TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Luft- und Raumfahrt

Constrained Model Predictive Control for Real-Time Tele-Operation Motion Planning

Mingming Wang

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. dr. ir. Daniel J. Rixen

Prüfer der Dissertation:

1. Univ.-Prof. Dr. rer. nat. Ulrich Walter

2. Univ.-Prof. Dr.-Ing. Alin Albu-Schäffer

Die Dissertation wurde am 14.01.2015 bei der Technische Universität München eingereicht und durch die Fakultät für Maschinenwesen am 17.06.2015 angenommen.

Acknowledgements

The 4 years' Ph.D life is a special and important experience in my life. It is full of passion, hope, desperation, struggle and happiness. As a foreigner in Germany, countless people from different countries have given me their hands to support and encourage me to successfully get through my Ph.D life.

The first gratitude is devoted to Prof. Walter and Prof. Luo. Prof. Walter accepted me as one of the members in his institute and gave me great freedom to do what I am really interested in. As my primal supervisor, Prof. Luo guides me in my research and encourages me to go further. I thank Roberto Lampariello who gave me lots of suggestions at the beginning of my first touch with Robotics. Many thanks to Prof. Igenbergs for his encouragement and support. I also give my appreciation to Dr. Höhn and Dr. Rott for their helpful counsel and enduring my Chinglish. I thank Lars for his support on computer. Thanks to Claas for his picking me up from airport. He and Jan can always lift my spirit in the institute. I also thank Hein and Andi for sharing their experience with me. Further thanks to all the colleagues from Institute of Astronautics, it is the common experience with you makes my Ph.D journey more colourful.

My German friends are a constant source of support. Evelyn, Herta, Ring, Family Yin and Family Babara gave me lots of help. They introduced Germany to me in detail and helped me to pass through the dark period. The time spent with them will be a beautiful memory in my life.

The last gratitude is devoted to my family. My parents encourage me to become the one who I want to be and always support me to go further. My wife Ling, who perfectly manage our family and takes care of our little daughter Emily, owns my constant gratitude. It is your love, tolerance, endurance and optimism that supports me to finish this work.

Mingming Wang
Garching, 01.07.2014

Zusammenfassung

Zunehmende Anforderungen an den Satelliten-Service in der Erdumlaufbahn (On Orbit Servicing, OOS) erfordern neue Technologien, um diese auszuführen. Wegen ihrer Flexibilität, Multifunktionalität und Erweiterbarkeit sind Weltraum-Roboter eine der vielversprechendsten Lösungen dafür. In der vorliegenden Arbeit wird für die Teleoperations-Aufgaben eines Weltraumroboters eine Architektur mit verteilter Echtzeit-Simulation basierend auf einem Data Distribution Service (DDS) vorgestellt. Außerdem wird für den frei schwebenden Roboter ein Model Predictive Control (MPC) Framework entworfen, das Kollisionen und Singularitäten bei Roboterbewegungen vermeidet. Ziel der vorliegenden Arbeit ist, für so eine Simulation eine allgemeine Architektur zu entwerfen, die dem Boden-Operateur eine intuitive Erkennung ermöglicht, und ein neues MPC Framework, das die vielfachen Einschränkungen beim Betrieb des Weltraumroboters in Betracht zieht und so die Leistung und Effektivität verbessert.

Zu diesem Zweck wurde eine neue verteilte Echtzeitsimulation, RACOON, entwickelt, die auf einem DDS in Matlab/Simulink/Stateflow basiert. Sie liefert dem Boden-Operateur eine intuitive Ansicht des Roboters und erweitert die Simulations-Architektur für kollaborative Teleoperation, Mehrkörperdynamik, Autonomous Mission Management (AMM), Weg- und Bahn-Planung, Bewegungskontrolle, virtuelle Realität, usw., für heutige komplexe Robotikmissionen.

Zunächst wird die Dynamik des Raumroboters mit einer Baumstruktur unter Nutzung von Konzepten aus der Graphentheorie und räumlichen Beschreibungen dargestellt, um damit ein Nonlinear Model Predictive Control (NMPC) Model zu verwirklichen. Aus der Topologie des Roboters wird eine Inertia Mapping Matrix (IMM) hergeleitet werden, um die Spärlichkeit eines JSIM und die Komplexität von CRBA Algorithmen zu untersuchen und die Dekomposition des JSIM zu unterstützen.

Danach werden für redundante Manipulatoren unter Berücksichtigung der Priorität von Redundanz und Aufgaben die Singularitäts- und Kollisions-Probleme untersucht. Für die Vermeidung von Singularitäten wird basierend auf dem Konzept von Handhabbarkeits-Ellipsoiden eine sogenannte STR-Methode entwickelt. Zur Kollisionsvermeidung, wird eine neue Zwei-Kontrollpunkte-Strategie vorgeschlagen, die Schwingung der Gelenkgeschwindigkeit dämpft und so eine glattere Bewegungsbahn des Roboterarms erzeugt.

In einer Anwendungsstudie wird gezeigt, wie bei der Ergreifung eines nicht kooperativen Zielsatelliten ein so gestalteter NMPC einen Manipulatorarm kollisions- und singularitätsfrei bewegt. Die Wirksamkeit und die Leistungsfähigkeit der hier neuen vorgeschlagenen Methoden werden mit traditionellen Methoden verglichen. Diese Arbeit zeigt die Machbarkeit und die berlegenheit eines eingeschränkten MPC für den Einsatz eines Weltraumroboters.

Abstract

The increasing demands of On-Orbit Servicing (OOS) require new technologies to complete the OOS mission. Space robot, since it is flexible, multi-functional and extendable, is one of the most promising solutions for OOS. In this thesis, a distributed real-time simulation architecture based on DDS has been presented for space robotic tele-operation tasks. Besides, a constrained MPC framework considering system input/output, anti-collision and anti-singularity constraints has been developed for the free-floating space robot. The objective of this thesis is to provide a general simulation architecture with intuitive perception for the operators on ground, and a new control framework considering multiple constraints for the application of the space robot while the performance and effectiveness are improved.

For that purpose, a new distributed real-time simulation architecture, RACOON, has been implemented based on DDS in the environment Matlab/Simulink/Stateflow. The new simulation architecture provides the operator on ground an intuitive view of space robot and makes the simulation architecture open for collaborative tele-operation. As a complete simulation system, the multi-body dynamics, AMM, path & trajectory planning, motion control, virtual reality etc. subsystems, integrate together seamlessly to complete the whole space robotic missions.

In order to realize the new control framework based on NMPC, firstly, the dynamics of the space robot with tree structure using the concepts of graph theory and spatial notation is introduced. A new IMM is derived from the topology of the space robot, which can be employed to analyse the sparsity of the JSIM and the complexity of the CRBA algorithm, and assist the decomposition of the JSIM. Secondly, the singularity and collision avoidance issues of redundant manipulators are investigated considering the redundancy and task priority. A so-called STR method is developed based on the concept of manipulability ellipsoid for singularity avoidance. For collision avoidance, a new strategy combining two control points is proposed to restrain the vibration of joint velocity and generate smoother joint trajectory reference. Thirdly, application of NMPC to space manipulator in capturing an uncooperative target satellite is investigated. The system input/output, collision/singularity constraints in practice imposed on decision variables are translated into linear inequalities as part of NMPC. An on-line QP algorithm with prioritized constraints is adopted to find the optimal control efforts.

The effectiveness and performance of the new proposed methods in this thesis are demonstrated by comparison to traditional methods. Well-designed end-effector path, together with the NMPC guarantees the success of the space manipulator to complete the capture of the un-cooperative target satellite. This work shows the feasibility and validity of constrained MPC applied in the field of space robot. Furthermore, it can be used to support further OOS mission and space exploration.

Contents

Acknowledgements V

Zusammenfassung VII

Abstract IX

List of Figures XV

List of Tables XVII

Nomenclature XIX
Acronyms . . . XIX
Symbols . . . XXII
Indices . . . XXIV

1. Introduction 1
1.1. Motivation . . . 2
1.1.1. On-Orbit Servicing . . . 2
1.1.2. Why Space Robotics . . . 3
1.2. State of the art . . . 4
1.2.1. OOS Technology Demonstrators . . . 4
1.2.2. Space Robotics Demonstrators . . . 7
1.2.3. Technical Challenges of Space Robotics . . . 10
1.3. Hypothesis and Problem Statements . . . 12
1.4. Scope of Work . . . 13
1.4.1. Research Scope within Space Robotics . . . 13
1.4.2. Thesis Roadmap . . . 13

2. Simulation System Design 17
2.1. Mission Profile . . . 17
2.2. Simulation Overall Design . . . 19
2.2.1. Racoon Design . . . 19
2.2.2. Simulation Environment . . . 20
2.2.3. Data Distribution Service . . . 22
2.3. RacoonSim Design . . . 24
2.3.1. Multi-body Dynamics . . . 25
2.3.2. Autonomous Mission Management . . . 26
2.3.3. Path & Trajectory Planning . . . 28
2.3.4. Motion Control . . . 29
2.3.5. Virtual Reality & Head-Up Display . . . 29
2.4. Summary . . . 31

3. Multibody Dynamics **33**
3.1. Graphy Theory . . . 34
3.2. Spatial Notation . . . 35
3.3. Dynamics Algorithm . . . 36
3.3.1. Lagrange Formulation . . . 36
3.3.2. Composite Rigid Body Algorithm . . . 37
3.3.3. Inertia Mapping Matrix . . . 38
3.3.4. Modified CRBA . . . 39
3.4. IMM Application . . . 39
3.4.1. Branch-induced Sparsity . . . 40
3.4.2. CRBA Computational Cost Analysis . . . 40
3.4.3. JSIM Factorization Analysis . . . 42
3.5. Case Study . . . 44
3.6. Summary . . . 47

4. Kinematic Control of Manipulator **49**
4.1. Inverse Kinematics . . . 49
4.1.1. Redundancy and Task Priority . . . 49
4.1.2. General Solution for Inverse Kinematics . . . 50
4.2. Singularity Avoidance . . . 51
4.2.1. Manipulability Ellipsoid . . . 52
4.2.2. Singular Task Reconstruction . . . 53
4.2.3. STR with Multiple Subtasks . . . 55
4.2.4. Simulation Results . . . 57
4.3. Obstacle Detection and Avoidance . . . 64
4.3.1. Collision Detection . . . 64
4.3.2. Collision Avoidance Strategy . . . 67
4.3.3. Simulation Results . . . 70
4.4. Summary . . . 74

5. Model Predictive Control **77**
5.1. Model Predictive Control . . . 77
5.1.1. Principle and Formulation . . . 78
5.1.2. MPC Properties . . . 80
5.2. NMPC Applied to Space Robot . . . 81
5.2.1. Free-Floating Space Robot . . . 82
5.2.2. Feedback Linearization . . . 82
5.2.3. Observer Design . . . 84
5.2.4. Optimization Index . . . 86
5.3. Inequality Constraints . . . 87
5.3.1. Input/Output Constraints . . . 87
5.3.2. Obstacle/Singularity Constraints . . . 89
5.4. Quadratic Programming . . . 92
5.4.1. KKT Conditions . . . 92
5.4.2. QP with Prioritized Constraints . . . 93
5.5. Simulation Study . . . 95
5.5.1. Simulation Set-up . . . 95
5.5.2. Approach to the Target . . . 95
5.5.3. Tracking a Predefined Path . . . 97

5.5.4. Tracking a Point on the Target 99
5.6. Summary 102

6. Conclusions and Future Research **105**
6.1. Conclusions 105
6.2. Future Research 106

A. Bibliography **109**

List of Figures

1.1. Engineering design MODPV loop . . . 1
1.2. OOS manned projects . . . 5
1.3. OOS unmanned projects . . . 7
1.4. Space roboitcs projects . . . 9
1.5. OOS demonstrators classification . . . 10
1.6. Structure of the work . . . 15

2.1. Mission profile of space robot . . . 19
2.2. Subsystems of Racoon Lab . . . 20
2.3. Simulation environment of Racoon . . . 21
2.4. DDS Structure . . . 22
2.5. Racoon system set-up . . . 24
2.6. Racoon simulation system set-up . . . 25
2.7. Workflow of high-level OOS missions . . . 27
2.8. Workflow of capture phase . . . 27
2.9. VRML mechanism and its interfaces . . . 30
2.10. GUI, VR and HUD in MCC . . . 31

3.1. Example of tree structure manipulators . . . 34
3.2. Generalized center of mass of link i . . . 35
3.3. Relationship among path matrix, IMM and JSIM . . . 38
3.4. Tree configuration and corresponding JSIM of one base with 4 links . . . 40
3.5. Humanoid and corresponding un-branched chain . . . 44
3.6. Operations numbers of CRBA and ABA . . . 46
3.7. Cost ratio of CRBA and ABA . . . 46

4.1. Schematic diagram of inverse kinematics based on task priority . . . 51
4.2. 3 revolute planar manipulator and its manipulability ellipsoid . . . 53
4.3. Schematic diagram of singular task reconstruction . . . 54
4.4. The functional behaviour of the turning parameters for singularity avoidance 55
4.5. Block scheme of STR for multiple subtasks . . . 56
4.6. 3 revolute planar manipulator and subtasks . . . 57
4.7. Singularity avoidance with single task by STR method . . . 58
4.8. Singularity avoidance with single task by DLS method . . . 59
4.9. Singularity avoidance with two subtasks by STR method . . . 60
4.10. Singularity avoidance with two subtasks by DLS method . . . 61
4.11. Singularity avoidance with two subtasks and higher gain by STR method . . . 62
4.12. Singularity avoidance with two subtasks and higher gain by DLS method . . . 63
4.13. The relationship of two line segments . . . 65
4.14. The relationship between line segment and triangle . . . 66
4.15. Collision avoidance with two control points . . . 67

4.16. The functional behaviour of the turning parameters for collision avoidance . . 69
4.17. 10 revolute planar manipulator and obstacles 71
4.18. Path tracking without collision avoidance 72
4.19. Path tracking with one control point for collision avoidance 73
4.20. Path tracking with two control points for collision avoidance 74

5.1. Schematic diagram of MPC fundamental principle 78
5.2. Basic NMPC control loop 79
5.3. The mismatch between open-loop prediction and closed-loop behaviour . . . 81
5.4. Schematic diagram of space robot 82
5.5. Linear feedback with NMPC for space robot 84
5.6. Schematic diagram of anti-collision 89
5.7. Schematic diagram of anti-singularity 90
5.8. Unfolding the space manipulator using RMAC method 96
5.9. Unfolding the space manipulator using NMPC method 97
5.10. Tracking infinite ring with anti-collision constraints 98
5.11. Tracking line with anti-singularity constraints 99
5.12. Relative relationship of end-effector frame and capture point frame 100
5.13. Tracking capture point on target satellite 101
5.14. Screenshot of tracking the capture point on target 102

List of Tables

1.1. OOS overview and its typical applications (Waltz (1993)) 3
1.2. OOS manned projects 5
1.3. OOS unmanned projects 6

3.1. Modified composite rigid body algorithm 39
3.2. Required operation cost of different transforms 42
3.3. LTL and LTDL factorization algorithms 43

4.1. Collision detection algorithms 66

Nomenclature

Acronyms

ABA . . . . . Articulated Body Algorithm
AFRL . . . . . Air Force Research Laboratory
AI . . . . . Artificial Intelligence
AMM . . . . . Autonomous Mission Management
API . . . . . Application Programming Interface
ASTRO . . . . . Autonomous Space Transport Robotic Operations
ATV . . . . . Autonomous Transfer Vehicle

BVH . . . . . Bounding Volume Hierarchy

CAD . . . . . Computer Aided Design
CESSORS . . . . Chinese Experimental Space System for On-Orbit Robotistic Services
CNES . . . . . Centre National d'Etudes Spatiales
CRBA . . . . . Composite Rigid Body Algorithm
CSA . . . . . Canadian Space Agency

DARPA . . . . . Defence Advanced Research Projects Agency
DART . . . . . Demonstration of Autonomous Rendezvous Technology
DCPS . . . . . Data-Centric Publish/Subscribe
DDS . . . . . Data Distribution Service
DEOS . . . . . DEutsche Orbitale Servicing Mission
DH . . . . . Denavit-Hartenberg
DLR . . . . . Deutsche Luft und Raumfahrttechnik
DLRL . . . . . Data Local Reconstruction Layer
DLS . . . . . Damped Least-Squares
DMC . . . . . Dynamic Matrix Control
DOF . . . . . degree-of-freedom

DOR . . . . . degree-of-redundancy

ESA . . . . . European Space Agency
ETS-VII . . . . . Engineering Test Satellite-VII
EVA . . . . . Extra-Vehicular Activities

FREND . . . . . Front-end Robotics Enabling Near-Term Demonstration
FSM . . . . . Finite State Machine

GEO . . . . . Geostationary Orbit
GJM . . . . . Generalized Jacobian Matrix
GPC . . . . . Generalized Predictive Control
GPS . . . . . Global Positioning System
GUI . . . . . Graphical User Interface

HIL . . . . . Hardware-in-loop
HST . . . . . Hubble Space Telescope
HTV . . . . . H-II Transfer Vehicle
HUD . . . . . Head-Up Display

IDL . . . . . Interface Definition Language
IMM . . . . . Inertia Mapping Matrix
IMU . . . . . Inertia Measurement Unit
ISS . . . . . International Space Station

JAXA . . . . . Japan Aerospace Exploration Agency
JSIM . . . . . Joint-Space Inertia Matrix

KF . . . . . Kalman Filter
KKT . . . . . Karush-Kuhn-Tucker

LEO . . . . . Low Earth Orbit
LP . . . . . Linear Programming

MCC . . . . . Mission Control Center
MPC . . . . . Model Predictive Control
MPD . . . . . Maximum Projection Distance

MSS . . . . . Mobile Servicing System

NASA . . . . . National Aeronautics and Space Administration
NASDA . . . . . NAtional Space Development Agency of Japan
NMPC . . . . . Nonlinear Model Predictive Control
NRL . . . . . Naval Research Laboratory

OBB . . . . . Oriented Bounding Box
OBDH . . . . . On-Board Data Handling
OE . . . . . Orbital Express
OLEV . . . . . Orbital Life Extension Vehicle
OMG . . . . . Object Management Group
OOS . . . . . On-Orbit Servicing
ORU . . . . . Orbital Replacement Unit

PRISMA . . . . . Prototype Research Instruments and Space Mission technology Advancement
PUMA . . . . . Programmable Universal Machine for Assembly

QoS . . . . . Quality of Service
QP . . . . . Quadratic Programming

RACOON . . . . Robotic Actuation, Control and On Orbit Navigation Laboratory
RacoonSim . . . Racoon Simulation
RHC . . . . . Receding Horizon Control
RMAC . . . . . Resolved Motion Acceleration Control
RNEA . . . . . Recursive Newton-Euler Algorithm
ROKVISS . . . . RObotic Components Verification on the ISS
ROTEX . . . . . Robot Technology Experiment

SMMR . . . . . Solar Maximum Mission Repair
SPDM . . . . . Special Purpose Dexterous Manipulator
SRMS . . . . . Shuttle Remote Manipulator System
SSC . . . . . Swedish Space Corporation
SSL . . . . . Space Systems Laboratory
SSN . . . . . Space Surveillance Network

SSTC . . . . . Shenzhen Space Technology Center

STR . . . . . Singular Task Reconstruction

STS . . . . . Space Transportation System

SVD . . . . . Singular Value Decomposition

TFX . . . . . Telerobotic Flight Experiment

TM/TC . . . . . Telemetry/Telecommand

VR . . . . . Virtual Reality

VRML . . . . . Virtual Reality Modelling Language

XML . . . . . Extensible Markup Language

XSS . . . . . Experimental Satellite System

Symbols

$\boldsymbol{a}$ . . . . . position vector of mass center
α_0 . . . . . escaping gain for singularity avoidance
α_c . . . . . turning parameter of repulsive motion for collision avoidance
α_d . . . . . weight parameter for singularity avoidance
α_e . . . . . turning parameter of cancelling primary task
α_h . . . . . turning parameter of repulsive motion for singularity avoidance
α_v . . . . . weight parameter for singularity avoidance
$\boldsymbol{c}$ . . . . . bias vector
$\boldsymbol{c}a$ . . . . . composite rigid body add
$\boldsymbol{c}t$ . . . . . composite rigid body transform
$\boldsymbol{c}t_a$ . . . . . composite rigid body transform for DH node
$\boldsymbol{c}t_b$ . . . . . composite rigid body transform for non-DH node
$\boldsymbol{c}t_c$ . . . . . composite rigid body transform for node connecting direct to the base
D_0 . . . . . numbers of moving links without connecting to the base
D_{0a} . . . . . numbers of moving links without connecting to the base for DH node
D_{0b} . . . . . numbers of moving links without connecting to the base for non-DH node
D_{0c} . . . . . number of saving operations in composite rigid body transform
D_1 . . . . . numbers of non-zero elements above the diagonal of IMM
D_{1a} . . . . . numbers of non-zero elements above the diagonal of IMM for DH node
D_{1b} . . . . . numbers of non-zero elements above the diagonal of IMM for non-DH node
D_{1c} . . . . . number of saving operations in vector transform
D_2 . . . . . sum of elements above the diagonal of IMM
d_{if} . . . . . influence zone of collision
$\mathbf{D}$ . . . . . diagonal matrix
d_{sr} . . . . . security zone of collision

d_{uf} . . . . . unsafe zone of collision
$\mathbf{E}$ . . . . . identity matrix
$\mathcal{E}$ . . . . . edge set
ϵ_{DLS} . . . . . threshold to active the DLS method
η_{ac} . . . . . damper coefficient of anti-collision constraint
η_{as} . . . . . damper coefficient of anti-singularity constraint
$\bar{\boldsymbol{f}}$ . . . . . spatial force
$\boldsymbol{f}^x$ . . . . . external force/torque vector
$\boldsymbol{f}$ . . . . . force vector
Γ . . . . . cost function
$\boldsymbol{h}$ . . . . . arbitrary vector
$\mathbf{H}$ . . . . . joint-space inertia matrix
$\bar{\mathbf{H}}$ . . . . . spatial inertia matrix
$\bar{\mathbf{I}}$ . . . . . inertia matrix
$\mathbf{J}$ . . . . . Jacobian matrix
$\bar{\mathbf{J}}$ . . . . . matrix projection onto the null-space
$\mathbf{J}^+$ . . . . . pseudo-inverse of Jacobian matrix
$\mathcal{K}$ . . . . . computational cost of algorithm
$\mathbf{K}$ . . . . . gain matrix
$\mathcal{L}$ . . . . . Lagrangian
$\mathbf{L}$ . . . . . lower triangular matrix
λ_{ac} . . . . . switching function for anti-collision constraint
λ_{as} . . . . . switching function for anti-singularity constraint
$\boldsymbol{\lambda}(\cdot)$ . . . . . parent array
$\mathbf{M}$ . . . . . inertia mapping matrix
m . . . . . mass
$\boldsymbol{m}$ . . . . . moment vector
$\mathbf{M}^*$ . . . . . augmented inertia mapping matrix
N_c . . . . . control horizon
n_{nDH} . . . . . number of non-DH nodes
N_p . . . . . prediction horizon
O . . . . . algorithm complexity representation
$\mathcal{O}$ . . . . . obstacles set
$\mathcal{P}$ . . . . . link set
$\boldsymbol{q}$ . . . . . generalized coordinate
$\dot{\boldsymbol{q}}$ . . . . . generalized velocity
$\ddot{\boldsymbol{q}}$ . . . . . generalized acceleration
${}^j\mathbf{R}_i$ . . . . . coordinate transformation matrix
${}^j\boldsymbol{r}_i$ . . . . . position vector from the origin of frame i to that of frame j
$\mathbb{R}$ . . . . . real set
$\mathcal{S}_{act}$ . . . . . active set
$\boldsymbol{s}$ . . . . . motion axis of generalized link
$\mathbf{S}$ . . . . . cross product operator
σ . . . . . singular value
σ_{if} . . . . . influence zone of singularity
σ_m . . . . . minimum singular value
σ_{sr} . . . . . security zone of singularity
σ_{uf} . . . . . unsafe zone of singularity

$\boldsymbol{\tau}$ . . . . . internal joint torque vector
$\mathcal{T}$ . . . . . kinetic energy
$\mathbf{T}$ . . . . . path matrix
$\boldsymbol{\theta}$ . . . . . joint position vector
$\dot{\boldsymbol{\theta}}$ . . . . . joint velocity vector
$\mathbf{T}^*$ . . . . . augmented path matrix
$\boldsymbol{u}$ . . . . . control input vector
$\boldsymbol{u}_m$ . . . . . left singular vector correspond to minimum singular value
$\mathcal{U}$ . . . . . potential energy
$\mathcal{V}$ . . . . . vertex set
$\mathcal{VR}$ . . . . . Voronoi region
$\bar{\boldsymbol{v}}$ . . . . . spatial velocity
$\boldsymbol{v}$ . . . . . translational velocity
$\boldsymbol{v}t$ . . . . . vector transform
$\boldsymbol{v}t_a$ . . . . . vector transform for DH node
$\boldsymbol{v}t_b$ . . . . . vector transform for non-DH node
$\boldsymbol{v}t_c$ . . . . . vector transform for node connecting direct to the base
$\boldsymbol{\omega}$ . . . . . angular velocity
$\boldsymbol{x}$ . . . . . system states vector
$\boldsymbol{x}_d$ . . . . . desired end-effector path
$\dot{\boldsymbol{x}}_c$ . . . . . end-effector velocity with feedback
$\dot{\boldsymbol{x}}_d$ . . . . . desired end-effector velocity
$\dot{\boldsymbol{x}}_e$ . . . . . end-effector velocity
$\dot{\boldsymbol{x}}_o$ . . . . . obstacle avoidance velocity
$\dot{\boldsymbol{x}}_p$ . . . . . end-effector modified velocity
$\boldsymbol{x}_e$ . . . . . end-effector path
$\boldsymbol{\xi}$ . . . . . generalized force
${}^j\mathbf{X}_i$ . . . . . spatial transformation matrix
$\boldsymbol{y}$ . . . . . control output vector

Indices

x . . . . . scalar
$\overline{x}$ . . . . . maximum within interval
$\underline{x}$ . . . . . minimum within interval
$\hat{x}$ . . . . . value from estimation
x^* . . . . . values in optimum
$\boldsymbol{x}$ . . . . . vector
$\mathbf{X}$ . . . . . matrix
$[\cdot]^{\mathrm{T}}$ . . . . . transpose of $\cdot$

1. Introduction

> [On President Bush's plan to get to Mars in 10 years] *Stupid. Robots would do a better job and be much cheaper because you don't have to bring them back.*
>
> —Stephen W. Hawking

It's the human demands and desires that stimulate the scientists and engineers to do scientific research to explore the unknown world and build new machines working for us. Normally, the general research procedure can be illustrated as in Figure 1.1.

People want to invent/build something new based on their demands or the motivation. The next step would be orientation, where generally series of solutions are emerged. The orientation phase will make our thoughts divergence, to search all possible feasible solutions. As a matter of fact, since there are financing, technical, human capital etc. issues, it is impossible to implement all of the possible solutions. Some of the resolution will be chosen to move forward based on the specific criteria among these. A decision phase is critical to evaluate all the possible solutions and find out the potentially best ones. Subsequently, a model, which is also named as prototype, will be established both in digital and physical environment. This prototyping design is used to check the feasibility of the solution, and demonstrate the technologies and performance of the proposed design. All aspects of the prototype, such as its hardware, software, functionality, etc. will be tested in the verification phase. These tests will indicate the cons and pros of the design and show up whether it meets the proposed demands and requirements. All these five steps finally form an engineering design loop, which will also show in this thesis. If the prototype cannot pass the test and fulfil the proposed demands, the design loop will proceed iteratively until the design makes the human satisfied, where then can stop the loop.

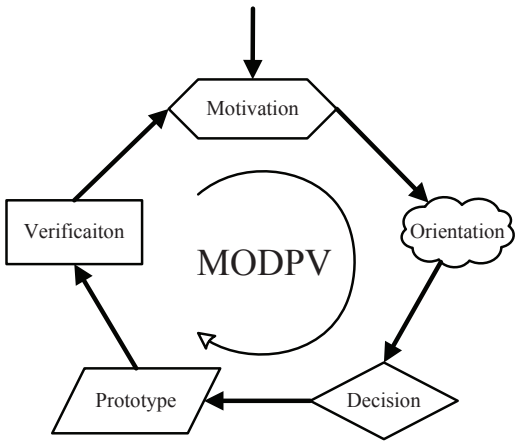


Figure 1.1.: Engineering design MODPV loop

1.1. Motivation

The motivation of this research origins from two aspects. Objectively speaking, there are thousands of malfunctioning satellites are currently floating in outer space which are harmful for future satellite launch and space operations. In the past two decades, some of the most extraordinary successes in space exploration have emphasized the growing importance of On-Orbit Servicing (OOS). The challenges have moved beyond simply launching complex spacecraft and system. What we need to face are the fully exploitation of the flight systems, construction of large structures in situ to enable new scientific ventures, and provide systems that reliably and cost-effectively support the next step in space exploration. The vision for OOS is straightforward to refuel, repair, or upgrade satellites after launched.

The other aspect of research motivation comes from the space race of our competitors which are hostile to us. Just as was illustrated in *Destruction and Creation* by John R. Boyd in Boyd (1987):

> *In a real world of limited resources and skills, individuals and groups form, dissolve and reform their cooperative or competitive postures in a continuous struggle to remove or overcome physical and social environmental obstacles. In a cooperative sense, where skills and talents are pooled, the removal or overcoming of obstacles represents an improved capacity for independent action for all concerned. In a competitive sense, where individuals and groups compete for scarce resources and skills, an improved capacity for independent action achieved by some individuals or groups constrains that capacity for other individuals or groups. Naturally, such a combination of real world scarcity and goal striving to overcome this scarcity intensifies the struggle of individuals and groups to cope with both their physical and social environments.*

From above illustration, one can derive that it is the cooperation and competition between individuals or organizations that motivate us to gain an improved capacity which intensifies the struggle of individuals and groups to enhance self-development and research activities. From this viewpoint, OOS has already become such an ability which extensively investigated and strengthened by world wild research and academic agencies. A list of space projects about OOS proceeded in the past and planned in the future will be described in section 1.2.

1.1.1. On-Orbit Servicing

Since a wide spectrum of using cases exists in space, OOS would be of great benefit for space exploration, which can reduce risk of mission failure and mission cost, increase mission performance and flexibility, and enable new missions. OOS includes a variety of applications, which can be grouped into five main operations.

Table 1.1.: OOS overview and its typical applications (Waltz (1993))

Operations	Typical Applications
Assembly	Spacecraft constructions Spacecraft update Deployment of appendages
Orbit transfer	Orbit corrections Retrieval from orbit Earth return
Re-supply	Consumables Components
Maintenance and repair	Inspection Modification Cleaning and resurfacing Tests and checkout
Special	Emergencey operations Scavenging Attitute control

Accordingly, On-Orbit Servicing (OOS) can be defined as follows refer to Wikipedia (2014)

> OOS includes installation, maintenance, and repair work on an orbiting man-made object (satellite, space station, space vehicle, etc.) with the aim of extending the useful life of the target object and/or enhancing the capability of the target (upgrade).

1.1.2. Why Space Robotics

Robotics is the branch of technology that deals with the design, construction, operation, structural disposition, manufacture and application of robots. it is concerned with the study of those machines which can replace human beings in the execution of a task, as regards both physical activity and decision making.

With the advancement of science and technology, the researchers are coming up with innovative ideas to construct robots that could simply the sophisticated tasks. Currently a vast spectrum of robots working around our daily life, such as on shop-floor, in the operation room, in rehabilitation centres and even at home. There are also many other applications of robotics in area where use of human is impractical and undesirable, and these are undersea and planetary exploration, satellite retrieval and repair, defusing of explosive devices, and work in radioactive environments, which induced different kinds of robots, such as manipulators, motion generators, loco-motors, swimming robots and flying robots that can be used to meet above specific requirements.

Refer to Table 1.3, where an OOS spacecraft is uncrewed, and with broad variety of missions to be executed on-orbit, in combination with the unpredictable nature of the serving missions, which call for a flexible and multi-functional flight segment. Robotic system is the

only means available today to fulfil these requirements. The application of space robotics in outer space has the following major advantages:

- The human operator has no needs to be on-orbit which greatly reduces the risk of human and launch cost;
- robot with dexterous end-effector can perform different types of tasks, sch as grasping, observation, ORU exchange, assembly, etc., while other OOS technology can not possess all these capacities simultaneously;
- With predefined interface, robot can be integrated into the spacecraft as an independent module;
- The workspace of robot is predictable and controllable, and the operation in its workspace is much more accurate;
- With pre-designed system or tele-operation, the space robot has the complete autonomy and flexibility to solve collision avoidance and emergency situation.

1.2. State of the art

After the first satellite launched into space 60 years ago, thousands of various types of spacecraft, such as communication, telemetry, observation etc. have been sent into space by different countries and organizations. According to the United States SSN, until the end of 2013, there are more than 21,000 objects larger than 10 cm orbiting the earth, while only 1071 satellites are operational Cain (2013). However, there are still no routine OOS procedures provided to remove or repair these defunct objects. As a matter of fact, most malfunctioning spacecraft require only a minor maintenance operation on-orbit to return to its operation status. Therefore, the accomplishment of OOS missions would be of great benefit for spacecraft operations, since a wide spectrum of applications exist as illustrated in Table 1.1.

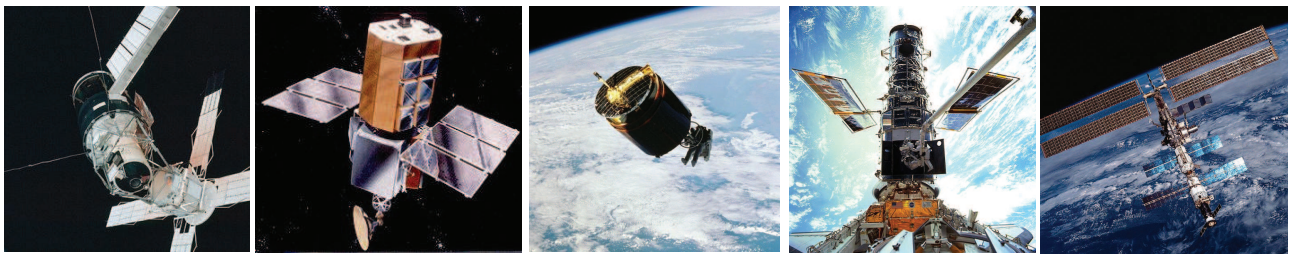
Before going deep into discussing the topics, some of the seminal missions in the past, present and future which helped us to define OOS are reviewed. There are two kinds of OOS technology, one is human involved spacecraft, the other is unmanned space demonstrators, both of which will be illustrated in the follows. Through this review, it becomes apparent that, OOS has become a technical developing tendency as specific critical needs.

1.2.1. OOS Technology Demonstrators

The first planned OOS mission was performed in 1973 on Skylab, through EVA. After that, refer to NASA (2010), numbers of OOS projects have been conducted with human-in-the-loop which are listed in Table 1.2 and Figure 1.2.

Table 1.2.: OOS manned projects

Project	Time	Operation Agency	Project Goals
Skylab	1973	NASA	Demonstration of on-orbit repair Provide a replacement thermal shield of Skylab Deploy the failed solar arrays
SMMR	1980	NASA	Orbit corrections Make the standard spacecraft parts modular Repair and/or replace in space Ground integration and test
Palapa B2 & Westar 6	1984	NASA	Retrieved the two errant spacecrafts through EVA Take the satellites back to Earth for refurbishment
HST	1990	NASA	Replace Orbital Replacement Instruments (ORIs) Exchange Orbital Replacement Units (ORUs)
ISS	1998	NASA, CSA, JAXA, etc.	Construction Installation Extension



Skylab SMMR Palapa B2 & Westar 6 HST ISS

Figure 1.2.: OOS manned projects

In contrast to manned OOS missions, unmanned OOS mission for malfunctioning or obsolete spacecraft, which reduces the risk of astronauts and has enormous economic value, has gained significant attention. A list of the historical and recent examples of technology demonstration activities are shown in Table 1.3 and Figure 1.3.

Table 1.3.: OOS unmanned projects

Project	Time	Operation Agency	Project Goals
ETS-VII	1997	NASDA	Rendezvous and docking, multi-body dynamics Tele-operation latency, ORU exchange and assembly of space structure
XSS-10	2003	AFRL	Demonstrate technology for line-of-sight guidance of spacecraft
XSS-11	2005	AFRL	Demonstrate technology for autonomous rendezvous and proximity maneuvers Tracking and navigation
DART	2005	NASA	Autonomous rendezvous Proximity operations
OE	2007	DARPA	Autonomous docking ORU exchange
ATV	2008	ESA	Autonomous rendezvous and docking ISS attitute reboot Deliver food, air and water to ISS
HTV	2009	JAXA	Rendezvous and berthing Deliver food, air and water to ISS
PRISMA	2010	SSC, DLR, CNES	Rendezvous and proximity operations Sustained formation flying
Ranger TFX	On-going	NASA, SSL	Rendezvous and docking Free flying telerobotic servicers for space operations
OLEV	On-going	DLR	Prolong the life of tele-communication satellites Rescue the spacecraft in a wrong orbit
FREND	On-going	DARPA, NRL	Test stereo photogrammetric imaging Debris removal
DEOS	On-going	DLR	Far range formation flying Autonomous rendezvous Capture and berthing with target
CESSORS	On-going	SSTC	On-orbit maneuvering Repair or retrieve malfuction satellites

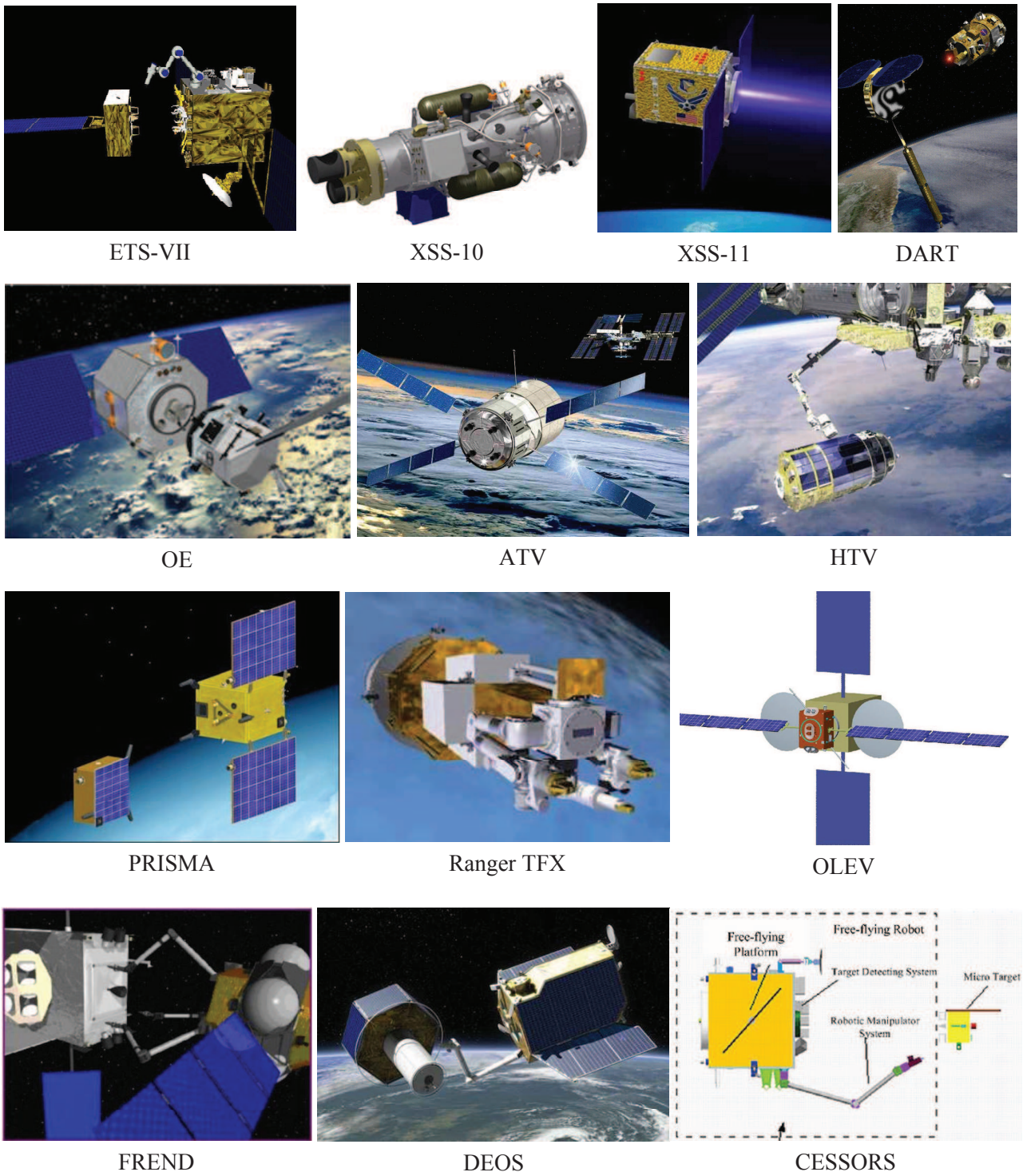


Figure 1.3.: OOS unmanned projects

1.2.2. Space Robotics Demonstrators

It has already been decades since the first robotic manipulator arm – space shuttle remote manipulator system applied in the STS-2 mission in 1981. Here we will give a brief overview of the past and undergoing space robotic demonstrators.

(1993) Robot Technology Experiment (ROTEX) ROTEX (Hirzinger et al. (1994)) was a starting shot for Germanys participation in space automation and robotics which contained as much sensor-based on-board autonomy as possible. The space experiment was led by DLR and ROTEX is a small, 6-axis robot mounted inside a space-lab rack, its main objective is verification of the man-machine interface as well as the joint control and the sensor based hand controllers of the robotic application under micro gravity. During the periods of the experiment, 4 operational modes were verified, which were automatic, tele-operation on-board, tele-operation from ground using predictive computer graphics and tele-sensor-programming. ROTEX showed that complex multi-sensory space robot system can be successfully operated in a variety of different modes, which also proved that a human operator and a robotic application can be accomplished under different levels of robot autonomy.

(1993–2005) Ranger Telerobotic Flight Experiment (TFX) Ranger (Bon and Seraji (1996, 1997))is a tele-operated space robot developed at the University of Marylands SSL. Ranger consists of two 7 DOF manipulators with interchangeable end-effectors to perform such tasks as change out of ORU in orbit. Its main objective was to design a servicing vehicle capable of flying on a Pegasus launch vehicle and then constructing the neutral buoyancy equivalent. Ranger has been redesigned for a Space Shuttle experiment, but until now has not yet been manifested on a flight.

(1997) Engineering Test Satellite-VII (ETS-VII) ETS-VII (Inaba and Oda (2000), Oda et al. (1996)) was another milestone in the development of space robotics technology, particularly in the area of OOS. ETS-VII was an unmanned spacecraft developed and launched by JAXA in November 1997. Its main objective was to test free-flying robotics technology and to demonstrate its utility in unmanned orbital operation and servicing tasks. To obtain a global coverage of communication in LEO operations, the signals were relayed by GEO communication satellites, which induced a larger delay in the case of ETS-VII mission. ETS-VII validated the concepts and theories for free-flying space robots.

(2001) Canadarm2 and Dextre Canadarm (Gibbs and Sachdev (2002)), designed by CSA for space applications, also called SRMS which stands for shuttle remote manipulator system, is a mechanical arm that maneuvers a payload from the payload bay of the Space Shuttle orbiter to its deployment position and then releases it. It can also grapple a free-flying payload and berth it to the payload bay of the orbiter. SRMS was first used on STS-2 mission in 1981. Since then it has been used more than 100 times during Space Shuttle flight missions. Canadarm2 along with Dextre, the SPDM, is the next generation of the SRMS, for use on the ISS. It was launched in 2001 during STS-100, both mounted on the MSS, a module of the ISS. They are mainly used in station assembly and maintenance: moving equipments and supplies around the station, support astronauts working in space, and servicing instruments and other payloads attached to the space station.

(2005–2008) RObotic Components Verification on the ISS (ROKVISS) ROKVISS is a German space technology experiment (Reintsema et al. (2007)) lead by DLR, which was launched by an unmanned Russian Progress transport vehicle in 2004 and installed on the outer platform of the Russian segment of ISS in early 2005. The goal of ROKVISS is the verification of mechatronic light-weight robot joint unit for use in the OOS. In addition, a haptic-visual tele-presence operation on the basis of a direct radio link between the space station and the ground station has been tested and finally established.

(2007) Orbital Express (OE) The OE (Ogilvie et al. (2008)) space project was led by DARPA, which was developed to verify technical feasibility of robotic on-orbit refuelling and re-

configuration of satellites, as well as autonomous rendezvous, docking, and manipulator berthing. It was launched in March 2007, after that various mission scenarios have been conducted. The system consists of the ASTRO vehicle, and a prototype modular of next-generation serviceable satellite, named NextSat. The ASTRO vehicle is equipped with a robotic arm to perform satellite capture and ORU exchange operations. All the mission scenarios were successfully completed by July 2007.

(2013) Robonaut2 Robonaut (Peters et al. (2003)) is a dexterous humanoid robot build and designed at NASA Johnson Space Center in Houston, Texas. It is the latest generation of the astronaut helpers, launched to the ISS aboard space shuttle Discovery on the STS-133 mission. It is the first humanoid robot in space. Its main goal is teaching engineers how dexterous robots behave in space. Eventually, a variant of Robonaut 2 may be used as an astronaut aid on the outside of ISS, which is capable of handling a wide range of EVA tools and interfaces. Now, Robonaut2 has successfully moved for the first time in space on October 13th, 2011.

(2014) DEutsche Orbitale Servicing Mission (DEOS) Due to a programmatic reorientation of the former TECSAS (Technology Satellite for the demonstration and verification of Space systems) project, it is now known as DEOS (Rupp et al. (2009)), which also leads by DLR, served as a test bed for OOS technologies, such as key robotics hard- and software elements for advanced space maintenance and servicing. The mission will demonstrate various OOS scenarios such as rendezvous, docking, formation flight, capture, stabilization and controlled de-orbiting of the target and servicing compound.

(2014) Front-end Robotics Enabling Near-Term Demonstration (FREND) As an successor of SUMO project (Spacecraft for the Universal Modification of Orbits), FREND (Bosse et al. (2004)) was created under DARPA sponsorship to prove the capability of autonomously executing an unaided grapple of a spacecraft which was never designed to be serviced. The capability allows nearly any satellite on-orbit to be repositioned and provides additional benefits such as satellite life extension, refuelling, ORUs etc. Now the project is focusing on the ground demonstration and verification for future performing on-orbit.

(2014) Chinese Experimental Space System for On-Orbit Robotistic Services (CESSORS) CESSORS (Liang et al. (2006)) is developed by SSTC inspired by the success of previous space robotic projects. The aim of this project is to fabricate a small satellite mounted with robotics system to complete orbit maneuvering and implement unmanned robotic servicing tasks, such as repair, retrieve malfunctioning satellites. Until now, CESSORS is still under constructing.

ROTEX ROKVISS Canadarm2 with DEXTRE Robonaut2

Figure 1.4.: Space roboitcs projects

1.2.3. Technical Challenges of Space Robotics

A possible classification of the OOS demonstrators depicted in Section 1.2 can be found in Figure 1.5. The classification is based on two criteria. One criterion for classification is the autonomy extent of the OOS demonstrators. The other criterion is the options for communication of OOS demonstrators in respective orbit. Here the direct and relay to LEO stands for a direct or relay communication link for OOS demonstrators, while direct to GEO means a direct communication link with the satellites in the GEO orbit.

