Inaba, M. “Dynamic Walking Pattern Generation for a Humanoid Robot Based on Optimal Gradient Method”. In: *IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*. Tokyo, Japan, Oct. 1999, pp. 908–913. DOI: 10.1109/ICSMC.1999.816673.
- [318] Nakamura, Y., Hanafusa, H., and Yoshikawa, T. “Task-Priority Based Redundancy Control of Robot Manipulators”. In: *The International Journal of Robotics Research* 6.2 (1987), pp. 3–15. DOI: 10.1177/027836498700600201.
- [319] Nakamura, Y. *Advanced Robotics: Redundancy and Optimization*. Boston, MA, USA: Addison-Wesley Publishing Company, Inc., 1991. ISBN: 978-0-201-15198-5.
- [320] NASA, United States of America. *Man-Systems Integration Standards (NASA-STD-3000 Volume I)*. July 1995. URL: <https://msis.jsc.nasa.gov/Volume1.htm> (visited on 06/17/2022).
- [321] Nelson, G., Saunders, A., and Playter, R. “The PETMAN and Atlas Robots at Boston Dynamics”. In: *Humanoid Robotics: A Reference*. Ed. by Goswami, A. and Vadakkepat, P. Dordrecht: Springer Netherlands, 2019, pp. 169–186. DOI: 10.1007/978-94-007-6046-2_15.
- [322] Nguyen, N. T. *Model-Reference Adaptive Control*. 1st ed. Cham, Switzerland: Springer International, 2018. DOI: 10.1007/978-3-319-56393-0.

- [323] Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M., and Inoue, H. “Online Generation of Humanoid Walking Motion based on a Fast Generation Method of Motion Pattern that Follows Desired ZMP”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. EPFL, Lausanne, Switzerland, Oct. 2002, pp. 2684–2689. DOI: 10.1109/IRDS.2002.1041675.
- [324] Nishiwaki, K., Kuffner, J., Kagami, S., Inaba, M., and Inoue, H. “The experimental humanoid robot H7: a research platform for autonomous behaviour”. In: *Philosophical Transactions of the Royal Society A* 365.1850 (2007), pp. 79–107. DOI: 10.1098/rsta.2006.1921.
- [325] Nishiwaki, K. and Kagami, S. “Online Design of Torso Height Trajectories for Walking Patterns that takes Future Kinematic Limits into Consideration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. May 2011, pp. 2029–2034. DOI: 10.1109/ICRA.2011.5979922.
- [326] Nishiwaki, K., Chestnutt, J., and Kagami, S. “Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor”. In: *The International Journal of Robotics Research* 31.11 (2012), pp. 1251–1262. DOI: 10.1177/0278364912455720.
- [327] Ogura, Y., Aikawa, H., Shimomura, K., Kondo, H., Morishima, A., Lim, H.-o., and Takanishi, A. “Development of a New Humanoid Robot WABIAN-2”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Orlando, FL, USA, May 2006, pp. 76–81. DOI: 10.1109/ROBOT.2006.1641164.
- [328] Okada, K., Inaba, M., and Inoue, H. “Walking Navigation System of Humanoid Robot using Stereo Vision based Floor Recognition and Path Planning with Multi-Layered Body Image”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, Oct. 2003, pp. 2155–2160. DOI: 10.1109/IROS.2003.1249190.
- [329] OpenMP Architecture Review Board. *Open Multi-Processing (OpenMP)*. URL: <https://www.openmp.org/> (visited on 02/06/2023).
- [330] Open Source Robotics Foundation. *ROS: Robot Operating System*. URL: <https://www.ros.org> (visited on 03/17/2022).
- [331] Ott, C. et al. “A Humanoid Two-Arm System for Dexterous Manipulation”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Genova, Italy, Dec. 2006, pp. 276–283. DOI: 10.1109/ICHR.2006.321397.
- [332] PAL Robotics. *IEEE Robots: Reem-C*. 2013. URL: <https://robots.ieee.org/robots/reemc> (visited on 03/02/2022).
- [333] PAL Robotics. *IEEE Robots: Talos*. 2017. URL: <https://robots.ieee.org/robots/talos> (visited on 03/02/2022).
