TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Echtzeitsysteme und Robotik

A New Model To Design Software Architectures For Mobile Service Robots

Martin Wojtczyk

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Darius Burschka

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Alois Knoll
2. Univ.-Prof. Dr. Bernd Radig (i. R.)

Die Dissertation wurde am 7.10.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 8.12.2010 angenommen.

Zusammenfassung

Sowohl Roboter als auch Personal Computer haben neue Märkte generiert und wurden Mitte der 1970er Jahre zu Massenprodukten. Sie wurden zu Schlüsseltechnologien in der Automation und Informationstechnik. Während man jedoch Personal Computer heutzutage in fast jedem Haushalt findet, werden Roboter hauptsächlich im industriellen Umfeld eingesetzt. Aufgrund der physikalischen Wirkungsmöglichkeiten eines Roboters, ist ein sicheres Design essentiell, das den meisten heutzutage hergestellten Manipulatoren immer noch fehlt und so deren Einsatz für den persönlichen Gebrauch verhindert. Es ist jedoch ein neuer Trend feststellbar. Immer mehr Roboter werden für die Ausführung spezieller Dienste in mit Menschen geteilten Umgebungen entwickelt. Diese Art Roboter wird gemeinhin als Service Roboter bezeichnet.

Die Entwicklung der am Lehrstuhl für Echtzeitsysteme und Robotik der Technischen Universität München entstandenen Service Roboter ist durch verschiedene reale Anwendungsszenarien für autonome mobile Roboter in Biotechnologielaboren, veränderlichen Fabriken, TV Studios und für den ausbildungs- als auch den persönlichen Gebrauch motiviert. Im Gegensatz zu industriellen Manipulatoren, sind die meisten Service Roboter mit weitaus mehr Sensorik und Rechenkraft ausgestattet, um ihre Umwelt wahrzunehmen und die ermittelten Sensordaten für autonomes Verhalten auszuwerten. Die Vielfalt der verwendeten Hardware und die sehr unterschiedlichen Anwendungsfälle für Service Roboter machen aus ihnen komplexe, heterogene und verteilte IT Systeme. Um die Neuentwicklung von systemspezifischen Softwarearchitekturen für jede neue Service Roboter Variante zu vermeiden, ist es notwendig Softwarekomponenten und ihre Schnittstellen zu standardisieren.

Diese Dissertation stellt daher ein neuartiges Modell vor, um die Hard- und Softwarekomponenten autonomer Service Roboter zu klassifizieren und diskutiert ihre Schnittstellen, Generalisierungen und Spezialisierungen. Ein großer Teil dieser Arbeit ist dem Design und der Implementierung verschiedener Wahrnehmungsmodule gewidmet, da diese für Service Roboter essentiell sind. Zusammengefasst umschließt das Modell Sensoren, Aktuatoren, die entsprechenden Busse und Netzwerke sowie die darüberliegenden Software Gegenstücke für Kommunikation, Geräteklassen und die Softwarekomponenten für Wahrnehmung, Planung und Applikationen. Der Ergebnisteil präsentiert die erfolgreiche Anwendung des entwickelten Modells in realen Service Roboter Projekten die an unserem Lehrstuhl entwickelt worden und Stand der Technik sind.

Abstract

Both Robots and Personal Computers established new markets and became mass-products in the mid-1970s. They were enabling factors in Automation and Information Technology respectively. However, while you can see Personal Computers in almost every home nowadays, the domain of Robots is mostly restricted to industrial automation. Due to the physical impact of robots, a safe design is essential which most manipulators still lack of today and therefore prevent their application for personal use. A slow transition can be noticed however by the introduction of dedicated robots for specific tasks in environments shared with humans. These are classified as service robots.

TUM's Department for Robotics and Embedded Systems approach to service robotics was driven by several real world application scenarios for autonomous mobile robots in life science laboratories, changeable factories, TV studios and educational as well as domestic application domains. Opposed to manipulators for industrial automation, most service robots carry much more sensor equipment and computing power to perceive their environment and to process the acquired sensor data for autonomous behaviour. The variety of utilised hardware and the versatile use cases for service robots turn them into complex, heterogeneous, and distributed IT systems. To avoid inventing custom software architectures for every newly created service robot, standardisation of software components and interfaces is key for their development.

This thesis proposes a novel model to classify the hard- and software components of autonomous service robots and discusses their interfaces, generalisations, and specialisations. A large part of this work is dedicated to the design and implementation of perception modules as they are essential for service robots. In summary, the model covers the sensors, the actuators and the corresponding busses and networks as well as the overlying software counterparts for the communication channels, device classes, and the software components for perception, task planning, and applications. The result section discusses its successful application in state of the art projects developed at our department.

Acknowledgement

I want to thank my advisor Professor Alois Knoll, for giving me the opportunity to prepare this thesis and for supporting my work with ideas, interesting and challenging projects, criticism, and guidance.

I thank Dr. Gerhard Schrott for valuable advice throughout the past years.

Thank you to Amy, Gila, Monika, and all the colleagues at the Department for Robotics and Embedded Systems at the Technische Universität München for everyones' support, collaboration, motivation and friendship.

Special thanks to Dr. Rüdiger Heidemann from Bayer HealthCare for strongly supporting my projects and my work in Berkeley, California.

Many thanks also to: Chetan Goudar, Paul Wu, Harald Dinter, Chun Zhang, Klaus Joeris, Mark Burnett, Tom Monica, Konstantin Konstantinov for supporting my main project at Bayer HealthCare: the mobile robot for the biotechnology lab. I thank Mehdi for helping with experiments, Carol for the chocolate supplies and everyone else for being great colleagues and friends.

I am very grateful for the substantial funding of the lab automation projects by Bayer HealthCare and for having had the opportunity to work on site in Berkeley, California.

Special thanks to my parents Willi and Monika as well as my brothers Christoph and Michael for their support and for being the best family.

Finally, many thanks to Devy, for her patience and understanding, for taking care of me, cooking numerous delicious dinners and making sure that I eat three times a day, for her support and encouragement during this time.

Contents

Chapter 1

Introduction

Personal Computers and Robots both have revolutionised our modern lives since they became mass-products in the mid-1970s. Personal Computers dramatically changed the way and speed of how we process information, be it at work or at home. Robots on the other hand revolutionised the production of mass products along conveyor belts in big factories and allow quick program controlled customisation of products up to a certain extent. Yet their impact for domestic use is currently limited. Similar to the evolution of Personal Computers from big mainframes to nearly every home and office, the existence of robots in the daily lives of common people is on the horizon.

Service Robots are intended to carry out tasks for human beings. As opposed to industrial factories where robots are located in dedicated work cells, service robots are commonly expected to share their workspace with humans in homes or labs. The shared workspace however raises new challenges, since a service robot often has to deal with large and changing environments. They need to be equipped with the necessary safety measurements for a shared workspace and sensors to acquire data about their surroundings. They need to localise themselves in unknown environments and to locate and detect every day objects for the personal use of humans. A service robot's controlling computer spends most of its time on the acquisition and moreover the processing and interpretation of sensor data.

At our department the development of Service Robotics was driven by several application scenarios. After an introduction to these scenarios in chapter 2, related work is presented in chapter 3. Chapter 4 – Systems Design addresses the complex task of developing sustainable software components for Service Robots. For this reason I discuss important design principles in Software Engineering and propose a new model for the classification of robotic hard- and software components. The model is applied to mobile service robots, that I have personally worked on and that we developed with our industrial partners. Afterwards I present several methods to overcome different challenges in the context of mobile service robots – from collaboration in developer teams and cross platform aspects to robotic perception algorithms for changing environments. Finally, in the Results chapter I present solutions that address the given application scenarios and that have been implemented in real world applications.

Chapter 2

Requirements

Requirements analysis is the task of determining the needs and conditions for a new or altered product. Requirement analysis is critical to the success of a development process and must be related to identified business needs or opportunities and defined to a level of detail sufficient for systems design. The following application scenarios demonstrate common challenges and requirements for mobile service robots. These scenarios are not and are not supposed to be complete and to cover the entire field of mobile service robotics. However they represent frequently addressed applications, that I personally came across during his work.

2.1 Biotech Lab Scenario

Biotech laboratories are a very dynamic working environment, although several of the tasks in a life science lab are simple and repetitive and could hence be carried out by a mobile service robot freeing up highly paid scientists for more important work and experiments.

Sample management for example is an inevitable and time-consuming part during the development and production of biopharmaceuticals to keep track of growth parameters and to adjust these as it becomes necessary. Sampling and maintenance of cell culture processes are labor intensive and especially continuous perfusion operations require constant monitoring on a 24/7 basis. Production setups in this kind of labs are very large and a robot would has to travel long distances between distinct bioreactors and analysis devices. This challenge implies a mobile robot platform, as long as the given processes can not be optimised for short distances.

To support human lab personnel, the robot needs to be able to carry out the same tasks as a human and to serve the same or at least similar devices, without changing the entire lab equipment. Thus a robot for this application should be an autonomous, mobile platform capable of localising, navigating and avoiding dynamic obstacles. The robot should be powered by batteries to make it independent of any power supply for a certain period of time. It should have a manipulator to enable the robot to pick up, carry and place different sizes of sample vials even in close-packed areas. Precise interaction with lab devices should be possible without damaging them by physical contact.

Figure 2.1 shows a typical life science laboratory for cell culture development. Noticeable are in particular the numerous moveable obstacles. Their positions can change several times a day, asking for highly adaptive robots.