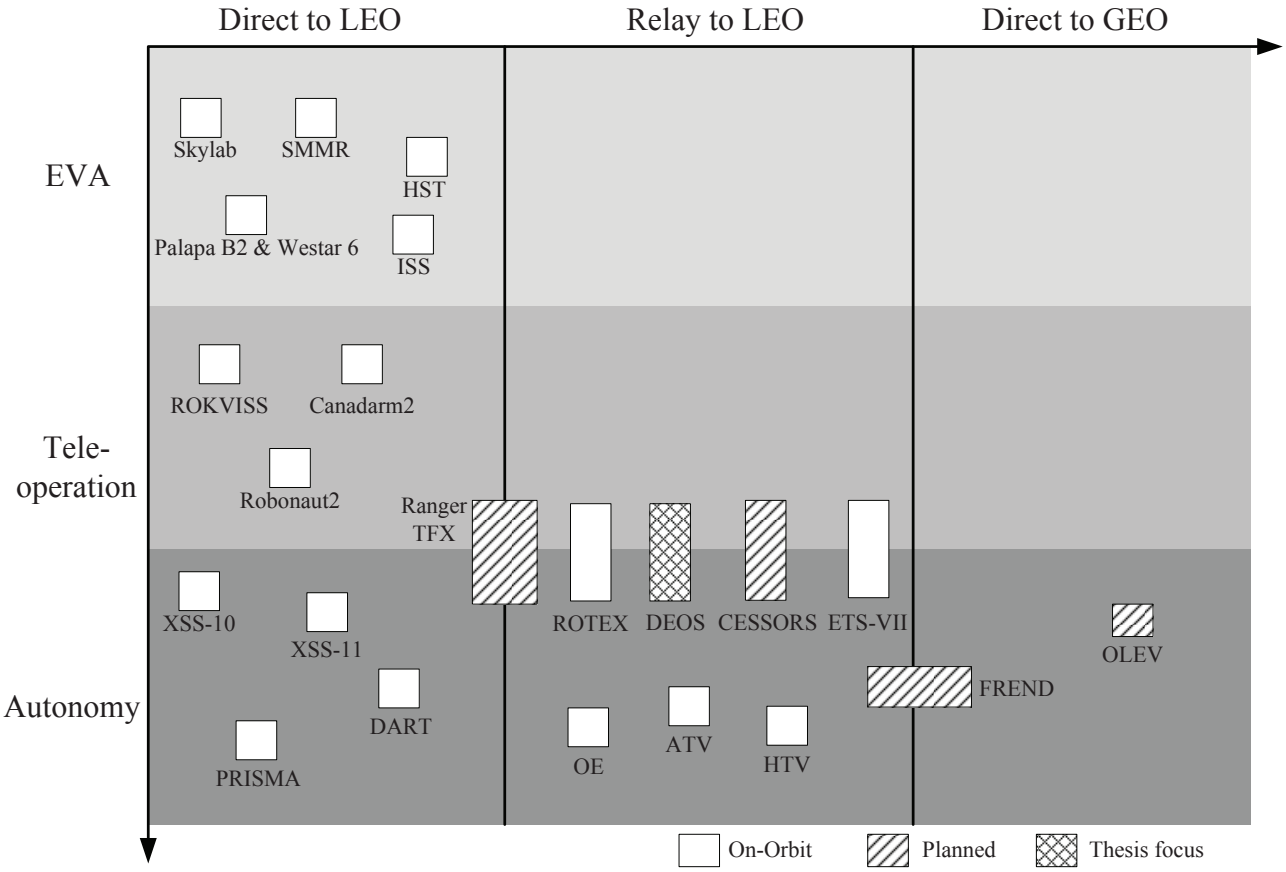


Figure 1.5.: OOS demonstrators classification

Undoubtedly, robotics, tele-robotics and autonomous systems will have a considerable potential for a wide spectrum of applications in providing a space-based infrastructure. However, in order to complete the space missions, we have to face the fact that there are still some technical challenges in the field of space robotics which will show us the possible research directions and stimulate our endeavours to figure them out. Here we list some of primary challenges in capturing a tumbling target with space robot.

- **Rendezvous, proximity operations and docking to a target satellite in extreme conditions**

  Rendezvous includes flyby of target satellite and formation keeping. Proximity operations require zero relative velocity at working position. Docking drives latching mechanisms and electrical/fluid interface into a mated condition. Major challenges

include completing above missions optimally in all range of lighting conditions, without collision and violating the actuators' work limits.

- **Object recognition and state estimation under space environment**

  To search and find an un-cooperative target without communication link in space is an intractable task. After the target is detected, since there is no prior knowledge about target, its geometry must be reconstructed with sensory data to estimate the states of the target and determine the grasping point and capturing strategies.

- **Full immersion tele-presence with haptic and multi modal sensor feedback**

  Tele-presence will provide the operator a physical sense of working at the place of space robot. It must be as real as the robot working site which need multiple modal sensor feedback, such as fully immersion displays, sound, touch, etc.

- **Supervised autonomy with time-delay**

  Complete autonomy of robotics still relies on the advancement of AI. So a supervised autonomy or the tele-operation would be a realistic choice. But the communication time-delay in the control loop will degrade the ability to tele-operation. Challenges include the run-time states prediction, visualization prediction and ability to work ahead of real-time.

- **Understanding the distinctive characteristics of space robotic dynamics**

  Space robotics with a floating base possesses some particular properties compared to fixed base robot. The interaction between space manipulator and it base (spacecraft) will induce some difficulties in controller design and path planning.

- **Optimized motion planning and control with various constraints during capturing phase**

  All real systems have some constraints, such as input and output boundaries, possible collision, etc. During capturing, it would be of great benefit if the system can handle all these constraints optimally while achieving some optimization index. Major challenges include developing a general framework to realize on-line path planning, constraints handling, and optimization simultaneously.

- **Verification and validation of autonomous system on-board**

  The software of space robots running on-orbit autonomously is a big software engineering project. It includes different subsystems from various programmers. Exhaustive and deep exploration of the codes has to be done before launch. Verification and validation techniques are required to more fully assure the feasibility and reliability in all conditions.

1.3. Hypothesis and Problem Statements

As illustrated in section 1.2.3, in order to complete capturing an un-cooperative target satellite using space robot, the major objective of this work is how to control the space robot to execute the required missions in complex space environment. Traditional control schemes can be classified by different objectives:

Resolved Motion Rate Controller was first proposed for a free-floating space manipulator in Umetani and Yoshida (1989) using the GJM. Later, similar controllers designed for kinematically non-redundant and redundant manipulators can be found in Masutani et al. (1989), Nenchev (1993), Papadopoulos and Moosavian (1994), Rekleitis et al. (2007), Xu and Kanade (1993).

Adaptive Controller is another widely developed strategy since it can be used to overcome the uncertainties and parameters variation issues. However, since the high non-linearity of the dynamics of space robot, the design of the adaptive control law is not easy. The application of adaptive control can be found in Abiko and Hirzinger (2007), Gu and Xu (1993), Wang and Xie (2009), Xu et al. (1992), Xu (1991).

Robust Controller provides another feasible controller solution for space robot to handle the parameters variation and unknown dynamics effects. The application of robust control for single or dual-arm can be found in Huang et al. (2007), Pathak et al. (2008), Tang et al. (2011), Xu et al. (1995).

Among aforementioned controllers, the possible collision and singularity issues during motion of space robot are out of consideration, moreover, input/output constraints of the system are also difficult to incorporate into the control schemes. Accordingly, the hypothesis of this work is:

The conventional control framework based on traditional control strategies are not adequate to solve the motion control issue of space manipulator in capturing another tumbling target when obstacle and singularity issues are taken into account.

Originated from chemical processing industries, Model Predictive Control (MPC), also referred to Receding Horizon Control (RHC), has gradually expanded its application to the field of aerospace such as spacecraft formation keeping Manikonda et al. (1999), spacecraft trajectory planning Richards et al. (2002), rendezvous and docking with a tumbling target Park et al. (2011), etc. As an effective control strategy, MPC has the capacity to handle the constraints and perform on-line optimization. Nevertheless, the application of MPC requires an explicit system model for states prediction. Similarly, if collision and singularity issues are considered in the design of MPC, they must be handled in advance and integrated into the framework of controller. Therefore, the problems for this work can be listed as follows:

- The dynamics equations of the space robotic system must be derived to represent the relationship between the states acceleration response and the given forces and torques;
- Computational efficient collision detection algorithm which can deal with multiple convex and non-convex obstacles has to be completed when space robot works in a complex environment;
- Singularity issue during the motion of space robot must be taken into account to impede the joint from generating enormous velocity;

- With the derived dynamics equations of space robotics system, MPC strategy can be implemented for the space robot to perform the capturing mission.

1.4. Scope of Work

1.4.1. Research Scope within Space Robotics

With the advancement of science and technology, especially great progress in the field of computer science, the extent of autonomy in OOS is increasing, while human operator, as an indispensable role, will play as a monitor or supervisor role without involving direct operation too much. In light of space robots currently planned by world wide space agencies, an increase in the number and the capacity of robots applied in space missions will be a foregone conclusion in the coming years. In summary, the concept of tele-operated and full autonomy of space robotics, as main concern in this thesis, seems advantageous for OOS operation, since it alleviates the operators' workload and avoids the human involving in the direct operations. Accordingly, refer to section 1.3, we can break our main goal down to its particulars into the following aspects:

- Development and design of a simulation and test environment, representative for space robotics;
- Evaluation the influence of geometry to the dynamics of space robotics;
- Provision of specific algorithms for obstacle and singularity avoidance in inverse kinematics control;
- Construction of a general control framework based on MPC with various constraints for autonomous space robotics;
- Assessment of the feasibility of the proposed general control framework.

1.4.2. Thesis Roadmap

The overview of the thesis structure is presented in Figure 1.6. An introductory part about the research motivation, state of the art, and research scope within space robotics are included in Chapter 1.

Chapter 2 presents a new distributed real-time simulation architecture based on DDS for space robotic autonomous and tele-operation tasks. The space mission profile and background of space robotic tele-operation are firstly recalled. Within this context, a closer look into RACOON system overall design and simulation environment are described. The detailed characters of DDS are subsequently exhibited. Next, RacoonSim system and its related subsystems are introduced. Additionally, a user-friendly VR user interface for coexistence of working operator and space robot is developed.

Chapter 3 focuses on a modelling scheme that uses the concepts of graph theory and spatial notation for calculating the joint-space dynamics of general tree structure space manipulator systems. This chapter is relevant to the development of appropriate trajectory planning and control algorithms in the following chapters.

Chapter 4 concerns the inverse kinematics issue in velocity level. The task-priority based inverse kinematic solution is derived from the concept of the null-space, some dexterity measurements are also proposed for the future analysis. Moreover, obstacle and singularity detection and avoidance are also dealt in this chapter.

Chapter 5 investigates the use of NMPC for the motion control of a space manipulator to approach an un-cooperative target satellite in space.

Chapter 6 closes the treatment of the group verification of autonomous space robotics with concluding remarks and future directions for continuing research.

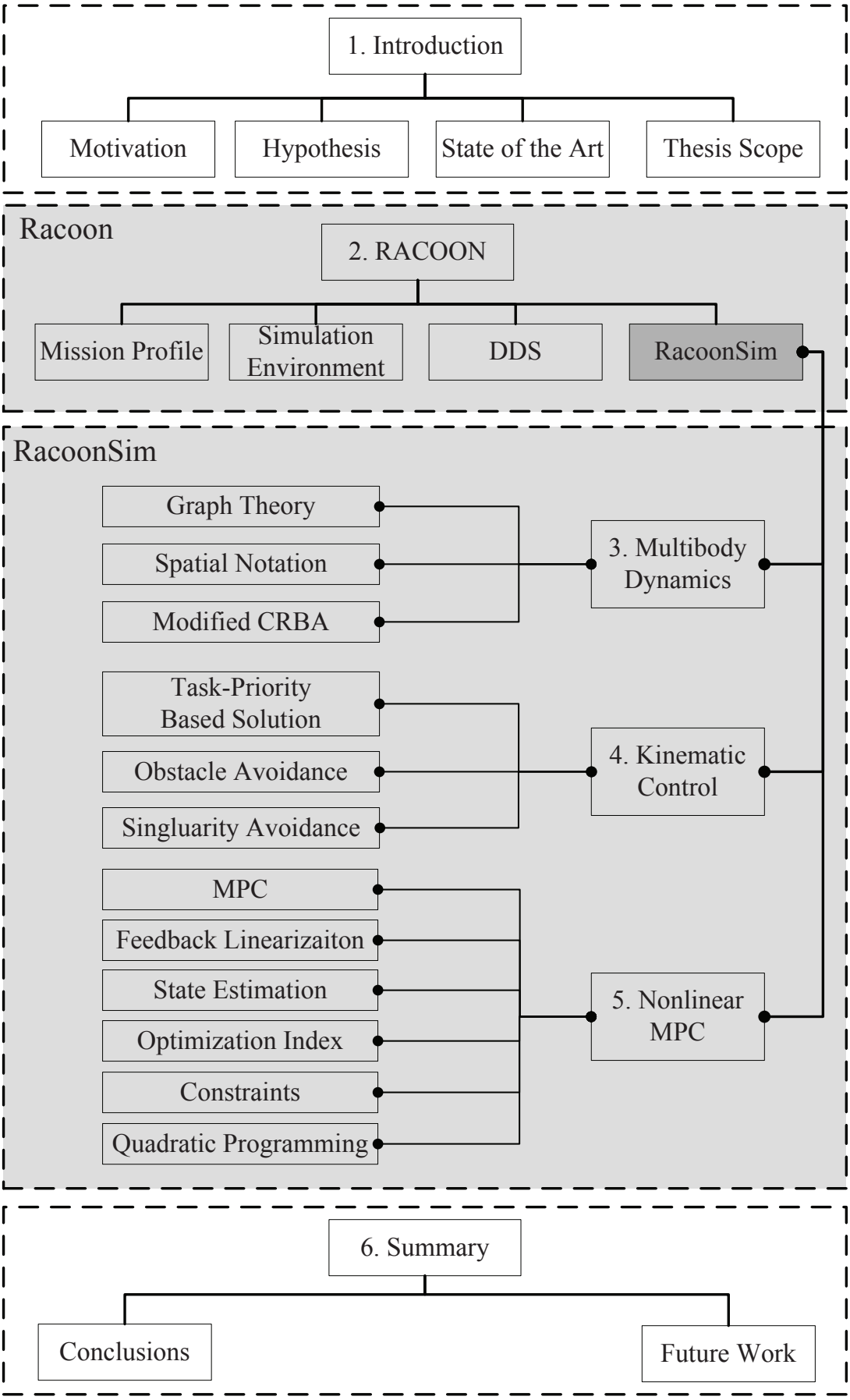


Figure 1.6.: Structure of the work

2. Simulation System Design

I do not fear computers. I fear the lack of them.

—Isaac Asimov

The increasing demands of satellite maintenance, on-orbit assembly and space debris removal etc. call for applications of space robot to perform tasks in the particular harsh space environment. However, it is still impossible to develop a fully autonomous space robot with the present robotic technology until now. For this reason, an operator tele-operating space robot from ground station becomes an option. To lengthen in duration and enlarge the scope of tele-operation, a Geostationary Orbit (GEO) relay satellite is employed to address the communication issue between ground station and servicer satellite. This chapter presents a new distributed real-time simulation architecture based on Data Distribution Service (DDS) for space robotic tele-operation tasks. The objective is to make the simulation architecture open for collaborative tele-operation research and provide the operator an intuitive view of space robotic tele-operation in a wide set of scenarios. The mission profile and background of space robotic tele-operation are firstly recalled. Within this context, a closer look into Robotic Actuation, Control and On Orbit Navigation Laboratory (RACOON) system overall design and simulation environment are described. Secondly, the detailed characters of DDS, including DDS specification and its core idea of data distribution, are exhibited. Thirdly, Racoon Simulation (RacoonSim) system, which comprises multi-body dynamics, Autonomous Mission Management (AMM), path & trajectory planning and control subsystems of space manipulator, is introduced. Additionally, a user-friendly Virtual Reality (VR) user interface for coexistence of working operator and space robot is developed, which is composed of 3 dimensional space mouse, joystick and Head-Up Display (HUD) as part of the Mission Control Center (MCC). Well-designed simulation system architecture makes the Hardware-in-loop (HIL) verification possible and can be extended easily in the future.

2.1. Mission Profile

A typical OOS mission is composed of a series of space missions. Before a specific OOS operation is executed, some operations about spacecraft must be conducted to decrease the relative distance between spacecraft and target satellite. During this period, lots of demonstration missions can also be performed such as flyby, formation flying to test the feasibility of new sensors and technologies. After the servicer satellite moves into the work scope of the space manipulator, space robot will take charge of the OOS mission to perform the subsequent mission flow. A typical OOS mission before capturing can be described as follows:

1. **Attitude control** The servicer satellite together with space manipulator is separated from the upper stage of the rocket and released into space. At the beginning of its orbital life, an attitude reconstruction procedure has to be performed to stabilize the servicer satellite. Attitude determination equipment such as infrared horizon sensor, sun sensor or gyroscope can be employed to assist the attitude reconstruction of the servicer satellite.

2. **Drift orbit** After the success of attitude control, the servicer satellite moves into the drift orbit phase. In this phase, orbit phasing will be conducted to regulate the orbit of servicer into the same orbit as the target satellite or a slightly lower orbit. The attitude of servicer satellite will be maintained, absolute navigation equipment like star trackers, IMU and GPS receivers will be used during this phase.

3. **Rendezvous** At the end of the drift orbit phase, the servicer satellite is at a position within a distance of 5 km to 300 m of the target. The phase will utilize the absolute and relative navigation equipment like GPS, IMU, radar or lidar to guide the servicer satellite into a range of 300 m for the future space missions.

4. **Proximity** When the servicer satellite moves within 300 m of the target satellite, a proximity procedure starts to drive the servicer to the target from 300 m to several meters. Besides the relative distance, the relative attitude and the relative translational and angular velocities between the two satellites have to be decreased for the coming docking mission. In this phase, the visual sensors can be added in for target tracking and monitoring. The MCC will observe the whole rendezvous and proximity phases. When the relative sensors lose the target or the service moves into an unsafe region, an emergence action has to be taken autonomously or commanded by the operators.

5. **Station keeping** Before the action of the space manipulator, a phase named station keeping has to be performed to guarantee the safety of the coming robotic operations. During this phase, the target satellite will be in the workspace of space manipulator. A control loop must be closed on board to keep the relative position and orientation out of collision.

Once the aforementioned tasks are completed, the servicer satellite is now in the neighbourhood of the target satellite and ready for the further OOS robotic operation. In order to perform the space missions using space robot, such as on-orbit assembly or capturing a tumbling target, the motions of space robot can be divided into 3 phases as shown in Figure 2.1: an approach phase, a tracking phase with grasping and a stabilization phase, which will be illustrated in the following.

1. **Approach** The approach phase starts from an observation distance and brings the manipulator mounted on the servicer to an optimal grasping pose. After that, a tracking phase will be proceeded. The approach phase moves space robot end-effector from an initial position to a final position under certain constraints. The final position of the approach phase will be the initial position of the tracking phase.

2. **Tracking** When space robot needs to capture or observe a certain point of a tumbling target, a tracking mission will be required. This tracking phase aims to minimize the residual relative velocity between the target interested point and the space robot end-effector. The duration of the tracking phase depends on the tracking controller performance, such that the end-effecotor can have sufficient time to regulate its pose and ensure the success of tracking the target interested point. At the end of tracking phase,

neither the end-effector nor the robot joints are at rest. Their states will be the initial value of the next phase.

3. **Stabilization** Once the relative pose between space robot end-effector and target interested point reduces to zero and the gripper on the end-effector is closed, in order to retain the stable attitude of the target, a stabilization phase must be carried out to decrease the angular velocity of the target to zero and be ready for further space operations. In the stabilization phase, by exerting external forces and torques or adjusting the internal torques of robot joints simultaneously, the relative motion between the servicer and the target will be brought to zero.

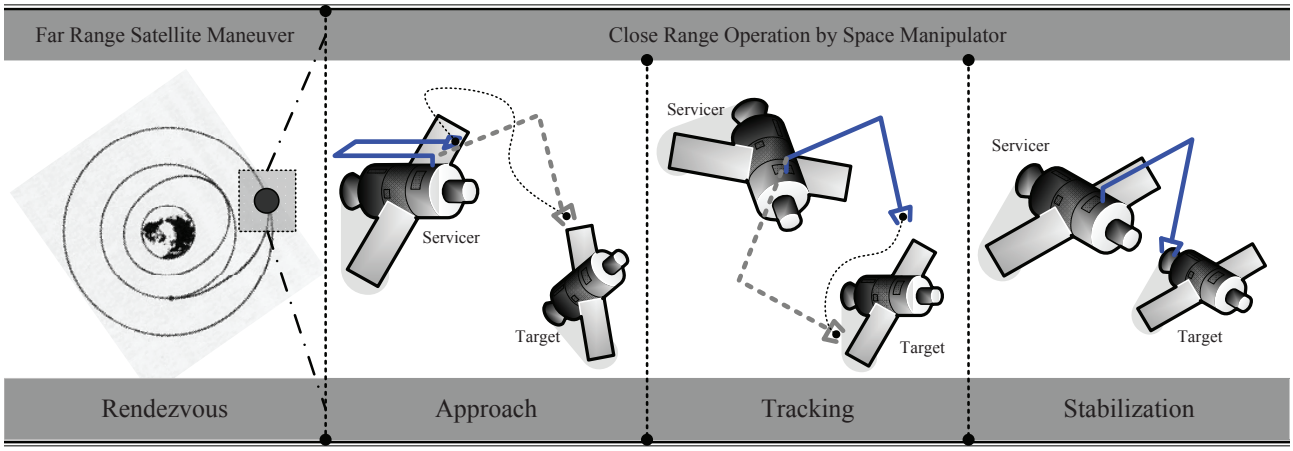


Figure 2.1.: Mission profile of space robot

After the space manipulator completing the capture, further space operations can be carried out such as inspection, de-orbit or ORU exchange, etc. In fact, besides above general uses of space robot, there are still some other potential applications of space robot. For example, as a result of dynamic coupling between spacecraft and space manipulator, it can be utilized to regulate spacecraft attitude for decreasing on-board fuel consumption. According to the external force and torque exerted on the spacecraft, the servicer satellite will be operated in three flying modes. If there are no forces and torques exert on the servicer, it is termed free-floating mode. When only torque is applied to the servicer, it is called free-flying mode. Another mode is auto-flying mode. Before-mentioned three kinds of space phases can be performed under these three flying modes. The flying modes switch can be controlled by the operator in terms of the particular requirements.

2.2. Simulation Overall Design

2.2.1. Racoon Design

The RACOON system is designed to assist the operator with intuitive view and validate the feasibility and the performance of the on-going space robot project. Hence, the following subsystems will be included in the RACOON system. A schematic diagram of RACOON system is shown in Figure 2.2.

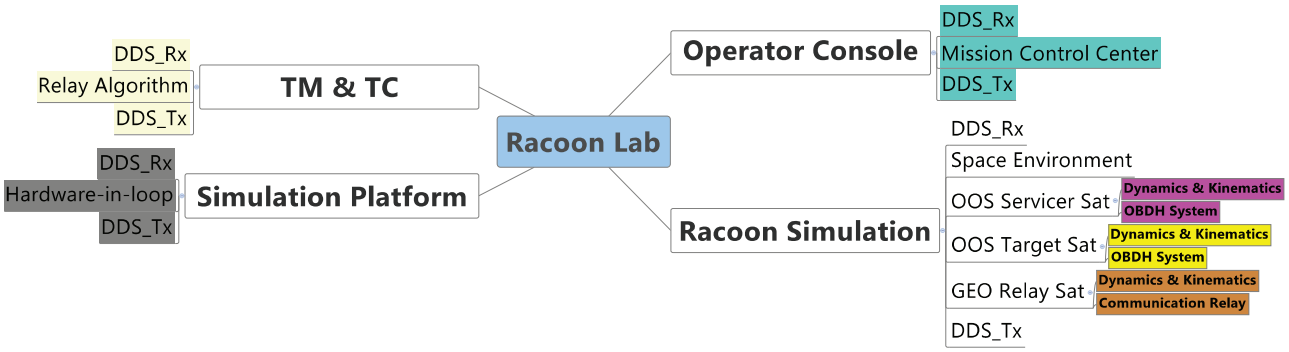


Figure 2.2.: Subsystems of Racoon Lab

Operator Console: It is an alias of MCC. The operator will send a series of commands from MCC uplink to the servicer satellite via the GEO relay satellite. The communication relay will be simulated by adding an individual node in the simulation system to relay the information. MCC will also receive the telemetry information collected from the states of space robot & target satellite and display them on a big screen to help the operator enhance the awareness and understand of the real-time situation in space. The operator can also control the space robot in real-time by joystick or 3 dimensional space mouse. A VR is also running when the operation is proceeding. These will be introduced in detail in the Section 2.3.5.

TM/TC: This subsystem is responsible for the signal relay between MCC and GEO relay satellite. It transmits the tele-commands from MCC to the GEO relay satellite and the telemetry information from servicer to the MCC. The information dissemination between various nodes is based on DDS.

RacoonSim: This system contains two subsystems, one is dynamics of servicer and target, and the other is OBDH subsystem. For detailed description of RacoonSim system design will be presented in Section 2.3.

Hardware-in-loop (HIL): This system can include different types of hardware, such as sensors, actuators, etc. Since the expandability and scalability of the simulation system, the proposed simulation architecture is easy to replace the software by particular hardware.

2.2.2. Simulation Environment

One of the first and major implementation solutions adopted was to build RACOON on top of Matlab/Simulink environment (see Figure 2.3). This environment integrates sufficient properties to cope with the aforementioned requirements, assuring high flexibility and availability of existed functionality.

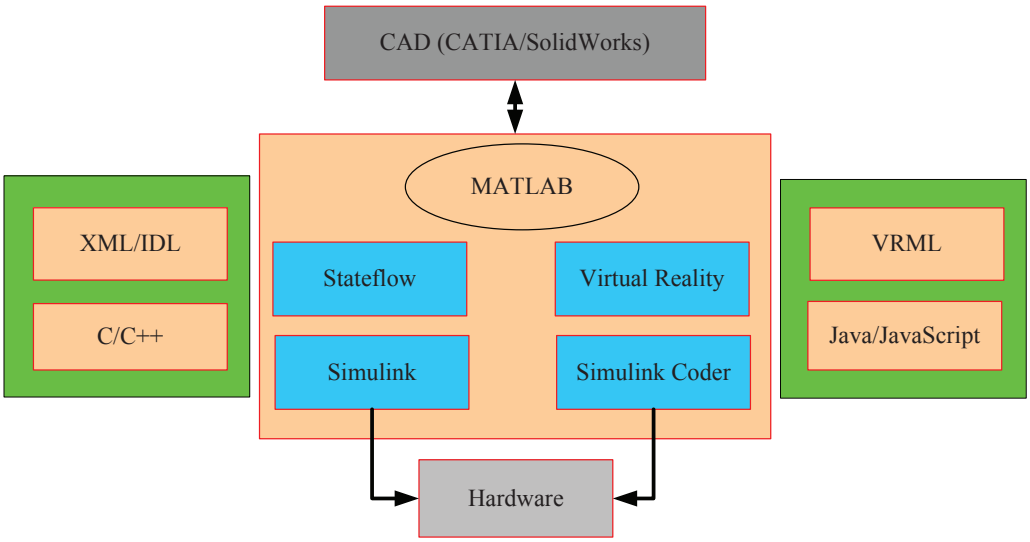


Figure 2.3.: Simulation environment of Racoon

Normally, modern engineering design starts from the application of CAD system, such as CATIA®or SolidWorks®. CAD can assist the designer in the creation, modification, analysis, or optimization of a design. On the one hand, it provides the designer an efficient and intuitive tool to improve the quality of the design and increase the productivity. On the other hand, the objects designed by CAD will be as a basic input of VR. In this paper, VRML is chosen to represent 3 dimensional interactive vector graphics. In order to increase the interactivity and immersion of the operator, Java/JavaScript is integrated into VRML to deal with the interactive event response issue. The transformation between CAD and VRML can be implemented by using different software.

XML is a mark-up language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. Its main design goals are simplicity, generality and usability over the Internet. In our design, XML is utilized to describe the configurations and initial states of the RACOON system. Another application for XML is DDS QoS configuration.

As a major programming language, C/C++ programming language can establish connections among various applications. That's why it is also employed here to deal with the different interfaces of relevant applications involved in RACOON. Before simulation starts, it proceeds the XML file to extract the configuration and initial states of RACOON simulation system. Furthermore, it is employed to process the IDL file which is subsequently involved in DDS.

At the core of RACOON simulation system is Matlab/Simulink/Stateflow/Simulink Coder, where Simulink is a data flow graphical programming environment for modeling and simulating multi-domain dynamic systems. A number of hardware and software products are available for use with Simulink. Stateflow, integrated in Simulink, provides a control logic tool to model reactive systems via state machines and flow charts. Coupled with Simulink Coder, Simulink model can automatically generate C/C++ source code for distributed real-time simulation.

2.2.3. Data Distribution Service

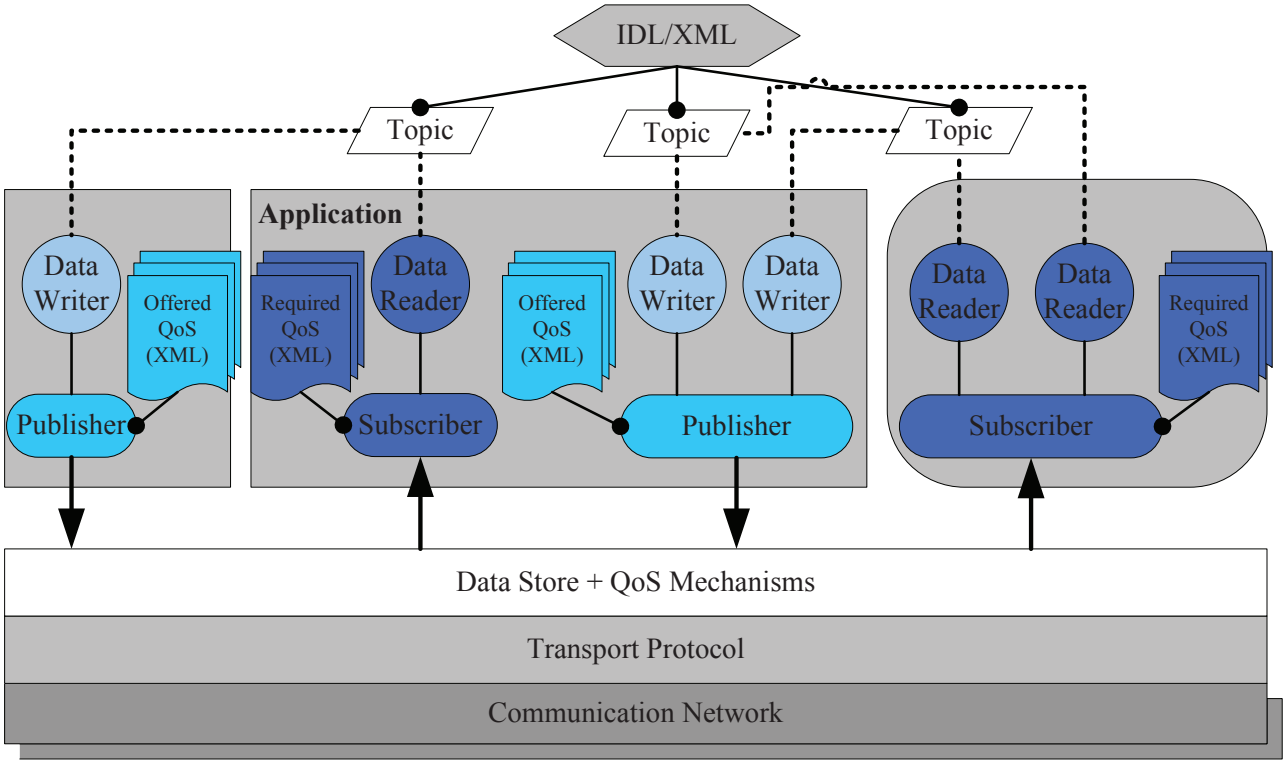


Figure 2.4.: DDS Structure

The DDS OMG (2007) for real-time system is an OMG Publish/Subscribe (P/S) standard that concentrates on scalability, real-time, dependability, high performance and interoperability of data exchange between publishers and subscribers. It provides the communications service programmers to distribute time-critical data between embedded and/or enterprise devices or nodes. As the DDS specification describes, there are two levels of DDS interface. One is the optional higher DLRL level, which can integrate DDS into the application layer simply. The other is the lower DCPS OMG (2010) level. As the core of DDS, DCPS is aimed to deliver the proper information from one node to the proper recipients efficiently. The underlying DCPS propagates data samples delivered by publishers into a global data space, where it is disseminated to interested subscribers. The DCPS model decouples the declaration of information access intent from the information access itself, therefore enabling the DDS middleware to support and optimize QoS enabled communication. As illustrated in Figure 2.4, a canonical DCPS module is comprised of the following entities that supply functionalities for a DDS application to publish/subscribe to data samples of interest.

Topic: Shared knowledge of the data types is a requirement for different application to communicate with DCPS. Data (of any data type) is uniquely distinguished using a name call a Topic. Topics interconnect DataWriters and DataReaders. The data samples associated with the Topic will be passed from DataWriter to the corresponding DataReader.

Domain and DomainParticipants: In order to isolate the affection of various applications, DCPS defines a concept termed Domain to address this issue. Domain represents logical isolated, communication network. Multiple applications running on the same sets

of hosts on different Domains are totally isolated from each other. Applications that want to exchange data must belong to the same Domain. To belong to a Domain, DCPS APIs are used to configure and create a DomainParticipant with a specific Domain Index. Domains are differentiated by the Domain Index. Applications that have created DomainParticapants with the same Domain Index belong to the same Domain. DomainParticipant includes Topics, Publishers and Subscribers. Therefore all the DCPS entities belong to a Domain.

Publisher and DataWriter: The data sending side used entities call publishers and DataWriters. A DataWriter is the actual object used to send data samples. It is associated with a single topic. A publisher is the DCPS object responsible for the actual sending data. Publisher own and manage a group of DataWriters with similar behaviour or QoS policies. When data is assigned to send to an application, it first calls the method on a DataWriter, the data sample is then passed to the publisher object which does actually dissemination of data on the network.

Subscriber and DataReader: The data receiving side uses objects call subscribers and DataReaders. A DataReader is the actual object used to receive the published data samples. It is associated with a single topic. A subscriber is the DCPS object responsible for the actual receiving data. Subscribers own and manage a group of DataReaders with similar behaviour or QoS policies. When data is sent to an application, it is first processed by a subscriber, the data sample is then stored in the appropriate DataReader.

QoS Policies: For real-time applications, the following features, such as efficiency, determinism, flexible delivery bandwidth, thread awareness and fault-tolerant operations are often required. DCPS, and thus DDS, are designed and implemented specifically to address these requirements through the configuration parameters know as QoS policies defined by DCPS standard. QoS policies control many aspects of how and when data are distributed between applications. It is a significant feature of DDS for the users to configure the QoS policies for each DDS entity or communication node. Until now, over 50 QoS policies categories are supported by DDS. Therefore, users can employ various QoS policies to control numbers of behaviour of communication entities (DomainParticipant, Topic, Publisher/-Subscriber, DataWriter/DataReader) utilized in your applications.

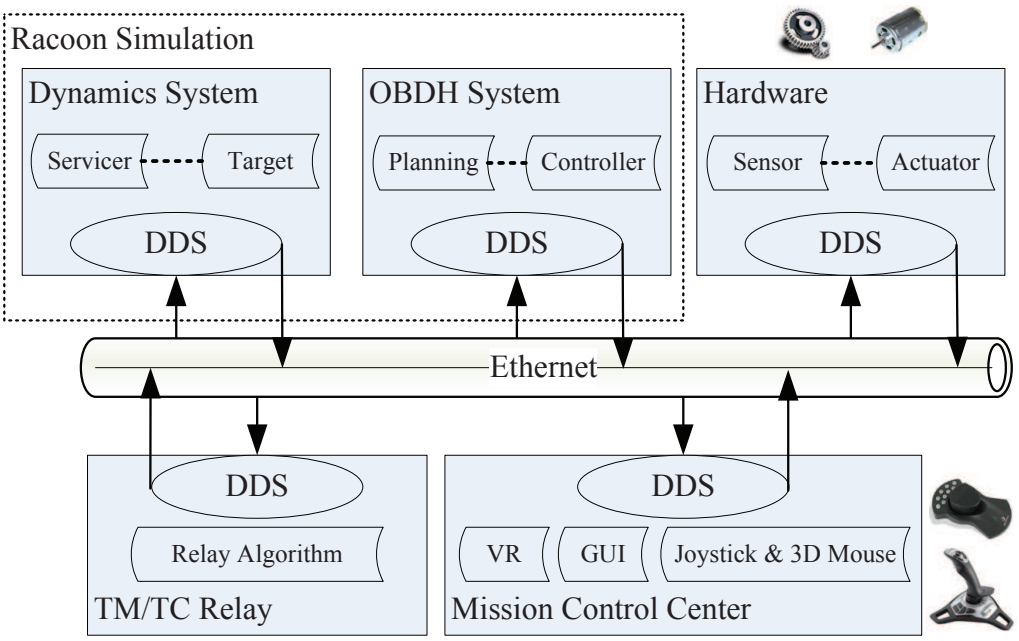


Figure 2.5.: Racoon system set-up

The entities and properties of DDS presented above bring important benefit to simulation applications, such as low latency, better communication determinism and runtime behaviour, enhanced QoS polices and wire interoperability, which makes the DDS middleware a necessary choice for building a distributed real-time simulation system. The publish-subscribe approach to distributed communications is a generic mechanism that can be employed by many different types of applications. Based on the DDS middleware used through Ethernet and the overall design in Section 2.2.1, we developed the space tele-operation simulator with 5 relevant subsystems running on the separate host computers. The two subsystems in dashed rectangle in Figure 2.5 are the core simulation part which is termed RacoonSim and will be illustrated in Section 2.3 in detail. MCC system acquires operator inputs and gives the force and scenery feedback to the operator. TM/TC relay system calculates the relative states among MCC, servicer and GEO relay satellite, and simulates the phenomenon of signal relay among them. Hardware system incorporates the instruments (sensors, actuators, etc.) and feedback to OBDH and Dynamic system, respectively.

2.3. RacoonSim Design

RacoonSim, the core of the RACOON laboratory, as shown in Figure 2.6, includes three primary parts: dynamics, OBDH, and human-machine interface. Dynamics algorithm is mainly responsible for simulation of 6 DOF target satellite and multiple DOF space manipulator. Human-machine interface provides a means for operator to interact with the RacoonSim system and intensifies the ability of perception and immension. OBDH system, which is the guarantee for completing the OOS mission, is composed of AMM, path & trajectory planning, and motion controller subsystems, which will be described in the following sections.

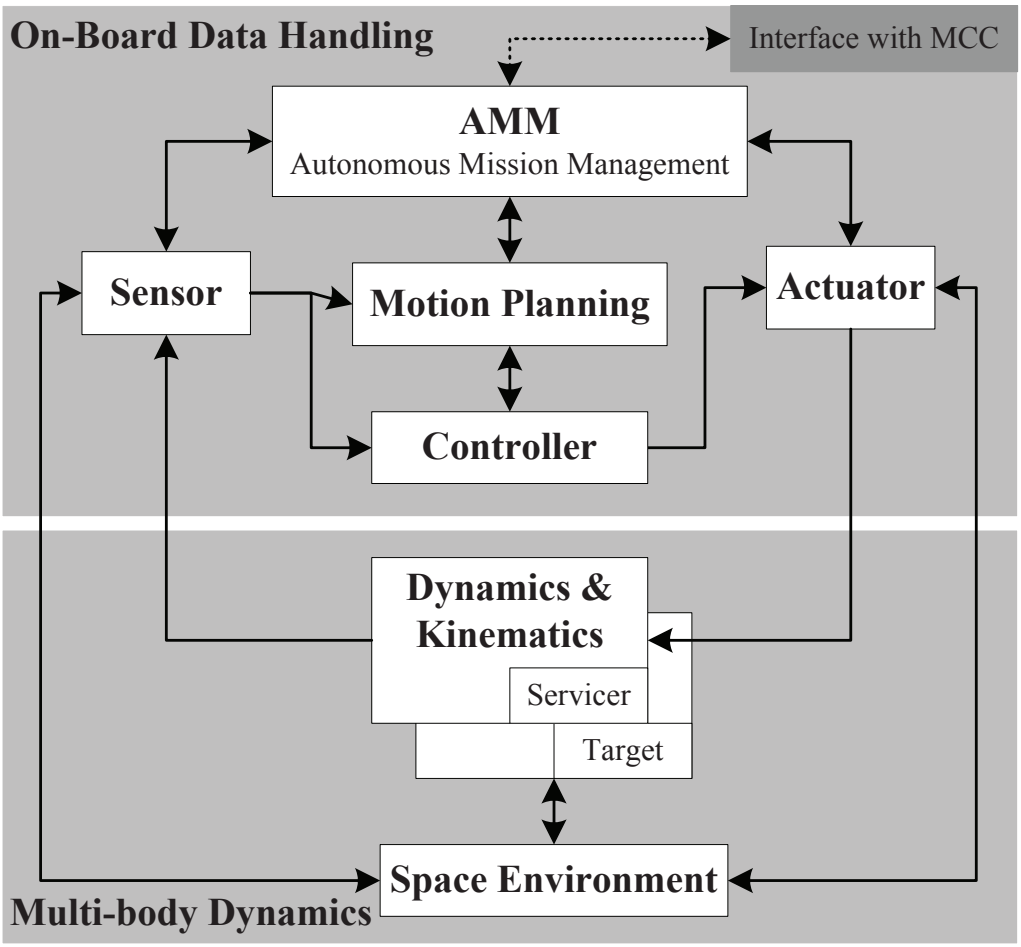


Figure 2.6.: Racoon simulation system set-up

2.3.1. Multi-body Dynamics

Multi-body dynamics modelling can be divided into two categories: operational-space dynamics and joint-space dynamics. Since the singularities in operational-space are unpredictable, especially when space robot involved in free-floating mode, thus joint-space dynamics modelling is adopted. As a matter of fact, there are two kinds of dynamics in terms of different applications: forward and inverse dynamics. In our simulation, forward and inverse dynamics are required to simulate the dynamics of RacoonSim system and to drive the virtual reality for human-machine interaction. The equation of motion can be described with the principle of Lagrangian mechanics, or the motion equations of D'Alembert, or Hamilton mechanics.

For a serial chain manipulator, its inverse dynamics can be obtained recursively with the computational cost of order n, $O(n)$ in Hollerbach (1980), Luh et al. (1980b), Siciliano (2009). On the contrary, the computational efficiency of forward dynamics depends on the notations, recursive layers, or the computer hardware architecture. Walker and Orin have proposed four methods in Walker and Orin (1982) expressed in normal notation for forward dynamics. The first of them is a $O(n^3)$ algorithm while the fourth is a $O(n^2)$ algorithm.

Balafoutis in Balafoutis and Patel (1989), Balafoutis et al. (1988) derived a more efficient algorithm by using orthogonal Cartesian tensors. With the concept of spatial notation, Lilly in Lilly and Orin (1991), Lilly and Bonaventura (1995), Featherstone in Featherstone and Orin (2000), Featherstone (2008) have developed a modified version of CRBA, which need less computing effort. With the aid of the decoupled natural orthogonal complement, Saha in Saha (1999) have obtained another set of motion equations based on Euler-Lagrange mechanics. In addition, Park, Bobrow and Sohl in Park et al. (1995), Sohl and Bobrow (2001) provided a recursive multi-body dynamics for branched kinematic chains using the technique and notation from the theory of Lie groups and Lie algebras. McMillan and Orin in McMillan and Orin (1995), McMillan et al. (1995) developed an ABA which produces a $O(n)$ forward dynamic algorithm without an explicit inverse of JSIM. On the other hand, Rodriguez and Jain in Rodriguez et al. (1991) applied Kalman Filtering and smoothing method to formulate the inverse and forward dynamics using spatial notation as Featherstone.

In the field of space robotics, Vafa and Dubowsky in Vafa and Dubowsky (1990) proposed a virtual manipulator technique to simplify the dynamics of space robotic system. Dubowsky and Papadopoulos in Dubowsky and Papadopoulos (1993), Papadopoulos and Moosavian in Papadopoulos and Moosavian (1994) developed a decoupled dynamics algorithm using a minimum set of body-fixed barycentric vectors. In Xu and Kanade (1993), Xu and Shum (1991) Xu derived the dynamics of planar manipulator with a free-floating base. Umetani and Yoshida in Umetani and Yoshida (1989) obtained a GJM for a free-floating space robot and provided a SpaceDyn Yoshida (1999) software for validation use. In this thesis, we will develop a unified equation of motion for a space manipulator with general tree-structure, based on the graph theory and spatial notation. The detailed information about multi-body dynamics will be illustrated in chapter 3.

2.3.2. Autonomous Mission Management

In the RacoonSim, the space tasks can be conducted in manual or autonomous operational modes. In the manual mode, an operator sends commands or utilizes joystick or other control instruments to perform the space mission. In the autonomous mode, the space mission is conducted fully autonomously with a minimum number of interventions from the operator. Therefore, an AMM subsystem is necessary to provide the ability of autonomy for the servicer satellite.

Refer to Brooks (1986), Rekleitis et al. (2007), one can see that AMM is fully at the core of the OBDH system of the servicer satellite. The fully implementation of AMM relies on the preloaded space missions, which are designed and encoded with a hierarchical FSM engine such as Matlab/Stateflow. The concept of the hierarchical FSM allows a high-level FSM to activate or invoke a lower-level. This supplies the capability to implement a hierarchical decomposition of a high-level task into a sequence of lower-level tasks. Various space missions are then considered as different states which can be switched from one to another when tele-commands sent by the operator are triggered or specific conditions are met. These specific conditions include the lighting condition, communication condition, thermal condition, or the tele-commands sent by operators, etc.

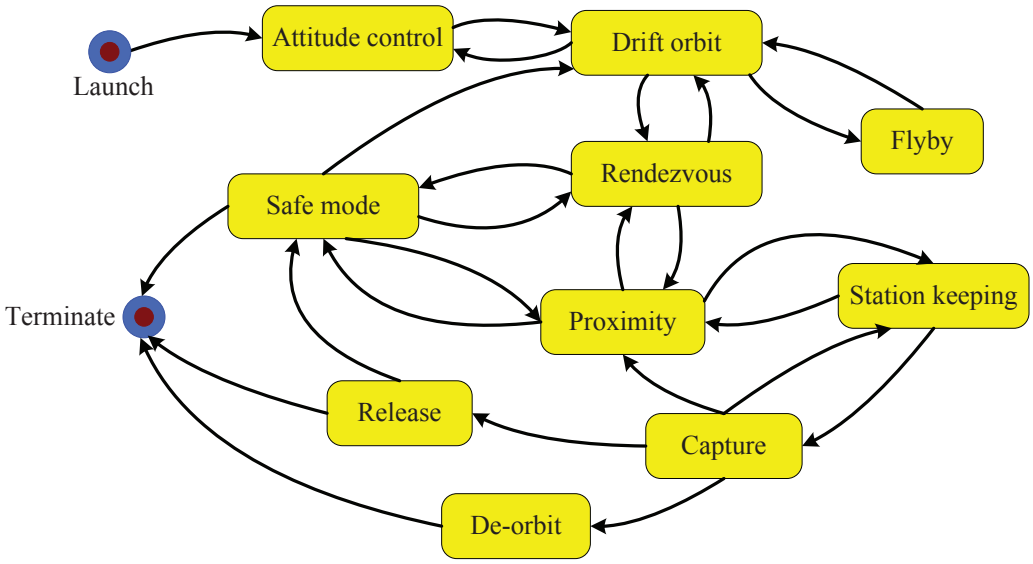


Figure 2.7.: Workflow of high-level OOS missions

Figure 2.7 depicts the operations sequence of the OOS missions as illustrated in section 2.1 employing FSM. Each phase is represented by a state with related transition conditions. When the conditions are met, transitions between phases will be performed to proceed the operations sequence. Figure 2.8 illustrates the implementation of capture phase of space manipulator with FSM. As proposed in Rekleitis et al. (2007), a soft-stop is used to pause the process and then restart it, while a hard-stop fully terminates the process.

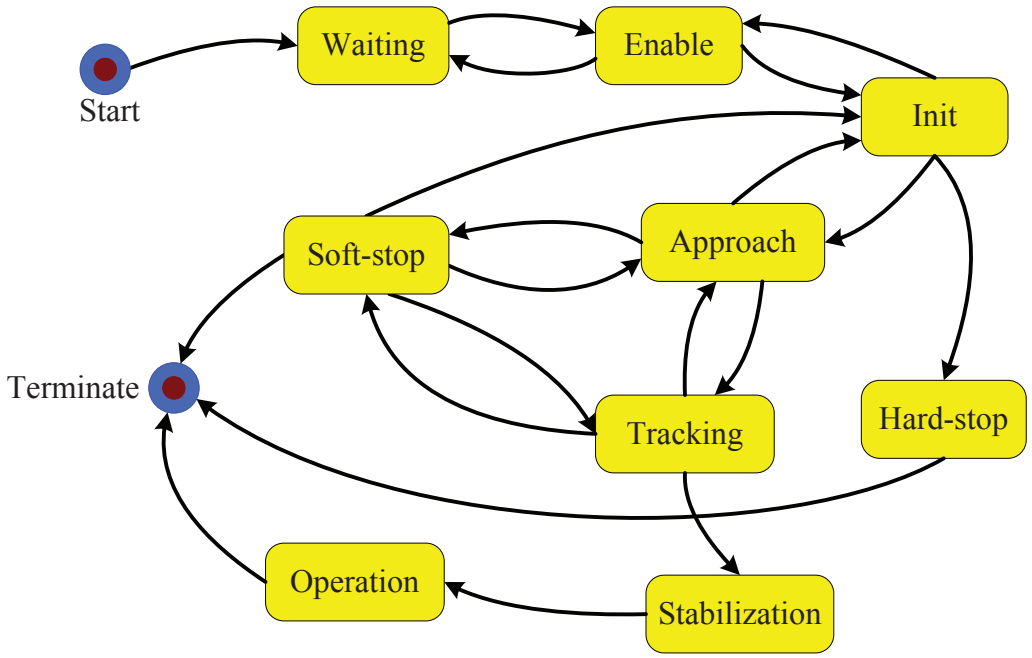


Figure 2.8.: Workflow of capture phase

The role of the operator is to initialize the space operations by submitting high-level command, conduct space mission manually in real-time by using specific instruments, and

monitor the process of the servicer performing tasks. In case of emergency, the operator could send a halt or abort command with highest priority to the autonomy engine.