- [334] Pandey, A. K. and Gelin, R. “A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind”. In: *IEEE Robotics & Automation Magazine* 25.3 (2018), pp. 40–48. DOI: 10.1109/MRA.2018.2833157.
- [335] Pang, J.-S. and Trinkle, J. “Stability Characterizations of Rigid Body Contact Problems with Coulomb Friction”. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 80.10 (2000), pp. 643–663. DOI: 10.1002/1521-4001(200010)80:10<643::AID-ZAMM643>3.0.CO;2-E.
- [336] Park, J. and Kim, K. “Biped Robot Walking Using Gravity-Compensated Inverted Pendulum Mode and Computed Torque Control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Leuven, Belgium, May 1998, pp. 3528–3533. DOI: 10.1109/ROBOT.1998.680985.

- [337] Park, H.-W. and Kim, S. “The MIT Cheetah, an Electrically-Powered Quadrupedal Robot for High-speed Running”. In: *Journal of the Robotics Society of Japan* 32.4 (2014), pp. 323–328. DOI: 10.7210/jrsj.32.323.
- [338] Park, H.-W., Wensing, P., and Kim, S. “High-speed bounding with the MIT Cheetah 2: Control design and experiments”. In: *The International Journal of Robotics Research* 36.2 (2017), pp. 167–192. DOI: 10.1177/0278364917694244.
- [339] Park, J., Delgado, R., and Choi, B. W. “Real-Time Characteristics of ROS 2.0 in Multiagent Robot Systems: An Empirical Study”. In: *IEEE Access* 8 (Aug. 2020). DOI: 10.1109/ACCESS.2020.3018122.
- [340] Parker Hannifin Corporation. *Motion Control Systems*. URL: <https://www.parkermotion.com> (visited on 07/25/2022).
- [341] Patel, S. and Sobh, T. “Manipulator Performance Measures - A Comprehensive Literature Survey”. In: *Journal of Intelligent & Robotic Systems* 77.3 (2015), pp. 547–570. DOI: 10.1007/s10846-014-0024-y.
- [342] Pfeiffer, F., Eltze, J., and Weidemann, H.-J. “The TUM-Walking Machine”. In: *Intelligent Automation & Soft Computing* 1.3 (1995), pp. 307–323.
- [343] Pfeiffer, F., Löffler, K., and Gienger, M. “The Concept of Jogging JOHNNIE”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Washington, DC, May 2002, pp. 3129–3135. DOI: 10.1109/ROBOT.2002.1013708.
- [344] Pfeiffer, F. “The TUM walking machines”. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 365.1850 (Nov. 2007), pp. 109–131. DOI: 10.1098/rsta.2006.1922.
- [345] Plasti Dip Europe GmbH. *Performix Plasti Dip Flüssiggummi*. URL: <https://plastidip-eu.com/produkte/fluessiggummi/> (visited on 07/27/2022).
- [346] Popovic, M. B., Goswami, A., and Herr, H. “Ground Reference Points in Legged Locomotion: Definitions, Biological Trajectories and Control Implications”. In: *The International Journal of Robotics Research* 24.12 (2005), pp. 1013–1032. DOI: 10.1177/0278364905058363.
- [347] Poskanzer, J. *Portable Anymap Format (PNM)*. 1988. URL: <http://netpbm.sourceforge.net/> (visited on 09/27/2022).
- [348] Pratt, G. A. and Williamson, M. M. “Series Elastic Actuators”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Pittsburgh, PA, USA, Aug. 1995, pp. 399–406. DOI: 10.1109/IROS.1995.525827.
- [349] Pratt, J., Carff, J., Drakunov, S., and Goswami, A. “Capture Point: A Step toward Humanoid Push Recovery”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Genova, Italy, Dec. 2006, pp. 200–207. DOI: 10.1109/ICHR.2006.321385.
- [350] Pratt, J. E. et al. “The Yobotics-IHMC Lower Body Humanoid Robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, Oct. 2009, pp. 410–411. DOI: 10.1109/IROS.2009.5354430.
- [351] Preston-Werner, T., Gedam, P., et al. *TOML: Tom’s Obvious Minimal Language*. Release v0.5.0. 2018. URL: <https://toml.io/> (visited on 09/27/2022).
- [352] Quarteroni, A., Sacco, R., and Saler, F. *Numerical Mathematics*. 2nd ed. Berlin Heidelberg: Springer, 2007. DOI: 10.1007/b98885.