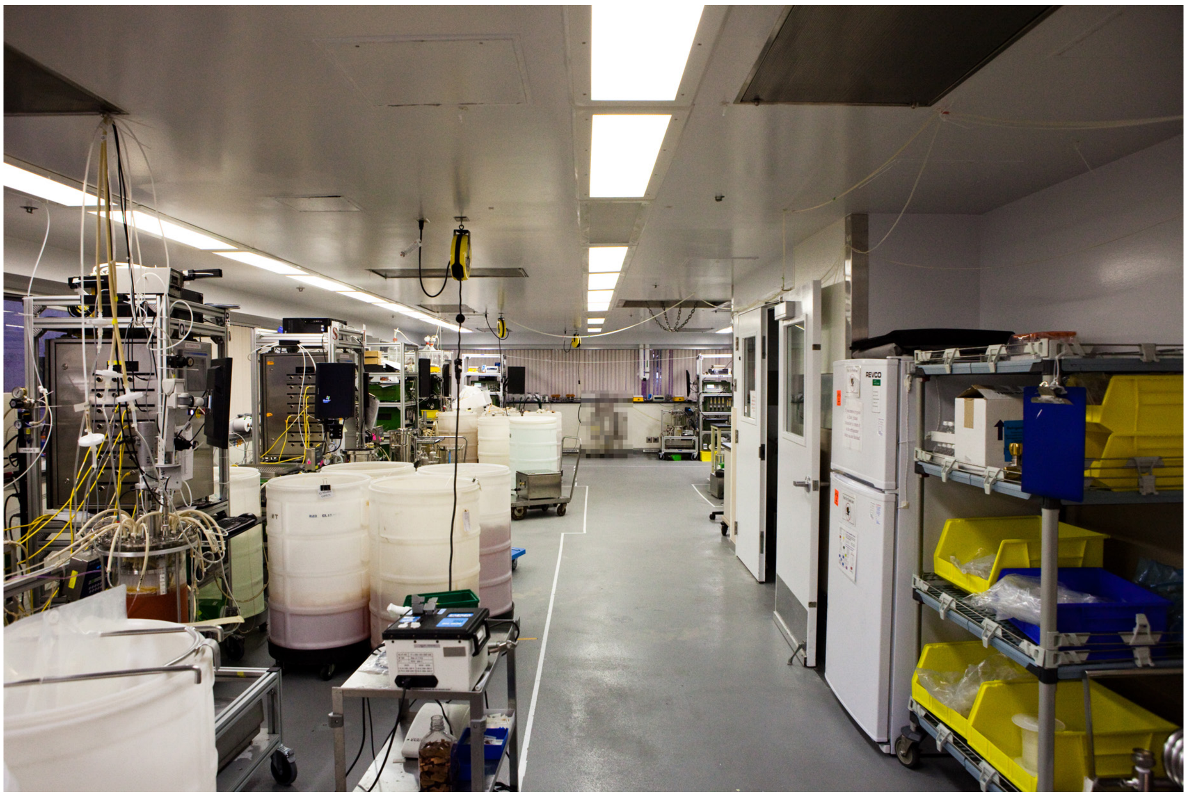


Figure 2.1: Typical environment in a life science laboratory with many moveable obstacles.

2.2 Blood Analysis Scenario

Blood analysis laboratories get blood samples at unpredictable time intervals – even at night. However hospitals can usually not afford nightshift coverage for possibly arriving blood samples. Therefore the blood analysis labs are often shut down at night. Based on previous work in a biotech laboratory, the idea came up, to utilise an autonomous mobile manipulator to carry out the blood analysis process in an institute for clinical chemistry. This way, blood samples could potentially be processed on a 24/7 basis.

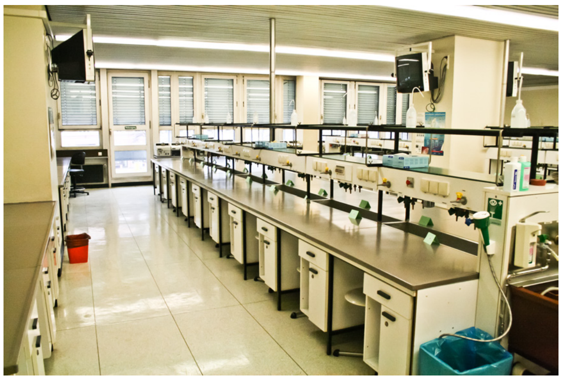


Figure 2.2: Typical blood analysis laboratory.

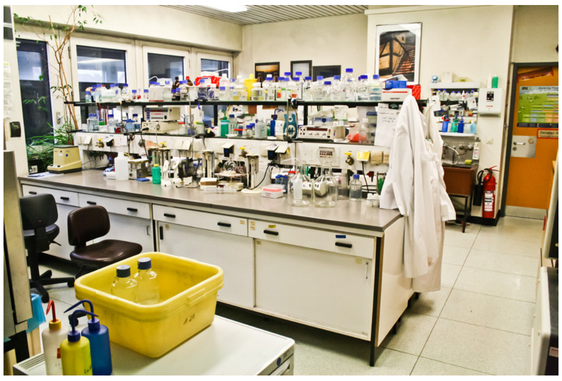


Figure 2.3: Another room in a typical blood analysis laboratory.

The requirements in the blood sample laboratory are very similar to the biotech laboratory: the robot should be able to perform the same tasks as humans and should serve the same or at least similar devices, without the need for changing the whole infrastructure in the laboratory. The robot should be autonomous and mobile and capable of localising itself within its environment. It should navigate safely and avoid dynamic obstacles and should work independently for a certain period of time from any power supply. It would require a manipulator with the necessary sensor equipment to precisely pick and place different sizes of sample vials without causing any damage by physical interaction.

In contrast to the biotech laboratory scenario however, where samples can potentially be retaken within a short time frame in case something goes wrong during the analysis process, blood samples are often delivered over far distances by couriers and repetition of the sampling process would be cumbersome and require a certified doctor or nurse to see the patient again. Therefore the blood samples require special care.

Figures 2.2 and 2.3 show pictures of a representative clinical analysis laboratory, Figure 2.4 shows a typical blood sample container and its content – a variety of barcoded vials in a plastic bag and a sample submission form. Figure 2.5 depicts one of many analysis devices in the laboratory.

2.3 Surveillance Scenario

Most surveillance robots, that are currently on the market, are remotely controlled by a human operator. Autonomous or semi-autonomous surveillance robots, patrolling a defined area by themselves on a 24/7 basis and analysing its environment with dedicated algorithms would be a dramatic progress. In addition to that, mobile indoor robots for laboratories or households would allow remote users to keep an eye on their experiments or equipment on demand.

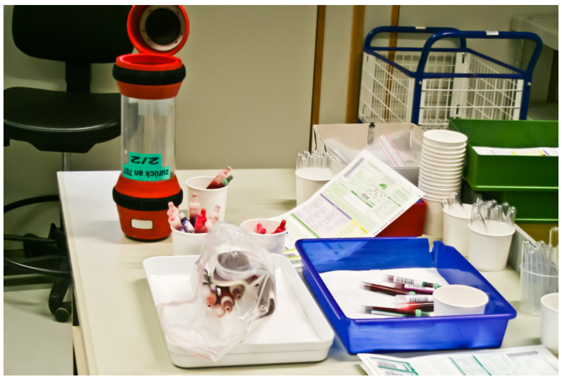


Figure 2.4: Blood sample container, barcoded sample vials, and sample submission form.

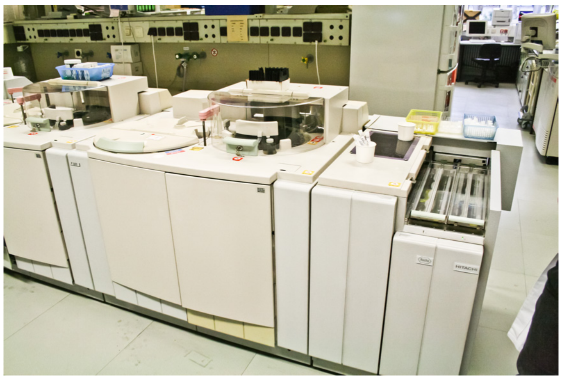


Figure 2.5: One of many blood analysis devices in the lab.

To define the requirements for surveillance robots, it is important to know its intended application domain, whether the surveillance tasks are carried out indoors, outdoors, submarine, aerial or above ground. The application domain predetermines the actuation and drives of the vehicle. Submarine robots most likely require propeller screws and rudders, aerial robots require airscrews or artificial wings while ground vehicles need artificial legs, caterpillar drives or wheels with the proper dimensions to cope with the environmental conditions. Combinations of GPS, laser range finders or ultrasonic sensors help surveillance robots to localise and navigate, cameras for visible and infrared light can record and monitor objects of interest.

While a manipulator is not imperatively necessary for surveillance tasks, a camera that is mounted on a robot arm provides additional flexibility and degrees of freedom. In addition to that a possibly attached gripper tool allows an autonomous robot or a remote user basic interaction with the environment. Autonomous surveillance vehicles are commonly driven by electronic motors and batteries or by combustion engines with gas tanks for energy storage.

2.4 Changeable Factory Scenario

Many optimised production facilities have a static layout along conveyor belts and robotic work cells nowadays and are specified for a predetermined throughput. These static production facilities however don't reflect possible market fluctuations or only up to a certain extent. A possible modular and changeable factory could rearrange its workstations and storages itself to minimise transportation distances, to move required resources to the consuming machines and to adapt to changing storage requirements. Mobile service robots could be the transportation link between self organising workstations and storage areas.

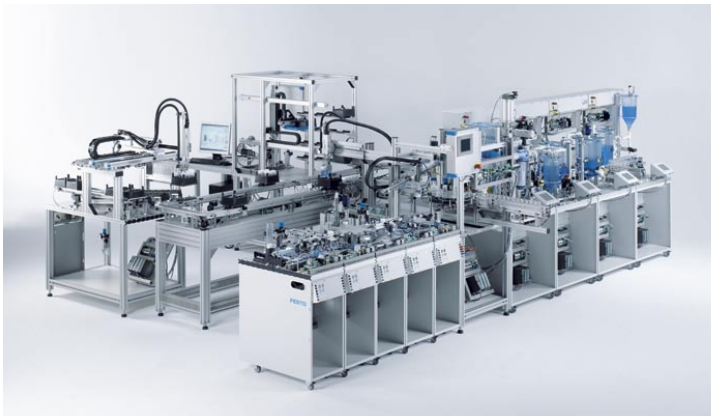


Figure 2.6: Changeable Factory consisting of several Festo Modular Production System workstations. Photograph: Festo Didactic.