2.3.3. Path & Trajectory Planning

Based on the knowledge or the preloaded task schedule, AMM determines the high-level action of the space robot, while the motion planning subsystem generates appropriate paths to accomplish the required tasks considering the un-structured environment and singularity issue. Generally, the motion planning problem can be categorized into two classes of global and local planning methods.

Global path and trajectory planning methods try to search an optimal motion which generates a safe path between initial configuration and goal configuration in the free configuration space. Agrawal and Xu (1994) presented a global optimum path planning scheme based on variational approach for redundant space manipulators. Papadopoulos and Poulakakis (2002) introduced a planning methodology for non-holonomic mobile platform using polynomials considering obstacles. Using genetic algorithms, Xu et al. (2008) developed a non-holonomic path planing technique with the advantages of smooth path and constrained motion of the manipulator and the disturbance to the base. In Lampariello (2010) a method of motion planning for capturing a tumbling target is presented based on non-linear optimization and collision avoidance. Order 4 B-spline is adopted to parametrize the joint trajectories. The merits of global planning methods are performance optimization and out of singularity issue, however, on-line optimization requires fairly computational effort and the actual end-effector path can not be predicted. When an off-line optimization is used, it can't deal with the un-structured environment.

Local planning methods require the inverse kinematics to reconstruct the time sequence of joint variables. Dubowsky and Torres (1991) employed an enhanced disturbance map to plan the manipulator's motion for reducing the disturbances to the base. Nenchev (1992), Nenchev and Uchiyama (1995) provided a reaction null-space to generate a motion without influencing the attitude of the base. Yoshida (2003) proposed a similar concept, named zero reaction manoeuvre to reduce the disturbance on the base. These local planning methods confront inevitable singularity problems. To overcome the dynamic singularity problem, one way is to adopt the global planning methods as illustrated before. The other way is to develop singularity avoidance strategies. Many works have been done in this field such as in Chiaverini (1997), Maciejewski and Klein (1988), Marani et al. (2002), Mayorga and Wong (1988), Nakamura and Hanafusa (1986), Qiu et al. (2006), Schreiber et al. (1999). The existed singularity avoidance strategies have the various disadvantages as introduced in Kim et al. (2006). In this thesis, a Singular Task Reconstruction (STR) method for singularity avoidance will be developed in chapter 4. Another problem that will meet in local planning methods is the mobile obstacles in workspace of the manipulator. Until now, like singularity issue, series of publications Faverjon and Tournassoud (1987), Glass et al. (1995), Kanehiro et al., Khatib and Burdick (1986), Kim and Khosla (1992), Maciejewski and Klein (1985), Seraji and Bon (1999), Yunong Zhang et al. (2003), Zlajpah and Nemec (2002) have emerged in the last 3 decades. A main problem to the existed collision avoidance strategy is, normally only the nearest point on the manipulator to the obstacles is adopted to execute the collision avoidance, which leads to vibration of the joint velocity when the control

point switches during its motion. Two control points are employed for collision avoidance to suppress the possible fluctuation and will be illustrate in chapter 4.

2.3.4. Motion Control

Controllers are designed and implemented for realization of planning problem by using actuators and sensors under control law. Depending on the control objectives, various control schemes have been developed for space manipulators.

A so-called Resolved Motion Acceleration Control (RMAC) for a free-floating space manipulator was developed in Umetani and Yoshida (1989) by using the GJM. This method is founded on two feedback loops, an inner loop based on space robot non-linear inverse dynamics and an outer loop operating on the tracking error. A similar control method was also developed in Masutani et al. (1989), Papadopoulos and Moosavian (1994). Nenchev (1993) introduced a resolved acceleration type controller based on a specific fixed-attitude restricted Jacobian matrix to suppress the disturbance of the manipulator to the base. Adaptive controller is another feasible solution to fit the uncertain space environment. Gu and Xu (1993), Xu et al. (1992), Xu (1991) proposed an adaptive control scheme for a space manipulator in Cartesian space with an extended manipulator model. The demerit of this method is that it requires the acceleration of the base. Ma and Huo (1995) introduced two adaptive control schemes, corresponding to availability and unavailability of joint acceleration measurement. It is shown that when the joint acceleration measurement is available, the zero end-effector velocity tracking error can be guaranteed. Wang and Xie (2009) presented a passivity based Jacobian tracking controller for free-floating space robot without involving the measurement of the joint acceleration.

However, the conventional controller design is incapable to address the input and output constraints issue and optimize the performance index on-line. More recently, controller scheme based on MPC has received significant attention. Hedjar and Boucher (2005) proposed a feedback non-linear predictive controller without on-line optimization for a two-link rigid robot. Vivas and Mosquera (2005) described an efficient predictive functional control of a PUMA robot. Combined with fuzzy logic, a new NMPC has been presented in Jasour and Farrokhi (2009). Chi-Ying Lin and Yen-Chung Liu (2012) applied MPC on precision tracking control and constraint handling of mechatronic servo system. In the field of space robotics, McCourt and Silva (2006) investigated the application of MPC for the capture of a target satellite using a deployable planar manipulator. The application of NMPC in the field of space robotics considering singularity and collision constraints will be illustrated in chapter 5.

2.3.5. Virtual Reality & Head-Up Display

In the RacoonSim design, how to intensify the interactivity between operator and machine really matters a great deal. In order to provide an immersion and intuitive feeling for the operator, development of a virtual reality system is significantly necessary. As a powerful tool, Virtual Reality (VR) is not only widely employed in the game industry, but also has the potential to enhance existing trainings and operations. The programmers can build virtual tools, vehicles, robots, even humans in a corresponding virtual environment, which

makes VR economical, reusable, and time-scalable. In fact, ISS, ETS-VII, ROKVISS etc. space projects adopt VR method as a training tool or for the tele-operation of space robot.

As explained in section 2.2.2, the virtual model and environment rely on the previous designed CAD model, which can be done by some commercial software, such as CATIA®, SolidWorks®, Autodesk®, etc. The virtual model will be transformed to a particular format which can be driven by the simulation data. A VRML language, with seamless interface to Matlab/Simulink is chosen in this thesis. A schematic diagram about the principle of VRML is depicted in Figure 2.9. Route manges the driving events, specific Java/Javascript applets are added into the model to proceed the complex events and to intensify the interactivity.

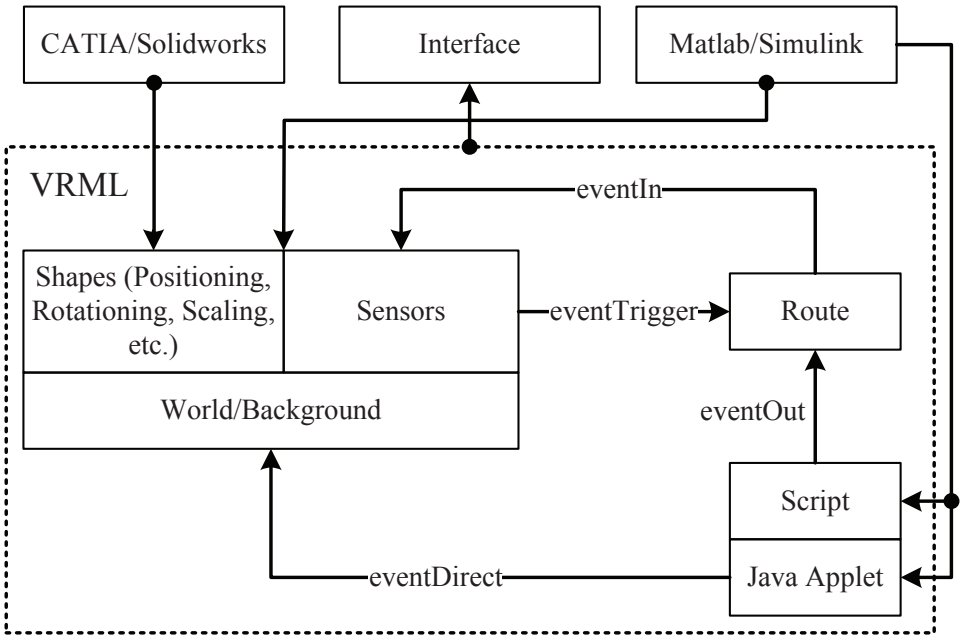


Figure 2.9.: VRML mechanism and its interfaces

For the control input of operator, joystick and 3 dimensional mouse are employed. When a space robot task is processing, it is important for the VR system to supply the full scene of space robot and the relative scene between the end-effector and target. The relative scene will offer a preview on top of end-effector to target satellite. Therefore, a HUD with full relative information and explicit vision of target satellite is built to assist the operator in the tele-operation tasks. It is also possible to provide the operator with complementary information to augment the VR model, such as alarm audio when collision nearly happen, or text information about the system states. A schematic diagram of GUI, VR and HUD as part of MCC is shown in Figure 2.10. These auxiliary subsystems play an important role in understanding the space environment and enhancing the immersion and interactivity.

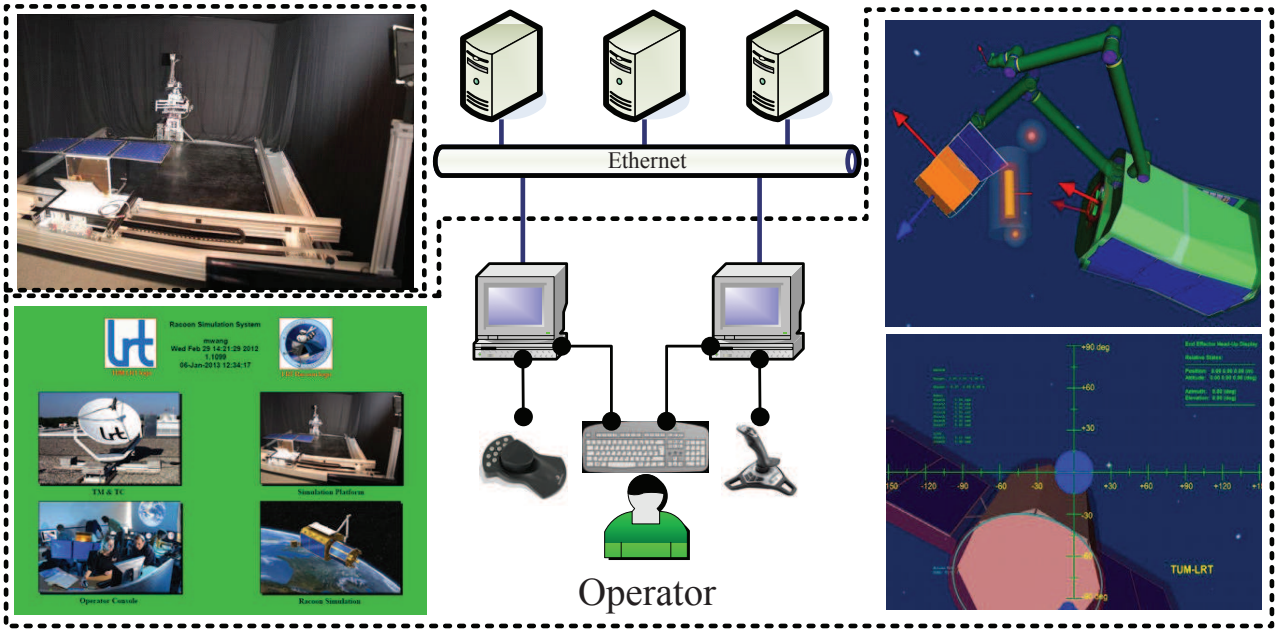


Figure 2.10.: GUI, VR and HUD in MCC

2.4. Summary

In this chapter, in terms of On-Orbit Servicing (OOS) missions using space robot, a real-time distributed simulation system has been developed for autonomous OOS operation or human-in-the-loop tele-operation. The proposed simulation architecture, integrated with Data Distribution Service (DDS), has the capacity of adaptivity, extendibility, and multiple goals. It is implemented in the environment of Matlab/Simulink/Stateflow, which can be extended easily while new hardware or software add into it. At the core of the simulation architecture is the Racoon Simulation (RacoonSim) to conduct the simulation of whole OOS operation using space manipulator. The multi-body dynamics, autonomous mission management, path and trajectory planning, and motion control subsystems are depicted in detail. For the sake of intensifying the immersion and interactivity of operators, an auxiliary Virtual Reality (VR) subsystem is also completed using Virtual Reality Modelling Language (VRML).

The application of Data Distribution Service (DDS) and the commercial software environment Matlab/Simulink/Stateflow decrease the communication latency among various simulators subsystems and speed up the construction of the ground verification system Robotic Actuation, Control and On Orbit Navigation Laboratory (RACOON). Moreover, the reusable simulator is easy to modify and replace by better modules, which provides expandability and scalability of this simulation architecture in the future. The RACOON system has demonstrated in its prototype the feasibility and effectiveness of such a complex but integrated environment. Future improvements will include extension to a wide range of space scenarios and enhancement of usability and scalability.

3. Multibody Dynamics

Accordingly, we find Euler and D'Alembert devoting their talent and their patience to the establishment of the laws of rotation of the solid bodies. Lagrange has incorporated his own analysis of the problem with his general treatment of mechanics, and since his time M. Poinsôt has brought the subject under the power of a more searching analysis than that of the calculus, in which ideas take the place of symbols, and intelligent propositions supersede equations.

—James Clerk Maxwell

Due to the increasing demands of On-Orbit Servicing (OOS) missions and the particular harsh environment of space, the application of space robot has received significant attention. In the last decades, numerous contributions have been made in the field of robot dynamics. From a systemic viewpoint, dynamics algorithm should be formulated with a compact set of equations for ease of development and implementation, while the greatest computational efficiency is obtained synchronously. Generally, the dynamics algorithms are mainly concerned for two particular calculations: forward dynamics and inverse dynamics. Forward dynamics, mainly required for simulation, is concerned about the calculation of the acceleration response when a given force exerts on a given rigid body, on the contrary, inverse dynamics is the calculation of the required force to drive a given rigid body to implement a given acceleration response. Inverse dynamics is of great importance in the controller design, trajectory planning, mechanical design, etc.

This chapter relevant to the development of appropriate trajectory planning and control algorithms, focuses on a modelling scheme that uses the concepts of graph theory and spatial notation for calculating the joint-space forward dynamics of tree structure space manipulator systems. Firstly, the configuration description of space manipulators using graph theory, the parent array, and path matrix are introduced. Secondly, based on the concept of generalized link, the spatial notation and composite rigid body are presented. Thirdly, the Composite Rigid Body Algorithm (CRBA) is exhibited. The Inertia Mapping Matrix (IMM) is then derived from the path matrix, which can be used to analyse the sparsity of the inertia matrix and the complexity of the CRBA algorithm. Within the context, a modified CRBA combining ideas of IMM and spatial vector is proposed to calculate the dynamics of tree structure space manipulators. Its computational procedure and complexity by using IMM are analysed. Finally, a case study and comparison by using a humanoid configuration for branched and un-branched chains particularly verify the effectiveness and potential of the proposed modified CRBA for the space manipulators.

3.1. Graphy Theory

Assumptions used in this chapter are firstly listed here. An N degree-of-freedom (DOF) manipulator with n rotational joints mounted on a spacecraft (floating base) is considered. Each rotational joint has n_i DOF, therefore $N = \sum_{i=1}^{n} n_i$. The whole mechanical chain is composed of $n+1$ rigid bodies, and has $N+6$ generalized coordinates. In this chapter, we assume that each joint has one single rotational axis along z axis, therefore $N = n$.

Description of interconnection structures of multi-body systems can be illustrated by graph theory. In order to reflect the affiliation between the joints and present a more compact inertia matrix, instead of using a regular numbering scheme, a depth-first numbering scheme is adopted in this chapter. A general manipulator with tree structure can be shown by a set of n links (vertices), numbered $1, \cdots, n$, a base link numbered 0 and a set of n joints (arcs), numbered $\hat{1}, \cdots, \hat{n}$. Each arc connects two vertices, the joints are numbered such that joint i connects body i to its parent. A general kinematic tree and a binary kinematic tree as two examples are displayed in Figure 3.1.

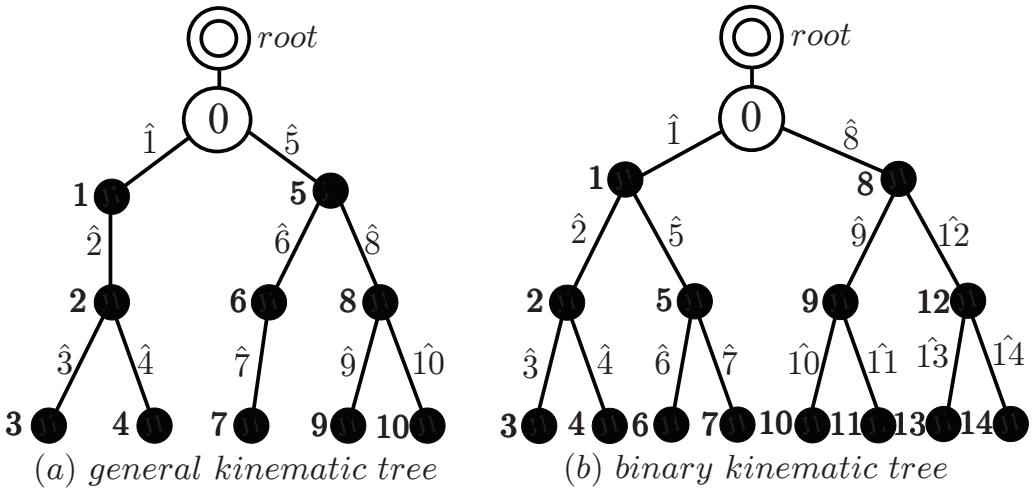


Figure 3.1.: Example of tree structure manipulators

The connectivity of a kinematic tree can be denoted by the parent array $\boldsymbol{\lambda}(\cdot)$. It has one entry for each mobile body, which indicates the body number of its parent. Hence, the parent array for a general kinematic tree as shown in Figure 3.1 is $\boldsymbol{\lambda}(\cdot) = \{0, 1, 2, 2, 0, 5, 6, 5, 8, 8\}$, and $\lambda(i)$ is the parent number of the body i and $0 \leq \lambda(i) < i$. For a specified tree structure graph, using only the parent array $\lambda(i)$ can express the configuration of the system.

One important matrix in graph theory used to illustrate the connectivity of tree structure is the path matrix $\mathbf{T}$, and its elements are defined as follows:

$$\mathrm{T}_{ij} = \begin{cases} -1 & (\hat{i} \text{ is on path between } 0 \text{ and } j) \\ 0 & (\hat{i} \text{ isn't on path between } 0 \text{ and } j) \end{cases} \tag{3.1}$$

The tree structure can uniquely be determined by the parent array $\boldsymbol{\lambda}(\cdot)$ or the path matrix $\mathbf{T}$. There is no column in the path matrix corresponding to vertex 0. From the definition of $\mathbf{T}$,

every row has at least one non-zero element. If T_{ij} is the only non-zero element in row i, the vertex j is a terminal vertex. We define $n_{\lambda(i)} = \sum_{j=1}^{n} |\mathrm{T}_{ji}|$ denotes the number of parent links of link i and $n_{\nu(i)} = \sum_{j=1}^{n} |\mathrm{T}_{ij}|$ denotes the number of offspring of link i for further application.

3.2. Spatial Notation

The so-called spatial notation was introduced and employed in Featherstone (2008). Both the angular and linear components of physical quantities like velocity, acceleration and force are dealt with a unified framework (e.g. 6×1 vector and 6×6 matrix). In the following, a variable with bar represents a spatial tensor. A spatial velocity $\bar{\boldsymbol{v}}$ and a spatial force $\bar{\boldsymbol{f}}$ of link i can be denoted as $\bar{\boldsymbol{v}}_i = [\boldsymbol{\omega}_i^{\mathrm{T}}, \boldsymbol{v}_i^{\mathrm{T}}]^{\mathrm{T}}$ and $\bar{\boldsymbol{f}}_i = [\boldsymbol{m}_i^{\mathrm{T}}, \boldsymbol{f}_i^{\mathrm{T}}]^{\mathrm{T}}$. The spatial transformation matrix is:

$$ {}^{j}\mathbf{X}_i = \begin{bmatrix} {}^{j}\mathbf{R}_i & \mathbf{0} \\ {}^{j}\mathbf{R}_i\mathbf{S}({}^{j}\boldsymbol{r}_i) & {}^{j}\mathbf{R}_i \end{bmatrix} \tag{3.2} $$

where ${}^{j}\mathbf{X}_i \in \mathbb{R}^{6\times6}$ represents the spatial transformation matrix, ${}^{j}\mathbf{R}_i \in \mathbb{R}^{3\times3}$ stands for the coordinate transformation matrix, from coordinate frame i to j. ${}^{j}\boldsymbol{r}_i \in \mathbb{R}^{3\times1}$ is a position vector from the origin of frame i to that of frame j expressed in frame i. $\mathbf{S} \in \mathbb{R}^{3\times3}$ is the cross product operator, for an arbitrary 3 dimensional vector $\boldsymbol{a} = [a_x, a_y, a_z]^{\mathrm{T}}$, it can be defined as

$$ \mathbf{S}(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{3.3} $$

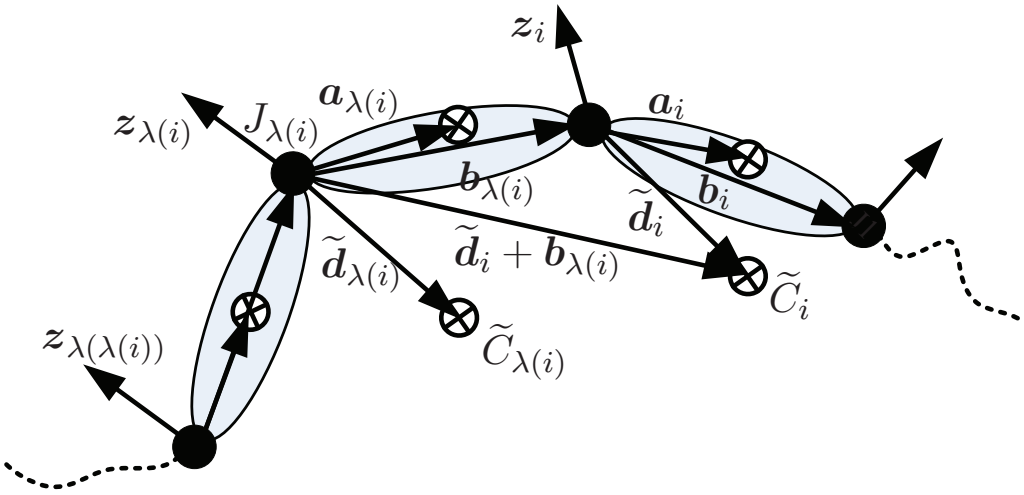


Figure 3.2.: Generalized center of mass of link i

Likewise, an inertia matrix is defined for each individual link in its own frame system. For link i, the spatial inertia matrix $\bar{\mathbf{H}}_i$, expressed in its own coordinate frame, can be represented as follows

$$ \bar{\mathbf{H}}_i = \begin{bmatrix} \bar{\mathbf{I}}_i + m_i\mathbf{S}(\boldsymbol{a}_i)^{\mathrm{T}}\mathbf{S}(\boldsymbol{a}_i) & m_i\mathbf{S}(\boldsymbol{a}_i) \\ m_i\mathbf{S}(\boldsymbol{a}_i)^{\mathrm{T}} & m_i\mathbf{E}_3 \end{bmatrix} \tag{3.4} $$

where $\bar{\mathbf{I}}_i$ is the inertia of link i with respect to its gravity center. $\boldsymbol{a}_i$ is the position vector from the origin of frame i to the mass center of link i, m_i is the mass of link i and $\mathbf{E}_3$ is a 3×3 identity matrix. For detailed description of spatial notation, refer to Featherstone (2008) and Lilly and Orin (1991). For a composite rigid body inertia expressed by spatial notation, it may be computed recursively based on the concept of generalized link (see Figure 3.2) using the following equation

$$\widetilde{\mathbf{H}}_{\lambda(i)} = {}^{i}\mathbf{X}_{\lambda(i)}^{\mathrm{T}}\widetilde{\mathbf{H}}_i{}^{i}\mathbf{X}_{\lambda(i)} + \bar{\mathbf{H}}_{\lambda(i)} \tag{3.5}$$

where $\widetilde{\mathbf{H}}_i$ and $\bar{\mathbf{H}}_{\lambda(i)}$ are the spatial composite rigid body inertia for body i and spatial inertia of body $\lambda(i)$, respectively.

3.3. Dynamics Algorithm

The dynamic model can be obtained from *Newtonian* or *Lagrange* mechanics. There are several different dynamic formulations applied in the field of multi-body dynamics: *Newton-Euler*, *Lagrange-Euler*, *D'Alembert*, *Hamilton*. Nevertheless, all these formulations with different representations are equivalent since they describe the same physical phenomenon. Generally speaking, the choose of dynamics formulation will depend on different objectives such as notation simplicity, computational cost, etc. Once the dynamic formulation is determined, corresponding motion equations about a specific multi-body system can be derived from the formulation. This section first reviews the basic steps of Composite Rigid Body Algorithm derived from the Lagrange formulation. Then the Inertia Mapping Matrix is obtained by using path matrix $\mathbf{T}$. A modified Composite Rigid Body Algorithm (CRBA) in terms of Inertia Mapping Matrix (IMM) is proposed based on the spatial notation.

3.3.1. Lagrange Formulation

Not like Newton's second law which denotes the relationship between the forces and movements, Lagrange formulation is represented by energy and generalized coordinates, which describes the relationship between forces and movements from energy aspect, has obtained considerable applications in the field of dynamics of multi-body and complex systems. With Lagrange formulation, the equations of motion about one complex system can be derived in a systematic way independent of the reference coordinate frame. The Lagrangian can be firstly represented by energy, like kinetic and potential energy, as a function of the generalized coordinates $\boldsymbol{q}_i (i = 1, 2, \cdots, n)$.

$$\mathcal{L} = \mathcal{T}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{U}(\boldsymbol{q}) \tag{3.6}$$

where $\mathcal{T}$ and $\mathcal{U}$ denote the kinetic and potential energies of the whole system. The Lagrange equations can be expressed by

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right)^{\mathrm{T}} - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}\right)^{\mathrm{T}} = \boldsymbol{\xi} \tag{3.7}$$

where $\boldsymbol{\xi}$ is the generalized force composed of the internal and external non-conservative forces, such as the joint actuator torques, the joint friction torques, and the joint torques induced by the contact with the environment.

In our analysis, since the space robot is in a micro gravitational environment, it is reasonable to set $\mathcal{U} = 0$. The kinetic energy $\mathcal{T}$ with a quadratic form of the generalized velocity $\dot{\boldsymbol{q}}$ can be expressed by:

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\mathbf{H}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.8}$$

Substitute equation 3.8 into equation 3.7, a general formulation derived from Lagrangian can be denoted as

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} + \boldsymbol{f}^x \tag{3.9}$$

where $\mathbf{H}(\boldsymbol{q})$ is the Joint-Space Inertia Matrix, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ are the vectors of joint position, joint velocity and joint acceleration, respectively. $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a bias vector containing Coriolis, centrifugal and gravity terms. For the space application, the gravity term can be omitted since the micro-gravity environment in space. $\boldsymbol{\tau}$ is an internal joint force/torque vector and $\boldsymbol{f}^x$ is the external force/torque acting on the space robot.

3.3.2. Composite Rigid Body Algorithm

As was illustrated in the beginning, dynamics algorithms can be divided into two categories: forward and inverse dynamics. To simplify the matter, the forward and inverse dynamics can be encapsulated into two functions, $FwdDyn$ and $InvDyn$:

$$\ddot{\boldsymbol{q}} = FwdDyn(model, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{f}^x) \tag{3.10}$$

$$\boldsymbol{\tau} = InvDyn(model, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{f}^x) \tag{3.11}$$

the symbol $model$ refers to a set of data that describes a particular rigid body system, such as its configuration, joint type, parameters of each link, etc. As explained in Walker and Orin (1982) and Featherstone (2008), the equations of motion for a tree structure rigid body system can be expressed in matrix form as expressed in equation 3.9. The inertia matrix $\mathbf{H}$ can be derived analytically or computed numerically. The goal is to find an efficient algorithm which is more efficient than $O(n^3)$ to compute the inertia matrix. If there is an inverse dynamics algorithm function $InvDyn$ as in equation 3.11, then the non-linear terms of Coriolis and centrifugal force vector $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be calculated by

$$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = InvDyn(model, \boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{0}, \mathbf{0}) \tag{3.12}$$

Comparing equation 3.11 with equation 3.12 and referring to equation 3.9, we can obtain

$$\begin{aligned} \mathbf{H}\ddot{\boldsymbol{q}} &= InvDyn(model, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{f}^x) + \boldsymbol{f}^x - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ &= InvDyn(model, \boldsymbol{q}, \mathbf{0}, \ddot{\boldsymbol{q}}, \mathbf{0}) - InvDyn(model, \boldsymbol{q}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \end{aligned} \tag{3.13}$$

Since the velocity term $\dot{\boldsymbol{q}}$ and external force term $\boldsymbol{f}^x$ in equation 3.13 can be cancelled out, these vectors can be set to $\mathbf{0}$. This equation induces a simple $O(n^2)$ algorithm for calculating $\mathbf{H}$, which is the first method proposed in Walker and Orin (1982). Featherstone named method 3 in Walker and Orin (1982) as Composite Rigid Body Algorithm (CRBA) to compute the inertia parameters of composite sets of rigid bodies at the outer end of the manipulator chain. By setting the sets of joint velocities to $\mathbf{0}$, and the joint accelerations to $\mathbf{0}$ or a unit vector, the column of the inertia matrix can be calculated efficiently through successive application of inverse dynamics algorithm such as Recursive Newton-Euler Algorithm (RNEA).

3.3.3. Inertia Mapping Matrix

Before a detailed forward dynamic algorithm is provided, some basic information about Joint-Space Inertia Matrix (JSIM) is presented. The path matrix expressed in equation 3.1 includes the basic configuration information of the system. It is also the reflection of the system mass distribution, so there must be some relations between path matrix and JSIM. Here we define the Inertia Mapping Matrix:

$$\mathbf{M} = \mathbf{T}^{\mathrm{T}}\mathbf{T} \tag{3.14}$$

Like the inertia matrix, $\mathbf{M}$ is also a symmetric and positive definite matrix. M_{ii} reveals the offspring number of link i, and therefore $M_{ii} = n_{\nu(i)}$. It also reflects the number of recursion to calculate the composite rigid body i. Next, each element in $\mathbf{M}$ is analysed in case of $i < j$ since $\mathbf{M}$ is symmetric. $M_{ij} = 0$ denotes that link i is not the parent node of link j, which also reflects the number of offspring links of link j, and it also represents the number of recursions necessary to compute H_{ij}, which can not be reduced any more. The relationship between parent array, path matrix, IMM and JSIM is shown in Figure 3.3.

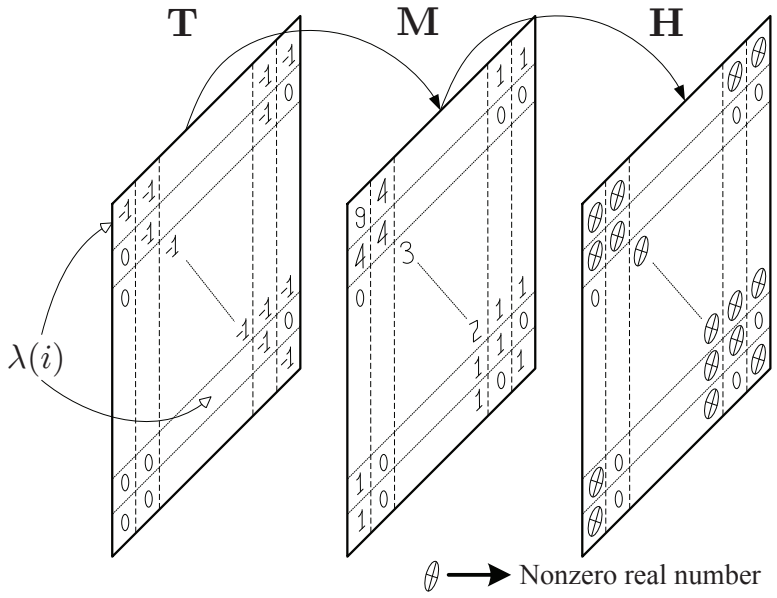


Figure 3.3.: Relationship among path matrix, IMM and JSIM

With a fixed base, the matrix $\mathbf{M}$ can be obtained using equation 3.1 and 3.14. For space applications, the base is not fixed but floating and requires further expansion. When the base is floating, and since every link in the tree structure is the offspring of the base, we can define the augmented path matrix $\mathbf{T}^*$ and augmented IMM $\mathbf{M}^*$ as follows

$$\mathbf{T} = \begin{bmatrix} -1 & -\mathbf{1}_{1\times n} \\ \mathbf{0}_{n\times 1} & \mathbf{T} \end{bmatrix} \tag{3.15}$$

$$\mathbf{M}^* = \mathbf{T}^{*\mathrm{T}}\mathbf{T}^* \tag{3.16}$$

M^*_{11} denotes the computational times of link 0 (base), and corresponds to a 6×6 sub-matrix in JSIM. It also indicates the total number of mobile joints in the kinematic tree, and therefore $M^*_{11} = n + 1$. More details about IMM and its application will be illustrated in section 3.4.

3.3.4. Modified CRBA

Refer to Lilly and Bonaventura (1995), Yoshida (1997) and equation 3.9, for a floating based robot, the general dynamics equation can be described by the following expression:

$$\begin{bmatrix} \mathbf{H}_b & \mathbf{H}_{bm} \\ \mathbf{H}_{bm}^{\mathrm{T}} & \mathbf{H}_m \end{bmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{\tau} \end{bmatrix} + \sum_k \mathbf{J}_e^{k\mathrm{T}} \boldsymbol{f}_e^k \tag{3.17}$$

where $\mathbf{H}_b$, $\mathbf{H}_{bm}$ and $\mathbf{H}_m$ are the base inertia matrix, dynamic coupling matrix and manipulator's inertia matrix, respectively. $\boldsymbol{c}_b$ and $\boldsymbol{c}_m$ are the Coriolis and centrifugal force vectors of the base and manipulator, respectively. $\boldsymbol{f}_b \in \mathbb{R}^{6\times 1}$ and $\boldsymbol{f}_e^k \in \mathbb{R}^{6\times 1}$ are the generalized forces applied on the base and end-effector k, respectively. $\mathbf{J}_e^k \in \mathbb{R}^{6\times n}$ is the Jacobian matrix of k^{th} end-effector.

Considering the IMM defined in section 3.3.3, a modified CRBA is proposed here based on spatial notation. From Figure 3.2, supposing only joint i is in motion and $\ddot{q}_i = 1$, the total force $\boldsymbol{f}_i$ and moment $\boldsymbol{m}_i$ exerted on link j $(j \le i)$ can be computed by inward recursion from the link i to the link 0. If we employ spatial notation, the spatial force $\bar{\boldsymbol{f}}_i$ exerted on generalized link i and the spatial force $\bar{\boldsymbol{f}}_i$ $(j \le i)$ can be computed by inward recursion. The formulae of calculating spatial force and corresponding elements in JSIM can be expressed by

$$\begin{cases} \bar{\boldsymbol{f}}_i = \mathbf{H}_i \boldsymbol{s}_i \\ \bar{\boldsymbol{f}}_{\lambda(j)} = {}^{\lambda(j)}\mathbf{X}_j^{\mathrm{T}} \bar{\boldsymbol{f}}_j \end{cases} \Longrightarrow \begin{cases} (\mathbf{H}_m)_{ij} = \boldsymbol{s}_j^{\mathrm{T}} \bar{\boldsymbol{f}}_j \\ (\mathbf{H}_{bm})_i = {}^{0}\mathbf{X}_i^{\mathrm{T}} \bar{\boldsymbol{f}}_i \end{cases} \tag{3.18}$$

where $\boldsymbol{s}_i \in \mathbb{R}^{6\times 1}$ represents the motion axis of joint i. $\boldsymbol{s}_i$ can be set to $[0\ 0\ 0\ 0\ 0\ 1]^{\mathrm{T}}$ or $[0\ 0\ 1\ 0\ 0\ 0]^{\mathrm{T}}$ for a prismatic joint or rotational joint, respectively. The basic steps of the modified CRBA are shown in Table 3.1.

Table 3.1.: Modified composite rigid body algorithm

Input: $\lambda(i), model, m_i, \bar{\mathbf{I}}_i, \boldsymbol{a}_i, \boldsymbol{b}_i, \mathbf{T}, \mathbf{M}, \boldsymbol{q}_{ini}, \dot{\boldsymbol{q}}_{ini}, \boldsymbol{\tau}, \boldsymbol{f}^x$

step 1: $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = InvDyn(model, \boldsymbol{q}, \dot{\boldsymbol{q}}, \mathbf{0}, \mathbf{0})$
step 2: for $i = n, \cdots, 0$
  do $\widetilde{\mathbf{H}}_i$, $\bar{\boldsymbol{f}}_i$
  for $j = i-1, \cdots, 0$
    if $T_{ij} \neq 0$ do $\bar{\boldsymbol{f}}_j$
    if $M_{ij} = 0$ $H_{(i+6)(j+6)} = 0$
    else $H_{(i+6)(j+6)} = (\mathbf{H}_m)_{ij}$
  end
 end
step 3: $\mathbf{H}_b = \widetilde{\mathbf{H}}_0$; $(\mathbf{H}_{bm})_i = {}^{0}\mathbf{X}_i^{\mathrm{T}} \bar{\boldsymbol{f}}_i$

Integrate: $\ddot{\boldsymbol{q}} = \mathbf{H}^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + \boldsymbol{f}^x - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}))$

3.4. IMM Application

IMM can be employed to explore the geometry of JSIM and cost analysis of CRBA. It also offers a general method to search the properties of JSIM and supplies an analytical tool to explore how different topologies affect the cost.

3.4.1. Branch-induced Sparsity

Once the topology and numbering scheme of an n single-axis joints manipulator is set, the geometry of the JSIM can be obtained from IMM immediately. For a kinematic tree with branches, certain elements of the JSIM will automatically be zero. The number of such zeros can be a large fraction of the total. This phenomenon is called branch-induced sparsity as defined in Featherstone (2008). The number of zeros depends on the topology of the kinematic tree, and the zero's location depends on the topology and the numbering scheme. Taking a space robot with 4 links mounted on a floating base as an example, its total spanning tree solutions and corresponding inertia matrix are displayed in Figure 3.4.

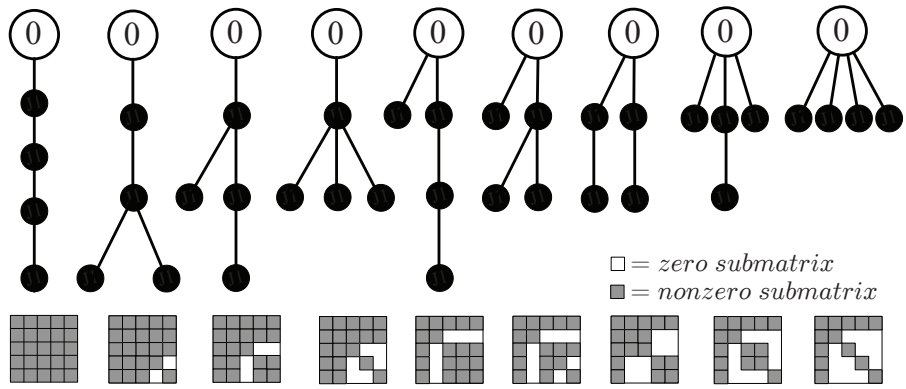


Figure 3.4.: Tree configuration and corresponding JSIM of one base with 4 links

From the IMM and its related JSIM it is easy to seek some other properties of JSIM. That is, for any arbitrary link i in the manipulator, its motion will only be controlled by its parent links and the base (In fact, base is the parent link of all the other links). The motion of link $j(j \neq i)$ is independent of link i if there are no accesses between j and i without passing through the base. The interrelationship of their motion is only dependent on the base, which means, if we want to use link j to influence the motion of link i, the only method is using the dynamic coupling between link j and the base, it can't control the motion of link i directly. When the base is fixed, these tow links will be totally independent of each other. The JSIM can therefore be decoupled.

3.4.2. CRBA Computational Cost Analysis

IMM can be used to explore the sparsity of inertia matrix as explained in section 3.4.1. It can be employed to analyse the computational cost of the CRBA as well. This section illustrates a cost formula for the CRBA, for the case of a tree structure manipulator with general geometry, general inertia parameters, revolute joints and an optional floating base.

As described in Table 3.1, if $M_{ij} = 0$, which indicates that link i is not in the branch of link j, consequently induces $H_{ij} = 0$. If $M_{ij} \neq 0$, H_{ij} requires M_{ij} times of recursive calculations from the outer ends of manipulator chain to link i itself. Consider the algorithm expressed in Table 3.1, it calculates every non-zero element of JSIM. In order to obtain the exact computational cost of CRBA, we first define $D_0 = \sum_{i=1}^{n} min(1, n_{\lambda(i)} - 1) = M_{11} - 1$ to indicate the

numbers of moving links without connectivity to the base and $D_1 = \sum_{i=1}^{n}(n_{\lambda(i)} - 1) = \sum_{i=2}^{n} M_{ii}$ to show the numbers of non-zero elements above the diagonal of IMM. The depth of the tree will influence the computational complexity. If $n_{\lambda(i)}$ has an upper limit d_{max}, then we can obtain $0 \leq D_0 \leq n-1$ and $0 \leq D_1 \leq n(d_{max}-1)$. The cost of CRBA represented by $\mathcal{K}_{CRBA}$ depends on D_0 and D_1 rather than directly on n and can be denoted as

$$\mathcal{K}_{CRBA} = D_0(\boldsymbol{ca} + \boldsymbol{ct}) + D_1\boldsymbol{vt} \tag{3.19}$$

where the symbol $\boldsymbol{ca}$, $\boldsymbol{ct}$ and $\boldsymbol{vt}$ stand for composite rigid body add, composite rigid body transform and vector transform, respectively. When $D_0 = 0$ happens, this means every mobile body connects to the fixed base directly. Then the JSIM is a diagonal matrix, which requires the least computational cost, and so the theoretical minimum complexity of CRBA is $O(1)$. When there are no branches in the kinematic tree, the JSIM is dense, $D_0 = n$ and $D_1 = n(n-1)/2$, which requires the maximum computational cost. This provides a computational cost estimation of CRBA

$$O(1) \leq \mathcal{K}_{CRBA} \leq O(n^3) \tag{3.20}$$

In fact, most of the practical system have a value of D_0 that is either equal to or less than the maximum possible value. For example, the asymptotic complexity of CRBA for a binary kinematic tree as shown in Figure 3.1 is $O(n\log(n))$ Featherstone (2005).

Many investigations have been made to find the minimum-cost implementations for operations $\boldsymbol{ca}$, $\boldsymbol{ct}$ and $\boldsymbol{vt}$. However, the minimum cost for $\boldsymbol{ct}$ and $\boldsymbol{vt}$ mainly depends on how the link coordinate frame are defined. This is another aspect that makes the situation more complicated for a tree structure robot. The minimum cost for $\boldsymbol{ct}$ and $\boldsymbol{vt}$ relied on the coordinate frame located in accordance with a set of Denavit-Hartenberg (DH) parameters (Denavit and Hartenberg (1955)), in which case the coordinate transformation expression by ${}^{\lambda(i)}\mathbf{R}_i$ can be realized via the successive application of two axis screw transforms: one aligns with the x-axis and the other aligns with z-axis. However, if a link connects more than one offspring link, then only one offspring link can possess the benefit of transformation with DH parameters. We define DH nodes and non-DH nodes, for those nodes have and do not have the benefit of DH parameters, respectively. The number of non-DH nodes can be determined by path matrix, if the offspring nodes of link i is $n_{\nu(i)}$, then the number of non-DH can be denoted by

$$n_{nDH} = \sum_{i=1}^{n} max(0, n_{\nu(i)} - 1) = n - 1 - \sum_{i=1}^{n-1} |T_{i,i+1}| \tag{3.21}$$

where the number of non-DH nodes n_{nDH} equals to the number of zeros in the superdiagonal of path matrix $\mathbf{T}$. For the un-branched case, $n_{\nu(i)} \leq 1$ denotes that there are no non-DH nodes in the tree. Using the subscripts a and b to represent the DH node and non-DH node, the cost of CRBA can be expressed by

$$\mathcal{K}_{CRBA} = D_{0a}(\boldsymbol{ca} + \boldsymbol{ct}_a) + D_{1a}\boldsymbol{vt}_a + D_{0b}(\boldsymbol{ca} + \boldsymbol{ct}_b) + D_{1b}\boldsymbol{vt}_b \tag{3.22}$$

Equation 3.19 and 3.22 are the cost formulae of computing JSIM with fixed base, however, when a floating base exists, some more reductions can be made since 3 of DH parameters between link i and its parent link $\lambda(i)$ can be set to 0, if link $\lambda(i)$ happened to be a floating base McMillan et al. (1995). Therefore the operations on each affected $\boldsymbol{ct}$ and $\boldsymbol{vt}$ transforms

can be decreased. We define here a new set to indicate the set of DH nodes that are offspring of a floating base. If we define D_{0c} and D_{1c} to count the number of saving operations in $\boldsymbol{ct}$ and $\boldsymbol{vt}$, respectively, then the final cost formula of $\mathcal{K}_{CRBA}$ with a floating base is then

$$\mathcal{K}_{CRBA} = D_{0a}(\boldsymbol{ca} + \boldsymbol{ct}_a) + D_{1a}\boldsymbol{vt}_a + D_{0b}(\boldsymbol{ca} + \boldsymbol{ct}_b) + D_{1b}\boldsymbol{vt}_b - D_{0c}\boldsymbol{ct}_c - D_{1c}\boldsymbol{vt}_c \tag{3.23}$$

Let mp and ad be the symbol to represent the operation of multiplication and addition, respectively. The minimum cost for $\boldsymbol{ca}$ operation is $10mp$ for spatial notation. The minimum cost operations for $\boldsymbol{ct}$ and $\boldsymbol{vt}$ of spatial notation are listed in Table 3.2. For further details on efficient implementations of $\boldsymbol{ct}$ and $\boldsymbol{vt}$ with different node types (DH nodes or non-DH nodes), see (Featherstone, 2008).

Table 3.2.: Required operation cost of different transforms

$\boldsymbol{ct}$	**Operation cost**	$\boldsymbol{vt}$	**Operation cost**
$\boldsymbol{ct}_a$	$32mp + 33ad$	$\boldsymbol{vt}_a$	$20mp + 12ad$
$\boldsymbol{ct}_b$	$47mp + 48ad$	$\boldsymbol{vt}_b$	$24mp + 18ad$
$\boldsymbol{ct}_c$	$18mp + 21ad$	$\boldsymbol{vt}_c$	$12mp + 8ad$

Considering the spatial notation, the cost of CRBA can be expressed as follows

$$\begin{aligned}\mathcal{K}_{CRBA} = {} & D_{0a}(32mp + 43ad) + D_{1a}(20mp + 12ad) \\ & + D_{0b}(47mp + 58ad) + D_{1b}(24mp + 18ad) \\ & - D_{0c}(18mp + 21ad) - D_{1c}(12mp + 8ad)\end{aligned} \tag{3.24}$$

Employing path matrix $\mathbf{T}$ and IMM $\mathbf{M}$ from section 3.1 and section 3.3.3, the expression for $D_{0a}, \cdots, D_{1c}$ can be denoted by

$$\begin{cases} D_{0a} = D_0 - D_{0b} = \sum_{i=1}^{n-1} |T_{i,i+1}| \\ D_{1a} = \sum_{i=1}^{n-1} M_{i,i+1} \\ D_{0b} = M_{11} - 1 - \sum_{i=1}^{n-1} |T_{i,i+1}| \\ D_{1b} = D_1 - D_{1a} = \sum_{i=2}^{n} M_{ii} - \sum_{i=1}^{n-1} M_{i,i+1} \\ D_{0c} = 1 \\ D_{1c} = max_{i\in[2,n]} M_{ii} \end{cases} \tag{3.25}$$

Substituting equation 3.25 into equation 3.24, the exact computational cost of CRBA based on spatial notation can be obtained. Their application and analysis will be illustrated in section 3.5.