- [353] Rader, S., Kaul, L., Fischbach, H., Vahrenkamp, N., and Asfour, T. “Design of a High-Performance Humanoid Dual Arm System with Inner Shoulder Joints”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 523–529. DOI: 10.1109/HUMANOIDS.2016.7803325.

- [354] Radford, N. A. et al. “Valkyrie: NASA’s First Bipedal Humanoid Robot”. In: *Journal of Field Robotics* 32.3 (2015), pp. 397–419. DOI: 10.1002/rob.21560.
- [355] Raibert, M. H. *Legged Robots That Balance*. Cambridge, MA, USA: The MIT Press, 1986. ISBN: 0-262-18117-7.
- [356] Raibert, M., Blankespoor, K., Nelson, G., Playter, R., et al. “BigDog, the Rough-Terrain Quadruped Robot”. In: *Proceedings of the 17th IFAC World Congress* 41.2 (July 2008), pp. 10822–10825. DOI: 10.3182/20080706-5-KR-1001.01833.
- [357] Ralph, P. and Wand, Y. “A Proposal for a Formal Definition of the Design Concept”. In: *Design Requirements Engineering: A Ten-Year Perspective*. Ed. by Lyytinen, K., Loucopoulos, P., Mylopoulos, J., and Robinson, B. Berlin, Heidelberg: Springer, 2009, pp. 103–136. DOI: 10.1007/978-3-540-92966-6_6.
- [358] Ramuzat, N., Buondonno, G., Boria, S., and Stasse, O. “Comparison of Position and Torque Whole-Body Control Schemes on the Humanoid Robot TALOS”. In: *International Conference on Advanced Robotics (ICAR)*. Ljubljana, Slovenia, Dec. 2021, pp. 785–792. DOI: 10.1109/ICAR53236.2021.9659380.
- [359] Reghenzani, F., Massari, G., and Fornaciari, W. “The Real-Time Linux Kernel: A Survey on PREEMPT_RT”. In: *ACM Computing Surveys* 52.1 (Feb. 2019). DOI: 10.1145/3297714.
- [360] Rohe, G., von Hundelshausen, F., and Wuensche, H.-J. “Sichtgeführte Tentakelnavigation für humanoide Roboter”. In: *Tagungsband 1. Interdisziplinärer Workshop Kognitive Systeme: Mensch, Teams, Systeme und Automaten*. Duisburg, Germany, Sept. 2011.
- [361] Roser, M., Ritchie, H., and Ortiz-Ospina, E. “World Population Growth”. In: *Our World in Data* (2013). URL: <https://ourworldindata.org/world-population-growth> (visited on 03/02/2022).
- [362] Russell, R. D. and Shampine, L. F. “A Collocation Method for Boundary Value Problems”. In: *Numerische Mathematik* 19.1 (1972), pp. 1–28. DOI: 10.1007/BF01395926.
- [363] Rusu, R. B. and Cousins, S. “3D is here: Point Cloud Library (PCL)”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.
- [364] Sabe, K., Fukuchi, M., Gutmann, J.-S., Ohashi, T., Kawamoto, K., and Yoshigahara, T. “Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA, Apr. 2004, pp. 592–597. DOI: 10.1109/ROBOT.2004.1307213.
- [365] Saff, E. B. and Kuijlaars, A. B. J. “Distributing Many Points on a Sphere”. In: *The Mathematical Intelligencer* 19.1 (Dec. 1977), pp. 5–11. DOI: 10.1007/BF03024331.
- [366] Saida, T., Yokokohji, Y., and Yoshikawa, T. “FSW (Feasible Solution of Wrench) for Multi-legged Robots”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Taipei, Taiwan, Sept. 2003, pp. 3815–3820. DOI: 10.1109/ROBOT.2003.1242182.
- [367] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., and Fujimura, K. “The intelligent ASIMO: System overview and integration”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. EPFL, Lausanne, Switzerland, Oct. 2002, pp. 2478–2483. DOI: 10.1109/IRDS.2002.1041641.
- [368] Schöps, T., Sattler, T., and Pollefeys, M. “SurfelMeshing: Online Surfel-Based Mesh Reconstruction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2020), pp. 2494–2507. DOI: 10.1109/TPAMI.2019.2947048.