The institute for industrial manufacturing and factory management of the university of Stuttgart (Institut für Industrielle Fertigung und Fabrikbetrieb der Universität Stuttgart)[1] is researching on digital and virtual tools for factory planning that are coupled to a physical changeable model factory to demonstrate the virtual planning in the real world. The physical model factory is implemented through the production system "iTRAME" manufactured by Festo Didactic GmbH[2]. The modules can be connected by conveyor belts, however as an alternative transportation method, an autonomous mobile robot platform is requested in the context of this project to dynamically overcome larger transportation distances in the changeable factory [23, 24].

A mobile platform for transportation tasks in a factory environment needs to meet similar requirements as in a laboratory in terms of autonomous localisation and navigation. When handing over factory goods from or to the mobile platform, an attached manipulator can increase the flexibility and reduce the necessary precision for docking a platform onto a factory station. If the platform is not equipped with a manipulator, the handover procedure demands a much higher precision of the localisation and navigation module.

2.5 Housekeeping Scenario

Housekeeping is a versatile application domain for service robots. There are many cumbersome and repetitive activities that people are willing to pay staff for. Research and development addresses these kind of tasks with service robots from two sides. Some teams work on dedicated service robots, designed to carry out a a single specific task. Other research teams work on versatile, more complex, and human-like robotic assistants, for one single machine to carry out all desired tasks.

[1]http://www.iff.uni-stuttgart.de/
[2]http://www.festo-didactic.com/

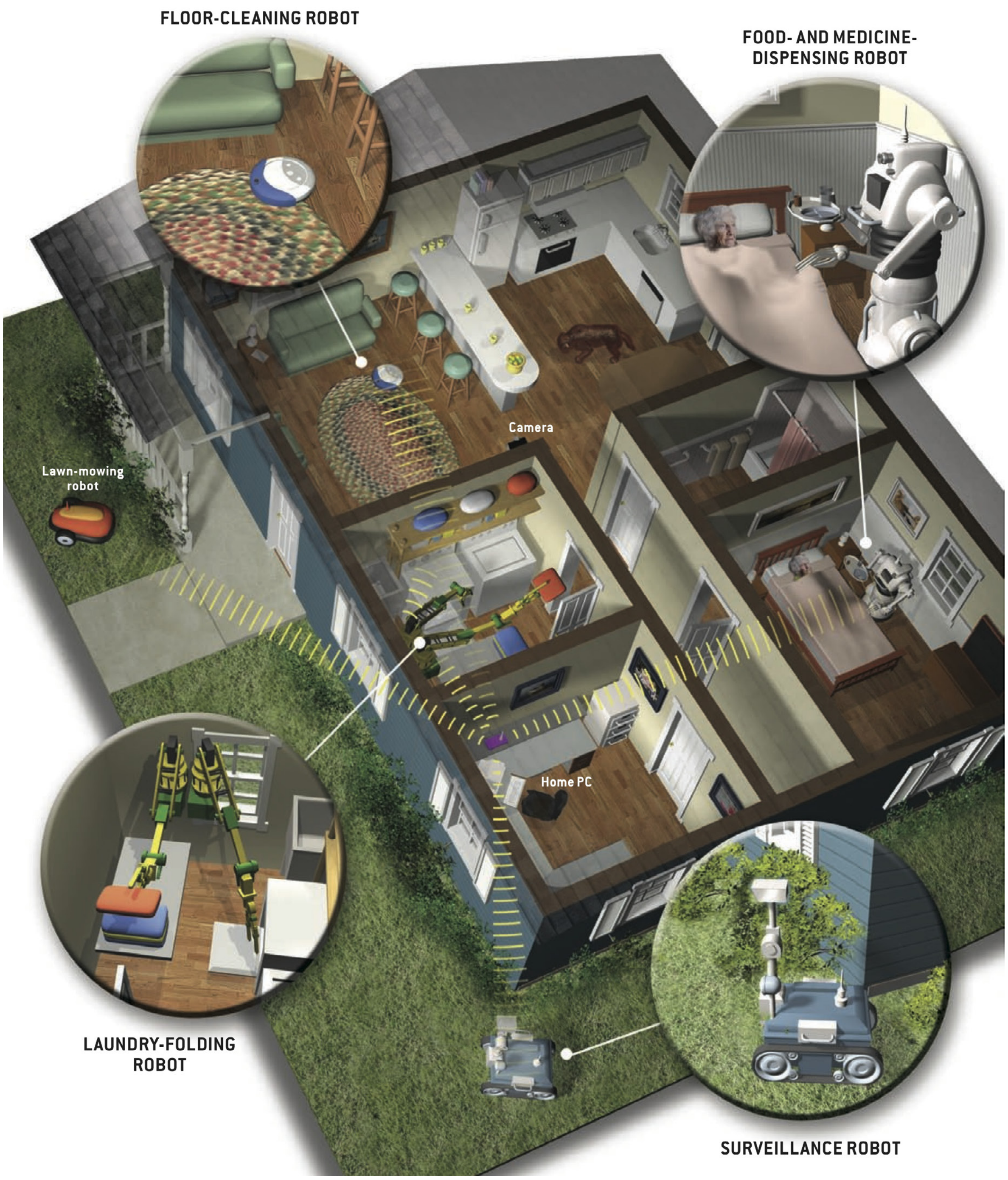


Figure 2.7: Possible applications for domestic service robots are manifold and cover outdoor and indoor tasks. Robots for domestic utilisation can be dedicated devices such as lawn-mowing, surveillance-, vacuum-cleaning, dish-washing or laundry-folding robots, or by far more complex, versatile and multipurpose human-like assistants. Image source: [25].

Dedicated service robots have been developed to clean floors, to mow the lawn, to clean windows or pools and to carry out surveillance activities. In addition to that experimental prototypes address laundry-folding and dish washing. Moreover there is a number of multipurpose robot prototypes intended to become human-like assistants, that are usually equipped with one or more manipulators similar to arms and something like a head. These human-like assistants would ideally be capable of understanding natural speech and gestures and carry out the desired task in the present context, just like a human assistant would.

Service robots for domestic use need to fulfil similar requirements as the previously mentioned lab robots. They have to be able to localise and navigate within an unknown and changing environment and to carry out the tasks, that they were designed for. While the environment in an industrial setup can at least be controlled to a certain extent, the customer of a housekeeping robot doesn't want to be restricted in his individuality and creativity when it comes to furniture, decoration, and utensils. In addition to that the price, the usability, and the reliability are important factors for domestic use, too. A domestic robot makes no sense if it is more expensive than human staff, that would do the same job more efficiently or if it requires a highly skilled technician for set up or for frequent repairs.

2.6 TV Studio Scenario

Virtual TV Studio sets for documentaries, news, and live productions commonly consist of a static set of furniture in a room with green walls, studio lights, and manually controlled TV cameras. The moderator of the TV production interacts within the static set and to the best of his knowledge within the virtual set. The quality of the Virtual Reality production highly depends on precise camera control, accurate determination of camera position and orientation, and virtual reality rendering. Figures 2.8 and 2.9 show a typical virtual reality TV studio and commonly utilised camera.

For decades, TV cameras have been mounted on movable stands such as ones produced by Shotoku[3], which allow the camera controller to bring cameras into position and enable direct control of parameters such as zoom and focus. These classical camera stands however also limit the directors' creativity for camera motions to the degrees of freedom of the utilised stands. The german company Robotics Technology Leaders GmbH (RTLeaders)[4] has addressed these limitations by introducing TV cameras mounted on mobile manipulators.

Opposed to the other mentioned application scenarios, the requirements for TV robots differ to a certain extent. For virtual reality productions and for high quality recordings the self localisation and localisation of the attached camera needs to be highly precise to enable accurate virtual reality rendering. Moreover the motions have to be very smooth, to avoid jitter in the movie records. In addition

[3]http://www.shotoku.co.jp/worldwide/
[4]http://www.rtleaders.com

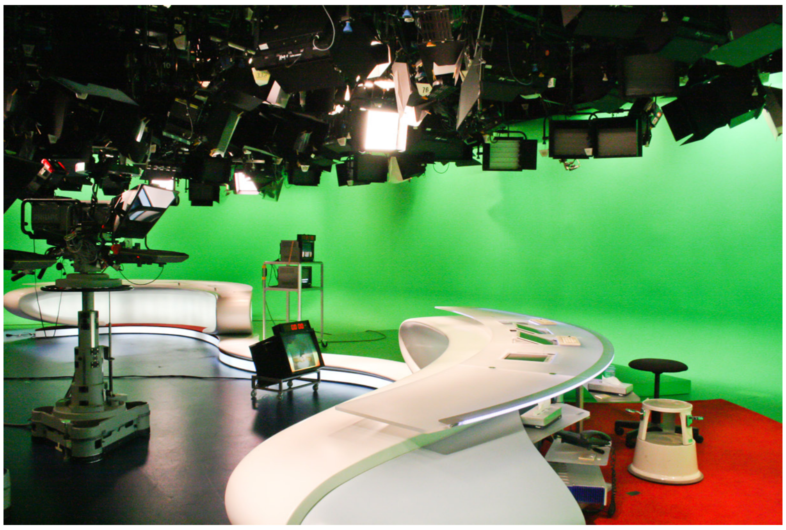


Figure 2.8: Virtual reality TV studio with static furniture, studio lights, TV camera and green walls.

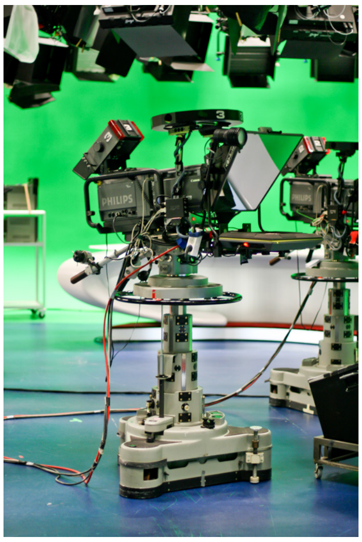


Figure 2.9: Commonly used TV Camera on a movable stand for virtual reality productions.

to that, the noise level has to be kept to a minimum, to enable synchronous live recording of the audio material as well. Since the manipulator is sharing the TV Studio with the moderator and behind the scenes staff, safety sensors are imperative to provide emergency stops whenever people enter the current work radius of manipulators.