3.4.3. JSIM Factorization Analysis

As illustrated in previous sections, in order to calculate the forward dynamics of space robot, three steps have to be adopted to complete the calculation in equation 3.9:

(1) Calculate the bias vector $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$;

(2) Compute JSIM $\mathbf{H}(\boldsymbol{q})$;

(3) Inverse of JSIM $\mathbf{H}^{-1}(\boldsymbol{q})$;

In general, the computational cost of $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\mathbf{H}(\boldsymbol{q})$ are $O(n)$ and $O(n^2)$, where their algorithms correspond to inverse dynamics algorithm – RNEA and forward dynamics algorithm – CRBA, respectively. For the sake of resolving step (3) (i.e. inverse of JSIM), a factorization process should be firstly taken into account. A UDUT factorization algorithm in Saha (1997) has been proposed to explore the various properties of the factorization, but without considering branch-induced sparsity. If we try to factorize JSIM $\mathbf{H}$ using a standard Cholesky or LDLT factorization ($\mathbf{H} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$ or $\mathbf{H} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$), then it will treat the resulting factor matrix as dense, which can't reflect the branch-induced sparsity clearly. Therefore, the equivalent LTL and LTDL factorization algorithms will be adopted here.

As illustrated in section 3.4.1, IMM can be used to explore the sparsity of inertia matrix induced by the system topology, it can assist matrix factorization as well. The relationship between JSIM and IMM can be established by $\mathbf{H} = \mathbf{L}^{\mathrm{T}}\mathbf{L}$ and $\mathbf{M} = \mathbf{T}\mathbf{T}^{\mathrm{T}}$. Which means, the path matrix $\mathbf{T}$, as an upper triangular matrix, has the same sparsity pattern as the matrix $\mathbf{L}^{\mathrm{T}}$. The special property of a LTL or LTDL factorization, when applied to a matrix with branch-induced sparsity, is that the factorization proceeds without *fill-in* as introduced in Featherstone (2005). In other word, every branch-induced zero element in JSIM remains zero throughout the factorization process. Therefore, the resulting factor matrix $\mathbf{L}$ is also sparse and the branch-induced zero element can be ignored directly during the factorization process. Given any $n \times n$ symmetric, positive-definite matrix $\mathbf{H}$ and its path matrix $\mathbf{T}$ defined by equation 3.1, the algorithm in Table 3.3 will perform an optimal, sparse LTL and LTDL factorization on $\mathbf{H}$ and will display how different topologies influence the computational cost from matrix factorization standpoint.

Table 3.3.: LTL and LTDL factorization algorithms

LTL factorization	**LTDL factorization**

for $k = n, \cdots, 1$ do
  $H_{kk} = \sqrt{H_{kk}}$
  for $i = \lambda(k), \cdots, 1$ do
  $H_{ki} = -T_{ik}H_{ki}/H_{kk}$
  end
  for $i = \lambda(k), \cdots, 1$
  for $j = i, \cdots, 1$ do
  $H_{ij} = H_{ij} + T_{ji}H_{ki}H_{kj}$
  end
  end
end
	for $k = n, \cdots, 1$ do
	  for $i = \lambda(k), \cdots, 1$ do
	  $a = -T_{ik}H_{ki}/H_{kk}$
	  for $j = i, \cdots, 1$ do
	  $H_{ij} = H_{ij} - aH_{kj}$
	  end
	  $H_{ki} = a$
	  end
	end

In order to obtain the exact computational cost of CRBA, D_0 and D_1 were defined in section 3.4.2. Additionally, a new quantity D_2 is defined to express the sum of elements above the diagonal of IMM as follows

$$D_2 = \sum_{i=1}^{n} n_{\lambda(i)}(n_{\lambda(i)} - 1)/2 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} M_{ij} \tag{3.26}$$

The value of D_2 reflects the computational cost of LTL or LTDL factorization. When there are no branches in the kinematic tree, the JSIM and IMM both are dense, resulting in D_2 =

$(n^3 - n)/6$, which requires the maximum computational cost for factorization. If $n_{\lambda(i)}$ has an upper limit d_{max}, then we can obtain $0 \leq D_2 \leq nd_{max}(d_{max} - 1)/2$. The computational cost of sparse LTL and LTDL factorization can be denoted by

$$\mathcal{K}_{LTL} = n \cdot sqrt + D_1 \cdot div + D_2(mp + ad) \tag{3.27}$$

$$\mathcal{K}_{LTDL} = D_1 \cdot div + D_2(mp + ad) \tag{3.28}$$

where the symbol $sqrt$ and div represent square-root calculations and divisions, respectively, while subtractions are treated as additions for cost purpose.

After factorization, another procedure to implement step (3) is back-substitution. The calculation of $\mathbf{L}^{-1}\boldsymbol{x}$ and $\mathbf{L}^{-T}\boldsymbol{x}$ will be considered, where $\boldsymbol{x}$ is a general vector. The computational cost of back-substitution for LTL factorization is $2n \cdot div + 2D_1(mp + ad)$, which for LTDL factorization is $n \cdot div + 2D_1(mp + ad)$. These computational cost analysis will be used in section 3.5 for comparison to verify the performance and effectiveness of the proposed CRBA.

3.5. Case Study

This section illustrates the effect of branches on the computational cost of forward dynamics based on aforementioned computational cost analysis. Two rigid body systems are considered: one is a humanoid composed of 1 single 6-DOF rigid torso and $4k$-DOF ($k = 1, 2, \cdots, 6$) limbs regarded as the branched system, the other is an un-branched chain system with a floating base as shown in Figure 3.5. Both systems have $4k + 1$ mobile bodies, $4k$ single-axis revolute joints, 1 floating base and in total $4k+6$ DOF. Therefore, the only difference between them is one is branched and the other is not.

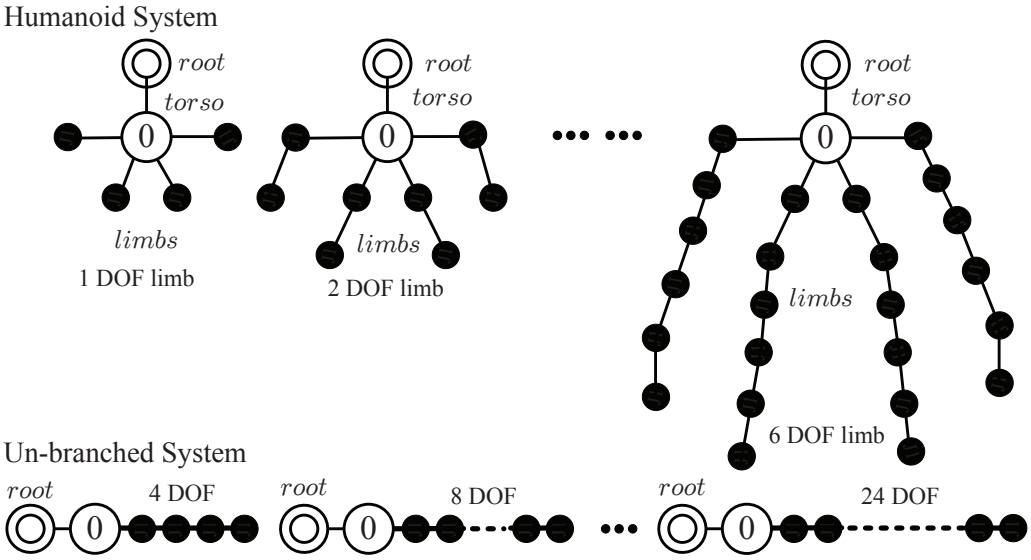


Figure 3.5.: Humanoid and corresponding un-branched chain

To calculate the exact operations of different algorithms, we first need to evaluate the various quantities for the branched and un-branched systems based on equation 3.25. According to the definition of augmented path matrix $\mathbf{T}^*$ and augment IMM $\mathbf{M}^*$, the quantities D_0, D_0, ..., D_{1c} for branched and un-branched systems can be computed as follows

$$D(branched) = \begin{cases} D_0 = 4k & D_1 = 2k(k+1) \\ D_{0a} = 4k-3 & D_{1a} = 2k^2 - k \\ D_{0b} = 3 & D_{1b} = 3k \\ D_{0c} = 1 & D_{1c} = k \end{cases} \tag{3.29}$$

$$D(un-branched) = \begin{cases} D_0 = 4k & D_1 = 2k(4k+1) \\ D_{0a} = 4k & D_{1a} = 2k(4k+1) \\ D_{0b} = 0 & D_{1b} = 0 \\ D_{0c} = 1 & D_{1c} = 4k \end{cases} \tag{3.30}$$

Substituting the various D quantities into equation 3.24, the computational cost of CRBA for humanoid and un-branched systems can be list as follows: $(40k^2 + 168k + 27)mp$ and $(24k^2 + 206k + 24)ad$ for humanoid; $(160k^2 + 120k - 18)mp$ and $(96k^2 + 164k - 21)ad$ for un-branched chain.

The comparison of CRBA applied to branched and un-branched systems is performed. Figure 3.6 shows the operations number of these calculations without considering the addition cost ($\#ad = 0$). One can see that, with small number of DOF, i.e. $n \leq 10$, the computational cost for branched and un-branched system is nearly the same. However, as the DOF increases, the computational cost for un-branched system increases more rapidly than that for branched system. Figure 3.7 shows the cost ratio of CRBA applied to branched and un-branched systems, which indicates that, the CRBA runs more quickly on branched system than on the equivalent un-branched chain. The cost ratio increases as the number of mobile joints increases. When $n \geq 22$, CRBA runs more than twice as quickly on the branched system than on the equivalent un-branched chain.

To verify the performance of proposed algorithm in this chapter, the ABA is utilized to compare with modified CRBA. As introduced in section 3.4.3, in order to solve the forward dynamics problem using CRBA, three steps have to be performed. The most efficient algorithm for calculating $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is RNEA proposed in Balafoutis and Patel (1989), which needs $(93n - 69)$ multiplications and $(81n - 66)$ additions. To calculate the cost of the inverse of JSIM $\mathbf{H}^{-1}(\boldsymbol{q})$, LTDL factorization and back-substitution algorithm expressed in section 3.4.3 is adopted. For an $n \times n$ general symmetric, positive matrix, the cost of factorization and back-substitution is $D_1 \cdot div + D_2(mp + ad)$ and $n \cdot div + 2D_1(mp + ad)$. When the branch-induced sparsity of JSIM is considered and $div = mp$ is assumed, the cost of JSIM inverse with sparsity is $(3D_1 + D_2 + n)$ multiplications and $(2D_1 + D_2)$ additions.

The algorithm of Articulated Body Algorithm (ABA) expressed in McMillan et al. (1995) is used for comparison reason. The computational cost formula for an un-branched chain with a floating base is $(224n - 30)mp + (205n - 37)ad$. To obtain an exact computational cost for the humanoid, a correction of $(66mp + 57ad)$ is applied to account for the additional transformation cost at each non-DH node.

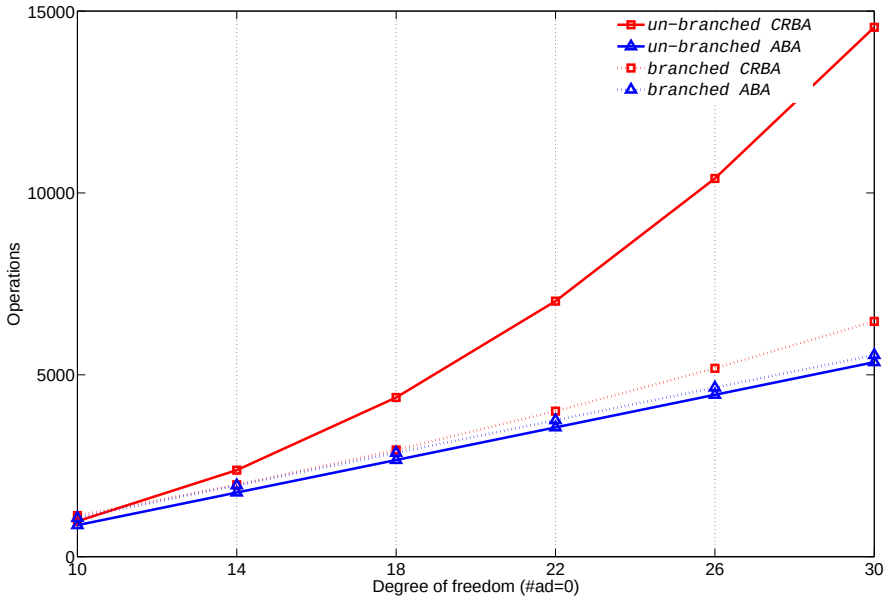


Figure 3.6.: Operations numbers of CRBA and ABA

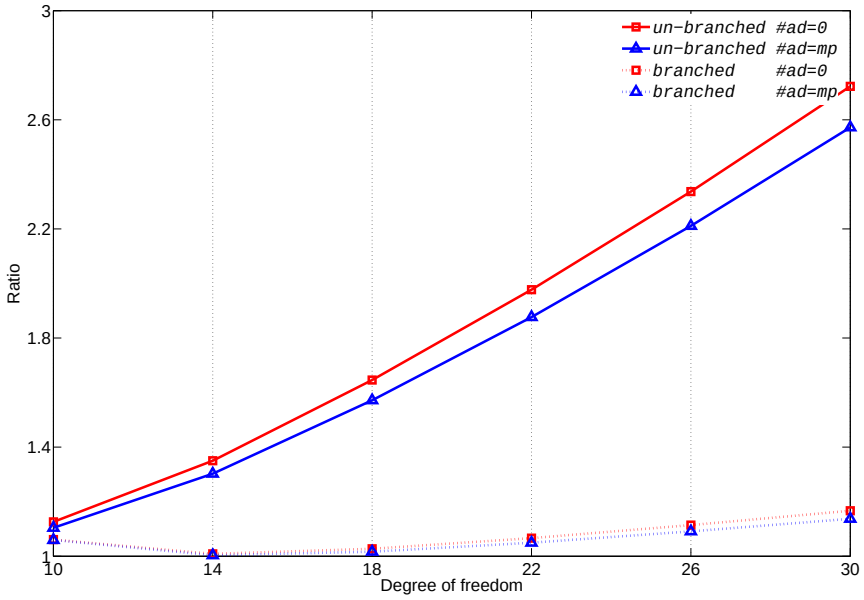


Figure 3.7.: Cost ratio of CRBA and ABA

Figure 3.6 shows the operations number of ABA and CRBA applied to branched and un-branched system without considering the addition cost ($\#ad = 0$). One can see that, when only small number of DOF is considered, i.e.$n \leq 10$, both CRBA and ABA applied to branched and un-branched system have almost the same computational efficiency. When only un-branched chain is considered, ABA is much faster than CRBA on condition that

$n > 10$, especially when $n \geq 14$. The cost ratio between CRBA and ABA for un-branched chain increases as the number of mobile joints increases. But if there are branches in the kinematic tree, ABA runs only slightly faster than CRBA even when $n = 30$. The ratio between CRBA and ABA for branched system increase slowly as the number of mobile joints increases. The CRBA is only about 14% slower than the ABA assuming $ad = mp$, or 17% slower assuming $ad = 0$ when $n = 30$. One can therefore conclude that, $O(n^3)$ algorithms are still competitive with $O(n)$ algorithms, providing there is sufficient branching in the kinematic tree.

3.6. Summary

This chapter presents a detailed JSIM algorithm and its computational analysis for a space robot with tree structure, and therefore makes the following contributions:

- It combines graph theory and spatial notation, and proposes an IMM and a modified CRBA;
- It employs IMM to explore branched-induced sparsity and explicitly offers the computational cost of JSIM;
- It provides a decomposition algorithm to assist the inverse of JSIM considering IMM. Its computational cost is also analysed.

The branched-induced sparsity of the inertia matrix depends on the system topology and numbering scheme. The complexity of calculating the JSIM is estimated between $O(1)$ and $O(n^3)$ by using IMM. The case study shows that for two systems with equal numbers of DOF, the calculation cost for the branched system is much lower than for the un-branched chain. According to the comparison, it shows that when there is efficient branching in the kinematic tree, $O(n^3)$ algorithms are still competitive with $O(n)$ algorithms even at high value of n. Furthermore, this approach provides a more intuitive way of understanding the properties of the JSIM and its computational complexity. The general dynamic formulation presented here is an essential element of the constrained motion model developed for space manipulators and will be employed in the following chapters for kinematic analysis and controller design.

4. Kinematic Control of Manipulator

A single idea, if it is right, save us the labor of an infinity of experiences.

—Jacques Maritain

This chapter focuses on the kinematics analysis of the redundant manipulator at velocity level considering singularity and obstacle avoidance issues. A manipulator moving in its workspace without enough degree-of-freedom (DOF) will have limited applications, therefore, redundant manipulator receives wide spread attention in the research academia and industry. The application of redundant manipulators often faces the following issues: completing multiple subtasks synchronously, possible emergence of singularity of the manipulator during its motion and existence of obstacles in the workspace of the manipulator, etc. In this chapter, inverse kinematics analysis with multiple prioritized subtasks is firstly reviewed. A so-called Singular Task Reconstruction (STR) method is developed based on the concept of manipulability ellipsoid for singularity avoidance. When the manipulator moves in an unstructured environment with obstacles, a collision avoidance strategy considering two control points is proposed to restrain the vibration of joint velocity and generate smoother joint trajectory reference. The proposed singularity and collision avoidance strategies are demonstrated by simulation works.

4.1. Inverse Kinematics

Inverse kinematics refers to determining the joint parameters to realize a required configuration of the manipulator or enable the end-effector to track a predefined path in task-space. In this section, we analyse the inverse kinematics of multiple prioritized subtasks at velocity level for redundant manipulator. As illustrated in section 3.1, an N DOF manipulator with n rotational joints is considered. In this chapter, we assume that each joint has one single rotational axis along z axis, therefore $N = n$.

4.1.1. Redundancy and Task Priority

The degree-of-redundancy (DOR) for fixed base robot has been defined as the difference between the DOF of the manipulator and the number of end-effector task variables. In general, 6 DOF manipulator can meet the end-effector task-space motion requirements. However, in practice, there are some other tasks as well as end-effector task to fulfil synchronously, such as joint range limits, singularity and collision avoidance, etc. When considering free-floating space manipulator, base motion control by using manipulator's redundancy can be regarded

as additional manipulator's task. These additional tasks call for system redundancy to meet the aforementioned requirements.

Supposing a general case at velocity level is taken into account. If there are m tasks required for the manipulator, each task need m_i task variables, then the total number of task variables $M = \sum_{i=1}^{m} m_i$. According to the definition of degree-of-redundancy (DOR), three cases of redundancy can be distinguished in reference to the total number of task variables M and the number of manipulator DOF N. The first case can be characterized by $N < min_{i\in[1,m]} m_i$, which indicates that, there are not sufficient DOF to satisfy even the simplest task with the minimum number of task variables, without mentioning residual redundancy available for other tasks. This case is highly unusual, and primary of theoretical interest. The second case can be denoted as $N \geq M$. In this case, besides the required task variables, there will be some additional redundancy left to satisfy certain criteria. For the third case, $min_{i\in[1,m]} m_i \leq N < M$, the available DOR will not suffice to accomplish all the tasks concurrently, although the manipulator may contain some redundancy with reference to one single task.

As aforementioned the third case, if it is impossible to accomplish all the tasks completely, since the shortage of DOF, then it would be reasonable to perform the most significant task preferentially and the less significant tasks as much as possible using the remaining DOR. Different tasks required can be categorized into a list of subtasks with various level of significance, and this was firstly defined as tasks with the order of priority in Nakamura et al. (1987). Even for a 6 DOF manipulator, the subtasks decomposition between position and orientation is highly favourable. It will enlarge the workspace of the primary subtask by allowing incompleteness for the second subtask. Redundancy analysis and application based on the concept of task priority will be discussed in the following section.

4.1.2. General Solution for Inverse Kinematics

Let $\dot{\boldsymbol{\theta}} = (\dot{\theta}_1, \cdots, \dot{\theta}_n)^{\mathrm{T}} \in \mathbb{R}^{n\times 1}$ and $\dot{\boldsymbol{x}} = (\dot{\boldsymbol{x}}_1, \cdots, \dot{\boldsymbol{x}}_m)^{\mathrm{T}} \in \mathbb{R}^{M\times 1}$ denote the joint-space and task-space manipulator, the relationship between these two spaces can be established by using Jacobian matrix, and hence can be expressed by the following formula:

$$\begin{cases} \mathbf{J}_1\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{x}}_1 \\ \mathbf{J}_2\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{x}}_2 \\ \quad \cdots \\ \mathbf{J}_m\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{x}}_m \end{cases} \tag{4.1}$$

where $\mathbf{J}_i \in \mathbb{R}^{m_i\times n}$ is the Jacobian matrix of the i^{th} manipulator task. If task 1 is chosen as the primary task, the other tasks will be treated as the secondary tasks. Generally, the primary task should be ensure to accomplish, and thereby holds the highest priority. Based on the significance of tasks, total m tasks can thus form a top-down hierarchy. For an arbitrary task m_i, suppose that there are sufficient DOF to perform this task, the solutions can be described as follows:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_i^{+}\dot{\boldsymbol{x}}_i + (\mathbf{E}_n - \mathbf{J}_i^{+}\mathbf{J}_i)\boldsymbol{h}_i \tag{4.2}$$

where $\mathbf{J}_i^+ \in \mathbb{R}^{n\times m_i}$ is the pseudo-inverse of $\mathbf{J}_i$. $(\mathbf{E}_n - \mathbf{J}_i^+\mathbf{J}_i)$ is the null-space of $\mathbf{J}_i$ and $\boldsymbol{h}_i \in \mathbb{R}^{n\times 1}$ is an arbitrary vector. If the priority of the tasks is taken into account, in order to fulfil all the m tasks expressed in equation 4.1, a recursive algorithm can be adopted to solve the priority based inverse kinematics problem at velocity level. In accordance with equation 4.2, enabling $i = 1$ and substituting this equation into the second task $\dot{\boldsymbol{x}}_2$ gives the solution of $\boldsymbol{h}_1$:

$$\boldsymbol{h} = \bar{\mathbf{J}}^+\bar{\boldsymbol{x}}_2 + (\mathbf{E}_n - \bar{\mathbf{J}}_2^+\bar{\mathbf{J}}_2)\boldsymbol{h}_2 \tag{4.3}$$

where $\bar{\mathbf{J}}_2 = \mathbf{J}_2(\mathbf{E}_n - \mathbf{J}_1^+\mathbf{J}_1)$ represents the projection of $\mathbf{J}_2$ onto the null-space of $\mathbf{J}_1$, $\bar{\boldsymbol{x}}_2 = \dot{\boldsymbol{x}}_2 - \mathbf{J}_2\mathbf{J}_1^+\dot{\boldsymbol{x}}_1$ indicates the modification of secondary task due to the existing of the first subtask. Accordingly, the solution sufficing to coordinate exactly subtask 1 and 2 in joint-space can be denoted by

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_1^+\dot{\boldsymbol{x}}_1 + \bar{\mathbf{J}}_2^+\dot{\bar{\boldsymbol{x}}}_2 + (\mathbf{E}_n - \mathbf{J}_1^+\mathbf{J}_1)(\mathbf{E}_n - \bar{\mathbf{J}}_2^+\bar{\mathbf{J}}_2)\boldsymbol{h}_2 \tag{4.4}$$

Employing equation 4.4, the second subtask will be performed without influencing the first one. Accordingly, substituting the obtained solutions from upper tasks into the successive task sequentially, the solution meeting total m subtasks can be represented as follows:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_1^+\dot{\boldsymbol{x}}_1 + \bar{\mathbf{J}}_2^+\dot{\bar{\boldsymbol{x}}}_2 + \bar{\mathbf{J}}_3^+\dot{\bar{\boldsymbol{x}}}_3 + \cdots + (\mathbf{E}_n - \mathbf{J}_1^+\mathbf{J}_1)(\mathbf{E}_n - \bar{\mathbf{J}}_m^+\bar{\mathbf{J}}_m)\boldsymbol{h}_m \tag{4.5}$$

The last term $\mathbf{E}_n - \bar{\mathbf{J}}_m^+\bar{\mathbf{J}}_m$ in equation 4.5 is the projection matrix with respect to $\mathbf{J}_1, \cdots, \mathbf{J}_m$, which is symmetric and idempotent. A schematic diagram of this algorithm is shown in Figure 4.1. Equation 4.5 reveals the inverse kinematics solution considering the priority of subtasks. This solution will be utilized in singularity and collision avoidance as a foundation in the following sections.

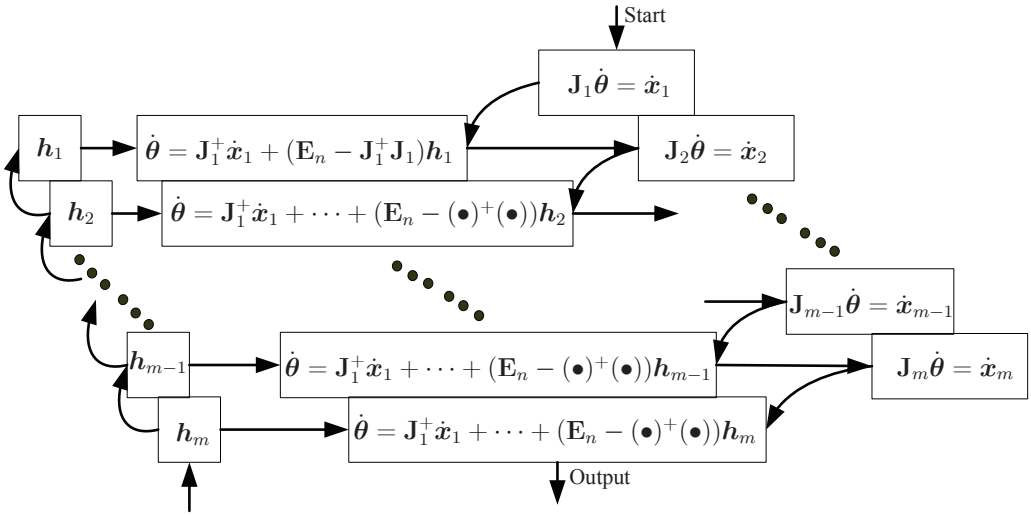


Figure 4.1.: Schematic diagram of inverse kinematics based on task priority

4.2. Singularity Avoidance

None consideration about the singularity of Jacobian matrix has been taken in section 4.1. Singularity appears when the set of joint configuration $\boldsymbol{\theta}^*$ make the Jacobian matrix $\mathbf{J}(\boldsymbol{\theta}^*)$

rank-deficient. It occurs not only at the boundary of the workspace, but also in the workspace of the manipulator. The pseudo-inverse of $\mathbf{J}(\boldsymbol{\theta}^*)$ cannot be determined at such configuration. Singularity limits the application of inverse kinematics to solve joint velocity near such configuration, moreover, in the neighbourhood of singularity, even a small velocity change in task-space requires an enormous joint velocity change in joint-space, which leads to large tracking error and is also harmful for the structure of manipulator.

As an inherent characteristic of articulated manipulator, singularity is an inevitable issue during the manipulator tracking a path defined in its task-space. In this section, a new Singular Task Reconstruction method was proposed based on the concept of manipulability ellipsoid to solve kinematic and algorithmic singularity problems. The basic idea is like the geometric method in Jinhyun Kim et al. (2004), Kim et al. (2006), Qiu et al. (2006): through projecting the desired task onto the direction of manipulability ellipsoid principal axis when the manipulator approaches its singular configuration, the modified task can drive the manipulator out of its singularity region, and guarantee path tracking performance simultaneously. Moreover, this algorithm can be simply extended in the framework of task-priority based method. Based on a real-time evaluation of singular value and eigenvector, this method does not require a preliminary knowledge of the singular configuration. The performance and effectiveness of the proposed singular task reconstruction algorithm are validated by simulation works. The method illustrated here is also the cornerstone of constraint conversion expressed in chapter 5.

4.2.1. Manipulability Ellipsoid

During the mechanical design and task execution of the manipulator, large dexterity is required for better manipulation. However, when the chosen dexterity approaches 0, the manipulator has entered into the neighbourhood of the singularity. Various of dexterity measures have been proposed, such as manipulability measure in Yoshikawa (1984), condition number in Klein and Blaho (1987), etc. A good review is listed in Klein and Blaho (1987). Among these measures, the most effective dexterity measure for singularity is the minimum singular value σ_m of Jacobian matrix $\mathbf{J}$ which can be gained through SVD:

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{4.6}$$

$$\boldsymbol{\Sigma} = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & \mathbf{0}_{m\times(n-m)} \\ & & \sigma_m & \end{array}\right] \tag{4.7}$$

where $\mathbf{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m]^{\mathrm{T}}$ is an $m \times m$ real unitary orthogonal matrix, $\sigma_1, \cdots, \sigma_m$ are the singular values of $\mathbf{J}$ with the relationship $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$, and $\mathbf{V}^{\mathrm{T}}$, the transpose of $\mathbf{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n]$ is an $n \times n$ real unitary matrix. The m columns of $\mathbf{U}$ and n columns of $\mathbf{V}$ are called left-singular vectors and right-singular vectors of $\mathbf{J}$, respectively.

Since $\mathbf{J}$ maps velocities from joint-space to task-space by linear transformation, for the joint velocity within unit norm $\|\dot{\boldsymbol{\theta}}\|^2 = \dot{\boldsymbol{\theta}}^{\mathrm{T}}\dot{\boldsymbol{\theta}} \geq 1$, refer to Walker (1994) a corresponding manipulability ellipsoid can be defined in $\mathbb{R}^m$:

$$\dot{\boldsymbol{x}}_e{}^{\mathrm{T}}(\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\dot{\boldsymbol{x}}_e \leq 1 \tag{4.8}$$

This definition forms an ellipsoid with its semi-axes in the directions of the columns of $\mathbf{U}$, and the length of the semi-axes are the corresponding singular values in $\mathbf{\Sigma}$. Figure 4.2 exemplifies the manipulability ellipsoid of a planar manipulator with 3 revolute joints.

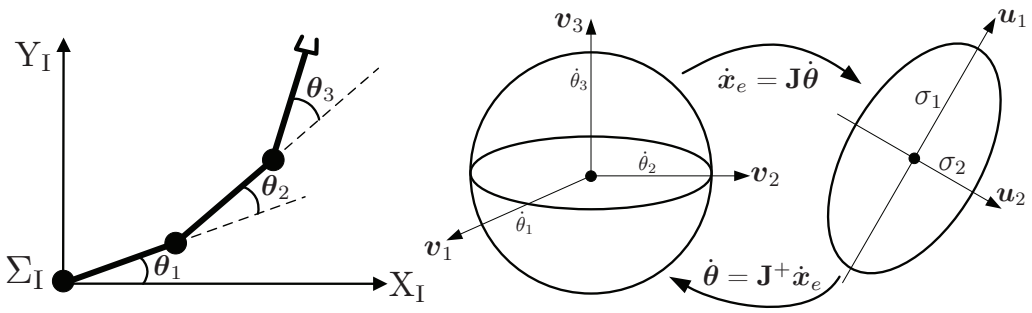


Figure 4.2.: 3 revolute planar manipulator and its manipulability ellipsoid

Along the direction of the major axes of the manipulability ellipsoid, larger end-effector velocity can be generated, on the contrary, smaller end-effector velocity can be obtained when along the direction of the minor axes of the manipulability ellipsoid. Therefore, if all the singular values are nearly equal, which means, the manipulability ellipsoid is almost a sphere, the end-effector can move isotropically along all the directions of the task-space. When the minimum value σ_m is 0, end-effector can not move along the direction of the minor axes, a singularity occurs at such configuration. Contrarily, when σ_m moves away from zero, no attention should be paid for the singularity issue. In the following section, the minimum singular value σ_m and it corresponding left-singular vector $\boldsymbol{u}_m$ will be utilized for the purpose of singularity avoidance.

4.2.2. Singular Task Reconstruction

From the definition of the manipulability ellipsoid in equation 4.8, it can be seen that a singularity emerges when $\sigma_i = 0$ along the i^{th} principal axis direction. Generally speaking, the minimum singular value σ_m of $\mathbf{J}$ has a special significance since it is the only accurate measure of proximity to a singularity. Physically, σ_m depicts the ratio of task-space velocity and joint-space velocity in the direction for which it is hardest to move. The direction is given by the corresponding output left-singular vector $\boldsymbol{u}_m$. Therefore, if we modify the path along the m^{th} principal axis, one can obtain end-effector's singularity-free path as follows:

$$\dot{\boldsymbol{x}}_p = \dot{\boldsymbol{x}}_e - (\dot{\boldsymbol{x}}_e \cdot \boldsymbol{u}_m)\boldsymbol{u}_m = (\mathbf{E}_m - \boldsymbol{u}_m \boldsymbol{u}_m{}^{\mathrm{T}})\dot{\boldsymbol{x}}_e \tag{4.9}$$

where $\dot{\boldsymbol{x}}_e \cdot \boldsymbol{u}_m$ represents the projection of the given task $\dot{\boldsymbol{x}}_e$ along the unit vector $\boldsymbol{u}_m$.

Equation 4.9 guarantees that the manipulator leaves the singularity region, but suffers a big performance loss. In order to improve the performance of singularity avoidance, a weight factor α_v is introduced to equation 4.9:

$$\dot{\boldsymbol{x}}_p = (\mathbf{E}_m - \alpha_v \boldsymbol{u}_m \boldsymbol{u}_m{}^{\mathrm{T}})\dot{\boldsymbol{x}}_e \tag{4.10}$$

where α_v is a turning parameter according to the quantity of the minimum singular value σ_m. As shown in Figure 4.3, if σ_{if} is a predefined threshold stands for the influence zone and

σ_{uf} for the unsafe region, then α_v can be defined as follows referring to Maciejewski and Klein (1985), Zlajpah and Nemec (2002):

$$\alpha_v = \begin{cases} 0 & \sigma_m > \sigma_{if} \\ \frac{1}{2}\left(1 + \cos(\pi \frac{\sigma_m - \sigma_{uf}}{\sigma_{if} - \sigma_{uf}})\right) & \sigma_{uf} < \sigma_m \leq \sigma_{if} \\ 1 & \sigma_m \leq \sigma_{uf} \end{cases} \tag{4.11}$$

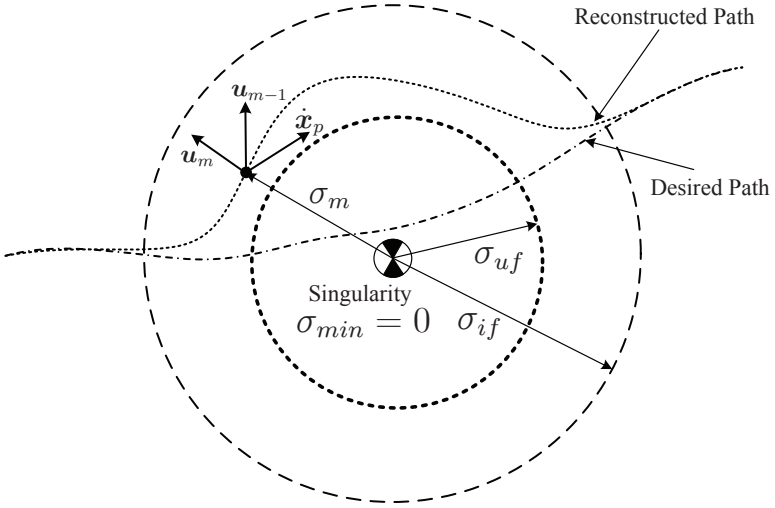


Figure 4.3.: Schematic diagram of singular task reconstruction

When σ_m initially is smaller than the unsafe threshold, equation 4.10 will not guarantee an escape from the singular region and will induce large tracking error. In order to overcome this drawback, we add a third term, a turning parameter α_h into equation 4.10:

$$\dot{\boldsymbol{x}}_p = (\mathbf{E}_m - \alpha_v \boldsymbol{u}_m \boldsymbol{u}_m{}^{\mathrm{T}})\dot{\boldsymbol{x}}_e + \alpha_h \boldsymbol{u}_m \tag{4.12}$$

which causes a repulsive action driving the manipulator out of the neighbourhood of the singularity. The turning parameter α_h can be defined as follows:

$$\alpha_h = \begin{cases} 0 & \sigma_m > \sigma_{uf} \\ \alpha_0(\sigma_{uf} - \sigma_m) & \sigma_m \leq \sigma_{uf} \end{cases} \tag{4.13}$$

where α_0 is the escaping gain for singularity avoidance. In fact, one should only consider the Singular Task Reconstruction when the scalar product $\dot{\boldsymbol{x}}_e \cdot \boldsymbol{u}_m \leq 0$ and $\sigma_m \leq \sigma_{if}$. Therefore, the Singular Task Reconstruction (STR) method can be expressed as follows:

$$\dot{\boldsymbol{x}}_p = \left(\mathbf{E}_m - \frac{1 - \mathrm{sgn}(\dot{\boldsymbol{x}}_e \cdot \boldsymbol{u}_m)}{2} \alpha_v \boldsymbol{u}_m \boldsymbol{u}_m{}^{\mathrm{T}}\right) \dot{\boldsymbol{x}}_e + \alpha_h \boldsymbol{u}_m \tag{4.14}$$

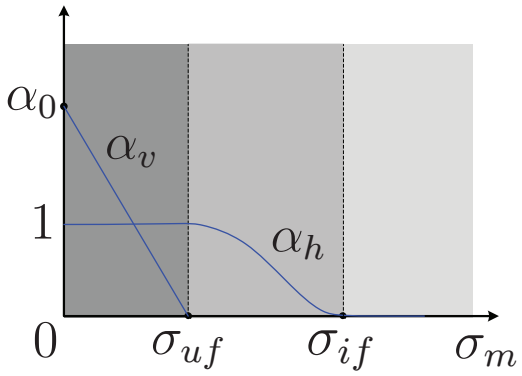


Figure 4.4.: The functional behaviour of the turning parameters for singularity avoidance

Equation 4.14 reflects a geometrical concept: no matter where the manipulator is initially in the unsafe zone or approaches to the singularity region, the modified task-space path will push the system away from the singular region autonomously. Neglecting the null-space motion, i.e. $\boldsymbol{h} = \mathbf{0}$, and considering single task case, the inverse kinematic solution in equation 4.2 becomes:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{+}\dot{\boldsymbol{x}}_p \tag{4.15}$$

4.2.3. STR with Multiple Subtasks

In section 4.1.2, we have derived inverse kinematics in velocity level for multiple subtasks with various task-priorities. Here the application of STR will extend to the inverse kinematics taking account of task-priority. First, two subtasks case are considered. Refer to equation 4.4, when $\mathbf{J}_1^{+}$ can not be determined, i.e. kinematic singularity of the first subtask occurs, or when $\bar{\mathbf{J}}_2$ meets its own singularities, the algorithm singularity issue emerges, it is absolutely critical to adopt necessary reactions to prevent such singularities from happening. In this chapter, the STR method will be applied to both subtasks recursively. Then the kinematic and algorithmic singularities of $\mathbf{J}_1$ and $\bar{\mathbf{J}}_2$ can be removed autonomously. For using the STR method, a Singular Value Decomposition (SVD) algorithm is required to obtain the minimum singular value σ_m and corresponding left-singular vector $\boldsymbol{u}_m$.

As depicted in equation 4.4, neglecting the null-space motion, the joint-space velocity can be divided into two parts:

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_1 + \dot{\boldsymbol{\theta}}_2 = \mathbf{J}_1^{+}\dot{\boldsymbol{x}}_{1p} + \bar{\mathbf{J}}_2^{+}\dot{\bar{\boldsymbol{x}}}_{2p} \tag{4.16}$$

The first subtask $\dot{\boldsymbol{x}}_1$ is reconstructed as $\dot{\bar{\boldsymbol{x}}}_{1p}$ by using STR method for singularity avoidance:

$$\mathbf{J}_1 = \mathbf{U}_1\boldsymbol{\Sigma}_1\mathbf{V}_1^{\mathrm{T}} \tag{4.17}$$

$$\dot{\boldsymbol{x}}_{1p} = \dot{\boldsymbol{x}}_1 - \frac{1-\operatorname{sgn}(\dot{\boldsymbol{x}}_1 \cdot \boldsymbol{u}_{m1})}{2}\alpha_v \boldsymbol{u}_{m1}\boldsymbol{u}_{m1}^{\mathrm{T}}\dot{\boldsymbol{x}}_1 + \alpha_h \boldsymbol{u}_{m1} \tag{4.18}$$

Likewise, the modified second subtask $\dot{\bar{\boldsymbol{x}}}_2$ can be reconstructed as:

$$\bar{\mathbf{J}}_2 = \bar{\mathbf{U}}_2\bar{\boldsymbol{\Sigma}}_2\bar{\mathbf{V}}_2^{\mathrm{T}} \tag{4.19}$$

$$\dot{\boldsymbol{x}}_{2p} = \dot{\boldsymbol{x}}_2 - \frac{1 - \operatorname{sgn}(\dot{\boldsymbol{x}}_2 \cdot \bar{\boldsymbol{u}}_{m2})}{2} \alpha_v \bar{\boldsymbol{u}}_{m2} \bar{\boldsymbol{u}}_{m2}^{\mathrm{T}} \dot{\boldsymbol{x}}_2 + \alpha_h \bar{\boldsymbol{u}}_{m2} \tag{4.20}$$

When we extend the STR method to multiple subtasks with task-priority, referring to equation 4.5, the generalized formulation can be denoted as:

$$\begin{cases} \dot{\boldsymbol{\theta}} = \sum_{i=0}^{m} \dot{\boldsymbol{\theta}}_i \\ \dot{\boldsymbol{\theta}}_i = \bar{\mathbf{J}}_i^{+} \dot{\bar{\boldsymbol{x}}}_{ip} \\ \bar{\mathbf{J}}_i = \mathbf{J}_i \mathbf{J}_i^{*} \\ \mathbf{J}_i^{*} = \mathbf{J}_{i-1}^{*} - \bar{\mathbf{J}}_{i-1}^{+} \bar{\mathbf{J}}_{i-1} \end{cases}, \begin{cases} \dot{\boldsymbol{\theta}}_0 = \mathbf{0} \\ \dot{\boldsymbol{x}}_0 = \mathbf{0} \\ \mathbf{J}_0 = \mathbf{0} \\ \mathbf{J}_0^{*} = \mathbf{E}_n \end{cases} \tag{4.21}$$

where $\dot{\bar{\boldsymbol{x}}}_i = \dot{\boldsymbol{x}}_i - \mathbf{J}_i \dot{\boldsymbol{\theta}}_{i-1}$ represents the i^{th} subtask modification with respect to its higher priority subtasks. When a singularity is detected in $\bar{\mathbf{J}}_i$, the STR approach will be executed on the subtask $\dot{\bar{\boldsymbol{x}}}_i$, which can be described as follows:

$$\bar{\mathbf{J}}_i = \bar{\mathbf{U}}_i \bar{\boldsymbol{\Sigma}}_i \bar{\mathbf{V}}_i^{\mathrm{T}} \tag{4.22}$$

$$\dot{\bar{\boldsymbol{x}}}_{ip} = \dot{\bar{\boldsymbol{x}}}_i - \frac{1 - \operatorname{sgn}(\dot{\bar{\boldsymbol{x}}}_i \cdot \bar{\boldsymbol{u}}_{mi})}{2} \alpha_v \bar{\boldsymbol{u}}_{mi} \bar{\boldsymbol{u}}_{mi}^{\mathrm{T}} \dot{\bar{\boldsymbol{x}}}_i + \alpha_h \bar{\boldsymbol{u}}_{mi} \tag{4.23}$$

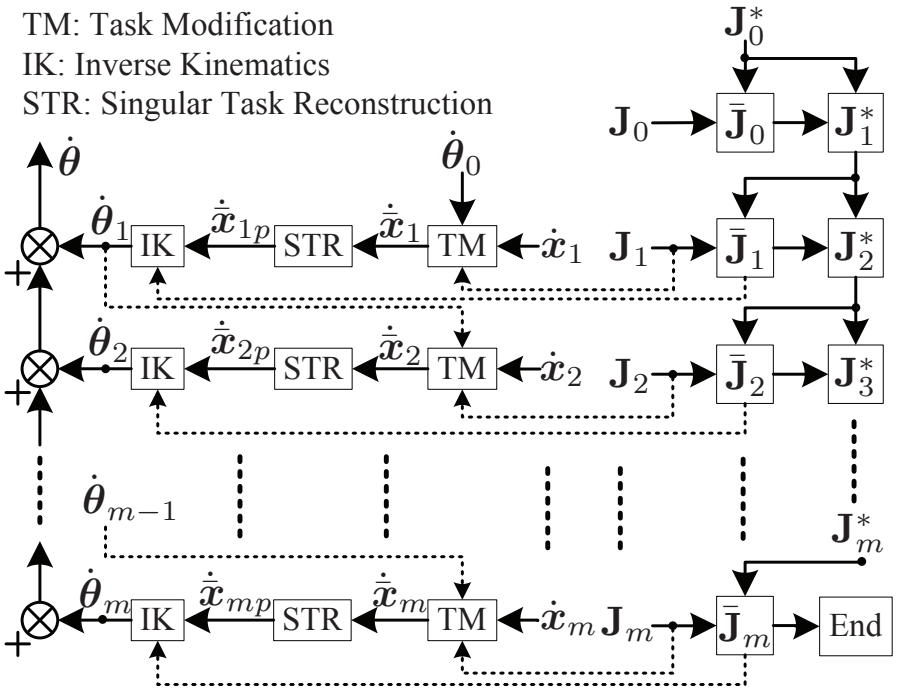


Figure 4.5.: Block scheme of STR for multiple subtasks

Accordingly, in the multiple subtasks cases, all of the singularities, no matter kinematic or algorithmic, can be handled by applying the STR method recursively. The schematic block diagram of the STR method for multiple subtasks referring to equation 4.5 is depicted in Figure 4.5.

4.2.4. Simulation Results

In this section, we demonstrate the effectiveness of our proposed STR method by presenting some simulation results. A 3 revolute manipulator was adopted in this study for verification purpose (see Figure 4.6). Two simulation cases were executed considering kinematic and algorithmic singularities. In order to show the performance of the STR method, widely-used singularity avoidance approach, i.e. Damped Least-Squares (DLS) method with numerical filtering is employed for comparison reason.

$$\mathbf{J}^{+}_{\mathrm{DLS}} = \mathbf{J}^{+}(\mathbf{J}\mathbf{J}^{\mathrm{T}} + \lambda^2 \boldsymbol{u}_m \boldsymbol{u}_m{}^{\mathrm{T}})^{-1} \tag{4.24}$$

$$\lambda^2 = \begin{cases} 0 & \sigma_m > \epsilon_{\mathrm{DLS}} \\ \left(1 - (\frac{\sigma_m}{\epsilon_{\mathrm{DLS}}})^2\right) \lambda^2_{max} & \sigma_m \leq \epsilon_{\mathrm{DLS}} \end{cases} \tag{4.25}$$

where λ is a weighting factor, also referred to as the damping factor, which can be applied to set the relative importance of the minimum residual criterion versus the norm of the solution. λ_{max} denotes the maximum value of damping factor λ. ϵ_{DLS} is the threshold to activate the DLS method for singularity avoidance.

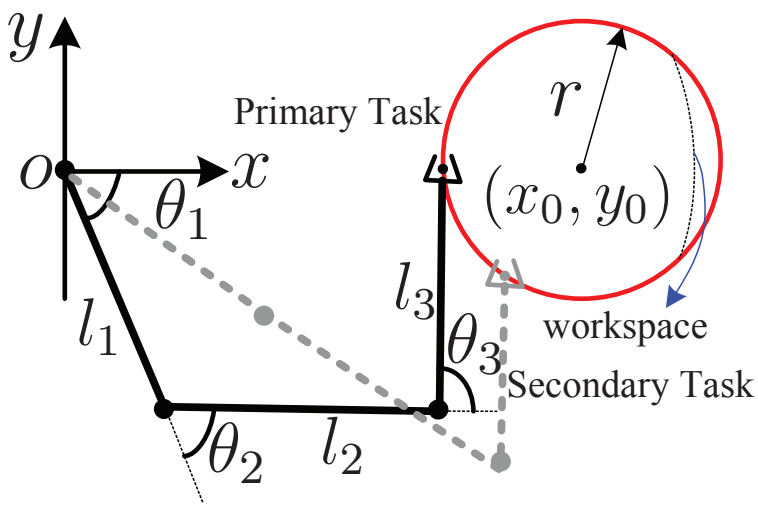


Figure 4.6.: 3 revolute planar manipulator and subtasks

In the simulation, the link length in Figure 4.6 is set to $l_1 = l_2 = 0.35$ and $l_3 = 0.26$. The primary task is tracking a circle in $x - y$ plane with center $(x_0, y_0) = (0.70, 0.00)$ and radius $r = 0.30$. The simulation sample time is 1 millisecond.

Due to the truncation error accumulation, the inverse kinematic solutions as given in section 4.1.2 are expected to suffer from typical numerical drift problem when implemented in discrete time. This drawback can be overcome by replacing the end-effector velocity $\dot{\boldsymbol{x}}_e$ with a feedback correction term as in Siciliano (2009):

$$\dot{\boldsymbol{x}}_c = \dot{\boldsymbol{x}}_d + \mathbf{K}_{ik}\boldsymbol{e} = \dot{\boldsymbol{x}}_d + \mathbf{K}_{ik}(\boldsymbol{x}_d - \boldsymbol{x}_e) \tag{4.26}$$

where $\mathbf{K}_{ik} \in \mathbb{R}^{n \times n}$ is a positive definite (usually diagonal) gain matrix, and $\boldsymbol{e}$ denotes the task-space error between the desired and the actual end-effector position.