- [369] Schunk GmbH & Co. KG. *FTE-AXIA80-DUAL SI-200-8/SI-500-20*. URL: https://schunk.com/de_en/gripping-systems/product/53291-1324514-fte-axia80-dual-si-200-8-si-500-20/ (visited on 07/26/2022).

- [370] Schwienbacher, M. “Entwicklung eines Kraft-Momentensensors für einen humanoiden Roboter”. Diploma thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2007.
- [371] Schwienbacher, M., Buschmann, T., Lohmeier, S., Favot, V., and Ulbrich, H. “Self-Collision Avoidance and Angular Momentum Compensation for a Biped Humanoid Robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 581–586. DOI: 10.1109/ICRA.2011.5980350.
- [372] Schwienbacher, M. “Efficient Algorithms for Biped Robots – Simulation, Collision Avoidance and Angular Momentum Tracking”. Dissertation. Germany: Technical University of Munich, 2013. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20140623-1175522-0-6>.
- [373] Scona, R., Nobili, S., Petillot, Y. R., and Fallon, M. “Direct Visual SLAM Fusing Proprioception for a Humanoid Robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada, Sept. 2017, pp. 1419–1426. DOI: 10.1109/IROS.2017.8205943.
- [374] Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F., and Caldwell, D. G. “Design of HyQ – a hydraulically and electrically actuated quadruped robot”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225.6 (2011), pp. 831–849. DOI: 10.1177/0959651811402275.
- [375] Semini, C., Barasuol, V., Goldsmith, J., Frigerio, M., Focchi, M., Gao, Y., and Caldwell, D. G. “Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max”. In: *IEEE/ASME Transactions on Mechatronics* 22.2 (2017), pp. 635–646. DOI: 10.1109/TMECH.2016.2616284.
- [376] Semini, C. et al. “Brief introduction to the quadruped robot HyQReal”. In: *Italian Conference on Robotics and Intelligent Machines (I-RIM)*. Rome, Italy, Oct. 2019, pp. 1–2. DOI: 10.5281/zenodo.4782613.
- [377] Sercos International. *Serial Realtime Communication System (Sercos)*. URL: <https://www.sercos.org> (visited on 03/11/2022).
- [378] Shoemake, K. “Animating Rotation with Quaternion Curves”. In: *SIGGRAPH Computer Graphics* 19.3 (July 1985), pp. 245–254. DOI: 10.1145/325165.325242.
- [379] Siciliano, B. and Slotine, J.-J. E. “A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems”. In: *IEEE International Conference on Advanced Robotics (ICAR)*. Pisa, Italy, June 1991, pp. 1211–1216. DOI: 10.1109/ICAR.1991.240390.
- [380] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. *Robotics: Modelling, Planning and Control*. Springer London, 2009. DOI: 10.1007/978-1-84628-642-1.
- [381] Siekmann, J., Green, K., Warila, J., Fern, A., and Hurst, J. “Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning”. In: *Robotics: Science and Systems*. July 2021. URL: <http://www.roboticsproceedings.org/rss17/p061.pdf> (visited on 03/02/2022).
- [382] Sodeyama, Y., Mizuuchi, I., Yoshikai, T., Nakanishi, Y., and Inaba, M. “A Shoulder Structure of Muscle-Driven Humanoid with Shoulder Blades”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, Canada, Aug. 2005, pp. 4028–4033. DOI: 10.1109/IROS.2005.1545123.
- [383] SoftBank / Aldebaran Robotics. *IEEE Robots: Nao*. 2008. URL: <https://robots.ieee.org/robots/nao> (visited on 03/02/2022).
- [384] Softbank / Aldebaran Robotics. *IEEE Robots: Pepper*. 2014. URL: <https://robots.ieee.org/robots/pepper> (visited on 03/03/2022).

- [385] Solà, J. “Quaternion kinematics for the error-state Kalman filter”. In: *arXiv* (Nov. 2017). DOI: 10.48550/ARXIV.1711.02508.
- [386] Sony. *IEEE Robots: Aibo*. 2018. URL: <https://robots.ieee.org/robots/aibo2018> (visited on 03/02/2022).