2.7 Robotic Education Scenario

As robots are appearing more and more not only in industrial automation but also in emerging markets, educational institutions need to provide didactic resources to teach robotics, which in turn is a very interdisciplinary field covering at least informatics, electrical engineering and mechanical engineering (see Figure 2.10) and can possibly extend to any arbitrary field like medicine, aerospace, psychology or agriculture, just to mention a few – depending on the concrete application for robotics. Furthermore students should be aware of the physical impact, that robots can have on their environment and about safety regulations and standards such as ISO 10218-1:2006 – "Robots for industrial environments" and in particular section 5.10. – "Collaborative operation requirements" as elementary guides of how to design safe robots [26].

In the field of *Informatics*, robotic education systems and training material should at least teach basic knowledge in sensor interfacing, data acquisition, and -processing, to be able to use the attached hardware components on a robot, such as laser rangefinders, force-/torque sensors and cameras, just

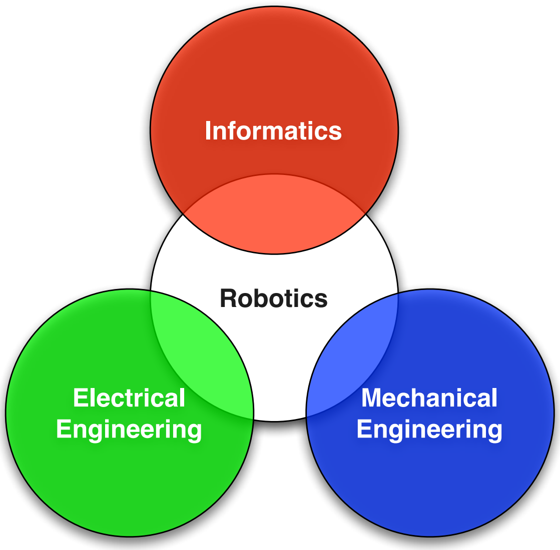


Figure 2.10: Robotics is a very interdisciplinary field and requires knowledge in Informatics, Electrical- and Mechanical Engineering among other disciplines depending on the concrete application.

to mention a few. Furthermore computer vision and image understanding became a very important field in service robotics, to enable robots to visually perceive the environment and changes herein. As service robots are often mobile and share the workspace with humans in an often cluttered and changeable environment, algorithms for autonomous behaviour such as collision-free path planning became imperative for navigating mobile robots, for moving manipulators, and for safely grasping objects with the attached tools. Moreover as service robots often consist of numerous microcontroller driven sensors and actuators, it is important to understand the underlying network infrastructure and communication interfaces.

With regard to *Electrical Engineering*, education programs should include basic knowledge about electricity, frequently used circuits, and electrical components, because things just break, as probably everyone has noticed, who worked on a complex service robot. The correct wiring of electrical components and proper soldering are skills that probably every robotocist will require at some point. An insight into the electrical control of drive units and basic knowledge in feedback control systems helps understanding the constraints and control of drives, servos, and many sensors.

In addition to informatics and electrical engineering, *Mechanical Engineering* plays an important role in robotics as every robot has a mechanical chassis and needs to carry a certain payload. Robotic education should cover basics about materials, drives, gears and the mechanical construction of robot systems and extension components.

Robotic education systems and training material should at least teach basic knowledge in:

- Safety
- Informatics
  - Sensor data acquisition and processing
  - Computer vision and image understanding
  - Communication interfaces and networks
  - Algorithms for Autonomy
- Electrical Engineering
  - Control of drive units
  - Wiring of electrical components
  - Feedback control systems
- Mechanical Engineering
  - Mechanical construction of a (mobile) robot system
  - Construction and mounting of robotic extensions

2.8 Summary

This chapter briefly described several different application scenarios for autonomous mobile service robots that I was personally confronted with and which shall be addressed in the following sections. When comparing the needs for the different scenarios, several particularities are noticeable due to the diversity of the application domains, nevertheless there are also challenges for robots, that all the scenarios share.

All scenarios have in common, that larger distances have to be travelled in changeable environments – at least distances, that stationary off-the-shelf manipulators would not be able to cover just by their workspaces. This asks for autonomous mobile robot platforms, that are able to come close to the locations of interest. Mobility however is attended by the challenge of self-localisation and autonomous, collision-free path planning and navigation in an unknown and changing environment. Since the space in the discussed scenarios is potentially shared with humans the robotic platforms need to be able to deal with dynamic obstacles as well to prevent damages and injuries.

However the scenarios also differ in some aspects. Some scenarios imply a platform with a many degree of freedom manipulator, whereas some others don't require a manipulator at all, in other cases a robotic arm is optional and can help overcome imprecise navigation. Robots for the laboratory scenarios and versatile domestic robots call for manipulators to precisely interact with the physical world and to safely pick and place objects. Also the precision requirements for the localisation and navigation differ from case to case. While the TV studio robot requires a highly precise localisation, the surveillance robot can deal with a more imprecise localisation as long as the locations of interest are covered by the onboard cameras.

Chapter 3

Related Work

This chapter reviews previous or similar work about the utilisation of mobile service robots in the formerly mentioned application scenarios. Since most of my work happened in the context of Lab Automation, related work is described more in detail for this scenario, while almost all other scenarios are well covered by established literature.

3.1 Lab Automation

Life Science laboratories are constantly being targeted by automation companies and research institutes, since every day lab work requires highly paid scientists to carry out many repetitive and cumbersome tasks. Many of these activities involve the handling and transportation of a series of samples or sample arrays and investigations on them through established lab devices. So far most approaches for lab automation addressed a certain task in the series of experiments and focused on increasing the throughput by quickly processing large arrays of uniform vials or sample arrays. Autonomous mobile service robots for life science laboratories could be the link between these dedicated lab devices and potentially free up some resources of the scientists to be able to focus on more important work and experiments.

3.1.1 Leonardo

Leonardo is a mobile service robot intended to assist human lab personnel in life science laboratories by taking over repetitive everyday tasks. The robot is a modular system and its tool was designed to handle sample vials in a cell culture laboratory to carry out a complete sample management process for a first application demonstration. Sample management is an important and time-consuming part during the development and production of biopharmaceuticals to keep track of growth parameters and to adjust these as it becomes necessary. Sampling and maintenance of cell culture processes are labor intensive and especially continuous perfusion operations require constant monitoring on a 24/7 basis. In 1998 this task has been addressed for the first time at the University of Bielefeld in Prof. Lehmann's and Prof. Knoll's groups involving a mobile manipulator [27, 28, 29].

After initial construction and development at the University of Bielefeld, the project was transferred to the Technische Universität München, Germany, where development continued in Prof. Knoll's newly formed department since 2002 [30]. The developed system is a battery driven mobile robot platform with laser range finders for localisation and a mounted robot arm with a camera, a force-/torque sensor, and a gripper attached to its tool for precise manipulation. Figure 3.1 shows the Leonardo robot as it was shipped to the customer in 2005.

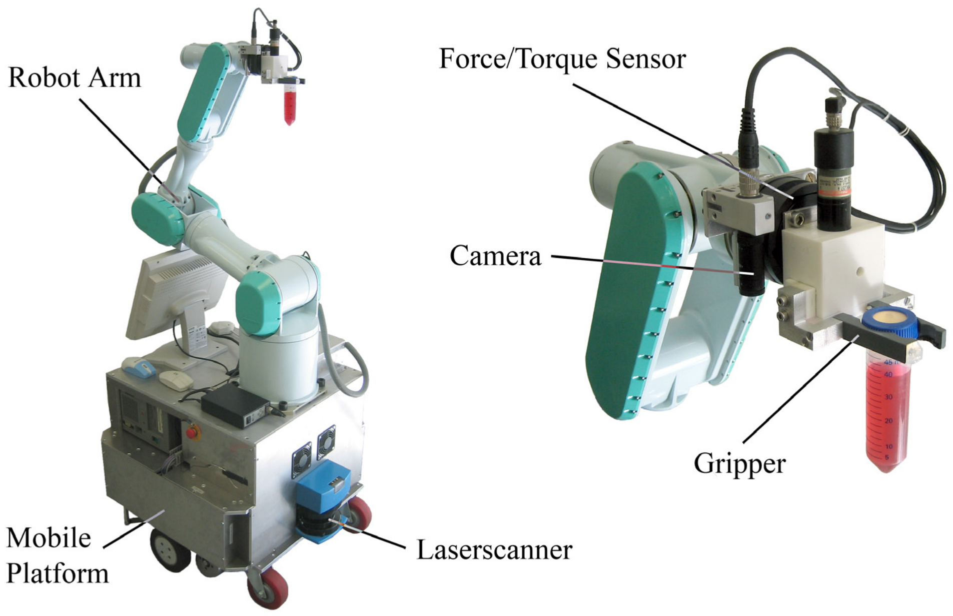


Figure 3.1: The Leonardo mobile robot platform as it was shipped to the customer.

3.1.1.1 Features

The software package addressed all necessary aspects for a pilot application in a biotechnology lab: accurate localisation, collision free path planning, and precise motion execution. The localisation is based on a known static, polygonal map and known landmarks, which are highly reflective tape stripes attached to walls and fixed objects. Furthermore the manipulator can be controlled precisely by a modified version of the *Robot Control C Library (RCCL)* to position the tool. The attached camera allows colour vision algorithms to recognise objects and to compute position displacements for visual servoing. The force-/torque sensor measures forces and torques applied to the gripper. This way, damage to lab equipment can be avoided by force controlled motions even if contact is required. All software components were integrated into an overlying scripting language to enable easier control of the platform, the manipulator, and the task sequences. Figure 3.2 depicts the developed software architecture installed on the platform.