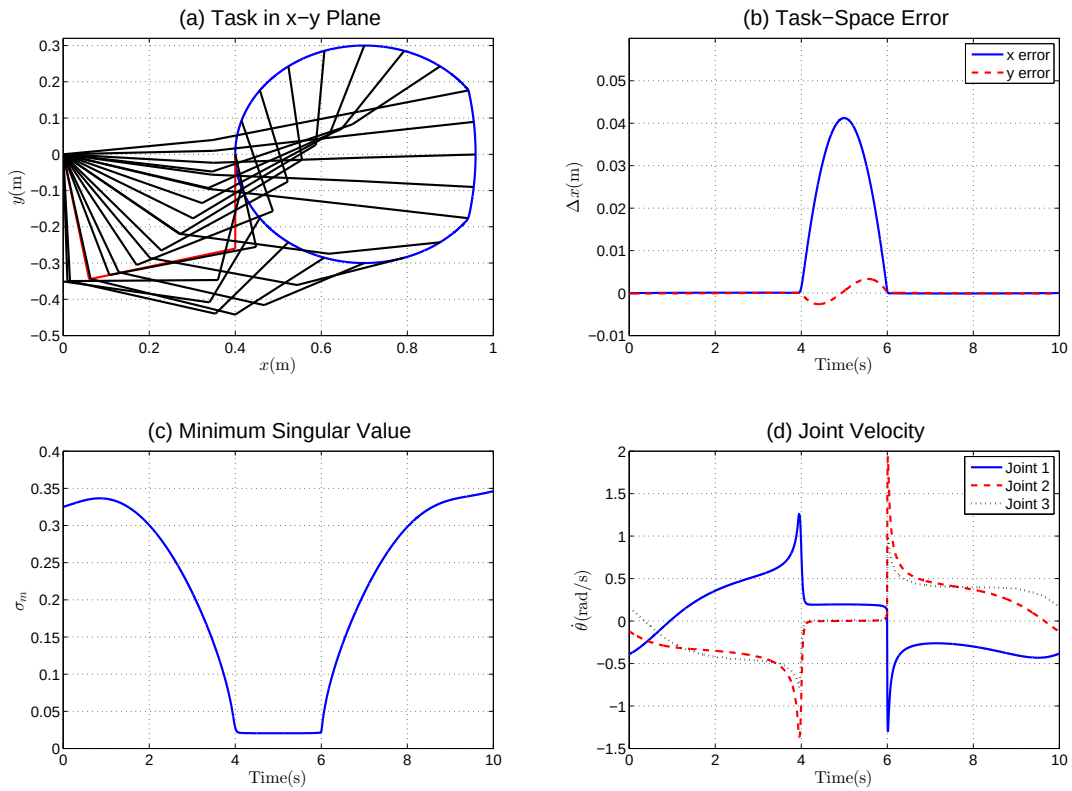


Figure 4.7.: Singularity avoidance with single task by STR method

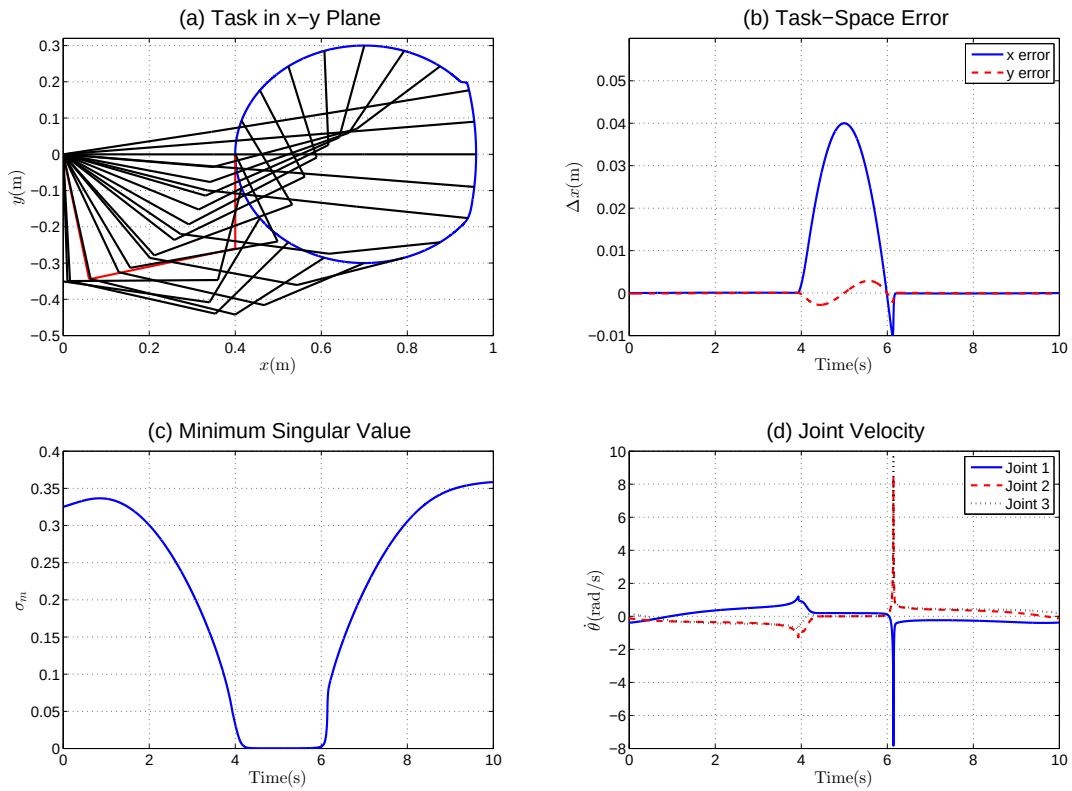


Figure 4.8.: Singularity avoidance with single task by DLS method

The first simulation is with only single task as the primary task depicted in Figure 4.6. The desired task is to track a circle in its workspace in 10 seconds. Some part of the desired task are lying outside the workspace, which ultimately results in encountering kinematic singularity. Simulations with the proposed STR method and DLS method are performed for comparison purpose.

We choose $\alpha_0 = 20$, $\sigma_{if} = 0.05$, $\sigma_{uf} = 0.02$, $\mathbf{K}_{ik} = \text{diag}(100, 100)$ as turning parameters of the STR method. For DLS approach with numerical filtering, $\epsilon_{\text{DLS}} = 0.05$ and $\lambda_{max} = 0.1$ are chosen for the best task performance.

Figure 4.7 and Figure 4.8 show the results of tracking one circle in task-space for the STR and DLS methods, respectively. Obviously, the task-space errors in x and y directions are almost the same for both methods. However, the DLS method suffers from an overshoot problem after the manipulator moves out of the singular region. In addition, the STR method maintains the minimum singular value of about 0.02 corresponding to the lower limits of σ_{uf}, the DLS method does not possess such an ability. Furthermore, the joint velocities from the DLS method are about 4 times larger as compared to the STR method when a singularity occurs. Of course, larger joint velocities are undesirable since they may harm the structure of the manipulator.

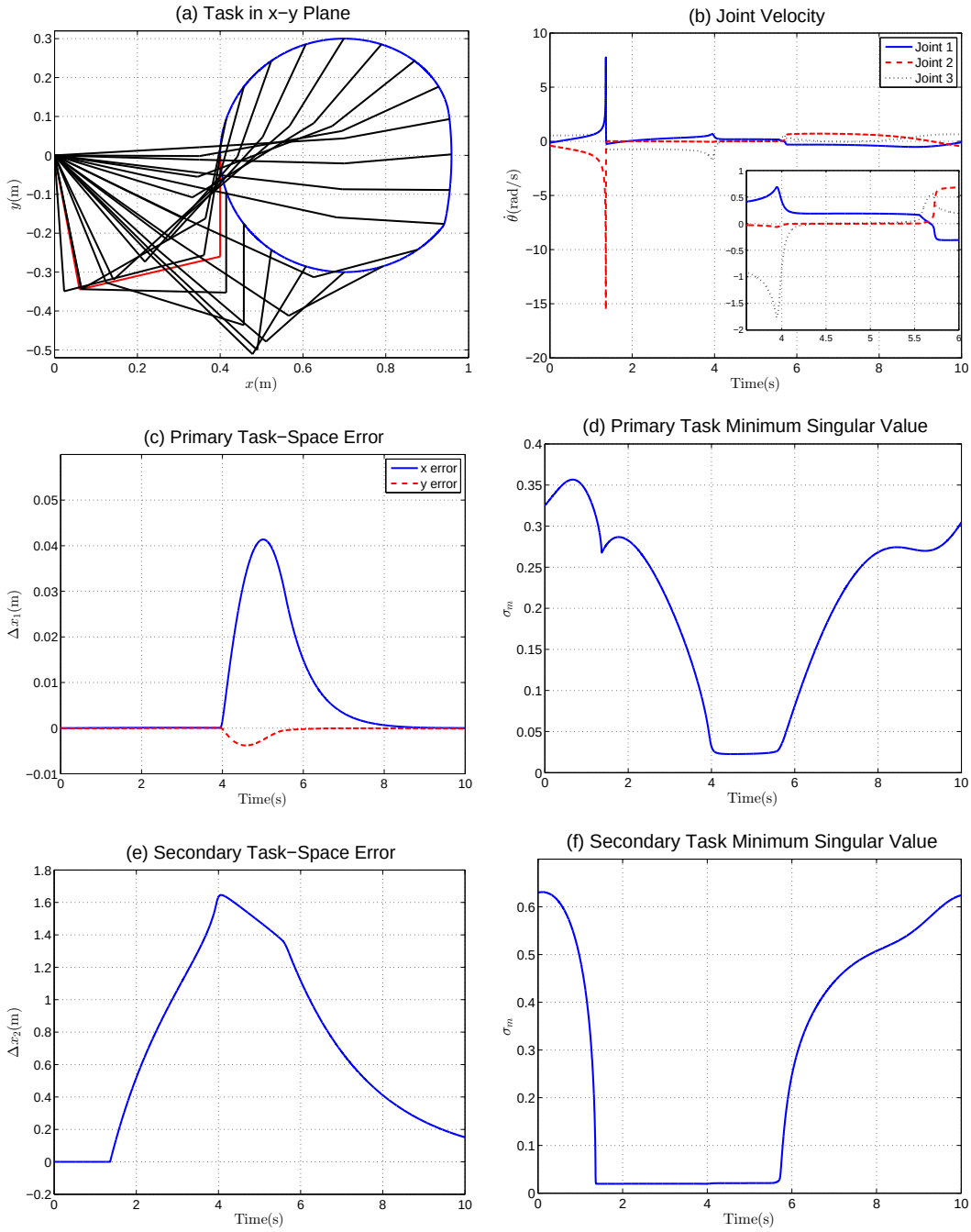


Figure 4.9.: Singularity avoidance with two subtasks by STR method

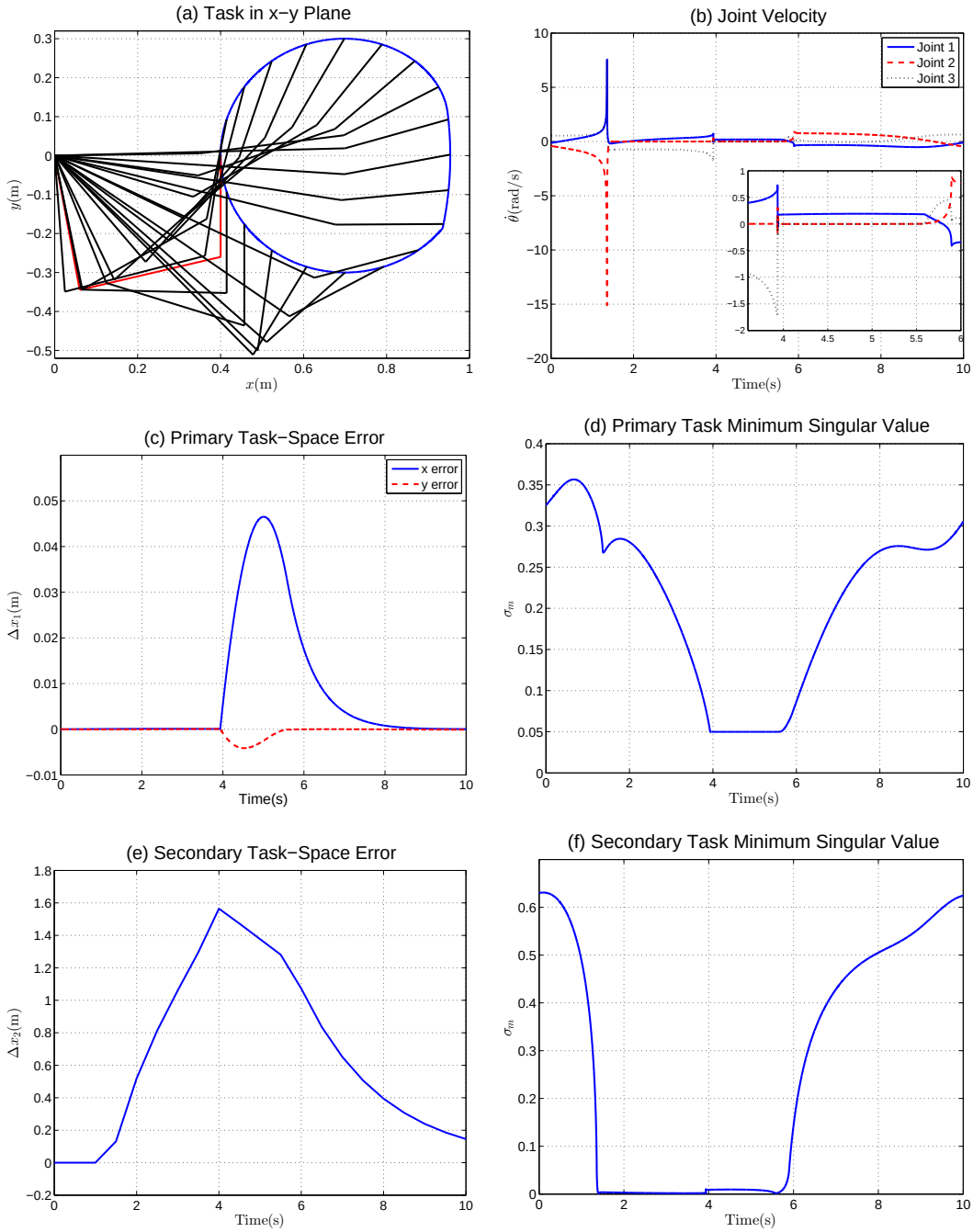


Figure 4.10.: Singularity avoidance with two subtasks by DLS method

To verify the performance of the STR method applying in multiple subtasks with task-priority, a secondary task is introduced as shown in Figure 4.6, while the primary task is still to track a circle as depicted previously. The secondary task $\boldsymbol{x}_2$ is to keep the orientation of the end-effector equal to $90\,°$. Figure 4.9 and Figure 4.10 show the results of completing two subtasks in task-space using the STR and DLS methods with control gains $\mathbf{K}_1 = \text{diag}(1.5, 1.5)$ and $\mathbf{K}_2 = 0.5$, respectively. One can see that, STR method generates smoother joint velocity than DLS approach. Moreover, primary task-space error is smaller for STR method, while the minimum singular value for $\mathbf{J}_1$ is maintained about 0.02 as defined beforehand. Since the conflict between the primary and secondary tasks, both kinematic and algorithmic singularities emerge, while STR method successfully drives the matrix $\bar{\mathbf{J}}_2$ out of the singular

region.

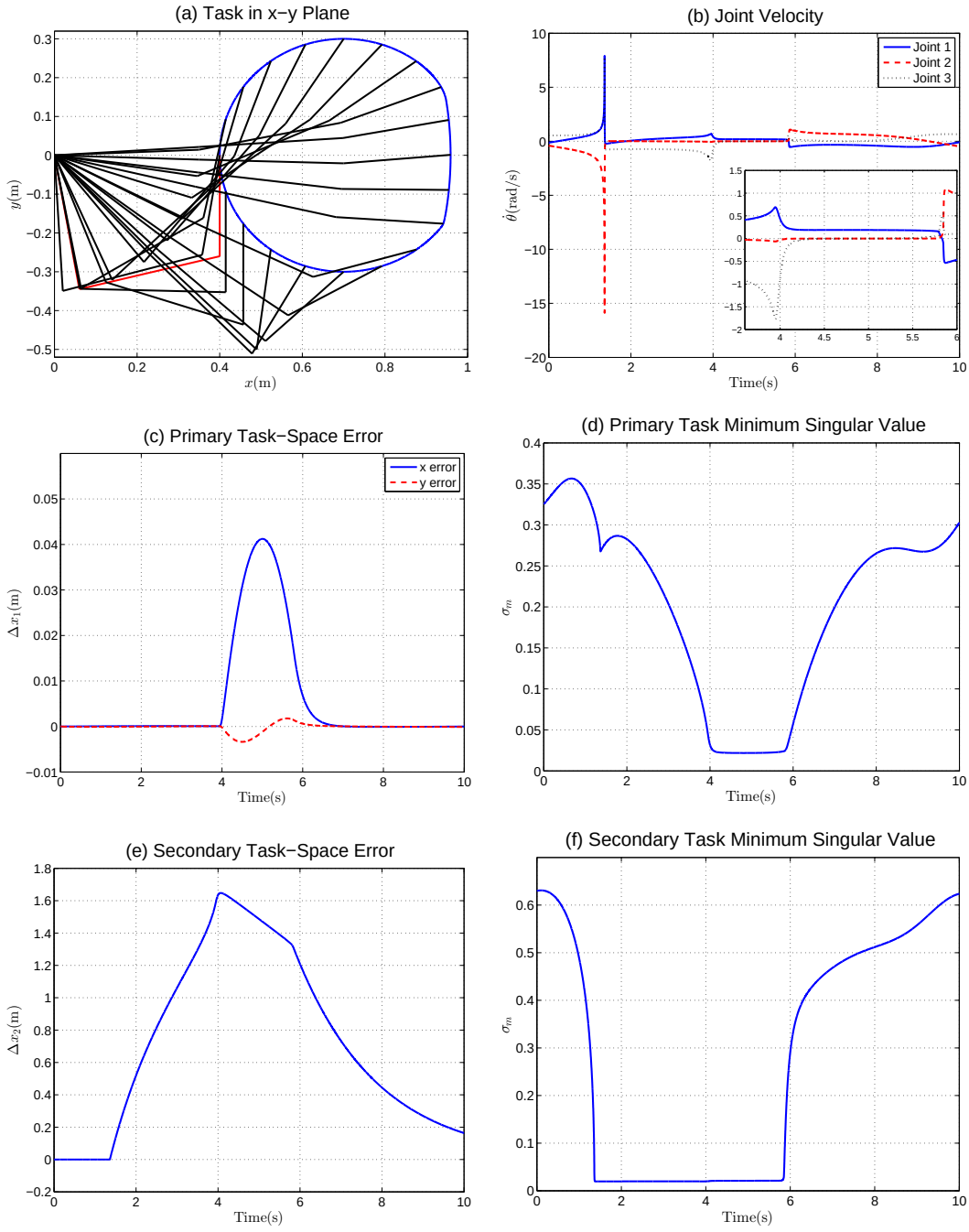


Figure 4.11.: Singularity avoidance with two subtasks and higher gain by STR method

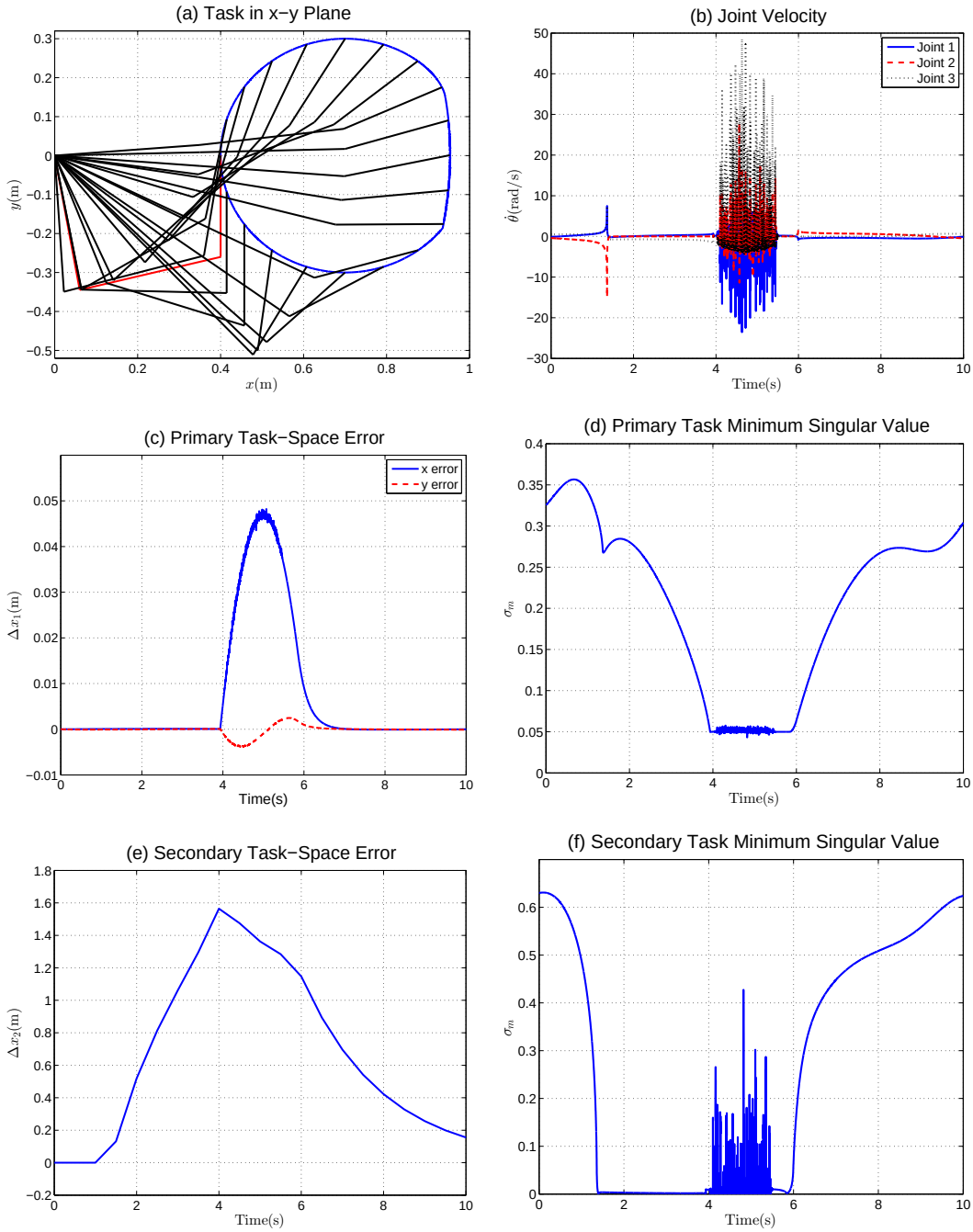


Figure 4.12.: Singularity avoidance with two subtasks and higher gain by DLS method

In order to test the robustness of the proposed STR method, another simulation is conducted with the same turning parameters as above while only the control gains $\mathbf{K}_1 = \text{diag}(4, 4)$ and $\mathbf{K}_2 = 0.5$ are reassigned for primary task and secondary task, respectively. The results are shown in Figure 4.11 and Figure 4.12 for the STR method and DLS method, respectively. A larger control gain speeds the task-space error convergence up and decreases the task-space error concurrently. One can see that, the STR method still possesses the advantages of smoother joint velocity, maintaining minimum singular value of primary and secondary tasks as preplanned. On the other hand, for the DLS method, an oscillation appears when kinematic and algorithmic singularities happen simultaneously, where the minimum singular value of $\bar{\mathbf{J}}_2$ varies drastically and the primary task-space error increases accordingly. The

rapid fluctuation in joint velocity requires huge joint acceleration to accomplish such task which is highly unwelcome in a real-world situation. This simulation reveals the robustness of the STR method while the DLS method is more sensitive to the control gains.

From the simulation results on single task and two subtasks with task-priorities, the effectiveness and robustness of the STR method are demonstrated through simulation studies. When the manipulator is in the neighbourhood of a singularity, the selected minimum singular value and its corresponding left singular vector guarantee the effectiveness and robustness of the proposed STR method.

4.3. Obstacle Detection and Avoidance

In order to perform various missions effectively, the motion control of robotic manipulators in the presence of workspace obstacles has been a prime concern over the past few decades. The existed solution to this problem, termed as collision avoidance, is normally considered as a planning problem can be roughly categorized into the two classes of global and local planning methods. As was described in section 2.3.3, global planning methods try to search an optimal collision-free motion for the robots which generates a safe path between initial and goal configuration in the free configuration space. Its computational cost is expensive and grows dramatically as the number of manipulator's joints increase. Moreover, global methods are only suitable for structural and static environments which aren't the case for general space missions. The un-modelled and moving obstacles call for the on-line planning and control which entails local planning strategy as a main research topic. In this section, we propose a new collision avoidance strategy with two control points to suppress the possible fluctuation problems induced by the control point switch. Two control points are chosen to prevent collision between the manipulator and the obstacles and vibration issues.

4.3.1. Collision Detection

Before a collision avoidance strategy taking effect, what matters most is how to detect the moving obstacles nearby. Normally, object detection algorithms involve 3 dimensional space modelling problem and can be classified into two categories of Bounding Volume Hierarchy (BVH) and space decomposition algorithms as illustrated in Ericson (2005). Both two algorithms use the hierarchical structure to reduce the number of intersection tests of the geometrical elements and enhance the on-line computational efficiency. Comparing space decomposition algorithm with BVH algorithm, the former one needs more memory to restore and less flexibility which often applied in the sparse environment with equally distributed objects, while BVH method, which is applicable to use in the complex environment to calculate the possible collision, has gained a wide spectrum of attention both in academic research or industrial application. Before we go into the detection algorithm in detail, let us first recall the definition of the strictly convex objects and Voronoi region.

Strictly convex objects. Let $\mathcal{O}$ represent a closed subset of $\mathbb{R}^3$ and $X(\mathcal{O})$ be the interior of (greater open subset of) $\mathcal{O}$. $\mathcal{O}$ is strictly convex if and only if $\lambda A + (1-\lambda)B \in X(\mathcal{O})$, $\forall A \in \mathcal{O}$, $\forall B \in \mathcal{O}$ and $0 < \lambda < 1$.

Voronoi region $\mathcal{VR}(Y)$ for feather Y. A Voronoi region associated with a feather Y is a set of points that are closer to Y than any other feather.

Strictly convex for an object is a strong condition and hardly always keeps true. For example, a convex polyhedron is not strictly convex, since the linkage of two points on one facet is not inside of the X($\mathcal{O}$). Normally, in the detection algorithm, only the minimum distance and corresponding control point are calculated for collision avoidance. This is adequate for strictly convex objects, however, not enough for the general objects, since in the physical environment, most of the objects can not be regarded as the strictly convex objects. Traditionally, only sphere, ellipsoid, etc. are strictly convex. The existence of none strictly convex objects makes the control point on object move not in a continuous way. If this happens frequently, a vibration will appear since the switch of the control point. A simple example about this phenomenon is introduced in Kanehiro et al..

In order to overcome this drawback, in literature Kanoun et al. (2009) the authors have employed multiple velocity damper constraints to suppress the possible robot arm vibration during its motion with obstacles in the workspace. However, this method requires an extensive computational cost for constraints processing. In this chapter, we developed a lightweight collision detection and avoidance algorithm referring to our daily life. Considering one people wants to take a dish out of a cabinet, when he puts his arm in the cabinet, he not only concerns the nearest point between his arm and cabinet, but also controls the other point's motion which is also on his arm. If both two points are out of collision, he does not care about the other part of his arm between this two points, because he is very sure that non collision will happen once the two different control points on his arm are without touching the cabinet. Therefore, before we develop our own collision avoidance strategy, an obstacle detection algorithm whose outputs are the two control points of minimum distance and so-called Maximum Projection Distance (MPD), should be firstly developed. Here we will first describe the detection algorithm between two line segments and then expand its application into more complex circumstances.

Figure 4.13 listed the possible relationship between two line segments. The line segment consists of one edge $\mathcal{E}$ and two vertices $\mathcal{V}_i(i = 1, 2)$, 3 dimensional space around the line segment can be divided into 3 Voronoi regions. A moving edge can be in each of the Voronoi regions, or lie in more than one Voronoi region at the same time. Consequently, the moving edge has to be separated into several line segments by clipping function VR_clip. The pseudo code of detection algorithm between two line segments is shown in Table 4.1.

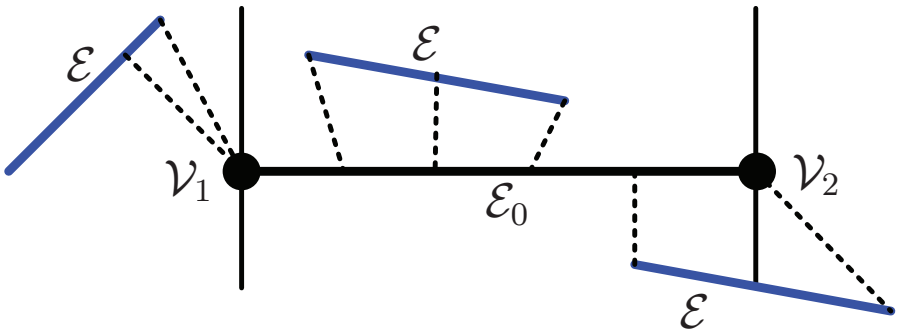


Figure 4.13.: The relationship of two line segments

The detection algorithm between two line segments is the foundation to devise further complex detection algorithms. When the obstacle is not a strictly convex object, it can be treated as a polyhedron composed of a set of basic facets like triangles. Figure 4.14 displays the relationship of a moving edge and a triangle. One can see that, the 3 dimensional space is divided into 7 Voronoi regions by the triangle. If the moving edge lies in the separate Voronoi region, one can calculate the two control points with different functions. When it lies in more than one Voronoi region, a clipping algorithm VR_clip has to be employed beforehand to search for the two control points. The detection algorithm between line segment and triangle is listed in Table 4.1.

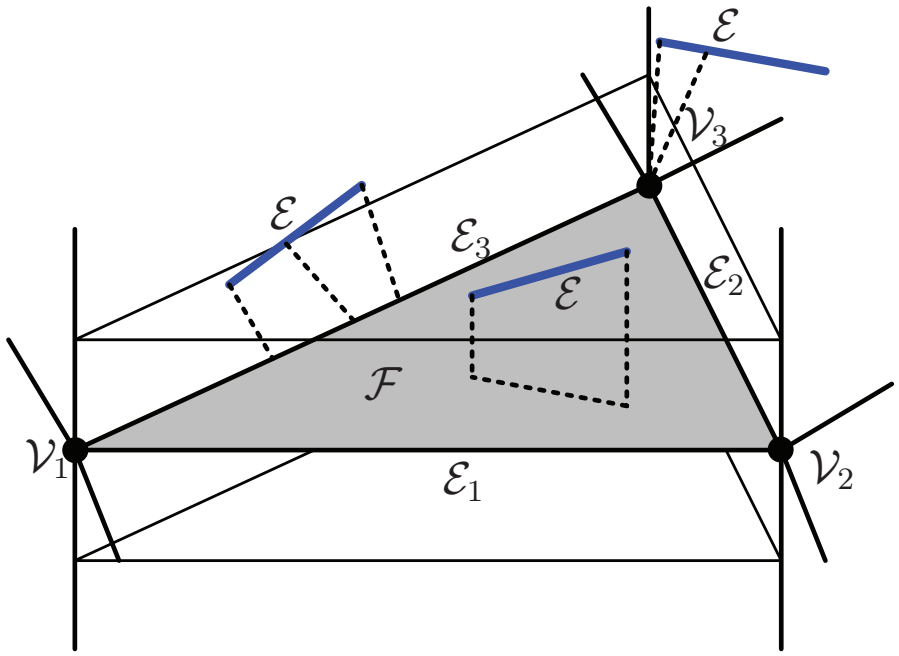


Figure 4.14.: The relationship between line segment and triangle

Table 4.1.: Collision detection algorithms

fun_lineseg_lineseg	**fun_lineseg_triangle**
Input: $\mathcal{E}, \mathcal{E}_0$	**Input:** $\mathcal{E}, \mathcal{T}$
Output: control pair (P_1, P_2)	**Output:** control pair (P_1, P_2)

pair = ∅, p_tmp = ∅;
for $\mathcal{X} \in \{\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}_0\}$
$\mathcal{Z}$ = VR_clip$(\mathcal{X}, \mathcal{E})$
If $\mathcal{Z} \in \{\mathcal{V}_1, \mathcal{V}_2\}$ do
p_tmp = p_tmp ∪ fun_point_edge$(\mathcal{Z}, \mathcal{E})$
else
p_tmp = p_tmp ∪ fun_edge_edge$(\mathcal{Z}, \mathcal{E})$
end
pair = min_max(p_tmp)
end
	pair = ∅, p_tmp = ∅;
	for $\mathcal{X} \in \{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{F}\}$
	$\mathcal{Z}$ = VR_clip$(\mathcal{X}, \mathcal{E})$
	If $\mathcal{Z} \in \{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3\}$ do
	p_tmp = p_tmp ∪ fun_point_edge$(\mathcal{Z}, \mathcal{E})$
	elseif $\mathcal{Z} \in \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\}$ do
	p_tmp = p_tmp ∪ fun_edge_edge$(\mathcal{Z}, \mathcal{E})$
	else
	p_tmp = p_tmp ∪ fun_face_edge$(\mathcal{Z}, \mathcal{E})$
	end
	pair = min_max(p_tmp)
	end

Once the fundamental detection algorithms are developed, for no strictly convex polyhedron, its interaction detection can be implemented by decomposing the polyhedron into a set of triangular faces with a hierarchical structure such as Oriented Bounding Box (OBB)-Tree. Then the algorithm proposed here can be used to calculate the two control points on different polyhedra. The algorithms will also be employed in section 5.3.2 as part of passive constraints integrated into Nonlinear Model Predictive Control (NMPC).

4.3.2. Collision Avoidance Strategy

As shown in Figure 4.15, let $\mathcal{P}$ and $\mathcal{O}$ represent robotic links and obstacles, respectively, and the pair (P_1, P_1') and (P_2, P_2') denotes the two control points regarding to the closest distance and MPD between $\mathcal{P}_2$ and $\mathcal{O}_1$. The collision avoidance strategy is to assign the control points P_1 and P_2 on the manipulator a motion component that moves the manipulator away from the obstacles. As depicted in Figure 4.15, the avoidance strategy is carried out only when the control points come into the neighbourhood of the obstacles. Here an influence distance d_{if} is defined to represent the range of activating avoidance strategy. From another aspect, the collision avoidance strategy should have minimum impact on the primary end-effector task. Since it is supposed that the manipulator is working in un-constructed and dynamic environment with obstacles, it is necessary to have some sensors used to detect the possible obstacles and implement the trajectory planning on-line.

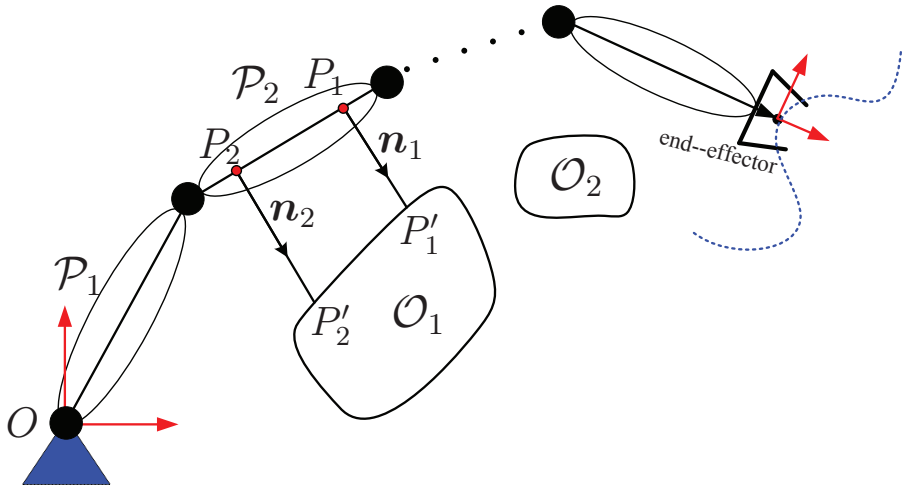


Figure 4.15.: Collision avoidance with two control points

We first consider one control point case. Refer to Maciejewski and Klein (1985), the primary task of end-effector $\dot{\boldsymbol{x}}_e$ and the secondary task of obstacle avoidance $\dot{\boldsymbol{x}}_o$ can be described by the following equations:

$$\begin{cases} \dot{\boldsymbol{x}}_e = \mathbf{J}_e \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{x}}_o = \mathbf{J}_o \dot{\boldsymbol{\theta}} \end{cases} \tag{4.27}$$

A common solution about multiple subtasks when task-priority is considered can be found in section 4.1.2. Here, a solution to meet both goals can be depicted as follows without

considering the null-space motion $\boldsymbol{h} = \mathbf{0}$:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_e^+ \dot{\boldsymbol{x}}_e + \bar{\mathbf{J}}_o^+ (\dot{\boldsymbol{x}}_o - \mathbf{J}_o \mathbf{J}_e^+ \dot{\boldsymbol{x}}_d) \tag{4.28}$$

where $\bar{\mathbf{J}}_o = \mathbf{J}_o(\mathbf{E}_n - \mathbf{J}_e^+ \mathbf{J}_e)$ is the projection of matrix $\bar{\mathbf{J}}_o$ on the null-space of $\mathbf{J}_e$. Each term in the proceeding equation has an simple interpretation. The first term $\mathbf{J}_e^+ \dot{\boldsymbol{x}}_e$ guarantees the implementation of the primary task (normally, $\dot{\boldsymbol{x}}_c = \dot{\boldsymbol{x}}_d + \mathbf{K}(\boldsymbol{x}_d - \boldsymbol{x}_e)$ is employed instead of $\dot{\boldsymbol{x}}_e$ for tracking stability reasons). The second term, which is projected on the null-space of the end-effector's Jacobian matrix, depicts the motion of the control point without influencing the motion of the end-effector. The term $\mathbf{J}_o \mathbf{J}_e^+ \dot{\boldsymbol{x}}_d$ describes the motion of control point induced by the end-effector motion, and the matrix $\bar{\mathbf{J}}_o$ which combines the null-space of the end-effector and the control point, is utilized to transform the assigned velocity of the control point from operational-space to the joint-space.

Considering the Cartesian space, if we want to activate the collision avoidance strategy and assign a velocity to the control point, then $\dot{\boldsymbol{x}}_o$ should be a 3 dimensional vector, which means $\mathbf{J}_o \in \mathbb{R}^{3\times n}$. This will require 3 additional DOR to complete the avoidance in a 3 dimensional Cartesian space. Nevertheless, not every manipulator has 3 DOR for the purpose of collision avoidance. This issue can be resolved by the geometric projection method as introduced in Zlajpah and Nemec (2002). In fact, the obstacle avoidance strategy only requires to control the escape velocity in the direction of the line connecting the control point with the correspondent point on the obstacle, this is just one dimensional constraint and one DOR would be sufficient to avoid the possible collision. The derivative of the distance $\|P_1 P_1'\|$ can be denoted by the following equation if the obstacle is steady:

$$\dot{d}_1 = \langle \mathbf{J}_o \dot{\boldsymbol{\theta}}, \boldsymbol{n}_1 \rangle = \langle \dot{\boldsymbol{\theta}}, \boldsymbol{n}_1^{\mathrm{T}} \mathbf{J}_o \rangle \tag{4.29}$$

$\boldsymbol{n}_1$ is the unit vector $\frac{P_1 P_1'}{\|P_1 P_1'\|}$ and notation $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ refers to the inner product of vector $\boldsymbol{u}$ and $\boldsymbol{v}$. Now we use $\mathbf{J}_d = \boldsymbol{n}_1^{\mathrm{T}} \mathbf{J}_o$ as a new Jacobian matrix to replace $\mathbf{J}_o$ in equation 4.28, then velocity $\dot{\boldsymbol{x}}_o$ and $\boldsymbol{n}_1^{\mathrm{T}} \mathbf{J}_o \mathbf{J}_e^+ \dot{\boldsymbol{x}}_d$ become scalar. Consequently, the calculating of Moore-Penrose pseudo-inverse of $\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o$ is much faster since now it can be expressed as follows:

$$(\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o)^+ = \bar{\mathbf{J}}_o^{\mathrm{T}} \boldsymbol{n}_1 (\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o \bar{\mathbf{J}}_o^{\mathrm{T}} \boldsymbol{n}_1)^{-1} \tag{4.30}$$

There is no need to inverse any matrix in computing the pseudo-inverse of $\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o$ since $\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o \bar{\mathbf{J}}_o^{\mathrm{T}} \boldsymbol{n}_1$ is scalar. This of course saves a lot of computation effort in the trajectory planning. Recall the equation 4.28, the new inverse kinematic solution to complete primary task and collision avoidance can be expressed by:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}_e^+ \dot{\boldsymbol{x}}_e + (\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o)^+ (\dot{x}_o - \boldsymbol{n}_1^{\mathrm{T}} \mathbf{J}_o \mathbf{J}_e^+ \dot{\boldsymbol{x}}_d) \tag{4.31}$$

A hidden issue of above solution is the possible singularity where the Jacobian matrix $\mathbf{J}_e$ or $\boldsymbol{n}_1^{\mathrm{T}} \bar{\mathbf{J}}_o$ lose its rank. When a singularity happens, the joint velocity will go to infinity which is not welcome. This problem can be tackled by employing the DLS method or the STR method developed in section 4.2.2.

In above analysis, only the nearest point between link and obstacle is taken into account, the switch between the control points results in bounce at some configurations when two objects are not strictly convex. However, we do not want to treat the control point P_2 as the third subtask. This is because normally for the simplicity and economical reasons, not so many DOR are available in a redundant manipulator. Hence, a better way would be,

choosing a new control point P_0 on the manipulator which combines the information of the two control points P_1 and P_2 for the purpose of collision avoidance. Furthermore, the new control point P_0 should be moved on the manipulator continuously. It can be calculated as follows:

$$P_0 = \frac{d_{if} - d_1}{2d_{if} - d_1 - d_2} P_1 + \frac{d_{if} - d_2}{2d_{if} - d_1 - d_2} P_2 \tag{4.32}$$

where $d_1 = \|P_1 P_1'\|$ and $d_2 = \|P_2 P_2'\|$ are the distance between P_1 and obstacle, P_2 and obstacle, respectively. The corresponding unit vector $\boldsymbol{n}_0$ for escaping from the obstacles is equal to $\boldsymbol{n}_1$ while the control point P_1 is more critical. The desired critical point velocity $\dot{x}_o = \alpha_c v_o$, v_o is the assigned nominal velocity and α_c is the obstacle avoidance gain which depends on the critical distance to the obstacles:

$$\alpha_c = \begin{cases} \left(\frac{d_{sr}}{d_1}\right)^2 - 1 & d_1 < d_{sr} \\ 0 & d_1 \geq d_{sr} \end{cases} \tag{4.33}$$

d_{sr} defines another critical distance named security distance. If the manipulator is too close to the obstacles $d_1 \leq d_{uf}$, the primary task $\dot{\boldsymbol{x}}_e$ should be cancelled which is controlled by turning parameter α_e. The unsafe distance d_{uf} can be a predefined quantity or be yielded according to the dynamic properties of the manipulator.

For smoother motion, the magnitude of $\dot{\boldsymbol{x}}_o$ should be 0 at d_{sr}. Then the solution for the inverse kinematics in conjunction with obstacles avoidance can be expressed as:

$$\dot{\boldsymbol{\theta}} = \alpha_e \mathbf{J}_e^+ \dot{\boldsymbol{x}}_e + \alpha_d \left(\mathbf{J}_d (\mathbf{E}_n - \mathbf{J}_e^+ \mathbf{J}_e)\right)^+ \left(\alpha_c v_o - \mathbf{J}_d \mathbf{J}_e^+ \dot{\boldsymbol{x}}_d\right) \tag{4.34}$$

The turning parameter α_d can be defined as follows:

$$\alpha_d = \begin{cases} 0 & d_1 \geq d_{if} \\ \frac{1}{2}\left(1 + \cos(\pi \frac{d_1 - d_{sr}}{d_{if} - d_{sr}})\right) & d_{if} \leq d_1 < d_{sr} \\ 1 & d_1 \leq d_{sr} \end{cases} \tag{4.35}$$

The functional behaviour of turning parameters α_c, α_d and α_e is shown in Figure 4.16.

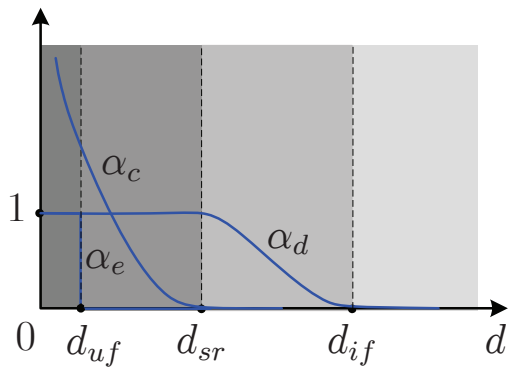


Figure 4.16.: The functional behaviour of the turning parameters for collision avoidance

From equation 4.34 and Figure 4.16, one can see that, after the control point P_1 enters into the influence zone, i.e. $d_1 < d_{if}$, the avoidance strategy begins to influence the configuration

of the manipulator. However, only part of the homogeneous solution is included. As the distance between control point and the obstacles decreases, more homogeneous solution is exerted which diminishes the relative velocity between control point and the obstacles from d_{if} to d_{sr}. Once the control point is in the range of the d_{sr}, a bounce velocity $\dot{\boldsymbol{x}}_o$ is assigned and the complete homogeneous solution is included. This avoidance strategy will assure the control point move out of the dangerous zone and guarantee the smooth transition even when the control point P_1 switches during the motion of the manipulator. Equation 4.34 is designed for a single obstacle, when there are more than one obstacle in the workspace, since the shortage of DOR, one can not treat different obstacles as multiple subtasks. If the equation 4.34 is still in use without modification, the frequent switching of the control points on the manipulator will also result in oscillation in joint velocities. This discontinuity of joint velocity is highly unwelcome since it maybe harmful for the structure and actuators of the manipulator. To improve the performance, a weighted sum of the homogeneous solution considering all the obstacles in the influence zone can be expressed by:

$$\dot{\boldsymbol{\theta}} = \alpha_e \mathbf{J}_e^{+} \dot{\boldsymbol{x}}_e + \sum_{i=1}^{n_o} w_i \alpha_{di} \left(\mathbf{J}_{di}(\mathbf{E}_n - \mathbf{J}_e^{+}\mathbf{J}_e)\right)^{+} \left(\alpha_{ci} v_o - \mathbf{J}_{di}\mathbf{J}_e^{+}\dot{\boldsymbol{x}}_d\right) \tag{4.36}$$

n_o is the number of the obstacles in the influence zone, the weighting factor w_i can be defined as:

$$w_i = \frac{d_{if} - d_{1i}}{\sum_{i=1}^{n_o}(d_{if} - d_{1i})} \tag{4.37}$$

The definition of weighting factor w_i shows the relative significance of the different obstacles in the influence zone. When one obstacle is much closer than the others, i.e. $d_{1i} \to 0$, then its corresponding weight factor increases and the velocity at that control point ramps up as well.

4.3.3. Simulation Results

To verify the effectiveness and performance of the proposed method in this chapter, simulation results are presented by using a 10 revolute planar manipulator. The primary task is to move the end-effector across a corridor. A schematic diagram of the simulation is illustrated in Figure 4.17. The widely used collision avoidance method in Zlajpah and Nemec (2002) is employed here with same turning parameters for comparison reason. For the simulation of our proposed collision avoidance method, the link length in Figure 4.17 is set to $l_1 = l_2 = \cdots = l_{10} = 0.1m$. The kinematic control gain $\mathbf{K} = \text{diag}(50, 50)$. The desired end-effector trajectory is $\boldsymbol{x}_d = [0.2266, 0.2539 + 0.11t]^{\mathrm{T}}$ during 5 seconds and the motion of the manipulator has been constrained by two obstacles. The simulation sample time is set to 10 milliseconds.