- [387] Sony. *IEEE Robots: Qrio*. 2003. URL: <https://robots.ieee.org/robots/qrio> (visited on 03/02/2022).
- [388] Sorge, K., Rossmann, T., Weidemann, H.-J., Funk, K., and Buschmann, T. *matvec*. Chair of Applied Mechanics, Technical University of Munich. 2009.
- [389] Stasse, O., Verrelst, B., Vanderborght, B., and Yokoi, K. “Strategies for Humanoid Robots to Dynamically Walk Over Large Obstacles”. In: *IEEE Transactions on Robotics* 25.4 (Aug. 2009), pp. 960–967. DOI: 10.1109/TRO.2009.2020354.
- [390] Stasse, O. et al. “TALOS: A new humanoid research platform targeted for industrial applications”. In: *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. Birmingham, UK, 2017, pp. 689–695. DOI: 10.1109/HUMANOIDS.2017.8246947.
- [391] Staufenberg, N.-S., Vielemeyer, J., Müller, R., Renjewski, D., and Rixen, D. J. “Virtual Pivot Point Analysis of the Humanoid Robot Lola”. In: *Dynamic Walking*. 2019.
- [392] Stephens, B. “Integral Control of Humanoid Balance”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Diego, CA, USA, Oct. 2007, pp. 4020–4027. DOI: 10.1109/IROS.2007.4399407.
- [393] Stephens, B. and Atkeson, C. “Modeling and Control of Periodic Humanoid Balance using the Linear Biped Model”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Paris, France, Dec. 2009, pp. 379–384. DOI: 10.1109/ICHR.2009.5379605.
- [394] Stephens, B. J. “State Estimation for Force-Controlled Humanoid Balance using Simple Models in the Presence of Modeling Error”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 3994–3999. DOI: 10.1109/ICRA.2011.5980358.
- [395] Stereolabs Inc. *ZED 2i – Industrial AI Stereo Camera*. URL: <https://www.stereolabs.com/zed-2i/> (visited on 08/09/2022).
- [396] Sterr, S. “Entwicklung einer taktilen Fußsohle für humanoide Roboter”. Bachelor’s thesis. Garching, Germany: Chair of Applied Mechanics, Technical University of Munich, 2020.
- [397] Stewart, D. “A Platform with Six Degrees of Freedom”. In: *Proceedings of the Institution of Mechanical Engineers* 180.1 (1965), pp. 371–386. DOI: 10.1243/PIME_PROC_1965_180_029_02.
- [398] Sugiura, H., Gienger, M., Janssen, H., and Goerick, C. “Real-Time Collision Avoidance with Whole Body Motion Control for Humanoid Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Diego, CA, USA, Oct. 2007, pp. 2053–2058. DOI: 10.1109/IROS.2007.4399062.
- [399] Sygulla, F., Ellensohn, F., Hildebrandt, A.-C., Wahrmann, D., and Rixen, D. “A Flexible and Low-Cost Tactile Sensor for Robotic Applications”. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. Munich, Germany, July 2017, pp. 58–63. DOI: 10.1109/AIM.2017.8013995.
- [400] Sygulla, F. and Rixen, D. “A force-control scheme for biped robots to walk over uneven terrain including partial footholds”. In: *International Journal of Advanced Robotic Systems* 17.1 (2020). DOI: 10.1177/1729881419897472.

- [401] Sygulla, F. “Dynamic Robot Walking on Unknown Terrain – Stabilization and Multi-Contact Control of Biped Robots in Uncertain Environments”. Dissertation. Germany: Technical University of Munich, 2022. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20220112-1614716-1-9>.
- [402] Tajima, R., Honda, D., and Suga, K. “Fast Running Experiments Involving a Humanoid Robot”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 1571–1576. DOI: 10.1109/ROBOT.2009.5152404.
- [403] Takanishi, A., Lim, H.-o., Tsuda, M., and Kato, I. “Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface”. In: *IEEE International Workshop on Intelligent Robots and Systems (IROS)*. Ibaraki, Japan, July 1990, pp. 323–330. DOI: 10.1109/IROS.1990.262408.
- [404] Takenaka, T., Matsumoto, T., and Yoshiike, T. “Real Time Motion Generation and Control for Biped Robot -1st Report: Walking Gait Pattern Generation-”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, Oct. 2009, pp. 1084–1091. DOI: 10.1109/IROS.2009.5354662.