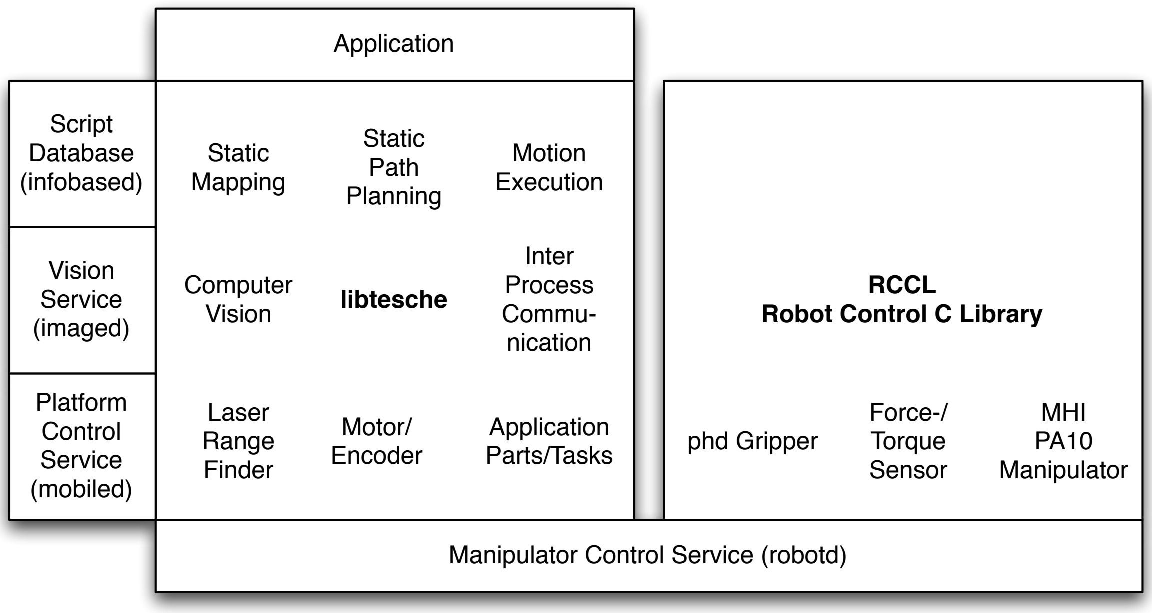


Figure 3.2: The previous software architecture of the Leonardo robot is consisting of two libraries: *libtesche* and *RCCL*, which comprise the functionality and key algorithms of the platform and the manipulator respectively. Several services and the executed applications build upon these libraries and communicate with each other.

Central elements are the two libraries *libtesche* and *RCCL* against which every application or service is linked. *libtesche* encloses the functionality of the platform hardware: the laser range finders, motors, wheel encoders and key algorithms such as localisation, mapping, path planning, motion generation, and computer vision. It also provides an inter process communication interface for different processes to exchange messages with each other to carry out tasks that involve several services. The modified *RCCL* contains the functionality to control the manipulator, gripper, and force-/torque sensor. Several daemons or services are built on top of these libraries. The Platform Control Service *mobiled* accesses the laser range finders and performs the accurate localisation and motion execution. The Vision Service *imaged* acquires images from the camera, detects known objects and computes position displacements. The Manipulator Control Service *robotd* moves the robot arm, opens and closes the gripper, and reads the force-/torque sensor to perform force controlled motions. The Script Database *infobased* contains positions of known devices, the color models for known objects, and task sequences to execute. An application links against *libtesche* to use the Inter Process Communication to send messages to the corresponding services and to trigger the task sequences. This way complex processes can be implemented. Experiments in the previously mentioned papers have shown good results in static laboratory environments.

3.1.1.2 Limitations

When investigating on the functionality of the software package, several limitations were encountered. The localisation and path planning are based on a precisely known, static map with landmarks. Sufficient landmarks have to be visible most of the time to ensure precise motions of the platform. If the robot encounters dynamic objects, it would stop until the obstacle is moved out of its way. Dynamic obstacles can potentially also cover some of the landmarks and confuse the localisation. The software architecture and image processing relies a lot on precisely known objects. While carrying out a sample management process, the robot would approach known lab devices, move the arm into a position where it would expect the camera to see a colour patch, the vision algorithm thereafter computes a two dimensional position and orientation displacement to accurately place the manipulator's tool into a known position and afterwards a sequence of tasks is performed. If at any point of time, the platform does not move precisely enough to put the camera into a position where a colour patch is visible, the result is undefined. These issues can be addressed with more complex task descriptions and step by step approaches to points of interest, which however would increase execution time to some extent.

3.1.1.3 Weaknesses

A closer look at the software architecture reveals some of its weaknesses. Although the different services leave an impression of modularisation, they are just processes accessing the functionalities of either of the two big libraries *libtesche* and *RCCL* resulting in an almost monolithic architecture. Applications need to link against *libtesche* as well to be able to access the Inter Process Communication and to transmit messages to the different services. If a hardware or software component needs to be replaced, changes have to be performed at multiple locations in the libraries and sometimes in the services and the applications as well. Replacing the static mapping or path planning by a more adaptive version would not be possible without reworking major parts of the library. Exchanging the computer vision algorithms would produce a similar burden. If functionality shall be added due to changes in the lab equipment, new tasks have to be defined in the script database, new message handlers have to be implemented in *libtesche* and new commands have to be implemented in the addressed services. In short, the software package performs well as long as it meets the requirements, however additional features or replacing key algorithms cannot be realised easily due to the monolithic software architecture.

3.1.2 LiSA

Noteworthy in the context of Lab Automation involving mobile service robots is LiSA – a Life Science Assistant, recently developed by the Fraunhofer Institute for Factory Operation and Automation IFF in Magdeburg, Germany since 2007. LiSA is "a mobile service robot that assists users in biological and pharmaceutical laboratories by carrying out routine jobs such as filling and transportation of microplates. The LiSA project is focused on integrating the latest research findings in a feasible service robot that meets the high level of safety demanded in real world human-robot collaboration" [31]. Figure 3.3 shows a picture of LiSA in a life science laboratory.

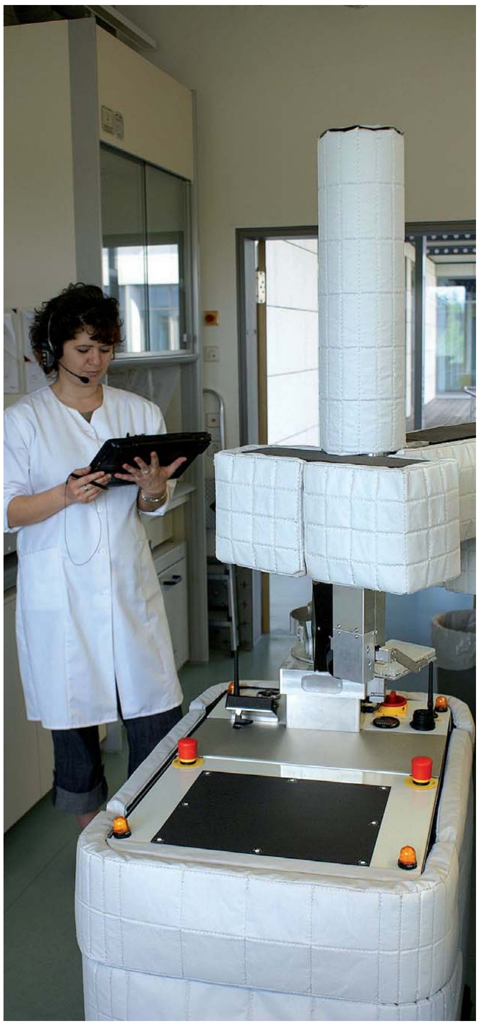


Figure 3.3: LiSA – Life Science Assistant. Photograph: Fraunhofer IFF.

3.1.2.1 Features

"Optical sensors identify and recognize the exchange positions and the microplates. To reliably position the gripper vis-à-vis objects being picked up, two digital cameras sample the immediate environment [...] The determination of exact 3-D position and orientation is based on a photogrammetric approach. Therefore, both digital cameras are used. Their positions and orientations are predetermined in a prior calibration step [...] A thermographic component is employed to ensure the safety of the manipulation process. This infrared component is part of a third combined camera device [and] is utilized to detect human interaction in front of the robotic arm and its gripper [...] The camera is mounted on a rotating stage and moves adaptively as the robotic arm moves."

"LiSA is designed to receive instructions directly from laboratory assistants. Their interaction with LiSA has to be intuitive, fast and easy. Interaction with LiSA is multimodal, i.e. spoken and touchpad input are possible. Speech recognition is speaker-independent. The commercial dialog engine used for LiSA supports mixed-initiative, natural language dialogs and conversation in full sentences."

"The main priority of [all project] objectives is safety, i.e. the robot may not harm any people or damage its environment." Therefore the "manipulator is covered by a pressure-sensitive artificial skin for collision detection. The skin's design enables localizing the collision area." Several laser scanners are combined to a 360°scanner, that serves two purposes: "localization in a prior map [... and] safety sensor [...] for constant avoidance of collisions with humans".

3.1.2.2 Limitations

Since I had no chance to personally experience the LiSA robot and its software package, little can be said about the robot's limitations and its behaviour in the real world. The utilisation of a classical SCARA manipulator however restricts the degrees of freedom and hence the versatility of the robot. The tool is designed to pick and place micro plates from and on to a planar surface like a table. Also it would be interesting to see, how task sequences are recorded and realised and how new lab processes can be implemented.

3.1.2.3 Weaknesses

Despite the safety features and the vision system, the given literature also points out a weakness of the implemented system: "The LiSA project does not include the autonomous exploration of the environment. In fact a prior navigation map of the environment is built for path planning and localization. Therefore, the environment is scanned in all three dimensions using the mobile robot Kurt3D [...] This robot is equipped with a 3-D laser scanner enabling it to obtain a three dimensional point cloud of the environment [...] This is achieved by iteratively taking 3-D scans and registering these scans to a consistent point cloud using a 6-D SLAM algorithm based on 'iterative closest points'. The point cloud is semi-automatically converted into a map for the USARSim robot simulation environment [...] The simulated environment is used to build a 2-D map for navigation (as a 2-D slice of the simulation refined manually) and is additionally integrated in the graphical user interface." While this separate and semi-automatical mapping process may perform well in static setups, real world experiments with the

Leonardo platform (see section 3.1.1) have shown, that a lot can change in an industrial life science lab during a single business day.