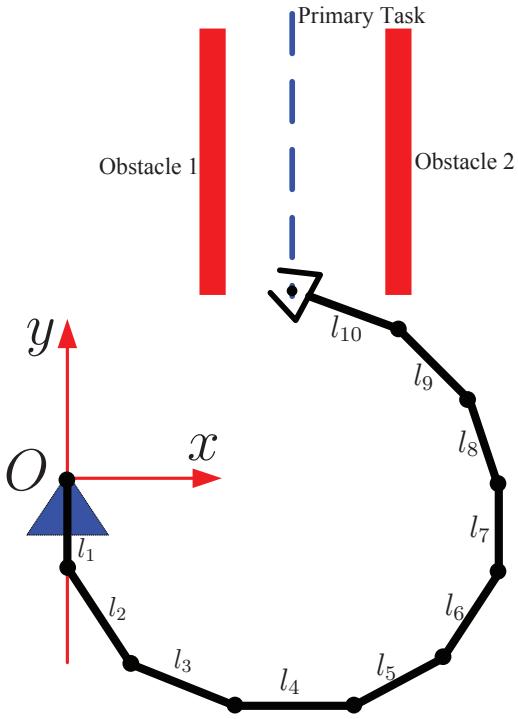


Figure 4.17.: 10 revolute planar manipulator and obstacles

The parameters of d_{if}, d_{sr}, d_{uf} are set to $0.1m$, $0.05m$ and $0.02m$. The assigned nominal velocity is $v_o = 5m/s$. The results of path tracking without control point, with one control point and with two control points are shown in Figure 4.18, Figure 4.19 and Figure 4.20, respectively. One can see that, when no collision avoidance strategy is applied, some links will collide with the second obstacle when the manipulator tracking the desired path. If one or two control points are considered, the manipulator can autonomously avoid the possible collision during its motion. The task-space tracking error with one control point, two control points and without control point are equally the same, which means, the successful collision avoidance strategy will not affect the primary task since it is completed in the null-space of the primary task's Jacobian matrix. No matter one control point or two, the minimum distance between different links and obstacles reveals the effectiveness of the avoidance strategies. The minimum distance between links and obstacles is constrained without moving into the predefined security zone d_{sr}. The main difference between the traditional collision avoidance method and the proposed method lies in the generation of joint velocity. One can see that, at $3.73s$, $4.56s$ and $4.80s$, the joint velocity endures a sharp change because of the control point switch. This discontinuity will result in large jerk in the acceleration level which is highly unexpected. While with the proposed method, in virtue of the autonomous regulation of control points, smoother joint velocities are generated without discontinuity. The proposed algorithm also provides us an intuitive way to understand and can be expanded to further application. The idea of regulating two control points will be adopted in section 5.3.2.

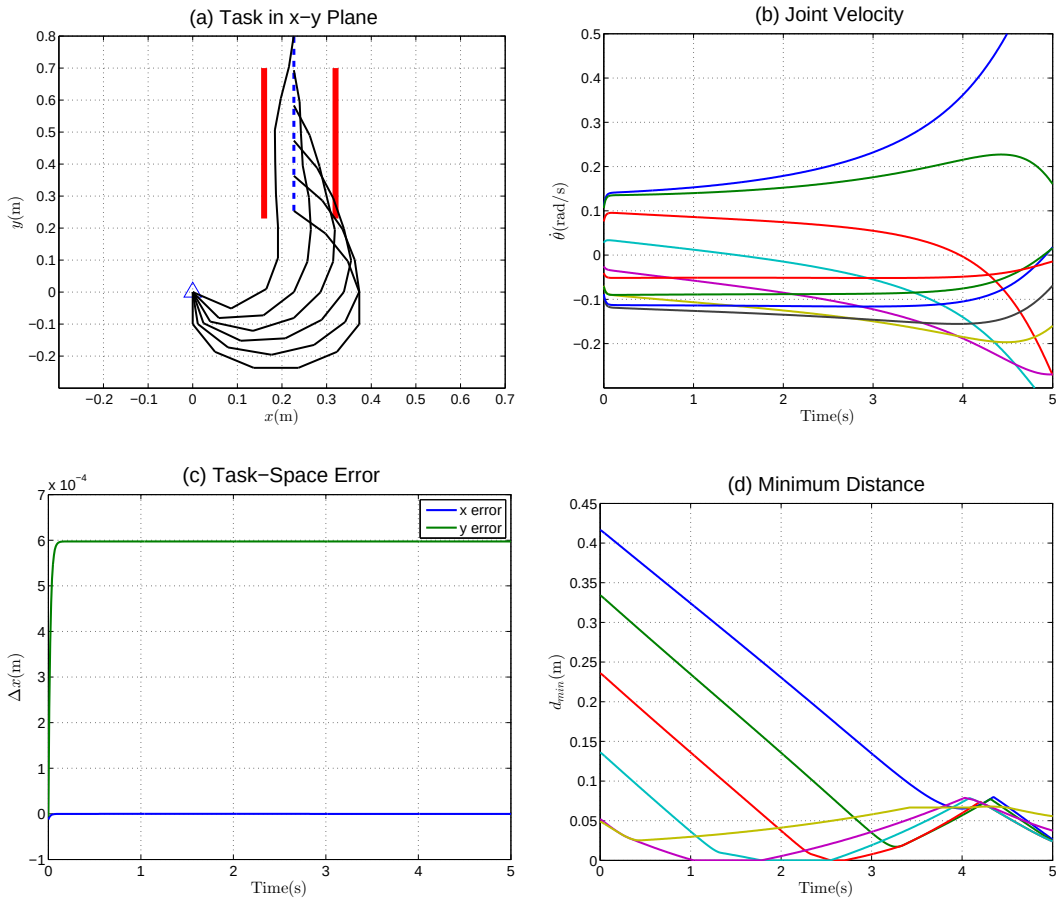


Figure 4.18.: Path tracking without collision avoidance

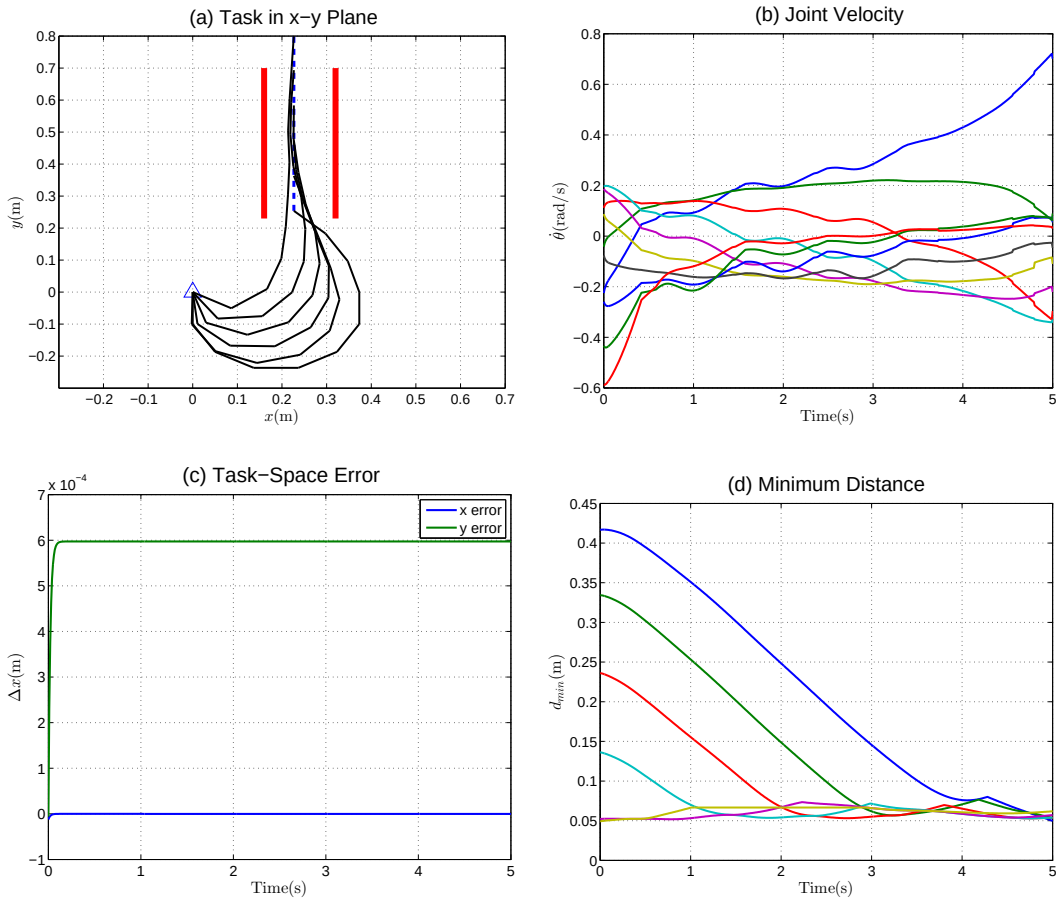


Figure 4.19.: Path tracking with one control point for collision avoidance

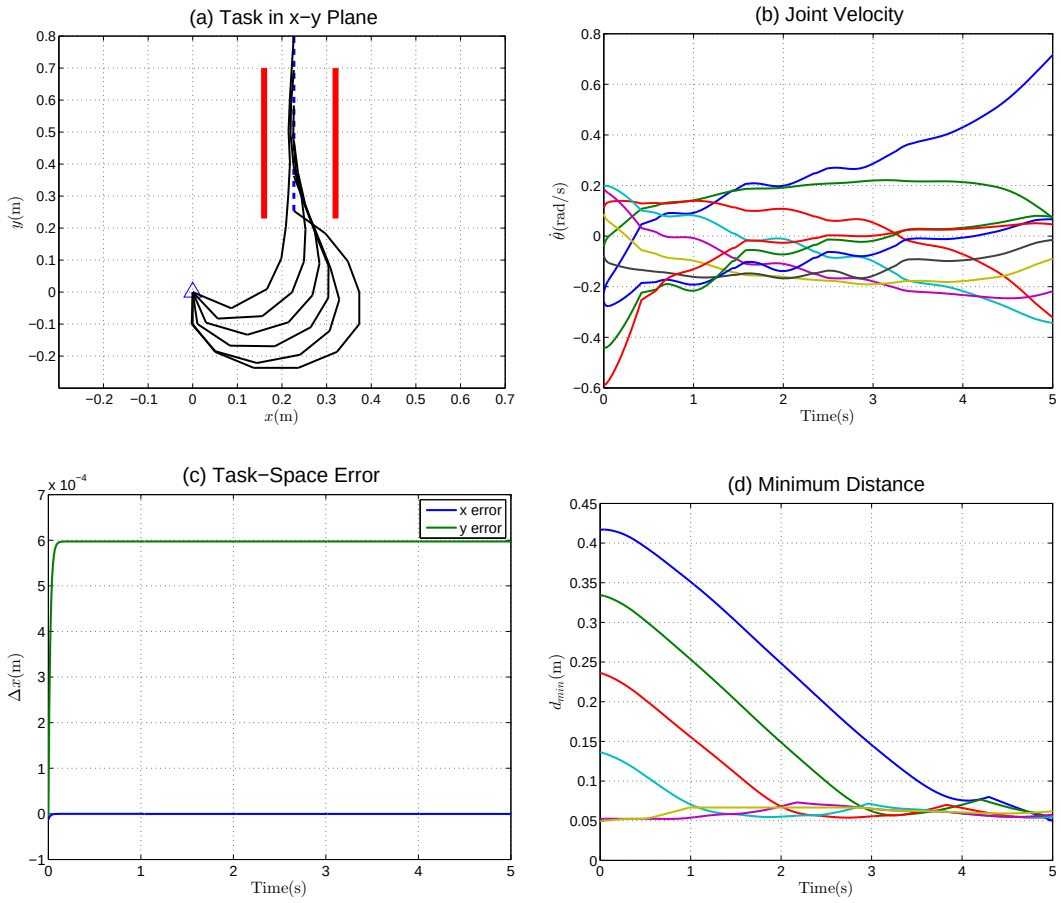


Figure 4.20.: Path tracking with two control points for collision avoidance

4.4. Summary

Two primary issues concerning for robotic manipulator have been resolved in this chapter based on the concept of task-priority and the pseudo-inverse redundancy resolution. A general recursive algorithm of inverse kinematics for multiple subtasks with different task-priorities is first presented and analysed. In view of the general inverse kinematics solution, singularity avoidance and collision avoidance are illustrated in this chapter and can be concluded as follows:

- The loss of independent DOF at singular position is an inherent property of articulated manipulators. A so-called Singular Task Reconstruction (STR) method based on the concept of velocity manipulability ellipsoid has been designed and applied for task-space path tracking. With the proposed STR method, less overshoot, more predictable minimum singular value and smaller & smoother joint velocities are achieved as compared to the typical Damped Least-Squares (DLS) approach. The simulation studies show significant improvements in single task or multiple subtasks with kinematic and algorithm singularities. With the intuitive geometric interpretation of manipulability ellipsoid, as a generalized and robust method, the STR method can be easily applied to non-redundant and redundant manipulators with different types of singularities.

- An inevitable issue during motion of space manipulator in its workspace is collision avoidance. Conventional methods have the demerits of oscillation of the manipulator due to the switch of the control point on the manipulator when the objects can not be treated as the strictly convex objects. An extra control point corresponding to the Maximum Projection Distance (MPD) together with the nearest point of the minimum distance is employed to suppress the possible fluctuation of the manipulator. The interaction test between line segment and line segment, rectangle, circle is implemented to provide the fully information of the two control points. An avoidance strategy is derived based on the general inverse kinematics solution using the two control points and the geometrical projection method. The new obstacle detection and avoidance strategy does not require too many DOR to complete the collision avoidance, even only 1 DOR can handle the collision avoidance problem with multiple obstacles. Simulation results demonstrate the performance of the proposed method when compared to the traditional collision avoidance algorithm.

5. Model Predictive Control

Authority in science exists to be questioned, since heresy is the spring from which new ideas flow.

—John C. Polanyi

This chapter investigates the application of Nonlinear Model Predictive Control (NMPC) to space manipulators in capturing an un-cooperative target satellite in space. How to control the motion of space manipulator with high accuracy and various constraints is a significant guarantee for the completion of the demanded space tasks. The fundamental idea and mathematical formulation of MPC is firstly introduced. In order to apply MPC to space robot, a feedback linearisation procedure is conducted to transform the non-linear system into a linear one. In addition, how to handle the different types of constraints in controller design is another primary concern. The constraints appear in practice imposed on decision variables are categorized into three classes: physical/security constraints, operational constraints and passive constraints. These constraints, together with varied priorities, constitute a new Quadratic Programming (QP) searching algorithm. A recursive state estimator regarding the minimum variance states estimation from Kalman Filter (KF) is derived. The controller has been implemented for a 7 DOF kinematically redundant manipulator installed on a 6 DOF free-floating satellite via simulation studies considering singularity and obstacles in the workspace of the space manipulator. Real-time end-effector approaching to target, path tracking and capturing target, particularly verify the effectiveness and prospect of the proposed NMPC strategy for space robot.

5.1. Model Predictive Control

MPC refers to a wide spread class of control strategies that apply an explicit model to predict the response of a plant. Originated from 1970s, MPC, also referred to RHC, has been widely employed in the field of chemical processing industries as an effective control strategy to deal with multi-variable constrained control problems. The interests in MPC and RHC started to surge only in the 1980s after exposition of the first papers on IDCOM Richalet et al. (1978) and DMC Cutler and Ramaker (1980), and the first comprehensive publication of GPC Clarke et al. (1987a,b). In the past two decades, various formulations about MPC have been developed for linear and non-linear systems Morari and H. Lee (1999), Qin and Badgwell (2003), that found successful applications especially in process industries Camacho and Bordons (2004), Maciejowski (2002).

5.1.1. Principle and Formulation

The fundamental principal of the MPC is fairly similar as our human interaction with the external environment. A prediction over finite horizon is always performed in our brains based on our knowledge about the environment, i.e. modelling of our environment, together with the real-time perception of the environment. An on-line open-loop optimization procedure employing the current (estimated) state as a new start point is conducted considering various constraints, such as our motion ability, without collision, etc. Subsequently, a decision will be made by our brains to determine an optimal solution of current situation. These procedures are performed all the time when we have some interactions with the others and the external environments. This predication-optimization-action pattern is also the cornerstone of the MPC strategy. A schematic diagram of the fundamental principle of the MPC can be depicted in Figure 5.1.

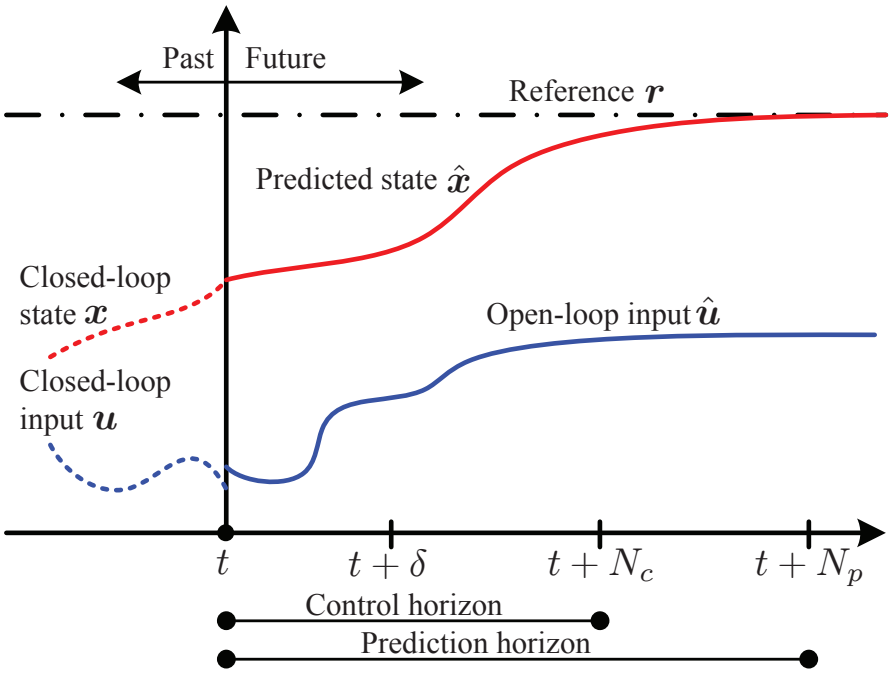


Figure 5.1.: Schematic diagram of MPC fundamental principle

As illustrated before, the fundamental idea of MPC is using a prediction model and numerical optimization methods on-line to obtain a sequence of control inputs that minimizes a pre-defined cost function over a finite time horizon, while subjects to certain constraints concurrently. Choosing moment t as a start point to analyse, at instant t, concerning new obtained measurements and the current system states estimated by the additional designed observer, the controller begins to predict the future dynamic behaviour of the system over a prediction horizon N_p, next, the optimal input over a control horizon N_c ($N_c \leq N_p$) is determined through optimizing a pre-determined open-loop cost function Γ. If there are no disturbance and no model mismatch, when N_p goes into infinite, a global optimal solution about control input will be generated. Then one can apply the control input sequence generated at $t = 0$ to the system and obtain an optimal performance. However, this is only an nominal case used to analyse which is hardly happen in practice. The external disturbances,

inaccuracy in modelling/identification will always exist in a real system which aggravate the difficulty of controller design.

Due to the external/internal disturbance and model-plant mismatch,the dynamic behaviour of the system prediction will differ from the true system. In order to incorporate feedback mechanism, only the first control input obtained through open-loop optimization will be adopted until the next measurement becomes available. When a new measurement is obtained at the time $t+\delta$. Such procedure: measurement-estimation-prediction-optimization will be repeated to search for the new optimal control input sequence in a "receding horizon "manner. Another aspect should be pay attention is the control input. In above analysis, the control input will be generated through an on-line optimization expressed in arbitrary function, one can also employ some basic functions to represent the control input as people in the Fourier transformation does. In Wang (op. 2009) Laguerre function is used as the basic function to construct the control input series. The basic NMPC control loop is shown in Figure 5.2.

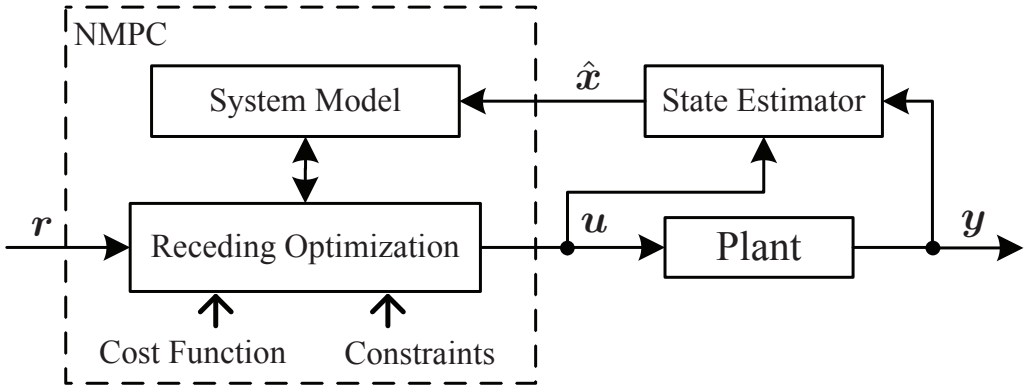


Figure 5.2.: Basic NMPC control loop

The response and output of a continuous time system with multiple states can be described by non-linear differential equations in a vector form:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}_c(\boldsymbol{x}) + \boldsymbol{g}_c(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{h}_c(\boldsymbol{x}, \boldsymbol{u}) \end{cases} \tag{5.1}$$

Continuous time system can be used to analyse while in simulation, a discrete time system has to be employed which can be depicted as follows:

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{f}_d(\boldsymbol{x}(k)) + \boldsymbol{g}_d(\boldsymbol{x}(k))\boldsymbol{u}(k) \\ \boldsymbol{y}(k+1) = \boldsymbol{h}_d(\boldsymbol{x}(k+1), \boldsymbol{u}(k+1)) \end{cases} \tag{5.2}$$

Normally, system outputs and control input can not have infinite value, they must yield some boundary constraints where $\boldsymbol{u}(t) \in \mathcal{U}$ and $\boldsymbol{y}(t) \in \mathcal{Y}$. In addition, when a system is operating in an environment, especially for a mechanical system, how to avoid the possible obstacles in workspace could constitute another class of constraints. Considering all these constraints, the general NMPC algorithm can be described from a theoretical point of view:

$$
\begin{aligned}
&\boldsymbol{u} = \arg_{\boldsymbol{u}} min\ \Gamma(k) \\
&\text{subject to} \\
&\begin{cases}
\boldsymbol{x}(k|k) = \boldsymbol{x}(k) \\
\boldsymbol{u}(k+j|k) = \boldsymbol{u}(k+N_c|k), j \geq N_c \\
\boldsymbol{x}(k+j+1|k) = \boldsymbol{f}_d(\boldsymbol{x}(k+j|k), \boldsymbol{u}(k+j|k)) \\
\boldsymbol{y}(k+j|k) = \boldsymbol{h}_d(\boldsymbol{x}(k+j|k), \boldsymbol{u}(k+j|k)) \\
\boldsymbol{y}_{min} \leq \boldsymbol{y}(k+j|k) \leq \boldsymbol{y}_{max} \\
\boldsymbol{u}_{min} \leq \boldsymbol{u}(k+j|k) \leq \boldsymbol{u}_{max} \\
\boldsymbol{g}(\boldsymbol{x}(k+j|k)) \leq \boldsymbol{0}
\end{cases}
\end{aligned}
\tag{5.3}
$$

where $j \in [0, N_p - 1]$. $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{u}$ represent system states, outputs and control inputs, respectively. The notation $\boldsymbol{a}(i|j)$ describes the value of vector $\boldsymbol{a}$ at the instant i predicted at the instant $j (j < i)$. The two functions, $\boldsymbol{f}_d(\boldsymbol{x}(k+j|k), \boldsymbol{u}(k+j|k))$ and $\boldsymbol{h}_d(\boldsymbol{x}(k+j|k), \boldsymbol{u}(k+j|k))$ stand for the discrete prediction model and measurement model, while the control input $\boldsymbol{u}$ and system output $\boldsymbol{y}$ yield to their corresponding lower and upper bound. The inequality equation $\boldsymbol{g}(\boldsymbol{x}(k+j|k)) \leq \boldsymbol{0}$ represents additional constraints, such as security constraints, terminal constraints, *etc*. The cost function $\Gamma(k)$ is a scalar function which is usually determined in terms of the control effort and the derivation between the predictive and the desired outputs. The optimization procedure must be solved at each sampling time k to obtain a sequence of optimal control inputs as $\{\boldsymbol{u}^*(k+1|k), \cdots, \boldsymbol{u}^*(k+N_c|k)\}$. When different constraints are taken into account during optimization, a QP procedure, as will be presented in section 5.4 must be employed to gain the optimal control inputs. When the optimal solution $\boldsymbol{u}^*(k)$ to the optimization issue described in equation 5.3 exists, the open-loop optimization will be proceeded at each sampling instant, the the nominal closed-loop system employing the optimal control input can be described by:

$$
\begin{cases}
\boldsymbol{x}(k+1) = \boldsymbol{f}_d(\boldsymbol{x}(k)) + \boldsymbol{g}_d(\boldsymbol{x}(k))\boldsymbol{u}^*(k) \\
\boldsymbol{y}(k+1) = \boldsymbol{h}_d(\boldsymbol{x}(k+1), \boldsymbol{u}^*(k+1))
\end{cases}
\tag{5.4}
$$

5.1.2. MPC Properties

In Mayne et al. (2000), the relationship between MPC and traditional optimal control is illustrated. It shows, MPC essentially solves the standard optimal control issue in a finite horizon. The main difference between MPC and other controllers is, MPC determines the optimal solution on-line for the current plant, rather than solve the control law off-line. When an open-loop prediction horizon is used, the trajectory of the input and output of the closed-loop system will differ from the predicted trajectories, even there are no disturbance and model mismatch. This is because the plant dynamics behaviour is predicted at the current sampling instant only for a certain time interval. The phenomenon is described in Figure 5.3 where the shade area is the feasible region defined by the constraints.

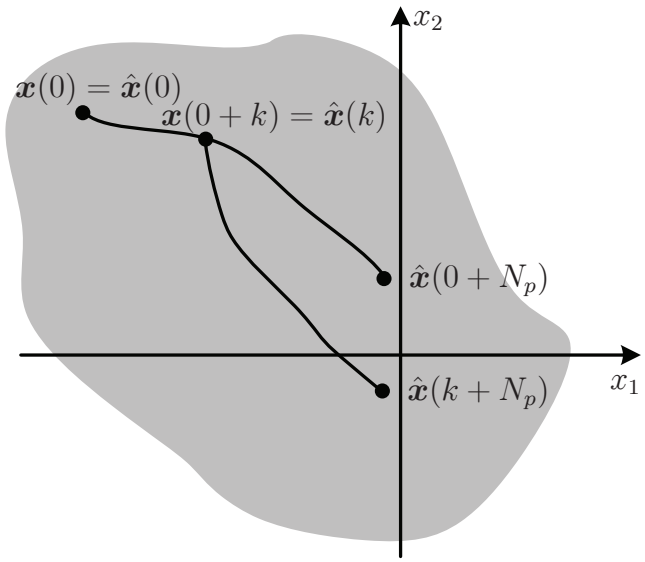


Figure 5.3.: The mismatch between open-loop prediction and closed-loop behaviour

The key difference of MPC and standard controller design makes MPC possesses a series of properties:

- An explicit non-linear model is required in NMPC for prediction;
- Various constraints such as input, output, collision, *etc* can be integrated into NMPC;
- A specialized cost function can be predisposed and optimized on-line in NMPC;
- An on-line optimization of the open-loop control issue is necessary for the application of NMPC;
- An open-loop prediction in NMPC differs from the closed-loop behaviour.

The characteristics of NMPC show us its advantages in industrial applications, such as the ability of constraints handling, multi-variable controller design and on-line optimization. However, on the other side, it also imports two consequences: the first one is the performance objective function to be minimized on-line over finite horizon does not guarantee the optimal solution of closed-loop system behaviour; the second problem is the closed-loop system maybe un-stable because of the difference between finite horizon prediction and closed-loop behaviour.

5.2. NMPC Applied to Space Robot

As illustrated in section 2.3, the application of NMPC in the field of space robotics is really rare. In Hirzinger et al. (1989), Inaba and Oda (2000), Oda et al. (1996), the prediction information was employed to assist the operators to complete the tele-operation without involving the control loop. In this chapter, NMPC will be applied to a free-floating space robot trying to capture a tumbling target satellite as shown in Figure 5.4.

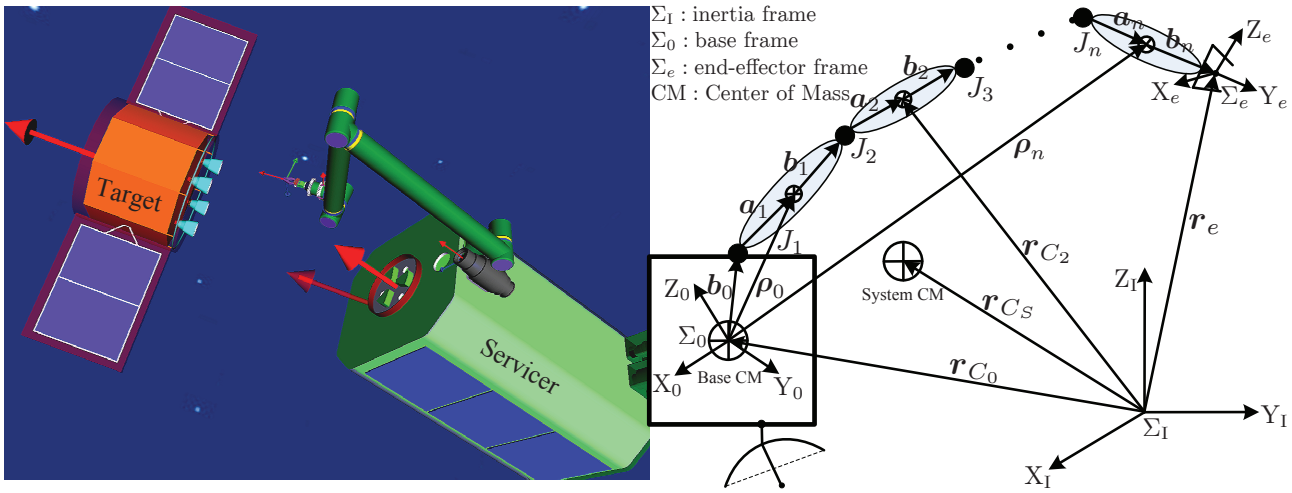


Figure 5.4.: Schematic diagram of space robot

5.2.1. Free-Floating Space Robot

The dynamics of space robot with general structure is introduced in chapter 3 and expressed by equation 3.17. Before a contact between servicer and target occurs, there is no external forces applied on the end-effector, i.e. $\boldsymbol{f}_e = \mathbf{0}$. If no actuators are activated to regulate the servicer satellite position and attitude, $\boldsymbol{f}_b = \mathbf{0}$, the whole space robot system (base and manipulator) is under the free-floating mode. The motion of the manipulator and the base is intricately coupled and governed by the law of momentum conservation. Let $\boldsymbol{l}_0 \in \mathbb{R}^{6\times 1}$ be the initial momentum of the system with respect to the inertial frame, it can be expressed by:

$$\boldsymbol{l}_0 = \mathbf{H}_b\dot{\boldsymbol{x}}_b + \mathbf{H}_{bm}\dot{\boldsymbol{\theta}} = \mathbf{0} \tag{5.5}$$

By substituting equation 5.5 into the kinematic mapping of the end-effector $\dot{\boldsymbol{x}}_e = \mathbf{J}_b\dot{\boldsymbol{x}}_b + \mathbf{J}_e\dot{\boldsymbol{\theta}}$, the velocity of the end-effector can be given as follows:

$$\dot{\boldsymbol{x}}_e = \mathbf{J}_g\dot{\boldsymbol{\theta}} = \mathbf{J}_e\dot{\boldsymbol{\theta}} - \mathbf{J}_b\mathbf{H}_b^{-1}\mathbf{H}_{bm}\dot{\boldsymbol{\theta}} \tag{5.6}$$

where $\mathbf{J}_g \in \mathbb{R}^{6\times n}$ is termed the Generalized Jacobian Matrix (GJM) in Umetani and Yoshida (1989), Yoshida (1997). The dynamics formulation of free-floating space robot can be described by:

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{c}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) \tag{5.7}$$

where $\mathbf{H}(\boldsymbol{\theta}) = \mathbf{H}_m - \mathbf{H}_{bm}^{\mathrm{T}}\mathbf{H}_b^{-1}\mathbf{H}_{bm} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{c}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) = \boldsymbol{c}_m - \mathbf{H}_{bm}^{\mathrm{T}}\mathbf{H}_b^{-1}\boldsymbol{c}_b \in \mathbb{R}^{n\times 1}$ are the generalized inertia matrix and the non-linear force and torque vector for free-floating space robot, respectively.

5.2.2. Feedback Linearization

Continuous or discrete non-linear model expressed in equation 5.1 and 5.2 can be employed to predict the future behaviour of the plant, however, it is still hard to integrate this non-linear model for NMPC design. A feedback linearisation procedure is conducted through

non-linear coordinate transform and non-linear state feedback to obtain a local linear model. Consider the continuous time state-space model expressed in equation 5.1, using Taylor expansion at the point $(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{y}_0)$, a so-called Jacobian linearisation method can be obtained:

$$\begin{cases} \dot{\boldsymbol{x}} = \left(\frac{\partial \boldsymbol{f}_c(\boldsymbol{x}_0)}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{g}_c(\boldsymbol{x}_0)}{\partial \boldsymbol{x}} \boldsymbol{u}_0 \right) (\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{g}_c(\boldsymbol{x}_0)(\boldsymbol{u} - \boldsymbol{u}_0) \\ \boldsymbol{y} - \boldsymbol{y}_0 = \frac{\partial \boldsymbol{h}_c(\boldsymbol{x}_0)}{\partial \boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_0) \end{cases} \tag{5.8}$$

One should note that Jacobian method is an exact approximation of the non-linear model at $(\boldsymbol{x}_0, \boldsymbol{u}_0)$, however, a control law based on the Jacobian model at such point may yield degraded performance and robustness problems at other points.

Feedback linearisation provides us a powerful technical tool to convert the original non-linear model to an exact linear one over a large set of operation conditions. There are two methods for feedback linearisation, one is input-output linearisation, the other is state-space linearisation. Both approaches rely on two operations: non-linear coordinate transformation and non-linear state feedback. In the input-output linearisation, the map between the actual output and the transformed input is linearised. When the dimension of transformed states variables is less than the system states, input-output linearisation method is restricted to use only the so-called zero dynamics are stable. The goal of state-space linearisation is to linear the map between the entire system states and the transformed inputs. This can be achieved when the dimension of the transformed states equals to the system states. A linearised controller can be then synthesized for the new linear input-state model.

Considering an n-dimensional non-linear system as depicted in equation 5.1 without using the subscript, if we assume $\boldsymbol{\beta} = \boldsymbol{\Phi}(\boldsymbol{x})$ and the artificial output $\hat{\boldsymbol{y}} = \hat{\boldsymbol{h}}(\boldsymbol{x})$, then the new coordinate β_k through transformation can be obtained by the following expression:

$$\boldsymbol{\beta}_k = \boldsymbol{\Phi}_k(\boldsymbol{x}) = L_{\boldsymbol{f}}^{k-1} \hat{\boldsymbol{h}}(\boldsymbol{x}) \quad 1 \leq k \leq n \tag{5.9}$$

The system model represented by the new coordinates can be described by:

$$\begin{aligned} \dot{\boldsymbol{\beta}}_1 &= \boldsymbol{\beta}_2 \\ \dot{\boldsymbol{\beta}}_2 &= \boldsymbol{\beta}_3 \\ &\vdots \quad \vdots \\ \dot{\boldsymbol{\beta}}_n &= \boldsymbol{b}(\boldsymbol{\beta}) + \boldsymbol{a}(\boldsymbol{\beta})\boldsymbol{u} \\ \hat{\boldsymbol{y}} &= \boldsymbol{\beta}_1 \end{aligned} \tag{5.10}$$

where $L_f^{k-1}\boldsymbol{g}(\boldsymbol{x})$ is the Lie derivative of $\boldsymbol{g}(\boldsymbol{x})$. $\boldsymbol{a}(\boldsymbol{\beta}) = L_{\boldsymbol{g}} L_f^{n-1} \hat{\boldsymbol{h}}[\boldsymbol{\Phi}^{-1}(\boldsymbol{\beta})]$ and $\boldsymbol{b}(\boldsymbol{\beta}) = L_f^n \hat{\boldsymbol{h}}[\boldsymbol{\Phi}^{-1}(\boldsymbol{\beta})]$. If the function $\boldsymbol{a}(\boldsymbol{\beta}) \neq \boldsymbol{0}$ throughout the operation region, the new control input can be expressed by:

$$\boldsymbol{u} = \frac{\dot{\boldsymbol{\beta}}_n - \boldsymbol{b}(\boldsymbol{\beta})}{\boldsymbol{a}(\boldsymbol{\beta})} \tag{5.11}$$

As a result, the new coordinate transformation $\boldsymbol{\beta} = \boldsymbol{\Phi}(\boldsymbol{x})$ linearises the map between the transformed input and each of the system states.

In our NMPC law design, in order to obtain the linear system model, above state-space feedback linearisation is proceed due to the existence of the highly non-linear effect and coupling terms in equation 5.7. The designer can then devise an outer loop control under the

new framework to meet the conventional controller design requirements, such as tracking error, model mismatch, and so forth. Obtaining such a linearised controller is guaranteed by the specific form of space robot dynamics since the inertia matrix $\mathbf{H}(\boldsymbol{\theta})$ can be inverted at any robot configuration. Through non-linear state feedback of equation 5.7, the control input $\boldsymbol{\tau}$ under linearised controller can be denoted by:

$$\boldsymbol{\tau} = \hat{\mathbf{H}}(\boldsymbol{\theta})\boldsymbol{u} + \hat{\boldsymbol{c}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \tag{5.12}$$

where $\boldsymbol{u} = \ddot{\boldsymbol{\theta}}$ stands for a new input vector. $\hat{\mathbf{H}}(\boldsymbol{\theta})$ and $\hat{\boldsymbol{c}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ are the estimation of inertia matrix $\mathbf{H}(\boldsymbol{\theta})$ and non-linear force vector $\boldsymbol{c}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, respectively. In the ideal case, i.e. without any model mismatch, $\hat{\mathbf{H}} = \mathbf{H}$ and $\hat{\boldsymbol{c}} = \boldsymbol{c}$. Therefore we obtain a linear and decoupled system with simple second-order differential dynamics. if we choose $\boldsymbol{x} = [\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}]^{\mathrm{T}}$ as the system states, the linear form of dynamic equation 5.7 can be given by:

$$\begin{cases} \dot{\boldsymbol{x}} = \mathbf{A}_c\boldsymbol{x} + \mathbf{B}_c\boldsymbol{u} \\ \boldsymbol{y} = \mathbf{C}_c\boldsymbol{x} + \mathbf{D}_c\boldsymbol{u} \end{cases} \tag{5.13}$$

where $\mathbf{A}_c = [\mathbf{0}_n, \mathbf{E}_n; \mathbf{0}_n, \mathbf{0}_n]$, $\mathbf{B}_c = [\mathbf{0}_n, \mathbf{E}_n]^{\mathrm{T}}$, $\mathbf{C}_c = \mathbf{E}_{2n}$, and $\mathbf{D}_c = [\mathbf{0}_n, \mathbf{0}_n]^{\mathrm{T}}$. The new control input $\boldsymbol{u} = \hat{\mathbf{H}}^{-1}(\boldsymbol{\tau} - \hat{\boldsymbol{c}})$. With the aforementioned linear form dynamics of space robot, a NMPC strategy with linear feedback for space robot is shown in Figure 5.5.

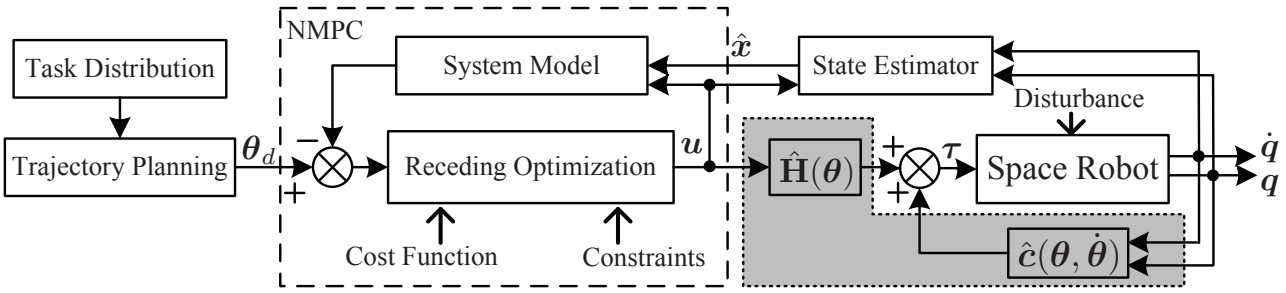


Figure 5.5.: Linear feedback with NMPC for space robot

5.2.3. Observer Design

In this chapter, state-space feedback linearisation technique is used to generate a corresponding linear system model. The current state (or state estimation) is applied as an initial condition to predict the system dynamic behaviour and integrate into the on-line optimization procedure to get an optimal control solution. Normally, since full states of the non-linear system can't be directly measured in most cases, how to reconstruct the system states from the current observations will be of the primary concern in this section. The method of estimating un-known system states based on the measurement, in a cybernetics context, is termed an observer. A system state observer is not inherently included in the implementation of NMPC which gives us free choice in design of state observer.

Consider the discrete linear system expressed in equation 5.13 with additional disturbance to the system states and measurements. The discrete linear system with disturbance

can be expressed by

$$\begin{cases} \boldsymbol{x}(k+1) = \mathbf{A}_d\boldsymbol{x}(k) + \mathbf{B}_d\boldsymbol{u}(k) + \boldsymbol{w}(k) \\ \boldsymbol{y}(k) = \mathbf{C}_d\boldsymbol{x}(k) + \boldsymbol{v}(k) \end{cases} \tag{5.14}$$

where $\boldsymbol{w}(k) \in \mathbb{R}^{2n\times 1}$ and $\boldsymbol{v}(k) \in \mathbb{R}^{n\times 1}$ are the disturbance vector to state and measurements, respectively. The state disturbance vector models un-determined disturbance to the system behaviour that affects the system states, while the measurement disturbance vector represents the measuring error caused by the sensor noise. The covariance matrix of $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ can be defined by:

$$\begin{cases} E\{\boldsymbol{w}(k)\ \boldsymbol{w}(j)^{\mathrm{T}}\} = \mathbf{\Lambda}\delta(k-j) \\ E\{\boldsymbol{v}(k)\ \boldsymbol{v}(j)^{\mathrm{T}}\} = \mathbf{\Gamma}\delta(k-j) \end{cases} \tag{5.15}$$

where $\delta(t)$ is the Dialac function. If the pair $(\mathbf{C}_d, \mathbf{A}_d)$ is observable, a sequential or recursive solution using the information only from the previous sample time can be constructed:

$$\begin{cases} \hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}(k|k-1) + \mathbf{K}_{ob}(k)(\boldsymbol{y}(k) - \mathbf{C}_d\hat{\boldsymbol{x}}(k|k-1)) \\ \hat{\boldsymbol{x}}(k|k-1) = \mathbf{A}_d\hat{\boldsymbol{x}}(k-1|k-1) + \mathbf{B}_d\boldsymbol{u}(k-1) \end{cases} \tag{5.16}$$

The system state estimations come from the predicted state estimation and the current output measurement. The differences between the predicted output $\mathbf{C}_d\hat{\boldsymbol{x}}(k|k-1)$ and the current output $\boldsymbol{y}(k)$ multiply the gain matrix $\mathbf{K}_{ob}(k)$ forms a correction to the predicted state estimation. How to design the gain matrix in terms of various criteria results in different state observers. Common pole placement or the Luenberger observer Maciejowski (2002), Wang (op. 2009) can be used to achieve required performance. However, if the poles are set too small, the rapid decay of the reconstruction error is satisfied, otherwise, it also amplify the measurement noise and modelling error. Therefore, the chosen of gain matrix $\mathbf{K}_{ob}$ and the poles of the observer must be very careful.

In this chapter, a Kalman Filter (KF) method Huang et al. (2009), Lee and Ricker (1994) of state estimation which takes measurement noise and modelling error into account is employed. It is a minimum mean square error method which uses series of observed measurement to produce more accurate estimates of system states containing noise. Consider the state disturbance and measurement noise expressed in equation 5.14, if we also assume that the initial state $\hat{\boldsymbol{x}}(0)$ is an independent, normally distributed random variable with covariance $\mathbf{P}(0)$, the recursive estimator in equation 5.16 produces state estimation with minimum variance. In the probabilistic formulation, the covariance matrix depicts the expected magnitudes of the disturbance to the system state and measurements. The ratio between the covariance of the system state disturbance and measurement noise will undoubtedly affect the determination of the gain matrix $\mathbf{K}_{ob}$. If the covariance of measurement noise is relatively large, the measurements are relatively uncertain and the feedback correction to the model prediction should be small. On the contrary, the measurements are relatively certain and should have more weight of the feedback correction. Consequently, the Kalman Filter gain $\mathbf{K}_{ob}(k)$ at sampling time k can be denoted as:

$$\mathbf{K}_{ob}(k) = \mathbf{P}(k)\mathbf{C}_d^{\mathrm{T}}(\mathbf{C}_d\mathbf{P}(k)\mathbf{C}_d^{\mathrm{T}} + \mathbf{\Gamma})^{-1} \tag{5.17}$$

The covariance matrix $\mathbf{P}(k)$ is propagated using the discrete filtering Riccati equation:

$$\mathbf{P}(k+1) = \mathbf{A}_d\mathbf{P}(k)\mathbf{A}_d^{\mathrm{T}} + \mathbf{\Lambda} - \mathbf{A}_d\mathbf{P}(k)\mathbf{C}_d^{\mathrm{T}}(\mathbf{C}_d\mathbf{P}(k)\mathbf{C}_d^{\mathrm{T}} + \mathbf{\Gamma})^{-1}\mathbf{C}_d\mathbf{P}(k)\mathbf{A}_d^{\mathrm{T}} \tag{5.18}$$

The initial covariance matrix $\mathbf{P}(0)$ can be determined by:

$$E\{[\boldsymbol{x}(0) - \hat{\boldsymbol{x}}(0)][\boldsymbol{x}(0) - \hat{\boldsymbol{x}}(0)]^{\mathrm{T}}\} = \mathbf{P}(0) \tag{5.19}$$

In the filtering Riccati equation, the first two terms $\mathbf{A}_d\mathbf{P}(k)\mathbf{A}_d^{\mathrm{T}} + \mathbf{\Lambda}$ represent a priori predicted estimate covariance of the system states at sampling time k. The remaining term describes the contribution of the output measurement in updating (a posteriori) estimate covariance, which generally decreases the covariance. As was illustrated before, if the pair $(\mathbf{C}_d, \mathbf{A}_d)$ is observable, $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ are positive definite, and $\mathbf{P}(0)$ is non-negative definite, as k goes to infinite, the recursive in equation 5.18 tends to a constant matrix.

$$\mathbf{P}(\infty) = \mathbf{A}_d(\mathbf{P}(\infty) - \mathbf{P}(\infty)\mathbf{C}_d^{\mathrm{T}}(\mathbf{C}_d\mathbf{P}(\infty)\mathbf{C}_d^{\mathrm{T}} + \mathbf{\Gamma})^{-1}\mathbf{C}_d\mathbf{P}(\infty))\mathbf{A}_d^{\mathrm{T}} + \mathbf{\Lambda} \tag{5.20}$$

This matrix is termed as steady-state discrete Raccati matrix. The KF gain $\mathbf{K}_{ob}(\infty)$ at steady-state can be then expressed by:

$$\mathbf{K}_{ob}(\infty) = \mathbf{P}(\infty)\mathbf{C}_d^{\mathrm{T}}(\mathbf{C}_d\mathbf{P}(\infty)\mathbf{C}_d^{\mathrm{T}} + \mathbf{\Gamma})^{-1} \tag{5.21}$$

Like in the design of Luenberger observer, all the eigenvalues of $(\mathbf{A}_d - \mathbf{K}_{ob}(\infty)\mathbf{C}_d)$ are guaranteed to be smaller than 1, which indicates, a nominal stable state observer is thereof well designed. The state estimator introduced in this section will be used in the following sections to provide a more accurate estimation of system states.