- [405] Takenaka, T., Matsumoto, T., Yoshiike, T., and Shirokura, S. “Real Time Motion Generation and Control for Biped Robot -2nd Report: Running Gait Pattern Generation-”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA, Oct. 2009, pp. 1092–1099. DOI: 10.1109/IROS.2009.5354654.
- [406] Tanguy, A., Gergondet, P., Comport, A. I., and Kheddar, A. “Closed-loop RGB-D SLAM Multi-Contact Control for Humanoid Robots”. In: *IEEE/SICE International Symposium on System Integration (SII)*. Sapporo, Japan, Dec. 2016, pp. 51–57. DOI: 10.1109/SII.2016.7843974.
- [407] The LLVM Team. *Clang C Language Family Frontend for LLVM*. URL: <https://clang.llvm.org/> (visited on 02/17/2023).
- [408] The Qt Company. *Qt – Cross-platform Software Design and Development Tools*. URL: <https://www.qt.io/> (visited on 02/17/2023).
- [409] Thomas, G. C. and Sentis, L. “Towards Computationally Efficient Planning of Dynamic Multi-Contact Locomotion”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, Oct. 2016, pp. 3879–3886. DOI: 10.1109/IROS.2016.7759571.
- [410] Thürrner, G. and Wüthrich, C. A. “Computing Vertex Normals from Polygonal Facets”. In: *Journal of Graphics Tools* 3.1 (1998), pp. 43–46. DOI: 10.1080/10867651.1998.10487487.
- [411] Tondu, B., Ippolito, S., Guiochet, J., and Daidié, A. “A Seven-degrees-of-freedom Robot-arm Driven by Pneumatic Artificial Muscles for Humanoid Robots”. In: *The International Journal of Robotics Research* 24.4 (2005), pp. 257–274. DOI: 10.1177/0278364905052437.
- [412] Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. “An Efficient Acyclic Contact Planner for Multiped Robots”. In: *IEEE Transactions on Robotics* 34.3 (2018), pp. 586–601. DOI: 10.1109/TRO.2018.2819658.
- [413] Tsagarakis, N. G., Morfey, S., Medrano Cerda, G., Zhibin, L., and Caldwell, D. G. “Compliant Humanoid COMAN: Optimal joint stiffness tuning for modal frequency control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany, May 2013, pp. 673–678. DOI: 10.1109/ICRA.2013.6630645.
- [414] Tsagarakis, N. G. et al. “WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments”. In: *Journal of Field Robotics* 34.7 (2017), pp. 1225–1259. DOI: 10.1002/rob.21702.

- [415] Tucker, V. A. “The Energetic Cost of Moving About”. In: *American Scientist* 63.4 (1975), pp. 413–419.
- [416] Turk, G. *The PLY Polygon File Format*. Tech. rep. Stanford University, USA, 1994. URL: <http://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf> (visited on 09/27/2022).
- [417] Ulbrich, H., Buschmann, T., and Lohmeier, S. “Development of the Humanoid Robot LOLA”. In: *Modern Practice in Stress and Vibration Analysis VI*. Vol. 5. Applied Mechanics and Materials. Trans Tech Publications, Oct. 2006, pp. 529–540. DOI: 10.4028/www.scientific.net/AMM.5-6.529.
- [418] Unitree Robotics. *Go1*. URL: <https://www.unitree.com/products/go1> (visited on 03/02/2022).
- [419] Usmani, R. A. “Smooth spline approximations for the solution of a boundary value problem with engineering applications”. In: *Journal of Computational and Applied Mathematics* 6.2 (1980), pp. 93–98. DOI: 10.1016/0771-050X(80)90002-9.
- [420] Vahrenkamp, N., Asfour, T., and Dillmann, R. “Robot Placement based on Reachability Inversion”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany, May 2013, pp. 1970–1975. DOI: 10.1109/ICRA.2013.6630839.
- [421] Vahrenkamp, N. and Asfour, T. “Representing the robot’s workspace through constrained manipulability analysis”. In: *Autonomous Robots* 38.1 (Jan. 2015), pp. 17–30. DOI: 10.1007/s10514-014-9394-z.
- [422] Valgrind Developers. *Valgrind*. URL: <https://valgrind.org/> (visited on 02/17/2023).