3.2 Surveillance

Surveillance has probably been among one of the first application scenarios for mobile service robots, either realised as simple remote controlled devices or as autonomous platforms. Mobile surveillance robots can be utilised in hazardous environments such as nuclear power plants or chemical production facilities, without exposing any humans to dangerous work spaces. Autonomous mobile surveillance robots can patrol predefined routes indoors or outdoors at certain periodic or unpredictable time intervals and provide a remote and mobile eye to home owners. Mobile surveillance robots have been developed by iRobot, Hitachi and others, however almost any mobile platform with cameras attached to it can be utilised as a surveillance platform.

3.3 Changeable Factory

Although autonomous mobile transportation vehicles have been in use in some modern car factories such as BMW in Dingolfing, Germany, their motions are restricted to follow installed inductor coils in the floor, which therefore prohibit their flexibility to adapt to a changeable factory layout as mentioned in the Changeable Factory Scenario (see section 2.4). To my knowledge, there has not been any previous development of mobile service robots in this field, that are capable of adapting to a changeable factory.

3.4 Housekeeping

Housekeeping with mobile robots has been addressed by several companies. Most efforts resulted in relatively cheap mobile cleaning robots, such as the: iRobot *Roomba*, Electrolux *Trilobyte 2.0*, Zucchetti *Orazio Plus*, Friendly Robotics *Friendly Vac*, RoboMop International *RoboMop*, Hanool Robotics *Ottoro*, LG *Roboking*, Samsung/Hauzen *VC-R560*, Yunjin *Iclebo*. Besides vacuum cleaners, also pool cleaning robots have been developed, such as the: Aqua Products *Aquabot*, Aquavac *TigerShark Pool Cleaner*, Maytronics *Dolphin Diagnostic 2001*, iRobot *Verro*. Outside the house, basic gardening tasks like lawn mowing have also been subject to robot development. In fact the first commercial lawn mowing robot was introduced to the mass market in 1995 – earlier than any vacuum cleaning robot. The Springer Handbook of Robotics contains a detailed survey of domestic service robots [32, pp. 1253–1282].

3.5 TV Studio

TV studio automation with a camera mounted on a robot manipulator has first been addressed by the german company Robotics Technology Leaders GmbH. They showed a first functional prototype of RoboKam at the IBC 2006 in Las Vegas. RoboKam is designed for live or automated control of film and TV cameras. It can be combined with a pan/tilt unit for additional degrees of freedom and fulfils highly precise localisation needs for virtual studio applications. Several RoboKam systems have been sold for use at RTL's n-tv news TV center in Cologne, and ZDF's recent news studio N1 in Mainz, Germany. RoboKam incorporates face and object tracking, which do not require a performer to wear dedicated tracking sensors. A director can record and combine an unlimited number of camera positions and motion sequences. Furthermore real-time motion control can be realised live through many degrees of freedom input devices such as 3Dconnexion's *SpaceBall*.

The GPS GmbH also developed a studio automation platform *Cinneo*, which is being utilised by Sat.1 Bayern in Germany since 2010.

3.6 Robot Education

Educational robot platforms fall into two categories: either expensive research platforms such as Honda's *Asimo* and Kuka's *youBot* or lower-cost and mainstream robot kits, such as the Lego *NXT*, Sony *Aibo*, or Aldebaran *Nao* [32, pp. 1283–1301]. Noteworthy in this context is also Festo's *Robotino* platform that evolved from *Robertino*, a development of Prof. Knoll's group.

3.7 Summary

This chapter gave a brief summary about robotic products, that are available on the market to address the application scenarios mentioned in chapter 2. While surveillance, housekeeping and robot education have been addressed by robot developers and companies for many years and are covered by common literature, lab automation and studiomation are more exotic application domains and only addressed by a few researchers and companies.

Chapter 4

Systems Design

Systems design is the process or art of defining the hardware and software architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Based on the requirements described in chapter 2 robot systems have been designed or selected and adapted to satisfy the needs in the given scenarios. After proposing a general description model for the components of robotic systems, the individual systems are described.

4.1 Design Principles

Different design principles used for the design specification can result in different system designs. Therefore it is important to state which design principles were followed, when creating a system design, to be able to eventually judge the quality of a system design.

4.1.1 Simplicity

"The simplest explanation or strategy tends to be the best one", is a common conclusion of Ockham's razor. Applied to software design, a software architecture should mimic the structure of the problem domain – if possible – and the intellectual distance between the software components and the real world problems should be minimised.

4.1.2 Modularity

Most complex problems can be broken down into smaller, manageable problems. Hence modularity is a direct result of applying simplicity to problem descriptions. It is the degree to which software can be understood by examining its components independently of one another. However things that belong together should also stick together. "Everything should be made as simple as possible, but no simpler" – Albert Einstein.

4.1.3 Reduced Dependence

Software systems are often designed with the goal to solve a *universal* problem. Therefore more and more features and more hardware support is included in a single software component and with each added feature or supported hardware the dependencies among software components or towards external libraries and hardware components grow stronger. The goal should be to strip down dependencies *to a minimum*, so that software components can be easily exchanged, easily compiled on a different platform and easily integrated into other projects. "Perfection is finally attained not when there is no longer anything to add but when there is no longer anything to take away" – Antoine de Saint Exupéry.

4.1.4 Interfaces

Software interfaces describe how the software elements communicate with each other, with other systems, and with human users. A good interface design should reduce the complexity of connections between modules and with the external environment. An interface should be simple, so developers or end users can easily develop software components for it, adapt to it or just use it. "Keep it simple and stupid" (KISS) - Clarence Leonard "Kelly" Johnson.

4.1.5 Data Models

Data model design requires at least as much attention as algorithm and interface design. Data structures represent the logical relationship among individual data elements. A good data structure should efficiently represent the appropriate data for a class of problems or a software component. Again, the same applies to data models like to interfaces: they should be kept as simple as possible, just like the interfaces, that use them.

4.1.6 Reusability

The amount of projects a software engineer can handle and the amount of software packages an engineer produces highly depends on the problems to solve, on the available time and tools and not least on the available software components that he can *reuse*. Quality of software or quality of code is often underrated and results in very customised program code, however reusability should be a major goal when developing software. This way a library of reusable software components is established and sustainable value is generated by the developers instead of reinventing the wheel over and over again.

4.1.7 Summary

Software engineering and systems design is a challenging task and should not be set equal to just programming. Many constraints need to be kept in mind and whole books and lectures are covering software and systems design for a good reason: software systems should be designed to create sustainable value [33, 34, 35]. In my opinion the above mentioned design aspects and principles can be summarised in two rules: *"Keep it damn simple!"* (KIDS) – so everyone can understand and use it, or *"Make it reusable!"* (MIR) – so it doesn't need to be done again. Moreover each of these two rules is inherently fulfilled when applying the other one. If a complex problem is broken down into small and simple components, these components are more likely to be reusable than a monolithic software package that does not cover unforeseen requirements. Software components, that are reusable tend to be easily adopted because they have such a simple design.

4.2 A Layer Model for Service Robots

Robotic Systems consist of a number of hardware components, such as motors, encoders, micro controllers, sensors, actuators and the different bus systems and interfaces to connect them to each other. Often enough I have personally experienced, that hardware components are randomly put together to fulfil a task in academia. Similarly the software packages are randomly put together or hacked afterwards, *as long as it works* and *as long as the deadlines are met*. This code is often produced in a more extreme way than even *Extreme Programming* suggests: by individuals as opposed to pairs of developers [36].

Unfortunately also often enough, the so produced code is not maintainable. As soon as new requirements raise, as soon as new features need to be implemented or new hardware needs to be included, the dedicated software package needs to undergo major changes, if parts of the package can be reused at all. *Reusability* however is one of the key goals software engineers should target at. A software component is only of sustainable value, if it can be used for multiple projects.

Hence I am proposing a layered model to classify hardware and software components of robotic systems to break down their complexity.

4.2.1 Hardware Layer

In this approach, the *Hardware Layer* consists of two sub layers: the Sensor/Actuator Layer and the Interface Layer, which makes the hardware components available to computer software.

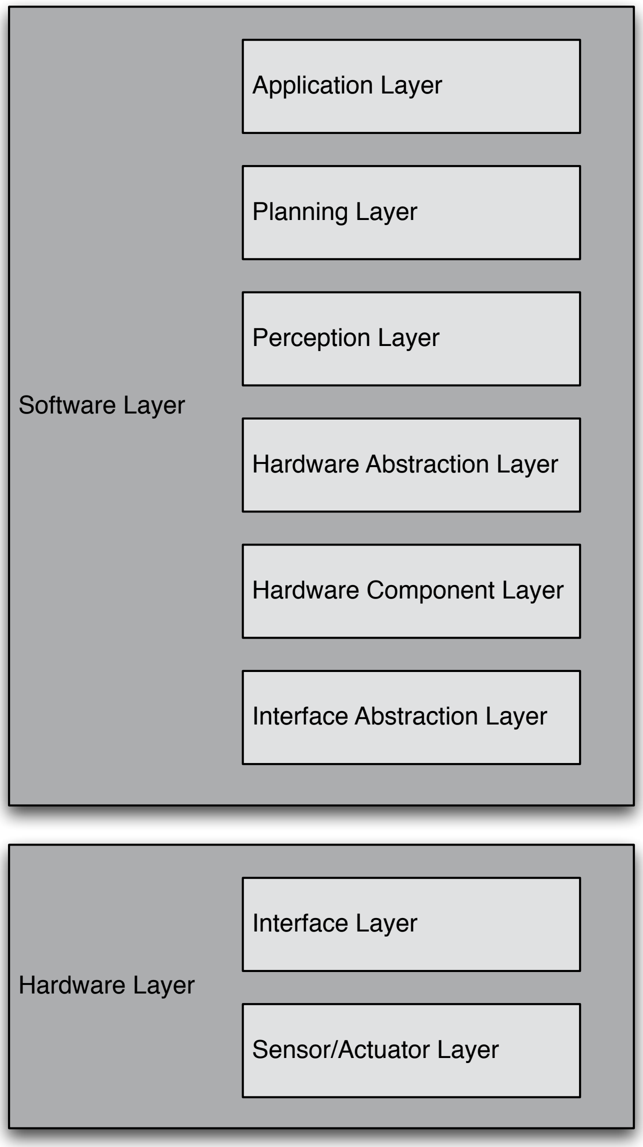


Figure 4.1: A Layer Model for Service Robots dividing the hardware and software components into functional layers. Ideally, the different layers talk to each other through standardised interfaces. This way hard- and software components can easily be exchanged.