5.2.4. Optimization Index

According to the NMPC algorithm expressed in equation 5.3, an appropriate cost function $\Gamma(k)$ must be chosen to obtain the local optimal control law. Generally, the cost function not only should include direct relation with the tracking error between the predictive controlled output $\hat{\boldsymbol{y}}(k+i|k)$ and the reference trajectory $\boldsymbol{r}(k+i|k)$, but also should contain the control inputs effort $\Delta\hat{\boldsymbol{u}}(k+i|k)$. Then a cost function with quadratic form can be described by:

$$\Gamma(k) = \sum_{i=1}^{N_p} \|\hat{\boldsymbol{y}}(k+i|k) - \boldsymbol{r}(k+i|k)\|^2_{\mathbf{Q}(i)} + \sum_{i=0}^{N_c-1} \|\Delta\hat{\boldsymbol{u}}(k+i|k)\|^2_{\mathbf{T}(i)} \tag{5.22}$$

where $\mathbf{Q}(i)$ and $\mathbf{T}(i)$ are the weight matrices of tracking error and control effort, respectively. If we define $\boldsymbol{Y}(k) = [\hat{\boldsymbol{y}}(k+1|k), \cdots, \hat{\boldsymbol{y}}(k+N_p|k)]^{\mathrm{T}}$, $\boldsymbol{R}(k) = [\boldsymbol{r}(k+1|k), \cdots, \boldsymbol{r}(k+N_p|k)]^{\mathrm{T}}$, and $\Delta\boldsymbol{U}(k) = [\Delta\hat{\boldsymbol{u}}(k|k), \cdots, \Delta\hat{\boldsymbol{u}}(k+N_c-1|k)]^{\mathrm{T}}$, then the cost function $\Gamma(k)$ can be written as follows:

$$\Gamma(k) = \|\boldsymbol{Y}(k) - \boldsymbol{R}(k)\|^2_{\mathbf{Q}} + \|\Delta\boldsymbol{U}(k)\|^2_{\mathbf{T}} \tag{5.23}$$

where $\mathbf{Q} = diag([\mathbf{Q}(1), \cdots, \mathbf{Q}(N_p)])$ and $\mathbf{T} = diag([\mathbf{T}(1), \cdots, \mathbf{T}(N_c-1)])$. Here if we define:

$$\boldsymbol{Y}(k) = \mathbf{\Phi}\boldsymbol{x}(k) + \mathbf{\Upsilon}\boldsymbol{u}(k-1) + \mathbf{\Theta}\Delta\boldsymbol{U}(k) \tag{5.24}$$

Consider the system discrete model expressed in equation , the matrices $\mathbf{\Phi}$, $\mathbf{\Upsilon}$ and $\mathbf{\Theta}$ can be expressed by:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{C}_d\mathbf{A}_d \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{N_c} \\ \mathbf{C}_d\mathbf{A}_d^{N_c+1} \\ \vdots \\ \mathbf{C}_d\mathbf{A}_d^{N_p} \end{bmatrix}, \mathbf{\Upsilon} = \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d \\ \vdots \\ \sum_{i=0}^{N_c-1}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d \\ \sum_{i=0}^{N_c}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d \\ \vdots \\ \sum_{i=0}^{N_p-1}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \mathbf{C}_d\mathbf{B}_d & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{N_c-1}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d & \cdots & \mathbf{C}_d\mathbf{B}_d \\ \sum_{i=0}^{N_c}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d & \cdots & \sum_{i=0}^{1}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{N_p-1}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d & \cdots & \sum_{i=0}^{N_p-N_c}\mathbf{C}_d\mathbf{A}_d^i\mathbf{B}_d \end{bmatrix} \tag{5.25}$$

In consideration of equation 5.24, let us define a new variable $\varepsilon(k)$ as follows:

$$\varepsilon(k) = \boldsymbol{R}(k) - \boldsymbol{\Phi}\boldsymbol{x}(k) - \boldsymbol{\Upsilon}\boldsymbol{u}(k-1) \tag{5.26}$$

then the cost function $\Gamma(k)$ can be denoted by the following equation:

$$\Gamma(k) = \|\boldsymbol{\Theta}\Delta\boldsymbol{U}(k) - \varepsilon(k)\|^2_{\mathbf{Q}} + \|\Delta\boldsymbol{U}(k)\|^2_{\mathbf{T}} = \Gamma_{const} + \Delta\boldsymbol{U}(k)^{\mathrm{T}}\boldsymbol{\vartheta} + \Delta\boldsymbol{U}(k)^{\mathrm{T}}\mathbf{M}\Delta\boldsymbol{U}(k) \tag{5.27}$$

where $\Gamma_{const} = \varepsilon(k)^{\mathrm{T}}\mathbf{Q}\varepsilon(k)$, $\boldsymbol{\vartheta} = -2\boldsymbol{\Theta}^{\mathrm{T}}\mathbf{Q}\varepsilon(k)$ and $\mathbf{M} = \boldsymbol{\Theta}^{\mathrm{T}}\mathbf{Q}\boldsymbol{\Theta} + \mathbf{T}$. Without considering the constraints imposed on $\Delta\boldsymbol{U}$, the optimal control $\Delta\boldsymbol{U}^* = -\frac{1}{2}\mathbf{M}^{-1}\boldsymbol{\vartheta}$ can be used. When various constraints about $\Delta\boldsymbol{U}$ are taken into account and expressed by linear inequalities $\mathbf{G}\Delta\boldsymbol{U} \leq \boldsymbol{g}$, the optimization of NMPC strategy turns out a Quadratic Programming (QP) problem which will be introduced in section 5.4 in detail.

5.3. Inequality Constraints

None constraints are considered in the previous sections. In practice, all processes are subject to certain constraints. Sensors have their own limited scopes, and actuators have limited field of action. Furthermore, security, environmental and operational conditions are often defined by the intersection of certain constraints for safety or economic reasons. Therefore, the control system will be implemented under a list of constraints. These facts make the introduction of constraints in the cost function to be minimized necessary. In view of the cost function expressed in equation 5.27, the optimal control issue with different kinds of constraints over the receding horizon is denoted as a Quadratic Programming problem:

$$\begin{aligned} &\Delta\boldsymbol{U}^*(k) = \min_{\Delta\boldsymbol{U}} \Delta\boldsymbol{U}(k)^{\mathrm{T}}\mathbf{M}\Delta\boldsymbol{U}(k) + \boldsymbol{\vartheta}^{\mathrm{T}}\Delta\boldsymbol{U}(k) \\ &\text{subject to} \qquad \mathbf{G}\Delta\boldsymbol{U}(k) \leq \boldsymbol{g} \end{aligned} \tag{5.28}$$

How to solve this QP problem will be depicted in section 5.4. Subsequently, we will concentrate on translating various constraints into inequality equations and obtained the matrix $\mathbf{G}$ and vector $\boldsymbol{g}$.

5.3.1. Input/Output Constraints

Generally, bound in the amplitude of the control input $\boldsymbol{\tau}(t)$ and limits in the output $\boldsymbol{y}(t)$ will be taken into account:

$$\begin{cases} \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_{max} & \forall t \\ \boldsymbol{y}_{min} \leq \boldsymbol{y}(t) \leq \boldsymbol{y}_{max} & \forall t \end{cases} \tag{5.29}$$

Considering feedback linearization in section 5.2.2 and control limits in equation 5.29, there is no linear relationship between new input $\boldsymbol{u}(t)$ and real control input $\boldsymbol{\tau}(t)$. Thanks to the positive symmetric inertia matrix $\mathbf{H}$, the new control input boundary $\boldsymbol{u}_{max} = \mathbf{H}^{-1}(\overline{\boldsymbol{\tau}} - \boldsymbol{c})$ and $\boldsymbol{u}_{min} = \mathbf{H}^{-1}(\underline{\boldsymbol{\tau}} - \boldsymbol{c})$, where $\overline{\boldsymbol{\tau}}$ and $\underline{\boldsymbol{\tau}}$ are the corresponding torque vectors fulfil control

input boundary expressed in equation 5.29. Since the JSIM $\mathbf{H}$ is symmetric, positive definite, it can be decomposed as follows:

$$\mathbf{H} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{p}_n^{\mathrm{T}} \end{bmatrix} \tag{5.30}$$

and its inverse can be described as:

$$\mathbf{H}^{-1} = (\mathbf{P}\mathbf{D}\mathbf{P}^{-1})^{-1} = \mathbf{P}\mathbf{D}^{-1}\mathbf{P}^{-1} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n] \begin{bmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{p}_n^{\mathrm{T}} \end{bmatrix} \tag{5.31}$$

If we project the vector $\boldsymbol{\tau}(t)$ on the direction of the eigenvector $\boldsymbol{p}_i$, the torque vector can be expressed by:

$$\boldsymbol{\tau} = \zeta_1 \boldsymbol{p}_1 + \zeta_2 \boldsymbol{p}_2 + \cdots + \zeta_n \boldsymbol{p}_n \tag{5.32}$$

where $\zeta_i = \langle \boldsymbol{\tau}, \boldsymbol{p}_i \rangle$. Since the term $\mathbf{H}^{-1}\boldsymbol{\tau}$ at sampling time k is constant, then the boundary of new control input can be translated into a Linear Programming (LP) problem:

$$\begin{cases} u_{i\,max} = \arg_{\boldsymbol{\zeta}}\ max\ \frac{1}{\sigma_i} u_i \cdot \boldsymbol{\zeta} \\ u_{i\,min} = \arg_{\boldsymbol{\zeta}}\ min\ \frac{1}{\sigma_i} u_i \cdot \boldsymbol{\zeta} \end{cases} \quad \text{subject to } \boldsymbol{\tau}_{min} \le \mathbf{P}\boldsymbol{\zeta} \le \boldsymbol{\tau}_{max} \tag{5.33}$$

The new control input vector $\boldsymbol{u}(t)$ expressed in discrete time at sampling time k yields its own amplitude boundaries can be described by

$$\boldsymbol{u}_{min} \le \boldsymbol{u}(k-1) + \Delta\boldsymbol{u}(k) \le \boldsymbol{u}_{max} \tag{5.34}$$

when all the control input vector over the receding horizon N_c are taken into account, the new control input constraints can be expressed as follows:

$$\begin{cases} \boldsymbol{\Omega}_{N_c}\boldsymbol{u}(k-1) + \boldsymbol{\Psi}\Delta\boldsymbol{U}(k) \le \boldsymbol{\Omega}_{N_c}\boldsymbol{u}_{max} \\ \boldsymbol{\Omega}_{N_c}\boldsymbol{u}(k-1) + \boldsymbol{\Psi}\Delta\boldsymbol{U}(k) \ge \boldsymbol{\Omega}_{N_c}\boldsymbol{u}_{min} \end{cases} \tag{5.35}$$

where the matrices $\boldsymbol{\Omega}_{N_c}$ and $\boldsymbol{\Psi}$ can be denoted as:

$$\boldsymbol{\Omega}_{N_c} = \begin{bmatrix} \mathbf{E}_n \\ \vdots \\ \mathbf{E}_n \end{bmatrix} \in \mathbb{R}^{nN_c \times n}, \quad \boldsymbol{\Psi} = \begin{bmatrix} \mathbf{E}_n & & \\ \vdots & \ddots & \\ \mathbf{E}_n & \cdots & \mathbf{E}_n \end{bmatrix} \in \mathbb{R}^{nN_c \times nN_c} \tag{5.36}$$

Likewise, consider the output boundaries in equation 5.29, suppose $\boldsymbol{Y}_p = \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{u}(k-1)$ and $\boldsymbol{\Omega}_{N_p} = [\mathbf{E}_{2n}, \cdots, \mathbf{E}_{2n}]^{\mathrm{T}} \in \mathbb{R}^{2nN_p \times 2n}$, the new input vector $\boldsymbol{u}(k)$ yields the boundaries of output $\boldsymbol{y}(t)$ can be denoted as:

$$\begin{cases} \boldsymbol{Y}_p + \boldsymbol{\Theta}\Delta\boldsymbol{U}(k) \le \boldsymbol{\Omega}_{N_p}\boldsymbol{y}_{max} \\ \boldsymbol{Y}_p + \boldsymbol{\Theta}\Delta\boldsymbol{U}(k) \ge \boldsymbol{\Omega}_{N_p}\boldsymbol{y}_{min} \end{cases} \tag{5.37}$$

5.3.2. Obstacle/Singularity Constraints

The presence of the obstacles and singularities in the workspace of the manipulator restricts the full ability of the manipulator. Generally, there are three classes of primary solutions for collision and singularity avoidance issues. The first solution is to treat them as a planning problem as was widely analysed in chapter 4. The second solution to solve these problems is adding an additional cost function in the optimization issue, as was discussed in Camacho and Bordons (2007), Jasour and Farrokhi (2009). The third solution, which is also the most reasonable one, is to regard them as motion constraints imposed on the optimization issue Fan-Tien Cheng et al. (1994), Faverjon and Tournassoud (1987), Kanehiro et al., Kanoun et al. (2011), since the existence of them limits the volume and form of the feasible workspace region. In this section, we will elaborate on how to translate the singularity and collision problems into different linear inequalities used in the optimization issue.

As was illustrated in chapter 4, when only the nearest point is considered in collision avoidance strategy, an oscillation will occur since the switching of the control point on the manipulator. In order to overcome such drawback, the other control point, named Maximum Projection Distance (MPD) point is used to adjust the control point on the manipulator. Here, these two points will also be employed to form two independent constraints imposed on the manipulator to suppress the possible vibration. A schematic diagram of the two constraints is shown in Figure 5.6.

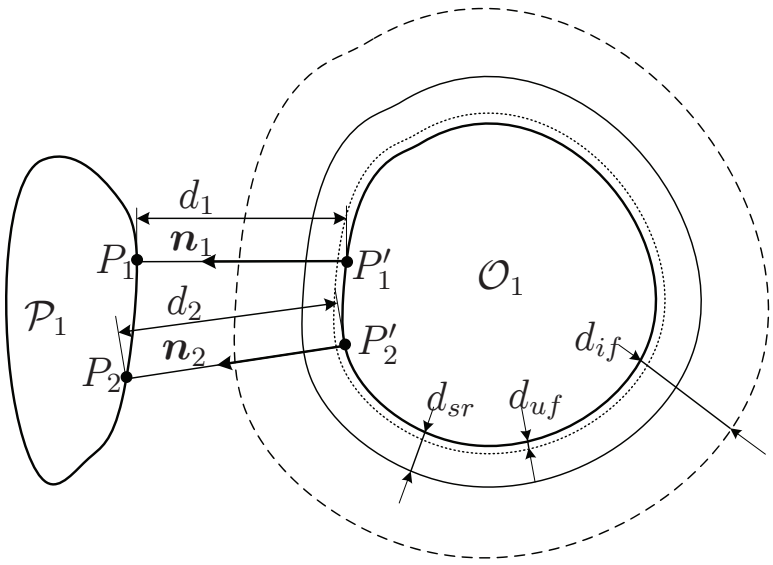


Figure 5.6.: Schematic diagram of anti-collision

Let us recall an inequality named velocity damper proposed in Faverjon and Tournassoud (1987), Kanehiro et al., refer to Figure 5.6, if the distance d between the control points and the obstacle enters into the influence zone defined by d_{if}, the following constraints can be imposed on the motion of the manipulator:

$$\lambda_{ac}\dot{d} \geq -\eta_{ac}\frac{d - d_{uf}}{d_{if} - d_{uf}} \tag{5.38}$$

where $0 < \eta_{ac} < 1$ is damper coefficient for adjusting convergence speed. λ_{ac} is a switching function to control whether current anti-collision constraint function uses or not, because in some cases, like in capturing phaser, the interact face between the gripper and the target satellite is constrained by the collision avoidance. The switching function can be expressed as:

$$\lambda_{ac} = \begin{cases} 0 & \text{constraint is not considered} \\ 1 & \text{constraint is considered} \end{cases} \tag{5.39}$$

Given the initial condition $d(0) \geq d_{uf}$, and in terms of $\lambda_{ac} = 1$, the following inequality can be derived:

$$d(t) \geq d_{uf} + (d(0) - d_{uf}) e^{-\frac{\eta_{ac} t}{d_{if} - d_{uf}}} \geq d_{uf} \quad \forall t > 0 \tag{5.40}$$

Expression in equation 5.40 assures that the distance between the control point and the obstacles constrained by velocity damper will be never smaller than d_{uf}. Similarly, d_{sr} is the security distance as defined in chapter 4, once $d < d_{sr}$, the inequality constraint expressed in equation 5.38 will become hard constraint and will be illustrated in section 5.4. Note that $\dot{d}$ is constrained by inequality 5.38, when the control point P_i enters into influence zone ($d \leq d_{if}$), a linear inequality constraint over the manipulator's velocity $\dot{\boldsymbol{\theta}}$ can be obtained:

$$\lambda_{ac} \langle \mathbf{J}_{P_i}^{\mathrm{T}} \boldsymbol{n}_i, \dot{\boldsymbol{\theta}} \rangle \geq \langle \dot{\boldsymbol{r}}_{P_i'}, \boldsymbol{n}_i \rangle - \eta_{ac} \frac{d_i - d_{uf}}{d_{if} - d_{uf}} \tag{5.41}$$

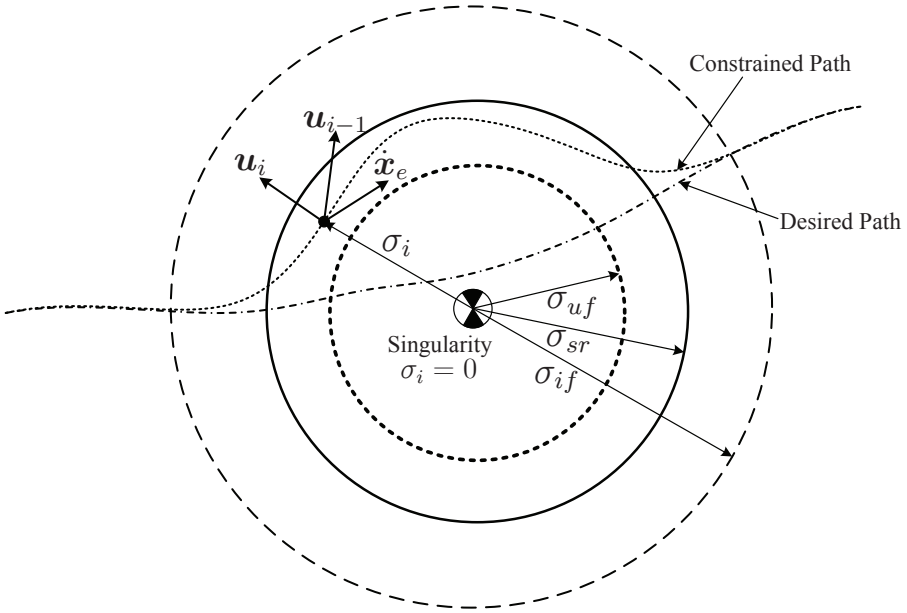


Figure 5.7.: Schematic diagram of anti-singularity

Likewise, we also want to translate the singularity issue into a linear inequality constraint as in equation 5.41. From the definition of the manipulability ellipsoid in equation 4.8, refer to Figure 5.7, it can be perceived that singularity emerges when the singular value of the Jacobian matrix $\sigma_i = 0$ along the i^{th} principal axis direction. It also means, one can drive

the manipulator far from singularity point through regulating the change rate of singular value appropriately as the obstacle avoidance. The change rate of singular value σ_i can be obtained through projecting the task-space velocity $\boldsymbol{x}_e$ in the direction of the principal axis $\boldsymbol{u}_i$:

$$\dot{\sigma}_i = \langle \mathbf{J}\dot{\boldsymbol{\theta}}, \boldsymbol{u}_i \rangle \tag{5.42}$$

If we define σ_{if}, σ_{sr}, σ_{uf} influence distance, security distance, and unsafe distance as been done in collision avoidance, recall the velocity damper the following constraint will be defined:

$$\lambda_{as}\dot{\sigma}_i \geq -\eta_{as}\frac{\sigma_i - \sigma_{uf}}{\sigma_{if} - \sigma_{uf}} \tag{5.43}$$

λ_{as} and η_{as} are the switching function and turning parameter for velocity damper. As illustrated in equation 5.40, give the initial condition $\sigma_i(0) > \sigma_{uf}$ and $\lambda_{as} = 1$, the following inequality can be derived:

$$\sigma_i(t) \geq \sigma_{uf} + (\sigma_i(0) - \sigma_{uf})e^{-\frac{\eta_{as}t}{\sigma_{if}-\sigma_{uf}}} \geq \sigma_{uf} \quad \forall t > 0 \tag{5.44}$$

Expression in equation 5.44 assures that the singular value σ_i will never smaller than σ_{uf}. A linear inequality constraint for singularity issue over the manipulator's velocity $\dot{\boldsymbol{\theta}}$ can be obtained:

$$\lambda_{as}\langle \mathbf{J}^{\mathrm{T}}\boldsymbol{u}_i, \dot{\boldsymbol{\theta}} \rangle \geq -\eta_{as}\frac{\sigma_i - \sigma_{uf}}{\sigma_{if} - \sigma_{uf}} \tag{5.45}$$

Next, we use the anti-collision and anti-singularity constraints to construct the matrix $\mathbf{G}$ and vector $\boldsymbol{g}$. Assuming that N_{ac}^k anti-collision constraints are activated at sampling time k, the new control input $\boldsymbol{u}(k)$ yields the velocity damper constraint in equation 5.41 can be expressed by:

$$\boldsymbol{\Theta}_{ac}^{v}\left[\mathbf{C}_{vd}\mathbf{A}_d\boldsymbol{x}(k) + \mathbf{C}_{vd}\mathbf{B}_d\boldsymbol{u}(k)\right] \geq \boldsymbol{D}_{uf} \tag{5.46}$$

where $\mathbf{C}_{vd} = [\mathbf{0}_n, \mathbf{E}_n]$, the matrix $\boldsymbol{\Theta}_{ac}^{v}$ and vector $\boldsymbol{D}_{uf}$ can be denoted as:

$$\boldsymbol{\Theta}_{ac}^{v} = \begin{bmatrix} \lambda_{ac}^{1}\boldsymbol{n}_1^{\mathrm{T}}\mathbf{J}_{P_1} \\ \vdots \\ \lambda_{ac}^{N_{ac}^k}\boldsymbol{n}_{N_{ac}^k}^{\mathrm{T}}\mathbf{J}_{P_{N_{ac}^k}} \end{bmatrix}, \quad \boldsymbol{D}_{uf} = \begin{bmatrix} \langle \dot{\boldsymbol{r}}_{P_1'}, \boldsymbol{n}_1 \rangle - \eta_{ac}\frac{d_1 - d_{uf}}{d_{if} - d_{uf}} \\ \vdots \\ \langle \dot{\boldsymbol{r}}_{P'_{N_{ac}^k}}, \boldsymbol{n}_{N_{ac}^k} \rangle - \eta_{ac}\frac{d_{N_{ac}^k} - d_{uf}}{d_{if} - d_{uf}} \end{bmatrix} \tag{5.47}$$

If we define $\boldsymbol{y}_{vp} = \mathbf{C}_{vd}\mathbf{A}_d\boldsymbol{x}(k) + \mathbf{C}_{vd}\mathbf{B}_d\boldsymbol{u}(k-1)$ and $\boldsymbol{\Theta}_{vb} = [\mathbf{C}_{vd}\mathbf{B}_d, \mathbf{0}_n, \cdots, \mathbf{0}_n] \in \mathbb{R}^{n \times nN_c}$, according to equation 5.46, let $\boldsymbol{D}_{vp} = \boldsymbol{\Theta}_{ac}^{v}\boldsymbol{y}_{vp}$ and $\boldsymbol{\Theta}_{ac} = \boldsymbol{\Theta}_{ac}^{v}\boldsymbol{\Theta}_{vb}$, equation 5.46 can be expressed by:

$$\boldsymbol{D}_{vp} + \boldsymbol{\Theta}_{ac}\Delta\boldsymbol{U}(k) \geq \boldsymbol{D}_{uf} \tag{5.48}$$

Similarly, the anti-singularity constraints can be treated as follows. If there are N_{as}^k anti-singularity constraints are activated at instant k, the control input $\boldsymbol{u}(k)$ yield the velocity damper constraints in equation 5.45 can be expressed by:

$$\boldsymbol{\Sigma}_{vp} + \boldsymbol{\Theta}_{as}\Delta\boldsymbol{U}(k) \geq \boldsymbol{\Sigma}_{uf} \tag{5.49}$$

where $\boldsymbol{\Sigma}_{vp} = \boldsymbol{\Theta}^{v}_{as}\boldsymbol{y}_{vp}$ and $\boldsymbol{\Theta}_{as} = \boldsymbol{\Theta}^{v}_{as}\boldsymbol{\Theta}_{vb}$. The matrix $\boldsymbol{\Theta}^{v}_{as}$ and vector $\boldsymbol{\Sigma}_{uf}$ can be expressed as follows:

$$\boldsymbol{\Theta}^{v}_{as} = \begin{bmatrix} \lambda^{1}_{as}\boldsymbol{n}^{\mathrm{T}}_{1}\mathbf{J} \\ \vdots \\ \lambda^{N^{k}_{as}}_{as}\boldsymbol{n}^{\mathrm{T}}_{N^{k}_{as}}\mathbf{J} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{uf} = \begin{bmatrix} -\eta_{as}\frac{\sigma_{1}-\sigma_{uf}}{\sigma_{if}-\sigma_{uf}} \\ \vdots \\ -\eta_{as}\frac{\sigma_{N^{k}_{as}}-\sigma_{uf}}{\sigma_{if}-\sigma_{uf}} \end{bmatrix} \tag{5.50}$$

After a variety of constraints are expressed by linear inequalities as in equations 5.35, 5.37, 5.48 and 5.49, the quantities of $\mathbf{G}$ and $\boldsymbol{g}$ can be detived as follows:

$$\mathbf{G} = \begin{bmatrix} \boldsymbol{\Psi} \\ -\boldsymbol{\Psi} \\ \boldsymbol{\Theta} \\ -\boldsymbol{\Theta} \\ -\boldsymbol{\Theta}_{ac} \\ -\boldsymbol{\Theta}_{as} \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} \boldsymbol{\Omega}_{N_c}u_{max} - \boldsymbol{\Omega}_{N_c}\boldsymbol{u}(k-1) \\ -\boldsymbol{\Omega}_{N_c}\boldsymbol{u}_{min} + \boldsymbol{\Omega}_{N_c}\boldsymbol{u}(k-1) \\ \boldsymbol{\Omega}_{N_p}\boldsymbol{y}_{max} - \boldsymbol{Y}_{p} \\ -\boldsymbol{\Omega}_{N_p}\boldsymbol{y}_{min} + \boldsymbol{Y}_{p} \\ \boldsymbol{D}_{vp} - \boldsymbol{D}_{uf} \\ \boldsymbol{\Sigma}_{vp} - \boldsymbol{\Sigma}_{uf} \end{bmatrix} \tag{5.51}$$

To complete a given task with optimization, the inequality constraints denoted by equations 5.35, 5.37, 5.48 and 5.49 should be integrated into the optimization problem. Of course, for collision avoidance, each link of the manipulator and every obstacle in workspace, also between two different links yield inequality 5.41, which implies series of anti-collision constraints will be imposed on the manipulator regarding to the relative relationship of two arbitrary mobile objects in workspace. For singularity avoidance, every singular value can be chosen to yield inequality 5.45, which indicates that anti-singularity constraints will be activated according to the relative distance to the singular configuration. These linear inequalities will be employed in QP problem as will be illustrated in section 5.4.

5.4. Quadratic Programming

As was indicated in section 5.1, the implementation of NMPC is not a trivial matter and is irrefutably a more complicated task than the commissioning of conventional control schemes. Leadingly, the designed controller has to solve a non-linear QP problem with linear constraints as shown in equation 5.28 to gain a sequence of optimal control effort $\Delta\boldsymbol{U}^{*}(k)$ through searching in the current feasible region. A variety of methods are commonly used including active set, interior point, augmented Lagrangian as introduced in Maciejowski (2002), Nocedal and Wright (2006), Wang (op. 2009). The choice of algorithm to solve QP needs to consider the special structure of the NMPC problem.

5.4.1. KKT Conditions

The Karush-Kuhn-Tucker (KKT) conditions are first order necessary conditions for a solution in non-linear optimization programming. The necessary conditions for QP problem in

equation 5.28 given by KKT conditions can be expressed as follows:

$$\begin{cases} \mathbf{M}\Delta\boldsymbol{U} + \boldsymbol{\vartheta} + \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda} = 0 \\ \mathbf{G}\Delta\boldsymbol{U} - \boldsymbol{g} \leq \mathbf{0} \\ \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{G}\Delta\boldsymbol{U} - \boldsymbol{g}) = 0 \\ \boldsymbol{\lambda} \geq 0 \end{cases} \tag{5.52}$$

where the vector $\boldsymbol{\lambda}$ is the Lagrangian multipliers. The inequality constraints expressed in equation 5.28 may be composed of inactive constraints and active constraints. The constraint $\mathbf{G}_i\Delta\boldsymbol{U} \leq \boldsymbol{g}_i$ is called active if $\mathbf{G}_i\Delta\boldsymbol{U} = \boldsymbol{g}_i$ and inactive if $\mathbf{G}_i\Delta\boldsymbol{U} < \boldsymbol{g}_i$. The active set $\mathcal{S}_{act}$ made up of the active constraints, plays a significant role in optimization theory since it determines which constraints will influence the finial result of optimization Nocedal and Wright (2006). If the active set $\mathcal{S}_{act}$ can be determined, the KKT conditions can be expressed in another way:

$$\begin{cases} \mathbf{M}\Delta\boldsymbol{U} + \boldsymbol{\vartheta} + \sum_{i\in\mathcal{S}_{act}} \mathbf{G}_{act}^{\mathrm{T}}\boldsymbol{\lambda}_{act} = 0 & \\ \mathbf{G}_i\Delta\boldsymbol{U} - \boldsymbol{g}_i = \mathbf{0} & i \in \mathcal{S}_{act} \\ \mathbf{G}_i\Delta\boldsymbol{U} - \boldsymbol{g}_i < \mathbf{0} & i \notin \mathcal{S}_{act} \\ \boldsymbol{\lambda}_i \geq 0 & i \in \mathcal{S}_{act} \\ \boldsymbol{\lambda}_i = 0 & i \notin \mathcal{S}_{act} \end{cases} \tag{5.53}$$

Obviously, if the i^{th} constraint is active, an equality constraint $\mathbf{G}_i\Delta\boldsymbol{U} = \boldsymbol{g}_i$ can be imposed on the cost function which can be solved with Lagrange multiplier method. On the contrary, $\mathbf{G}_i\Delta\boldsymbol{U} < \boldsymbol{g}_i$ denotes that the constraint is satisfied, thus it is inactive and can be discarded. An active set method, called Hildreth's Quadratic Programming procedure Wang (op. 2009), will be employed in our NMPC design.

5.4.2. QP with Prioritized Constraints

During the optimization operation, the feasible region defined in the decision variables by the active set of constraints may be empty due to the model-mismatch, external disturbance, and noise or artificially fault. Such a problem, will stop the standard QP procedure without output which is unacceptable for a controller in real-time provided to the space robot. In Qin and Badgwell (2003), it is depicted that, when a predicted input violates the boundary in the DMC-plus algorithm, it is set to its limits and the computation continues. Another solution to the infeasibility is first solve the QP problem without constraints and then clip to yield hard constraints. These methods prevent violation of hard constraints, but involves a loss of performance and the solutions do not satisfy the KKT necessary conditions for optimality.

The infeasibility problem, maybe lead to instability of the control-loop, motivates the academic community to develop new techniques as introduced in Rawlings and Muske (1993), Scokaert and Rawlings (1999), Vada et al. (2001), Zheng and Morari (1995) aimed at recovering feasibility without violating hard constraints. In these techniques, how to handle constraints is at the core of solving QP infeasibility problem. The linear constraints imposed on decision variables can be categorized into three classes:

- **Physical/Security constraints**: These limits should never be surpassed and are usually associated to the device physical functioning or the security warranty. For instance, the input bound of $\boldsymbol{\tau}(t)$ expressed in equation 5.35 pertains to this kinds of constraints and will be treated as hard constraints.
- **Operational constraints**: These limits are fixed by the operators as bounds within which the decision variables are expected to maintain appropriate operating conditions. They can be violated in certain circumstance and be treated as soft constraints such as the limits of output $\boldsymbol{y}(t)$ denoted in equation 5.37.
- **Passive constraints**: Unlike physical/security or operational constraints which are imposed on the decision variables at all the time, the passive limits will only be triggered when the decision variables slide into its influence zone, such as the distance between the link and obstacle enters into the collision influence zone, or the configuration of the manipulator comes into a influence zone of singularity. Under this circumstance, they can be considered as soft constraints and violated to some extent. Once the decision variables enter into the security zone, the corresponding constraints turn to hard constraints and can't be violated any more. Actually, anti-singularity and anti-collision constraints in the robotic motion can be translated into suchlike passive constraints.

Like the prioritized tasks illustrated in chapter 4, the constraints belong to different categories do not possess the same priority. If we use $\mathbf{G}_{hc}\Delta\boldsymbol{U}(k) \leq \boldsymbol{g}_{hc}$ and $\mathbf{G}_{sc}\Delta\boldsymbol{U}(k) \leq \boldsymbol{g}_{sc}$ to represent hard and soft constraints respectively, and assign different priorities to various soft constraints according to their relative significance, the QP problem defined in equation 5.28 can be rewritten as follows:

$$
\begin{aligned}
&\Delta\boldsymbol{U}^*(k) = \min_{\Delta\boldsymbol{U}} \Delta\boldsymbol{U}(k)^{\mathrm{T}}\mathbf{M}\Delta\boldsymbol{U}(k) + \boldsymbol{\vartheta}^{\mathrm{T}}\Delta U(k) + \rho\Delta\boldsymbol{g}_{sc}^{\mathrm{T}}\Delta\boldsymbol{g}_{sc} \\
&\text{subject to} \\
&\text{hard constraints} \begin{cases} \boldsymbol{x}(k|k) = \boldsymbol{x}(k) \\ \boldsymbol{x}(k+j+1|k) = \boldsymbol{f}_d(\boldsymbol{x}(k+j|k), \boldsymbol{u}(k+j|k)) \\ \boldsymbol{u}(k+j|k) = \boldsymbol{u}(k+N_c|k), \ j \geq N_c \\ \mathbf{G}_{hc}\Delta\boldsymbol{U}(k) \leq \boldsymbol{g}_{hc}, \qquad\quad j \leq N_p \end{cases} \\
&\text{soft constraints} \begin{cases} \mathcal{S}_1: \quad \mathbf{G}_{sc}^1\Delta\boldsymbol{U}(k) \leq \boldsymbol{g}_{sc}^1 \\ \vdots \qquad\qquad \vdots \qquad\qquad j \leq N_p \\ \mathcal{S}_{n_s}: \quad \mathbf{G}_{sc}^{n_s}\Delta\boldsymbol{U}(k) \leq \boldsymbol{g}_{sc}^{n_s} \end{cases}
\end{aligned}
\tag{5.54}
$$

where the constraint sets $\{\mathcal{S}_1, \cdots, \mathcal{S}_{n_s}\}$ are constructed while constraint set $\mathcal{S}_i$ has higher priority than $\mathcal{S}_{i+1}$. Refer to Nocedal and Wright (2006), Vada et al. (1999), Wang (op. 2009), an algorithm solving QP problem subject to the constraints with priorities is presented as follows:

(1) Employing Hildreth's QP procedure to solve the QP optimization problem defined in equation 5.54 only with hard constraints. If there is not a feasible solution, go to step (2); else, go to step (3).

(2) Checking existence of a solution to the first hard constraints at current sampling time without prediction, since the current hard constraints are more important than the predicted constraints. If no solution exists, solve the QP problem without constraints, then clip the solution to yield hard constraints. This ensures the continuation of the computation with some performance loss. Else, employ the solution as an input to the plant.

(3) Checking existence of a solution to the complete QP problem in equation 5.54. If there is a feasible solution, the optimal control effort is determined. Else go to step (4).

(4) Set $k = 1$, from the highest priority constraints set $\mathcal{S}_1$, check existence of a solution to hard constraints and constraint set $\{\mathcal{S}_1, \cdots, \mathcal{S}_k\}$, if a feasible solution is found, $k = k + 1$ and go to step (4). Else, if no feasible solution is found, $\{\mathcal{S}_1, \cdots, \mathcal{S}_k\}$ can't be satisfied simultaneously, go to step (5).

(5) Step (4) revealed $\{\mathcal{S}_1, \cdots, \mathcal{S}_{k-1}\}$ are satisfied while $\{\mathcal{S}_k, \cdots, \mathcal{S}_{n_s}\}$ can't be satisfied. Compute the optimal slack variables $\Delta \boldsymbol{g}_{sc}$ to minimize the violation of constraint sets $\{\mathcal{S}_k, \cdots, \mathcal{S}_{n_s}\}$ and replace $\{\mathcal{S}_k, \cdots, \mathcal{S}_{n_s}\}$ with $\{\mathcal{S}'_k, \cdots, \mathcal{S}'_{n_s}\}$, such that the renewed sets $\{\mathcal{S}_1, \cdots, \mathcal{S}_{k-1}, \mathcal{S}'_k, \cdots, \mathcal{S}'_{n_s}\}$ are satisfied. Go to step (6).

(6) At this step, there exists a solution which fulfils the constraint sets $\{\mathcal{S}_1, \cdots, \mathcal{S}_{k-1}, \mathcal{S}'_k, \cdots, \mathcal{S}'_{n_s}\}$. Now, we can minimize the performance index in equation 5.54 subject to these constraints and output the optimal solution $\Delta \boldsymbol{U}^*(k)$.

The softening constraints offer additional region for the control input, to some extent, it expands the feasible region of the decision variables. And, the constraints with priorities enable the most significant constraints execute as much as possible, which also enlarges the feasible region of the control input.

5.5. Simulation Study

In this section, the robotic mission from approach to capture the target using space robot will be implemented applying the proposed NMPC strategy. Four simulations, approach to the target, tracking a line with singularity, tracking a infinite ring with obstacles and tracking & capturing a point target, are executed to verify the performance and effectiveness of our NMPC method.

5.5.1. Simulation Set-up

The NMPC algorithm implemented with a non-linear model and receding optimization, is employed for the space robot to complete the approach, tracking and capturing target missions. The tuning parameters relevant to NMPC are the sampling period, prediction horizon N_p, control horizon N_c and the input and output weights matrices $\mathbf{Q}(i)$ and $\mathbf{T}(i)$. The sampling time is set to $0.01s$. According to the inaccuracy of long-term prediction and high burden of computation, the prediction horizon $N_p = 20$ and the control horizon $N_c = 5$. The ratio between weights matrices $\mathbf{Q}(i)$ and $\mathbf{T}(i)$ indicates the importance of tracking error and control effort. Here we choose $\mathbf{Q}(i) = \mathbf{E}_n$ and $\mathbf{T}(i) = 0.05\mathbf{E}_n$.

5.5.2. Approach to the Target

As illustrated in chapter 2, once the servicer satellite comes into the neighbourhood of the target satellite and ready for the robotic mission, the first robotic mission would be unfolding the manipulator to an optimal grasping pose. The space manipulator is initially at

its folding configuration $\boldsymbol{\theta}_0 = [0, -\frac{\pi}{2}, 0, 0, 0, 0, 0]^{\mathrm{T}}$. After receiving tele-commands from the ground station or certain requirements are satisfied, the motors will be activated and will deploy the manipulator to a specific position $\boldsymbol{\theta}_0 = [0, \frac{\pi}{4}, 0, \frac{2\pi}{5}, 0, -\frac{\pi}{6}, 0]^{\mathrm{T}}$ in 4 seconds ready for further robotic missions. For each joint motion, a quintic polynomial trajectory as set point expressed in the following equation is employed:

$$\boldsymbol{r}(t) = \boldsymbol{b}_0 + \boldsymbol{b}_1 t + \boldsymbol{b}_2 t^2 + \boldsymbol{b}_3 t^3 + \boldsymbol{b}_4 t^4 + \boldsymbol{b}_5 t^5 \tag{5.55}$$

where $\boldsymbol{b}_i (i = 0, \cdots, 5)$ represents the polynomial coefficients of the planned trajectory based on the inverse kinematics and the duration of the trajectory.

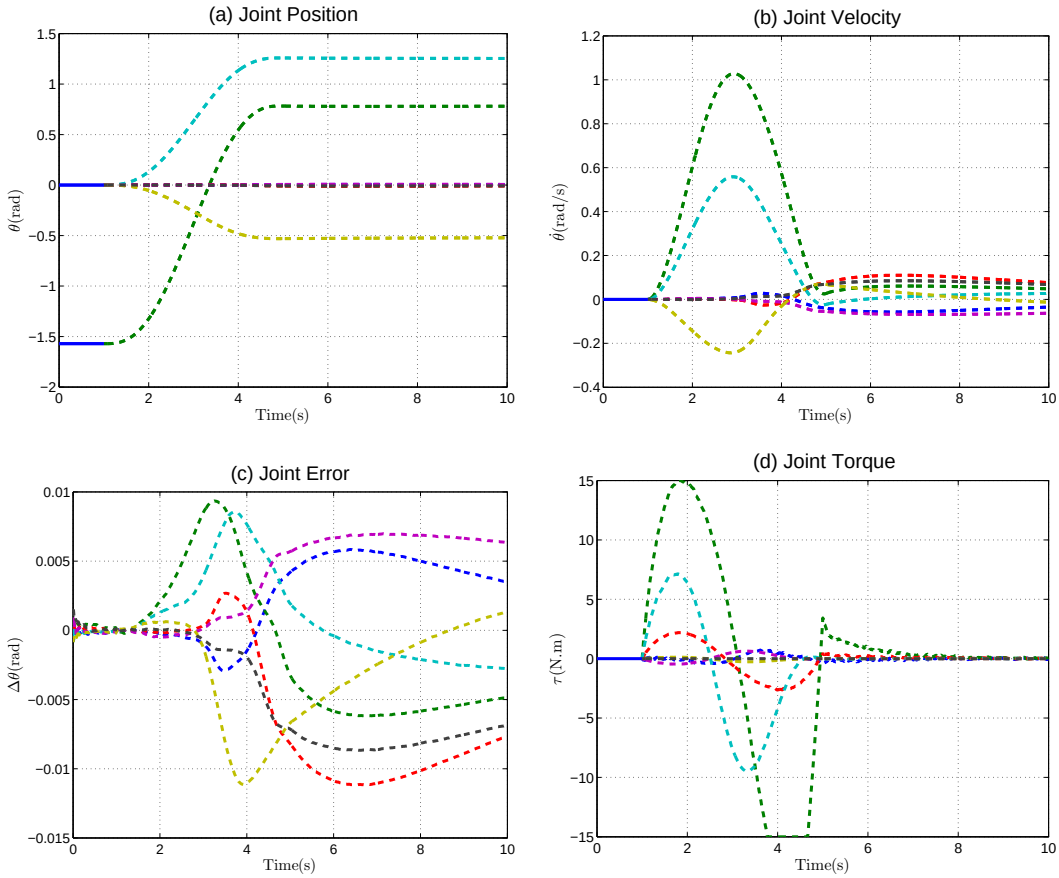


Figure 5.8.: Unfolding the space manipulator using RMAC method

The widely used RMAC strategy as introduced in Luh et al. (1980a), Siciliano (2009), Umetani and Yoshida (1989) is employed for comparison reason. Both two strategies are implemented in the simulation environment Matlab®/Simulink® as part of RACOON system. Figure 5.8 and Figure 5.9 show the simulation results of the RMAC and NMPC methods, respectively. One can see that, both methods can complete the unfolding task successfully, while NMPC strategy holds extra advantages. The residue error of joint position and velocity are larger using RMAC than using NMPC. Besides, joint toques violates its ranges using RMAC as comparing Figure 5.8(d) with Figure 5.9(d).

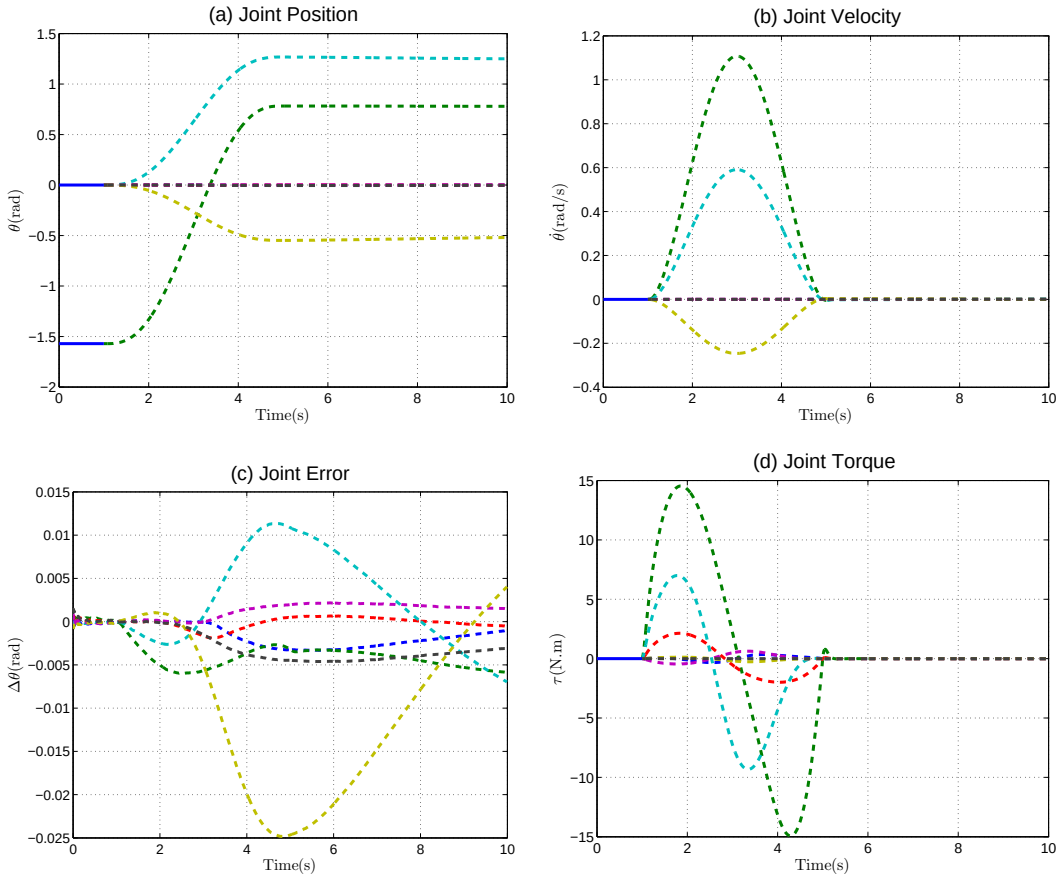


Figure 5.9.: Unfolding the space manipulator using NMPC method

5.5.3. Tracking a Predefined Path

Two simulations will be conducted independently in this section to show the collision and singularity avoidance ability of the proposed NMPC strategy. For demonstration, two pre-defined tasks and three artificial obstacles are employed to test the effect of collision and singularity avoidance.

In order to test the collision avoidance, three additional obstacles, two spheres and one cylinder are located in the workspace of the space manipulator. The primary task of the end-effector is to track a path with the form of infinite ring in 10 seconds. The simulation results are shown in Figure 5.10. One can see that, the tracking task does not completely conducted since the existence of the obstacles. When the manipulator approaches to the obstacles, the anti-collision constraints are activated to impede the occurrence of possible collision. The minimum distances between the manipulator and the obstacles are strictly constrained by the pre-defined unsafe distance d_{uf}. Three anti-collision constraints are activated during the tracking task. After collision avoidance, the NMPC strategy continues to conduct the required task as much as possible as shown in Figure 5.10(c) and (d). The joint position and joint velocity also fulfil the output constraints as shown in Figure 5.10(e) and (f). The simulation results demonstrate that the proposed constrained MPC algorithm is satisfactory for obstacle avoidance especially when more than one obstacles in workspace of the manipulator.

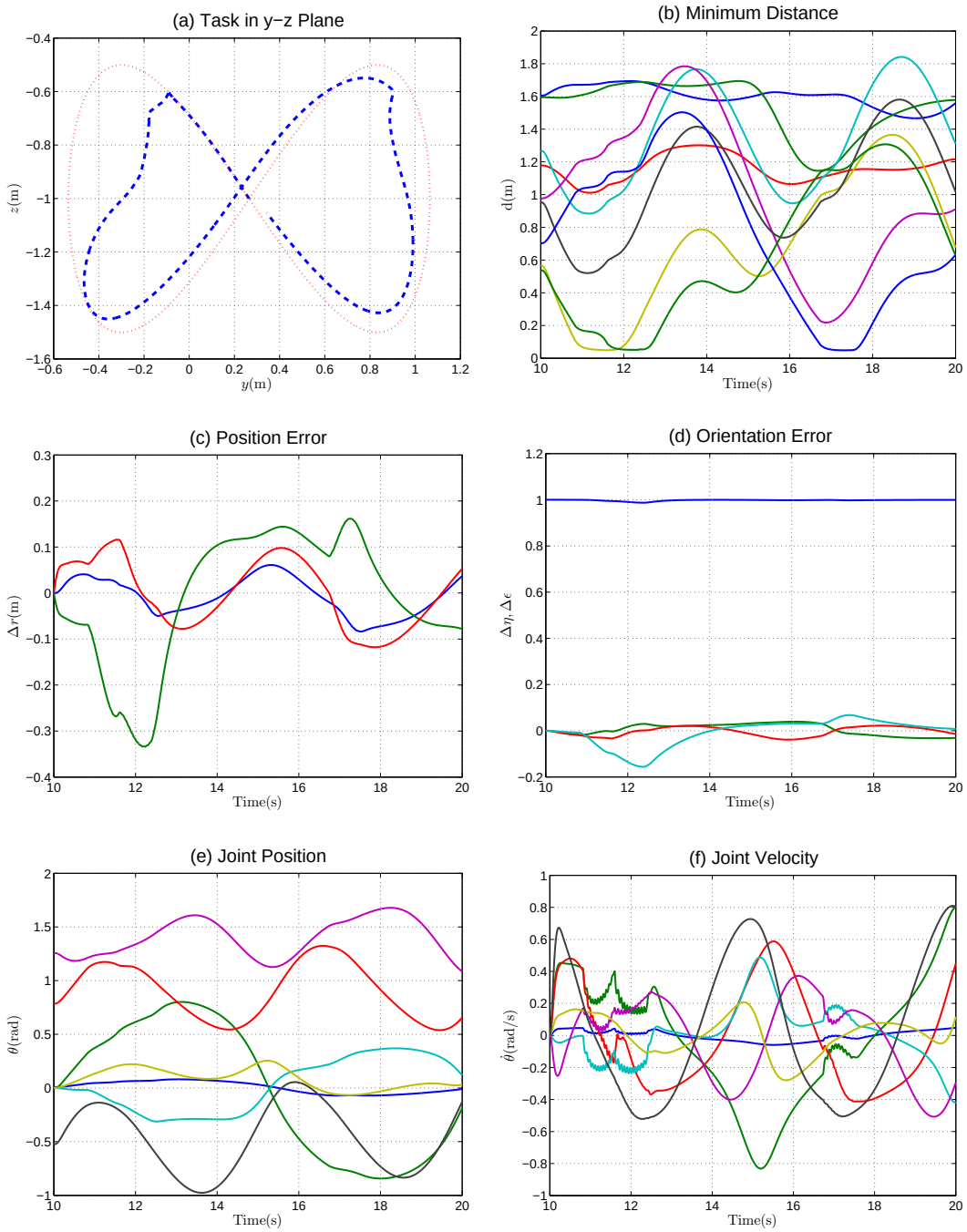


Figure 5.10.: Tracking infinite ring with anti-collision constraints

The second simulation is to test the singularity avoidance ability of the proposed NMPC strategy. The path is set to track a line in the workspace of space manipulator within 10 seconds. Without adopting any measure for singularity issue, the singularity will occur at 18^{th} second. Like the anti-collision constraints, an anti-singularity constraint is activated at that moment to prevent the manipulator from sliding into the singular configuration. The minimum singular value σ_m is restricted at its unsafe quantity σ_{uf}. The tracking error for position and orientation is promising, while the joint position and joint velocity also fulfil the output constraints as shown in Figure 5.11(e) and (f). The oscillation in joint velocity is induced by the damper coefficient λ_{as}. Some trade-off must be taken to choose the optimal value of λ_{as}.