- [423] van den Berg, J., Shah, R., Huang, A., and Goldberg, K. “ANA*: Anytime Nonparametric A*”. In: *AAAI: Annual Conference*. 2011.
- [424] van Heesch, D. *Doxygen – Generate Documentation from Source Code*. URL: <https://www.doxygen.nl/> (visited on 02/17/2023).
- [425] Varah, J. M. “On the Solution of Block-Tridiagonal Systems Arising from Certain Finite-Difference Equations”. In: *Mathematics of Computation* 26.120 (1972), pp. 859–868. DOI: 10.2307/2005868.
- [426] Vicon Motion Systems Limited. *Vicon – Award Winning Motion Capture Systems*. URL: <https://www.vicon.com/> (visited on 05/20/2022).
- [427] Vukobratović, M. and Stepanenko, J. “On The Stability of Anthropomorphic Systems”. In: *Mathematical Biosciences* 15.1 (1972), pp. 1–37. DOI: 10.1016/0025-5564(72)90061-2.
- [428] Vukobratović, M. and Borovac, B. “Zero-Moment Point – Thirty Five Years of its Life”. In: *International Journal of Humanoid Robotics* 1.1 (Mar. 2004), pp. 157–173. DOI: 10.1142/S0219843604000083.
- [429] Wahrmann, D., Hildebrandt, A.-C., Wittmann, R., Sygulla, F., Rixen, D., and Buschmann, T. “Fast Object Approximation for Real-Time 3D Obstacle Avoidance with Biped Robots”. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. Banff, Canada, July 2016, pp. 38–45. DOI: 10.1109/AIM.2016.7576740.
- [430] Wahrmann, D. “Autonomous Robot Walking in Unknown Scenarios. Perception, Modeling and Robustness in Dynamic Environments”. Dissertation. Germany: Technical University of Munich, 2018. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20181015-1395259-1-9>.
- [431] Wang, S. and Hauser, K. “Realization of a Real-time Optimal Control Strategy to Stabilize a Falling Humanoid Robot with Hand Contact”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, May 2018, pp. 3092–3098. DOI: 10.1109/ICRA.2018.8460500.

- [432] Werner, A., Henze, B., Rodriguez, D. A., Gabaret, J., Porges, O., and Roa, M. A. “Multi-Contact Planning and Control for a Torque-Controlled Humanoid Robot”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, Oct. 2016, pp. 5708–5715. DOI: 10.1109/IROS.2016.7759840.
- [433] Westervelt, E., Grizzle, J., and Koditschek, D. “Hybrid Zero Dynamics of Planar Biped Walkers”. In: *IEEE Transactions on Automatic Control* 48.1 (2003), pp. 42–56. DOI: 10.1109/TAC.2002.806653.
- [434] Whitney, D. E. “Resolved Motion Rate Control of Manipulators and Human Prostheses”. In: *IEEE Transactions on Man-Machine Systems* 10.2 (June 1969), pp. 47–53. DOI: 10.1109/TMMS.1969.299896.
- [435] Williams, J. W. J. “Algorithm 232: Heapsort”. In: *Communications of the ACM* 7.6 (1964), pp. 347–348. DOI: 10.1145/512274.512284.
- [436] Williams, T. and Kelley, C. *gnuplot*. URL: <https://gnuplot.sourceforge.net/> (visited on 02/28/2023).
- [437] Wind River Systems. *VxWorks RTOS*. URL: <https://www.windriver.com/products/vxworks> (visited on 03/17/2022).
- [438] Winter, D. A. *Biomechanics and Motor Control of Human Movement*. 4th ed. John Wiley & Sons, Inc., 2009. DOI: 10.1002/9780470549148.
- [439] Wittmann, R., Hildebrandt, A.-C., Ewald, A., and Buschmann, T. “An Estimation Model for Footstep Modifications of Biped Robots”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, IL, USA, Sept. 2014, pp. 2572–2578. DOI: 10.1109/IROS.2014.6942913.
- [440] Wittmann, R., Hildebrandt, A.-C., Wahrmann, D., Rixen, D., and Buschmann, T. “State Estimation for Biped Robots Using Multibody Dynamics”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, Sept. 2015, pp. 2166–2172. DOI: 10.1109/IROS.2015.7353667.