4.2.1.1 Sensor/Actuator Layer

The *Sensor/Actuator Layer* accommodates the different kinds of hardware components available and characteristic for a robotic system to operate, mainly sensors and actuators.

Classical sensors in the application domain of robotics are position or wheel encoders to determine the position of a mobile platform or a manipulator. In addition to that, service robots are also often equipped with laser range finders and gyroscopes or other positioning systems, monocular or stereo cameras for computer vision, force-/torque sensors and microphones.

Common actuators in robotics are for example off the shelf manipulators as used for many years in industrial automation, electrical or pneumatic grippers or motors to transport a robotic vehicle.

4.2.1.2 Hardware Interface Layer

The *Hardware Interface Layer* classifies device interfaces and bus systems connecting the sensors and actuators to the controlling computer. Common examples are the *Controller Area Network (CAN)*, the *Universal Serial Bus (USB)*, *IEEE 1394 / Firewire*, *RS-232/422/485* serial interfaces, *Process Field Bus (Profibus)*, *Profinet*, or *Ethernet*, just to mention a few.

4.2.2 Software Layer

The Software Layer has two analogous counter parts to the Hardware Layer to reflect the hardware interfaces and hardware components. Moreover four additional layers accommodate hardware abstraction, robotic perception, planning and application functionality.

4.2.2.1 Interface Abstraction Layer

The *Interface Abstraction Layer* is the lowest software layer communicating to the hardware interfaces, which in turn are connected to the hardware components. Often hardware interfaces are shared among components on field busses or chained on serial interfaces, so the software component abstracting the hardware interface often needs to handle concurrent access on hardware components. Common examples for this layer would be the software counterparts of the hardware interface layer: CAN Bus-, Serial-, Firewire- or USB interface libraries and so forth.

4.2.2.2 Hardware Component Layer

The *Hardware Component Layer* classifies the software counterpart of the *Sensor/Actuator Layer*. The software components herein provide a software interface to the hardware components' functionalities and can basically be regarded as device drivers. Communication to the devices is established through standardised interfaces at the *Interface Abstraction Layer*. This way, hardware interfaces can easily be exchanged, if devices support this feature.

4.2.2.3 Hardware Abstraction Layer

The purpose of the *Hardware Abstraction Layer* is to provide a generic interface to classes of devices with similar features. This way certain hardware devices can easily be taken out of the system and exchanged by a more suitable one. Typical software components for this layer would be abstract interface libraries, to unify access to range finders, motors, cameras, manipulators and grippers for example.

4.2.2.4 Perception Layer

According to the *New Oxford American Dictionary* perception is "the ability to see, hear, or become aware of something through the senses". Robots generally, but in particular service robots need to be aware of their environment, to act and interact properly in it without causing damage and without being a safety hazard. Software components in the *Perception Layer* make robots aware of their environment on a lower level, by reading data from the connected sensors, such as images from cameras or distance information from laser range finders and on a higher level, by interpreting and potentially fusioning the acquired sensor data to create or update a world model.

4.2.2.5 Planning Layer

Based on the perceived environment and the desired tasks to be carried out, a robotic system may need to adjust its current behaviour and its path. For example, when moving in a changeable lab or in a changeable household, the robot needs to be able to drive around obstacles which just have been detected by the laser range finders and which haven't been there before. When manipulating in a cluttered environment, the path for the robot arm needs to be planned carefully, not to damage any equipment and not to harm any people. This asks for software components on the *Planning Layer*, that can plan tasks with regard to the created world model.

4.2.2.6 Application Layer

The uppermost software layer accommodates software packages that contain the application logic. Based on the desired tasks, different processes may ask the planning layer to create an appropriate path or manipulation sequence. A simple application may just allow to directly control the robot system through an appropriate input device and may provide safety features based on the acquired sensor data. Another common application is Teleoperation, where a human expert knows about the abilities of a robotic system and remotely controls the mobile manipulator to carry out a task. However in addition to that, *autonomous* applications open up completely new future perspectives in robotics. Repetitive and time consuming tasks, like surveillance walkthroughs or the sample management process in a life science laboratory can just as well be carried out by a mobile manipulator.

4.2.3 Summary

This section discussed the proposed layer model for service robots. The purpose for each of the hard- and software layers was explained and common examples were given for where components fit. I am proposing this model, to categorise robotic hard- and software components to make development with respect to the presented design principles (see section 4.1) easier. I am aware of the fact, that this model also requires additional overhead for creating and complying to standardised software interfaces, but – given my personal experience with monolithic software architectures I inherited with previous projects – I am confident, that it is worth the effort and that it enables the creation of sustainable software components. The following sections will address five service robots and their components as well as how the discussed model fits in their context.

4.3 Leonardo

The Leonardo robot is a mobile platform which we used to address the lab automation and surveillance scenarios. It is a mobile platform type MP-L655, produced by GPS GmbH in Stuttgart, Germany with a mounted Mitsubishi Heavy Industries *PA10-7* manipulator with seven degrees of freedom. It is a wheeled vehicle with two laser range finders for localisation, navigation and obstacle detection and is powered by batteries. This makes the platform independent of any power supply for up to nine hours. The kinematically redundant industrial robot arm with seven joints enables the robot to pick up, carry and place different sizes of sample vials even in close-packed areas. For precise interaction with the lab devices, the robot is also equipped with a camera to detect and localise objects like vials and analysis devices. A force/torque-sensor allows force-controlled motions to prevent any damage by physical contact. All the sensors and actuators are controlled by an on-board computer which can receive user commands or give image, or other sensor feedback to an observing station or to invoke analysis processes at certain stations in a life science lab via wireless network. Figure 4.2 shows the latest state of the Leonardo robot and its sensors and actuators. Figures 4.3, 4.4 and Table 4.1 summarise the physical dimensions of the robot.

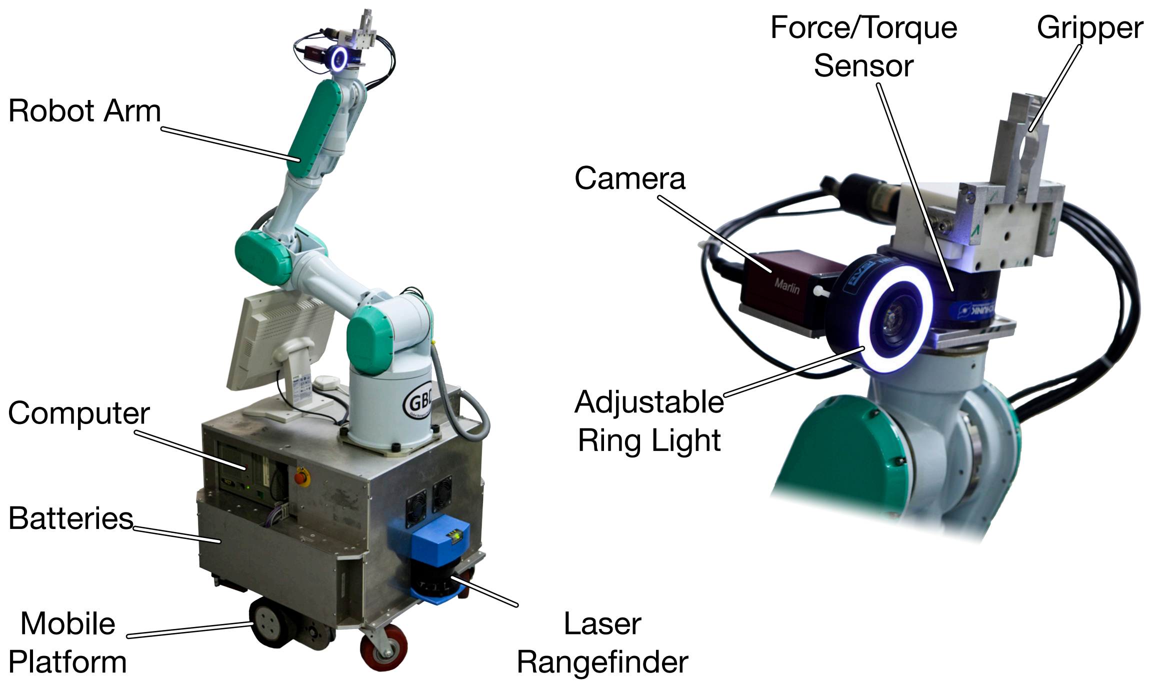


Figure 4.2: The Leonardo mobile robot platform with its sensors and actuators.

Length	796 mm
Width	656 mm
Height including housing	950 mm – 2056 mm
Overall weight	190 kg
Maximal payload	10 kg

Table 4.1: Dimensions of the Leonardo platform.

4.3.1 Robotic Components

The Leonardo platform moves and steers with two motorised wheels, which are located at the sides of the rotational axis. The motor controllers are connected to the robot's CAN-Bus interface. Two supporting wheels are located in the front and one supporting wheel is located at the back of the platform. Table 4.3 summarises the parameters of its differential drive system.