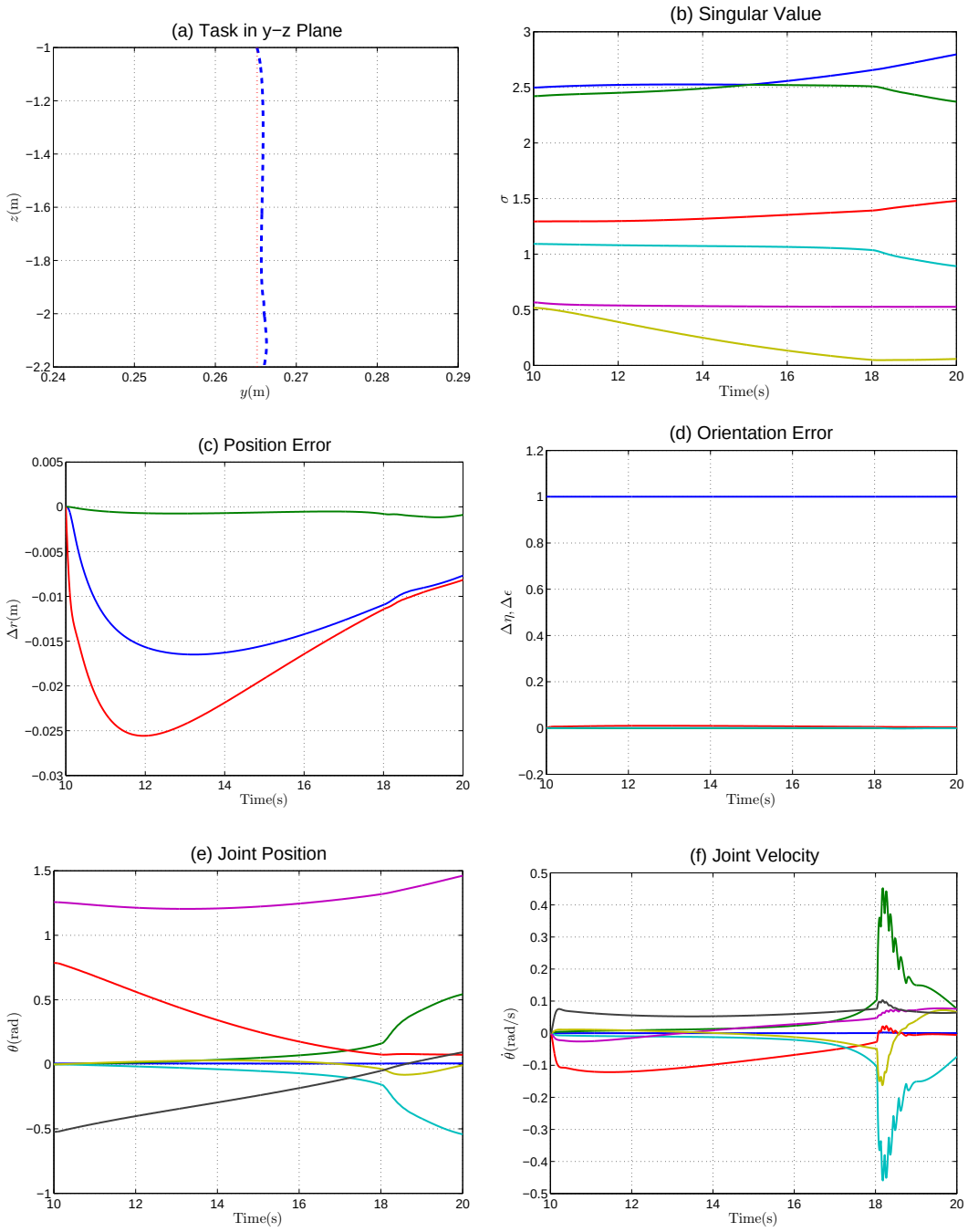


Figure 5.11.: Tracking line with anti-singularity constraints

5.5.4. Tracking a Point on the Target

After the space robot unfolding its configuration and approaching to the target, in order to capture one point on the target with the end-effector, a tracking mission that can guide the end-effector to the capture point with appropriate orientation should be planned very carefully. As shown in Figure 5.12, the objective of this phase is to adjust the relative position and orientation between the end-effector frame and capture point frame as close as possible. In this simulation we assume that the full states of the capture point can be obtained from the sensors mounted on the servicer satellite or the manipulator.

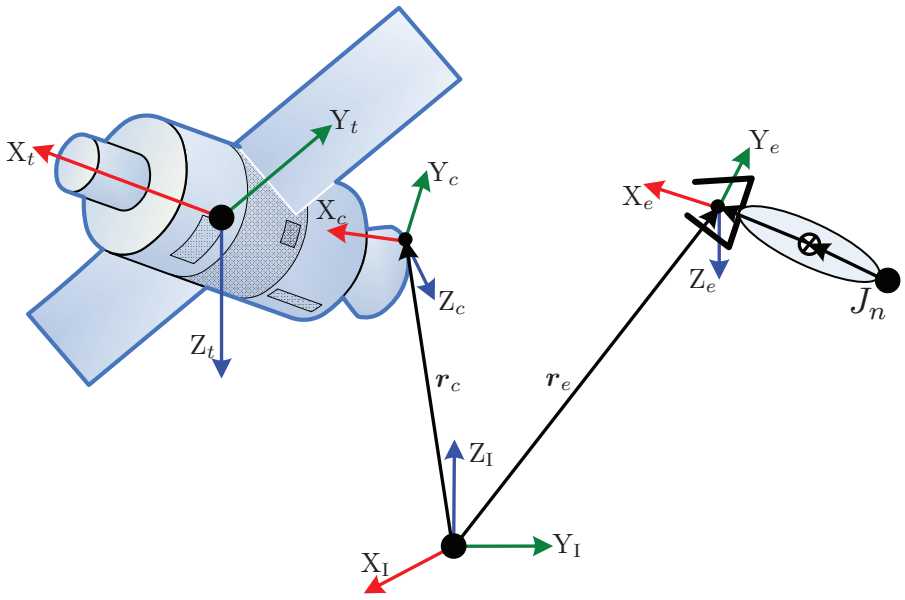


Figure 5.12.: Relative relationship of end-effector frame and capture point frame

For the sake of simplification, tracking the capture point is decomposed into two subtasks. The first subtask is tracking the capture point without considering the translational and rotational motion along x axis. This will drive the end-effector rightly locate in the rear of the capture point and be ready for the next step mission, such as surveillance, target state estimation, etc. If $\mathbf{R}_c = [\boldsymbol{n}_c\ \boldsymbol{s}_c\ \boldsymbol{a}_c]$ denotes the rotation matrix of the capture point and $\mathbf{R}_e = [\boldsymbol{n}_e\ \boldsymbol{s}_e\ \boldsymbol{a}_e]$ the rotation matrix of the end-effector frame, the first subtask can be represented as follows:

$$\begin{cases} \dot{\boldsymbol{r}}_e = \dot{\boldsymbol{r}}_c + \mathbf{K}_P(\boldsymbol{r}_c - \boldsymbol{r}_e) = [0, \dot{\boldsymbol{r}}_{cy} + \mathbf{K}_{Py}(\boldsymbol{r}_{cy} - \boldsymbol{r}_{ey}), \dot{\boldsymbol{r}}_{cz} + \mathbf{K}_{Pz}(\boldsymbol{r}_{cz} - \boldsymbol{r}_{ez})]^{\mathrm{T}} \\ \boldsymbol{\omega}_e = \boldsymbol{\omega}_c + \mathbf{K}_O(\boldsymbol{n}_e \times \boldsymbol{n}_c) = [0, \boldsymbol{\omega}_{cy} + \mathbf{K}_{Oy}(n_{c1}n_{e3} - n_{c3}n_{e1}), \boldsymbol{\omega}_{cz} + \mathbf{K}_{Oz}(n_{c2}n_{e1} - n_{c1}n_{e2})]^{\mathrm{T}} \end{cases} \tag{5.56}$$

where $\dot{\boldsymbol{r}}_c$ and $\dot{\boldsymbol{r}}_e$ are the velocity vectors of the capture point and end-effector, respectively. $\mathbf{K}_P$ and $\mathbf{K}_O$ are the control gain matrix for position and orientation, respectively.

After x axis of the end-effector frame aligns with x axis of the capture point frame, a second subtask can be activated to drive the end-effector close to the capture point. The finger of the end-effector will be activated to open for the coming capturing mission during this phase. Although the second subtask is mainly to reduce the relative translational and rotational error along x axis, since the motion of the target satellite, a feedback of motion along y and z axis are also required to compensate the drifting error. The designed path for the second subtask can be given by:

$$\begin{cases} \dot{\boldsymbol{r}}_e = \dot{\boldsymbol{r}}_c + \mathbf{K}_P(\boldsymbol{r}_c - \boldsymbol{r}_e) \\ \boldsymbol{\omega}_e = \boldsymbol{\omega}_c + \mathbf{K}_O(\eta_e\boldsymbol{\epsilon}_c - \eta_c\boldsymbol{\epsilon}_e - \mathbf{S}(\boldsymbol{\epsilon}_c)\boldsymbol{\epsilon}_e) \end{cases} \tag{5.57}$$

where $\{\eta_c, \boldsymbol{\epsilon}_c\}$ and $\{\eta_e, \boldsymbol{\epsilon}_e\}$ are the unit quaternions associated with the rotation matrix $\mathbf{R}_c$ and $\mathbf{R}_e$, respectively. During this phase, NMPC strategy with anti-collision and anti-singularity constraints developed in this chapter is employed to complete both subtasks. Figure 5.13

shows the simulation results of tracking a certain point on the target satellite. The reference path of the end-effector is generated by using equation 5.56 and equation 5.57. The relative position and relative quaternion between the frame of the end-effector and the capture point are shown in Figure 5.13(c) and (d). The two frames finally coincide with each other after the two subtasks are executed subsequently. The joint position and velocity fulfil the output constraints as shown in Figure 5.13(e) and (f). Figure 5.14 shows the screenshot of the tracking and capturing phase by using the proposed NMPC strategy. The end-effector successfully accomplishes the required task and captures the target.

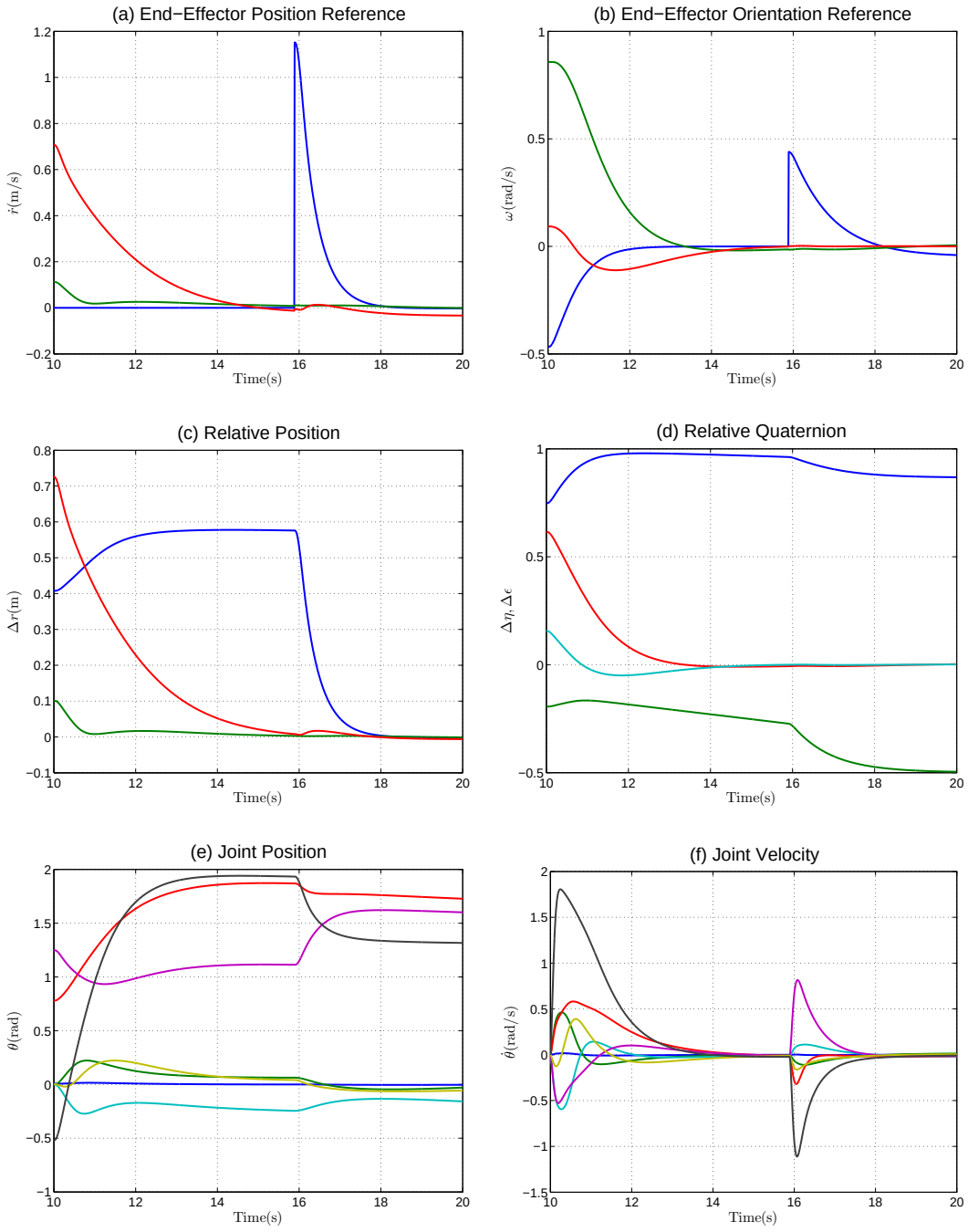


Figure 5.13.: Tracking capture point on target satellite

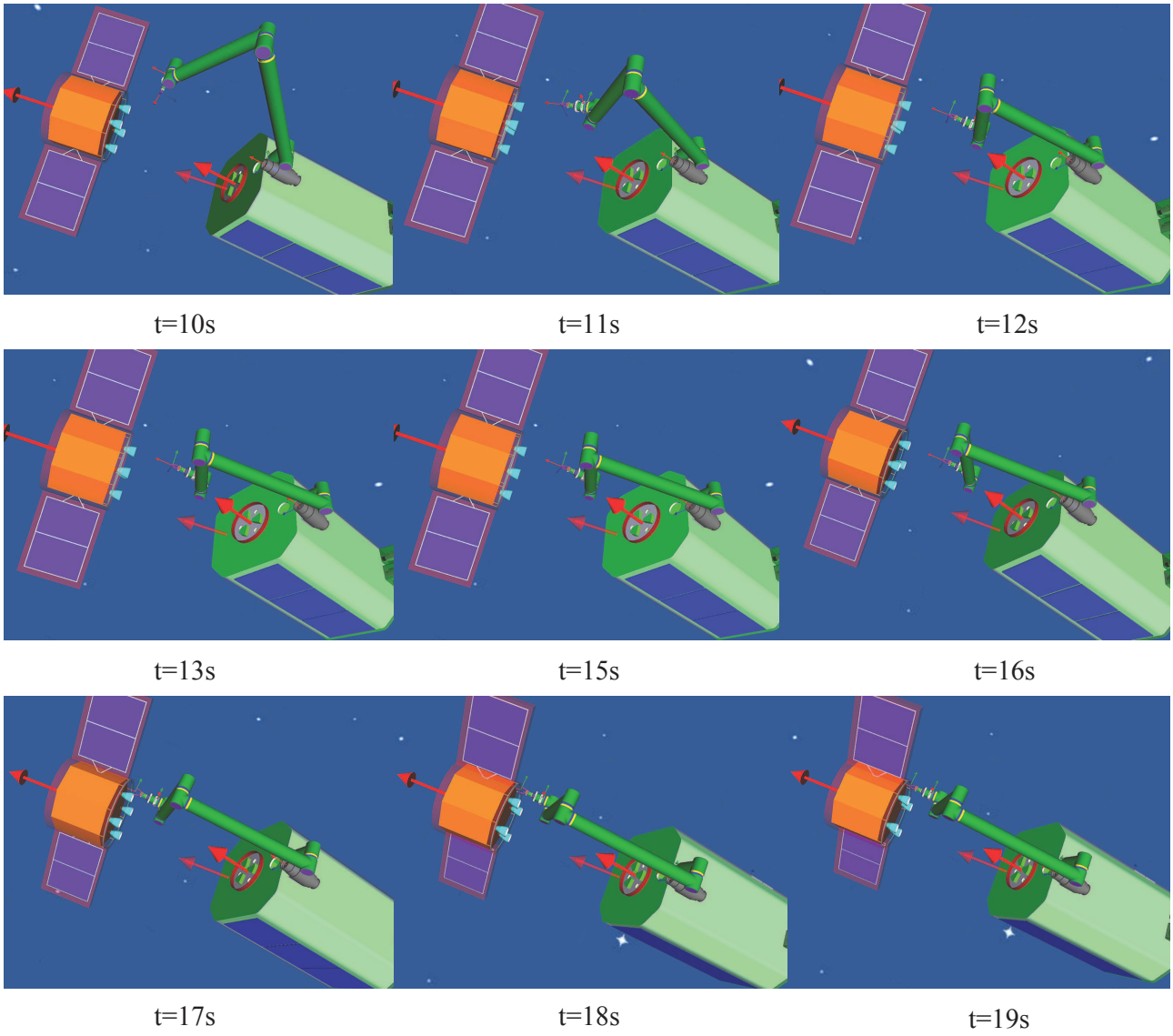


Figure 5.14.: Screenshot of tracking the capture point on target

The designed tracking motion for the end-effector guarantees the success of the end-effector moving to the capture point and adjusting the capture point in the middle of the end-effector. Once the end-effector frame coincides with the frame of capture point, the fingers of end-effector can be closed to complete the capturing phase.

5.6. Summary

The application of Nonlinear Model Predictive Control (NMPC) to capture another target satellite using space manipulator is investigated in this chapter. NMPC originates from chemical processing industry and spreads its applications to other fields. The principle and formulation of NMPC is illustrated in detail. Since the non-linearity of the free-floating space robot, feedback linearisation technique is used to decouple the non-linear system. System observer based on Kalman Filter (KF) algorithm is developed to estimate the system states

when modelling error and measurement error are taken into account. The NMPC strategy utilizes a quadratic function as its optimization index, and combines various of constraints, which forms an on-line constrained Quadratic Programming (QP) problem. In this chapter, input/output constraints, obstacle/singularity constraints are successfully translated into linear inequalities as part of the QP problem. Different constraints are categorized into diverse constraints with unequal priorities for dealing with the possible infeasibility during on-line optimization. Four simulations, approach to the target, tracking a infinite ring, tracking a line and tracking a point on the target are performed to verify the proposed method. As one can see that, not only the traditional input/output constraints are considered, but also the obstacle/singularity constraints are taken into account. This means, on the one hand, anti-collision and anti-singularity problems can be treated as planning problems as introduced in chapter 4, on the other hand, it can also be treated as control constraints to limit the control input in a reasonable way. The simulation results convincingly demonstrated the proposed control framework in the application of space robot.

6. Conclusions and Future Research

There are now three types of scientists: experimental, theoretical and computational.

—Silvan S. Schweber

As a high potential solution for future On-Orbit Servicing (OOS) missions, the use of space robot introduces series of challenging issues to implement such a system working in outer space. This chapter concludes and discusses the results in the previous chapters. The advantages and disadvantages of the proposed methods are analysed. Furthermore, the possible research directions in the future are listed to enhance and pursue this thesis.

6.1. Conclusions

Capturing an un-cooperative target satellite by using space manipulator is the main concern in this thesis. For that purpose, at the beginning of this work, a distributed real-time simulation system, RACOON has been established for space robotic technique demonstration. The proposed RACOON system can simulate the scenario of space robot and offer the operators on ground intuitive information, such as text, image and VR etc. which enhances the perception ability of the operators. Moreover, the reusable simulator is easy to modify and replace by better modules or related hardware, which provides expandability and scalability of this simulation architecture in the future.

As a complex multi-body system, the dynamics of space robot possesses specific dynamic equations. Considering the effect of the system topology to its dynamics, spatial notation and graph theory are used to construct the dynamics equation of the space robot with general tree structure. From path matrix and parent array, an IMM is derived which has the same pattern as the JSIM. The IMM reveals the influence of the system topology to its dynamics. As a symmetric, positive definite matrix, IMM can be used to explore the branched-induced sparsity and to analyse the computational cost of CRBA. The complexity of calculating the JSIM is estimated between $O(1)$ and $O(n^3)$ by using IMM. The proposed algorithm is competitive when there is sufficient branching in the kinematic tree.

The characteristics of the outer space introduce some special properties to the space robot. One of them is the dynamic coupling between the space manipulator and the spacecraft when none external forces and torques applied on the spacecraft. Besides, the moving target satellite requires real-time collision avoidance when the space manipulator approach and track the target in its workspace. Furthermore, the un-predictable singularity may occur when the manipulator tries to track the path defined in Cartesian space. Traditional methods consider the collision and singularity avoidance issues mainly as path planning problem without thinking the constraints of the system input and output. In this thesis, from path

planning and control aspects, the singularity and collision avoidance issues are investigated independently.

At the path planning level, the proposed STR method for singularity avoidance utilizes the minimum singular value σ_m as singular measurement and left-singular vector as projection direction to reconstruct the path expressed in task-space. With the proposed STR method, less overshoot, more predictable minimum singular value, robustness and smaller & smoother joint velocities are achieved as compared to the typical DLS method. For collision avoidance, a new control point, MPD point is introduced into the collision avoidance strategy when non strictly convex objects are considered to suppress the possible fluctuation of the joint velocities. The new collision avoidance strategy ensures the continuous change of the control point, which also guarantees the smoother joint velocities. The new approaches for collision and singularity avoidance can be expanded to fixed-base manipulator or space manipulator.

At the motion control level, NMPC strategy was chosen as the motion control method for free-floating space robot. As a finite horizon, on-line optimized control strategy, NMPC successfully resolves the system input and output boundary conditions. Furthermore, using velocity damper, the collision and singularity issues are also translated into linear inequalities and integrated into the NMPC. An on-line QP with prioritized constraints is adopted to search the optimal control effort over the prediction horizon. Simulation results show that the proposed NMPC strategy can successfully fulfil multiple constraints and complete the required task as much as possible. To capture a point on the moving target satellite, a two-stage path of the end-effector is generated and implemented using constrained MPC strategy. As a result, the issues in path planing become various constraints in motion control, which is fairly similar as our human interaction with external environment. One drawback of NMPC is its computational burden which is caused by the on-line predication and QP algorithm.

6.2. Future Research

How to realize capturing an un-cooperative target satellite is still a challenging issue until now. The simulation system RACOON and the control framework based on NMPC provide the first step into space robotic research field involving input/output boundary conditions, collision/singularity avoidance for the space robotic missions. The possible research questions arising from this work can be listed as follows:

- **Estimation and prediction of the states of the target satellites without prior knowledge**. Full and perfect states are assumed to be known during this work, which is not the case in practice. How to estimate the states of the target accurately is a big issue in capturing. The accuracy of the estimation will influence the success of the capture. One should widely use multiple sensors on-board and data integration technique to compute the kinematic and dynamic parameters of the target. Moreover, the prediction of the target is also a considerable aspect that can be imported to the control loop Lampariello and Hirzinger (2013).
- **Determination of the capture point**. It is another challenging issue for capture. When the target satellite is moving and rotating, how to find a feasible capture point is the

key to guarantee the success of capture. This involves collision, manipulability, input/output boundary conditions of the space robot.

- **Robust stability of the NMPC strategy**. The nominal stability of the MPC can be obtained by adding terminal constraint, terminal weighting matrix or contraction constraint as illustrated in Mayne et al. (2000), Rawlings and Muske (1993), Zheng and Morari (1995). The NMPC strategy in this work assumes that the plant to be controlled and the model for prediction are the same without considering model mismatch. The influence of the plant uncertainty to the robust stability should be carefully analysed and pursued. Some techniques that synthesis robust NMPC can be found in Kothare et al. (1996), Mayne et al. (2000)
- **Faster collision detection algorithm**. The collision detection algorithm is very time consuming especially when multiple non-convex obstacles are in the workspace. Another application of collision detection would be the contact between the space robot and the target satellite. These requires faster and more accurate collision detection algorithms.
- **Contact theory**. Once the frame of the capture point on the target coincides with the frame of the end-effector, the fingers of the end-effector will be activated to perform close action to complete the grasp. Nevertheless, due to the measurement noise, model mismatch or external disturbance, there will be still some residual relative velocities between end-effector and the capture point. A contact will occur when the fingers of the end-effector close. The effect of the contact to the space robot and the target satellite has to be investigated seriously to ensure safety and successful capture.
- **Post-capture stabilization and further operations**. After the capture is done, a post-capture stabilization should be performed to decrease the rotational velocity of the target satellite. During this phase, the attitude control of the servicer satellite has to be activated to realize the stabilization of servicer and target as a whole. The space manipulator will be used for further OOS operations, such as assembly, ORU exchange, repair etc. which require the developers to synthesis all the demanded mission elaborately.

In summary, the presented work in this dissertation provides a simulation tool for the coming space robotic operations and opens the window to a field of studies in MPC application to space robot.

A. Bibliography

S. Abiko and G. Hirzinger. An adaptive control for a free-floating space robot by using inverted chain approach. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2236–2241. IEEE, 2007. ISBN 978-1-4244-0912-9.

O.P Agrawal and Y. Xu. On the global optimum path planning for redundant space manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(9):1306–1316, 1994.

C.A Balafoutis and R.V Patel. Efficient computation of manipulator inertia matrices and the direct dynamics problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5): 1313–1321, 1989.

C.A Balafoutis, R.V Patel, and P. Misra. Efficient modeling and computation of manipulator dynamics using orthogonal Cartesian tensors. *IEEE Journal on Robotics and Automation*, 4 (6):665–676, 1988.

B. Bon and H. Seraji. On-line collision avoidance for the Ranger telerobotic flight experiment. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2041–2048. IEEE, 1996. ISBN 0-7803-2988-0.

B. Bon and H. Seraji. Real-time model-based obstacle detection for the NASA Ranger Telerobot. In *Proceedings of International Conference on Robotics and Automation*, pages 1580–1587. IEEE, 1997. ISBN 0-7803-3612-7.

Albert B. Bosse, W. J. Barnds, Michael A. Brown, N. G. Creamer, Andy Feerst, Carl G. Henshaw, Alan S. Hope, Bernard E. Kelm, Patricia A. Klein, Frank Pipitone, Bertrand E. Plourde, Brian P. Whalen, Jr.Peter Tchoryk, and Melissa Wright. SUMO: Spacecraft for the universal modification of orbit. In *Spacecraft Platforms and Infrastructure*, pages 36–46. SPIE, 2004.

John R. Boyd. *Destruction and creation*. The Operational level of war. U.S. Army Comand and General Staff College, [Ft. Leavenworth and Kan.], 1987.

R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

Fraser Cain. How Many Satellite are in Space?, 2013. URL `www.universetoday.com/42198/how-many-satellites-in-space/`.

Eduardo F. Camacho and Carlos Bordons. *Model predictive control*. Advanced textbooks in control and signal processing. Springer, London [u.a.], 2004. ISBN 978-0-85729-398-5.

Eduardo F. Camacho and Carlos Bordons. Nonlinear Model Predictive Control: An Introductory Review. In Rolf Findeisen, Frank Allgöwer, and Lorenz T. Biegler, editors, *Lecture Notes in Control and Information Sciences*, pages 1–16. Springer Berlin Heidelberg, Berlin and Heidelberg, 2007. ISBN 978-3-540-72698-2.

Chi-Ying Lin and Yen-Chung Liu. Precision Tracking Control and Constraint Handling of Mechatronic Servo Systems Using Model Predictive Control. *IEEE/ASME Transactions on Mechatronics*, 17(4):593–605, 2012.

S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

D.W Clarke, C. Mohtadi, and P.S Tuffs. Generalized Predictive Control—Part II Extensions and interpretations. *Automatica*, 23(2):149–160, 1987a.

D.W Clarke, C. Mohtadi, and P.S Tuffs. Generalized predictive control—Part I. The basic algorithm. *Automatica*, 23(2):137–148, 1987b.

C. R. Cutler and B. L. Ramaker. Dynamic Matrix Control - A Computer Control Algorithm. In *Joint Automatic Control Conference*, 1980.

J. Denavit and R. S. Hartenberg. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Trans. ASME, J. Appl. Mech.*, 22(2):215–221, 1955.

S. Dubowsky and E. Papadopoulos. The kinematics, dynamics, and control of free-flying and free-floating space robotic systems. *IEEE Transactions on Robotics and Automation*, 9(5): 531–543, 1993.

S. Dubowsky and M.A Torres. Path planning for space manipulators to minimize spacecraft attitude disturbances. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2522–2528. IEEE Comput. Soc. Press, 1991. ISBN 0-8186-2163-X.

Christer Ericson. *Real-time collision detection*. Morgan Kaufmann series in interactive 3D technology. Elsevier, Amsterdam and Boston, 2005. ISBN 1558607323.

Fan-Tien Cheng, Wei-Ming Wang, and Fan-Chu Kung. Priority considerations for multiple goals of redundant manipulators. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 264–269. IEEE, 1994. ISBN 0-7803-2129-4.

B. Faverjon and P. Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, pages 1152–1159. Institute of Electrical and Electronics Engineers, 1987.

R. Featherstone. Efficient Factorization of the Joint-Space Inertia Matrix for Branched Kinematic Trees. *The International Journal of Robotics Research*, 24(6):487–500, 2005.

R. Featherstone and D. Orin. Robot dynamics: equations and algorithms. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, pages 826–834. IEEE, 2000. ISBN 0-7803-5886-4.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, New York and N.Y, 2008. ISBN 0387743146.

Graham Gibbs and Savi Sachdev. Canada and the International Space Station program: overview and status. *Acta astronautica*, 51(1-9):591–600, 2002.

K. Glass, R. Colbaugh, D. Lim, and H. Seraji. Real-time collision avoidance for redundant manipulators. *IEEE Transactions on Robotics and Automation*, 11(3):448–457, 1995.

Y.-L Gu and Y. Xu. A normal form augmentation approach to adaptive control of space robot systems. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 731–737. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-3450-2.

R. Hedjar and P. Boucher. Nonlinear Receding-Horizon Control of Rigid Link Robot Manipulators. *International Journal of Advanced Robotic Systems*, page 1, 2005.

G. Hirzinger, J. Heindl, and K. Landzettel. Predictive and knowledge-based telerobotic control concepts. In *Proceedings. 1989 International Conference on Robotics and Automation*, pages 1768–1777. IEEE Comput. Soc. Press, 1989. ISBN 0-8186-1938-4.

G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. ROTEX-the first remotely controlled robot in space. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2604–2611. IEEE Comput. Soc. Press, 1994. ISBN 0-8186-5330-2.

John M. Hollerbach. A Recursive Lagrangian Formulation of Maniputator Dynamics and a Comparative Study of Dynamics Formulation Complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11):730–736, 1980.

P. Huang, J. Yan, J. Yuan, and Y. Xu. Robust control of space robot for capturing objects using optimal control method. In *2007 IEEE International Conference on Informaiton Acquisition*, pages 397–402. IEEE Comput. Soc. Press, 2007. ISBN 1-4244-1220-X.

Rui Huang, Sachin C. Patwardhan, and Lorenz T. Biegler. Robust extended Kalman filter based nonlinear model predictive control formulation. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8046–8051. IEEE, 2009. ISBN 978-1-4244-3871-6.

N. Inaba and M. Oda. Autonomous satellite capture by a space robot: world first on-orbit experiment on a Japanese robot satellite ETS-VII. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, pages 1169–1174. IEEE, 2000. ISBN 0-7803-5886-4.

Ashkan M. Jasour and Mohammad Farrokhi. Path tracking and obstacle avoidance for redundant robotic arms using fuzzy NMPC. In *2009 American Control Conference*, pages 1353–1358. IEEE, 2009. ISBN 978-1-4244-4523-3.

Jinhyun Kim, G. Marani, Wan Kyun Chung, and Junku Yuh. A general singularity avoidance framework for robot manipulators: task reconstruction method. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 4809–4814 Vol.5. IEEE, 2004. ISBN 0-7803-8232-3.

Fumio Kanehiro, Florent Lamiraux, Oussama Kanoun, Eiichi Yoshida, and Jean-Paul Laumond. A Local Collision Avoidance Method for Non-strictly Convex Polyhedra.

Oussama Kanoun, Florent Lamiraux, Pierre-Brice Wieber, Fumio Kanehiro, Eiichi Yoshida, and Jean-Paul Laumond. Prioritizing linear equality and inequality systems: Application to local motion planning for redundant robots. In *2009 IEEE International Conference on Robotics and Automation*, pages 2939–2944. IEEE, 2009. ISBN 978-1-4244-2788-8.

Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber. Kinematic Control of Redundant Manipulators: Generalizing the Task-Priority Framework to Inequality Task. *IEEE Transactions on Robotics*, 27(4):785–792, 2011.

O. Khatib and J. Burdick. Motion and force control of robot manipulators. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, pages 1381–1386. Institute of Electrical and Electronics Engineers, 1986.

J.-O Kim and P.K Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(3):338–349, 1992.

Jinhyun Kim, Giacomo Marani, Wan Kyun Chung, and Junku Yuh. Task reconstruction method for real-time singularity avoidance for robotic manipulators. *Advanced Robotics*, 20(4):453–481, 2006.

C. A. Klein and B. E. Blaho. Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

Mayuresh V. Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10): 1361–1379, 1996.

Roberto Lampariello. Motion Planning for the On-orbit Grasping of a Non-cooperative Target Satellite with Collision Avoidance. In *10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010)*, 2010.

Roberto Lampariello and Gerd Hirzinger. Generating feasible trajectories for autonomous on-orbit grasping of spinning debris in a useful time. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5652–5659. IEEE, 2013. ISBN 978-1-4673-6358-7.

Jay H. Lee and N. Lawrence Ricker. Extended Kalman Filter Based Nonlinear Model Predictive Control. *Industrial & Engineering Chemistry Research*, 33(6):1530–1541, 1994.

Bin Liang, Cheng Li, Lijun Xue, and Wenyi Qiang. A Chinese Small Intelligent Space Robotic System for On-Orbit Servicing. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4602–4607. IEEE, 2006. ISBN 1-4244-0258-1.

K. W. Lilly and D. E. Orin. Alternate Formulations for the Manipulator Inertia Matrix. *The International Journal of Robotics Research*, 10(1):64–74, 1991.

K.W Lilly and C.S Bonaventura. A generalized formulation for simulation of space robot constrained motion. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, pages 2835–2840. IEEE, 1995. ISBN 0-7803-1965-6.

J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980a.

J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-Line Computational Scheme for Mechanical Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69, 1980b.

B. Ma and W. Huo. Adaptive control of space robot system with an attitude controlled base. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, pages 1265–1270. IEEE, 1995. ISBN 0-7803-1965-6.

A. A. Maciejewski and C. A. Klein. Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985.

Anthony A. Maciejewski and Charles A. Klein. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems*, 5(6):527–552, 1988.

Jan M. Maciejowski. *Predictive control: With constraints*. Prentice Hall, Harlow and England and New York, 2002. ISBN 9780201398236.

V. Manikonda, P.O Arambel, M. Gopinathan, R.K Mehra, and F.Y Hadaegh. A model predictive control-based approach for spacecraft formation keeping and attitude control. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, pages 4258–4262. IEEE, 1999. ISBN 0-7803-4990-3.

G. Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pages 1973–1978. IEEE, 2002. ISBN 0-7803-7272-7.

Y. Masutani, F. Miyazaki, and S. Arimoto. Sensory feedback control for space manipulators. In *Proceedings. 1989 International Conference on Robotics and Automation*, pages 1346–1351. IEEE Comput. Soc. Press, 1989. ISBN 0-8186-1938-4.

D.Q Mayne, J.B Rawlings, C.V Rao, and P.O.M Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

R.V Mayorga and A.K.C Wong. A singularities avoidance approach for the optimal local path generation of redundant manipulators. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 49–54. IEEE Comput. Soc. Press, 1988. ISBN 0-8186-0852-8.

Richard A. McCourt and Clarence W. de Silva. Autonomous Robotic Capture of a Satellite Using Constrained Predictive Control. *IEEE/ASME Transactions on Mechatronics*, 11(6): 699–708, 2006.

S. McMillan and D.E Orin. Efficient computation of articulated-body inertias using successive axial screws. *IEEE Transactions on Robotics and Automation*, 11(4):606–611, 1995.

S. McMillan, D.E Orin, and R.B McGhee. Efficient dynamic simulation of an underwater vehicle with a robotic manipulator. *IEEE Transactions on Systems, Man, and Cybernetics*, 25 (8):1194–1206, 1995.

Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-Priority Based Redundancy Control of Robot Manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

Yoshihiko Nakamura and Hideo Hanafusa. Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163, 1986.

NASA. *On-orbit satellite servicing study: Project report*. National Aeronautics and Space Administration. Goddard Space Flight Center, 2010.

D. Nenchev. A controller for a redundant free-flying space robot with spacecraft attitude/-manipulator motion coordination. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, pages 2108–2114. IEEE, 1993. ISBN 0-7803-0823-9.

D. N. Nenchev. Restricted Jacobian Matrices of Redundant Manipulators in Constrained Motion Tasks. *The International Journal of Robotics Research*, 11(6):584–597, 1992.

D.N Nenchev and M. Uchiyama. Singularity-consistent path tracking: a null space based approach. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, pages 2482–2489. IEEE, 1995. ISBN 0-7803-1965-6.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, Berlin and New York, 2 edition, 2006. ISBN 978-0-387-30303-1.

M. Oda, K. Kibe, and F. Yamagata. ETS-VII, space robot in-orbit experiment satellite. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 739–744. IEEE, 1996. ISBN 0-7803-2988-0.

Andrew Ogilvie, Justin Allport, Hannah Michael, and John Lymer. Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System. In *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS '08)*, pages 25–29, 2008.

OMG. Data Distribution Service for Real-time Systems: Version 1.2, 2007. URL `http://www.omg.org/spec/DDS/1.2`.

OMG. The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specificiation (DDS-RTPS): Version 2.1, 2010. URL `http://www.omg.org/spec/DDS-RTPS/2.1/`.

E. Papadopoulos and I. Poulakakis. On path planning and obtacle avoidance for nonholonomic platforms with manipulator: a polynomial approach. *The International Journal of Robotics Research*, 21(4):367–383, 2002.

Evangelos Papadopoulos and S. Ali A. Moosavian. Dynamics and control of space free-flyers with multiple manipulators. *Advanced Robotics*, 9(6):603–624, 1994.

F.C Park, J.E Bobrow, and S.R Ploen. A Lie Group Formulation of Robot Dynamics. *The International Journal of Robotics Research*, 14(6):609–618, 1995.

H. Park, S. D. Cairano, and I. Kolmanovsky. Model predictive control for spacecraft rendezvous and docking with a rotating/tumbling platform and for debris avoidance. In *2011 American Control Conference*, pages 1922–1927. IEEE, 2011.

P. M. Pathak, R. P. Kumar, and A. Mukherjee. A scheme for robust trajectory control of space robots. *Simulation Modelling Practice and Theory*, 16(9):1337–1349, 2008.

R.A Peters, C.L Campbell, W.J Bluethmann, and E. Huber. Robonaut task learning through teleoperation. In *2003 IEEE International Conference on Robotics and Automation*, pages 2806–2811. IEEE, 2003. ISBN 0-7803-7736-2.

S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

Changwu Qiu, Qixin Cao, and Yijun Sun. Redundant Manipulator Control with Constraints for Subgoals. In *2006 IEEE International Conference on Automation Science and Engineering*, pages 212–217. IEEE, 2006. ISBN 1-4244-0310-3.

J.B Rawlings and K.R Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

Detlef Reintsema, Klaus Landzettel, and Gerd Hirzinger. DLR's Advanced Telerobotic Concepts and Experiments for On-Orbit Servicing. In Manuel Ferre, Martin Buss, Rafael Aracil, Claudio Melchiorri, and Carlos Balaguer, editors, *Springer Tracts in Advanced Robotics*, pages 323–345. Springer Berlin Heidelberg, Berlin and Heidelberg, 2007. ISBN 978-3-540-71363-0.

Ioannis Rekleitis, Eric Martin, Guy Rouleau, Régent L'Archevêque, Kourosh Parsa, and Eric Dupuis. Autonomous capture of a tumbling satellite. *Journal of Field Robotics*, 24(4):275–296, 2007.

J. Richalet, A. Rault, J.L Testud, and J. Papon. Model predictive heuristic control. *Automatica*, 14(5):413–428, 1978.

Arthur Richards, Tom Schouwenaars, Jonathan P. How, and Eric Feron. Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

G. Rodriguez, A. Jain, and K. Kreutz-Delgado. A Spatial Operator Algebra for Manipulator Modeling and Control. *The International Journal of Robotics Research*, 10(4):371–381, 1991.

Thomas Rupp, Toralf Boge, Reinhard Kiehling, and Sellmaier Florian. Flight Dynamics Challenges of the German On-Orbit Servicing Mission DEOS. In *21st International Symposium on Space Flight Dynamics*, 2009.

S. K. Saha. Dynamics of Serial Multibody Systems Using the Decoupled Natural Orthogonal Complement Matrices. *Journal of Applied Mechanics*, 66(4):986, 1999.

S.K Saha. A decomposition of the manipulator inertia matrix. *IEEE Transactions on Robotics and Automation*, 13(2):301–304, 1997.

G. Schreiber, M. Otter, and G. Hirzinger. Solving the singularity problem of non-redundant manipulators by constraint optimization. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, pages 1482–1488. IEEE, 1999. ISBN 0-7803-5184-3.

Pierre O. M. Scokaert and James B. Rawlings. Feasibility issues in linear model predictive control. *AIChE Journal*, 45(8):1649–1659, 1999.

H. Seraji and B. Bon. Real-time collision avoidance for position-controlled manipulators. *IEEE Transactions on Robotics and Automation*, 15(4):670–677, 1999.

Bruno Siciliano. *Robotics: Modelling, planning and control*. Advanced textbooks in control and signal processing. Springer, London, 2009. ISBN 9781846286414.

Garett A. Sohl and James E. Bobrow. A Recursive Multibody Dynamics and Sensitivity Algorithm for Branched Kinematic Chains. *Journal of Dynamic Systems, Measurement, and Control*, 123(3):391, 2001.

X. Tang, C. Tang, and H. Li. A backstepping robust control method for free-floating space robot system with dual-arms. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 761–765. IEEE Comput. Soc. Press, 2011. ISBN 978-1-4577-2136-6.

Y. Umetani and K. Yoshida. Resolved motion rate control of space manipulators with generalized Jacobian matrix. *IEEE Transactions on Robotics and Automation*, 5(3):303–314, 1989.

Jostein Vada, Olav Slupphaug, and Bjarne A. Foss. Infeasibility Handling in Linear MPC Subject to Prioritized Constraints. In *Proceedings of the 14th World Congress, IFAC*, IFAC conference proceedings, pages 163–168. Published for the International Federation of Automatic Control by Pergamon, 1999. ISBN 9780080427553.

Jostein Vada, Olav Slupphaug, and T. A. Johansen. Optimal Prioritized Infeasibility Handling in Model Predictive Control: Parametric Preemptive Multiobjective Linear Programming Approach. *Journal of Optimization Theory and Applications*, 109(2):385–413, 2001.

Z. Vafa and S. Dubowsky. The Kinematics and Dynamics of Space Manipulators: The Virtual Manipulator Approach. *The International Journal of Robotics Research*, 9(4):3–21, 1990.

A. Vivas and V. Mosquera. Predictive Functional Control of a PUMA Robot. In *ACSE-05 Conference Proceedings*, pages 35–40. ICGST, 2005.

I.D Walker. Impact configurations and measures for kinematically redundant and multiple armed robot systems. *IEEE Transactions on Robotics and Automation*, 10(5):670–683, 1994.

M. W. Walker and D. E. Orin. Efficient Dynamic Computer Simulation of Robotic Mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205, 1982.

Donald M. Waltz. *On-orbit servicing of space systems*. Orbit, a foundation series. Krieger Pub. Co., Malabar and Fla, original ed. edition, 1993. ISBN 9780894640025.

H. Wang and Y. Xie. Passivity based adaptive jacobian tracking for free-floating space manipulators without using spacecraft acceleration. *Automatica*, 45(6):1510–1517, 2009.

Liuping Wang. *Model predictive control system design and implementation using MATLAB ®*. Advances in industrial control. Springer, London, op. 2009.

Wikipedia. On-Orbit Servicing, 2014. URL de.wikipedia.org/wiki/On-Orbit_Servicing.

W. Xu, Y. Liu, B. Liang, Y. Xu, C. Li, and W. Qiang. Non-holonomic path planning of a free-floating space robotic system using genetic algorihtm. *Advanced Robotics*, 22(4):451–476, 2008.

Y. Xu, H.-Y Shum, J.-J Lee, and T. Kanade. Adaptive control of space robot system with an attitude controlled base. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2005–2010. IEEE Comput. Soc. Press, 1992. ISBN 0-8186-2720-4.

Y. Xu, Y. Gu, Y. Wu, and R. Sclabassi. Robust control of free-floating space robot systems. *International Journal of Control*, 61(2):261–277, 1995.

Yangsheng Xu. *Adaptive control of space robot system with an attitude controlled base*, volume CMU-RI-TR-91-14 of *Technical report. Carnegie Mellon University. The Robotics Institute*. Carnegie Mellon University, the Robotics Institute, Pittsburgh and Pa, 1991.

Yangsheng Xu and Takeo Kanade. *Space robotics: Dynamics and control*, volume 188 of *The Kluwer international series in engineering and computer science*. Kluwer Academic Publishers, Boston, 1993. ISBN 9780792392651.

Yangsheng Xu and Heung-Yeung Shum. *Dynamic control of a space robot system with no thrust jets controlled base*, volume CMU-RI-TR-91-33 of *Technical report. Carnegie Mellon University. The Robotics Institute*. Carnegie Mellon University, the Robotics Institute, Pittsburgh and Pa, 1991.

K. Yoshida. A general formulation for under-actuated manipulators. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97*, pages 1651–1657. IEEE, 1997. ISBN 0-7803-4119-8.

K. Yoshida. The SpaceDyn: a MATLAB toolbox for space and mobile robots. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, pages 1633–1638. IEEE, 1999. ISBN 0-7803-5184-3.

K. Yoshida. Engineering Test Satellite VII Flight Experiments for Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit. *The International Journal of Robotics Research*, 22(5):321–335, 2003.

T. Yoshikawa. Analysis and Control of Robot Manipulators with Redundancy. In M. Brady and R. Paul, editors, *Robotics Research The First International Symposium*, pages 735–747. MIT Press, 1984.

Yunong Zhang, Jun Wang, and Youshen Xia. A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits. *IEEE Transactions on Neural Networks*, 14(3):658–667, 2003.

A. Zheng and M. Morari. Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, 40(10):1818–1823, 1995.

L. Zlajpah and B. Nemec. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and System*, pages 1898–1903. IEEE, 2002. ISBN 0-7803-7398-7.