- [441] Wittmann, R., Hildebrandt, A.-C., Wahrmann, D., Rixen, D., and Buschmann, T. “Real-Time Nonlinear Model Predictive Footstep Optimization for Biped Robots”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Seoul, South Korea, Nov. 2015, pp. 711–717. DOI: 10.1109/HUMANOIDS.2015.7363432.
- [442] Wittmann, R., Hildebrandt, A.-C., Wahrmann, D., Sygulla, F., Rixen, D., and Buschmann, T. “Model-Based Predictive Bipedal Walking Stabilization”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico, Nov. 2016, pp. 718–724. DOI: 10.1109/HUMANOIDS.2016.7803353.
- [443] Wittmann, R. “Robust Walking Robots in Unknown Environments. Dynamic Models, State Estimation and Real-Time Trajectory Optimization”. Dissertation. Germany: Technical University of Munich, 2017. URL: <http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20171023-1352929-1-6>.
- [444] Wolpert, D. H. “The Lack of A Priori Distinctions Between Learning Algorithms”. In: *Neural Computation* 8.7 (Oct. 1996), pp. 1341–1390. DOI: 10.1162/neco.1996.8.7.1341.
- [445] Wright, S. J. “Stable Parallel Algorithms for Two-Point Boundary Value Problems”. In: *SIAM Journal on Scientific and Statistical Computing* 13.3 (1992), pp. 742–764. DOI: 10.1137/0913044.
- [446] Wu, S.-C., Tateno, K., Navab, N., and Tombari, F. “SCFusion: Real-time Incremental Scene Reconstruction with Semantic Completion”. In: *IEEE International Conference on 3D Vision (3DV)*. 2020, pp. 801–810. DOI: 10.1109/3DV50981.2020.00090.

- [447] Wu, S.-C., Tateno, K., Navab, N., and Tombari, F. *SCFusion: Semantic Completion Fusion*. 2020. URL: <https://github.com/ShunChengWu/SCFusion> (visited on 03/02/2022).
- [448] Wu, S.-C., Wald, J., Tateno, K., Navab, N., and Tombari, F. “SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7511–7521. DOI: 10.1109/CVPR46437.2021.00743.
- [449] Wu, S.-C., Wald, J., Tateno, K., Navab, N., and Tombari, F. *SGFusion: Scene Graph Fusion*. 2020. URL: <https://github.com/ShunChengWu/SceneGraphFusion> (visited on 03/02/2022).
- [450] Xinjilefu, X., Feng, S., and Atkeson, C. G. “Dynamic State Estimation using Quadratic Programming”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, IL, USA, Sept. 2014, pp. 989–994. DOI: 10.1109/IROS.2014.6942679.
- [451] Yoshiike, T., Kuroda, M., Ujino, R., Kanemoto, Y., Kaneko, H., Higuchi, H., Komura, S., Iwasaki, S., Asatani, M., and Koshiishi, T. “The Experimental Humanoid Robot E2-DR: A Design for Inspection and Disaster Response in Industrial Environments”. In: *IEEE Robotics & Automation Magazine* 26.4 (2019), pp. 46–58. DOI: 10.1109/MRA.2019.2941241.
- [452] Yoshikawa, T. “Manipulability of Robotic Mechanisms”. In: *The International Journal of Robotics Research* 4.2 (1985), pp. 3–9. DOI: 10.1177/027836498500400201.
- [453] Zacharias, F., Borst, C., Wolf, S., and Hirzinger, G. “The capability map: a tool to analyze robot arm workspaces”. In: *International Journal of Humanoid Robotics* 10.4 (2013). DOI: 10.1142/S021984361350031X.
- [454] Zhang, H., Han, X., and Yang, X. “Quintic B-spline collocation method for fourth order partial integro-differential equations with a weakly singular kernel”. In: *Applied Mathematics and Computation* 219.12 (2013), pp. 6565–6575. DOI: 10.1016/j.amc.2013.01.012.
- [455] Zhao, Y. and Sentis, L. “A Three Dimensional Foot Placement Planner for Locomotion in Very Rough Terrains”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Osaka, Japan, Nov. 2012, pp. 726–733. DOI: 10.1109/HUMANOIDS.2012.6651600.