Attached to the platform front and rear face are two SICK *LMS 200* laser rangefinders for localisation and mapping that are interfaced through the RS-422 ports. The platform also has a gyroscope sen-

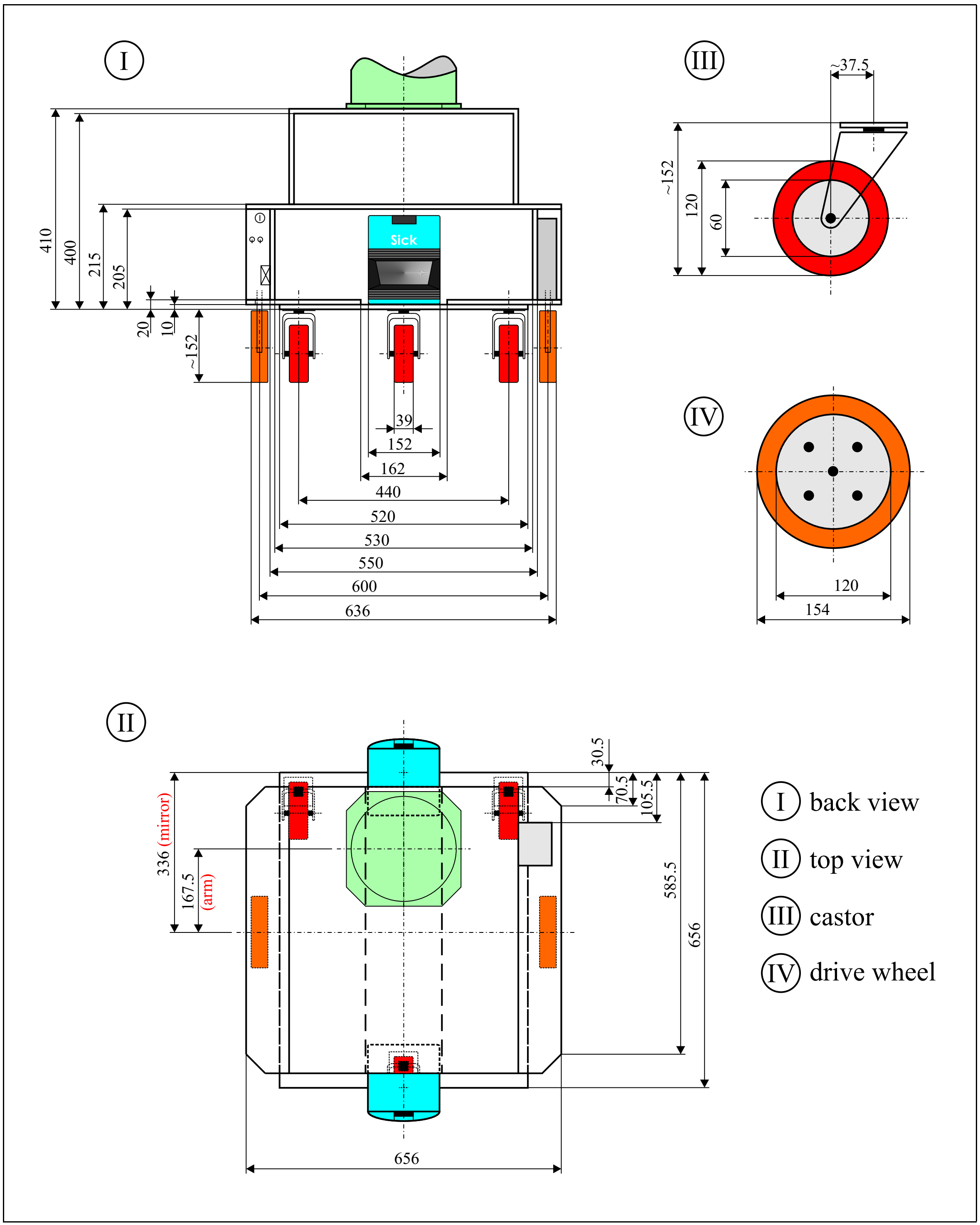


Figure 4.3: Dimensions of the Leonardo mobile robot platform [37].

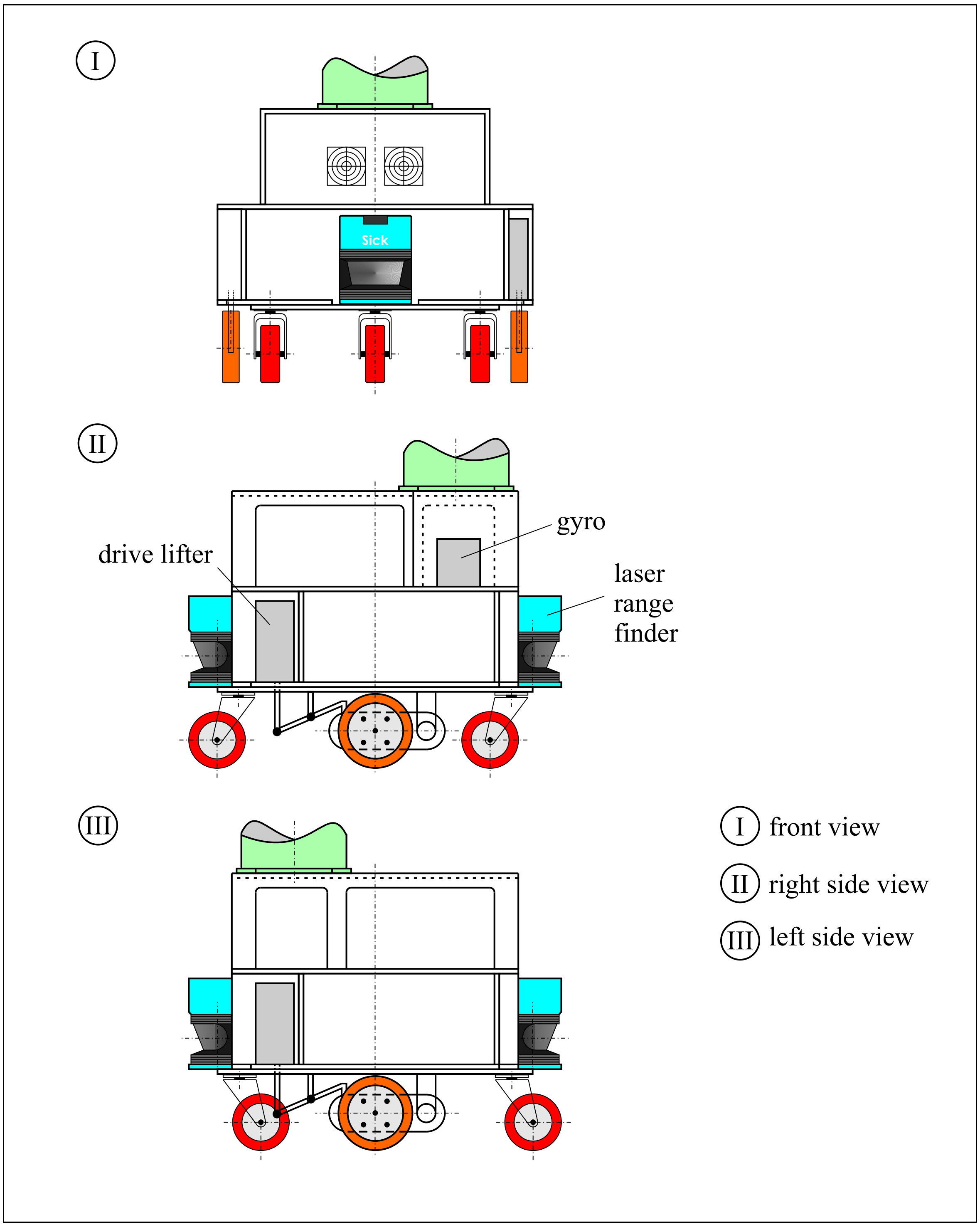


Figure 4.4: Front and side views of the Leonardo mobile robot platform [37].

Voltage supply	48 V DC
Arcnet interface	1
Ethernet interface	1
W-LAN interface	1
CAN bus interface	1
IEEE1394 / Firewire	2
USB 2.0	3
RS-422	2
RS-232	1

Table 4.2: Summary of the I/O interfaces of Leonardo.

Drive Type	differential
Motors	2, 9000 rpm
Gear	37:1
Motorised wheels	2, diameter: 154 mm
Encoder Resolution	4096
Support wheels	2 in the front, 1 in the back

Table 4.3: Drive system parameters of the Leonardo robot.

sor to measure its rotation and a voltage sensor to detect a low battery voltage. These two sensors are connected via the CAN-Bus interface. Furthermore it is equipped with a Mitsubishi Heavy Industries *PA10 Conventional* manipulator, with seven degrees of freedom, that is connected to the Arcnet interface card. The manipulator's tool carries an ATI force/torque-sensor which communicates via a dedicated interface card with the controlling computer, a phd electrical gripper, that is controlled via empty digital outputs of the force/torque-sensor's card and an AVT *Marlin F-145C2* Firewire camera. Table 4.4 summarises the robot's additional sensors and actuators.

1	MHI PA10 Conventional 7 DOF manipulator
1	AVT Marlin F-145C2 Firewire camera
2	SICK LMS 200 laser rangefinder
1	ATI force/torque sensor
1	phd gripper
1	gyroscope sensor
1	voltage sensor

Table 4.4: Summary of Leonardo's additional sensors and actuators.

4.3.2 Software Architecture

As the Leonardo robot was installed at the customer's location and first experiments were carried out in a dedicated pilot plant, the limitations of the installed software package became noticeable as discussed in the Related Work chapter (see section 3.1.1). Afterwards the platform was moved to a real Cell Culture Development Laboratory to carry out monitoring and surveillance tasks in a highly changeable environment, which the previous and the manufacturer's software packages were not able to cope with. For this reason I had to break down the previous software architecture, reengineer, and adjust it to the new requirements. The software package that was contained in two big libraries beforehand has been reimplemented according to the proposed model in section 4.2 and extended by new software components such as a simultaneous localisation and mapping (SLAM) and a visual tracking library. Figure 4.5 depicts the newly implemented software architecture.

At the *Application Layer* reside the implemented surveillance, sample management and a simple remote control and monitoring process. The *Planning Layer* includes a task and a path planner as well as a trajectory generator for the platform and the manipulator respectively. A simultaneous localisation and mapping module and a visual tracking library along with a robot platform library for sensor access form the *Perception Layer*. The *Hardware Abstraction Layer* contains standardised interfaces to classes of devices for the robot's sensors and actuators. In this case it includes laser range finder, light source control, motor and odometry, gyroscope, battery voltage sensor, manipulator, camera, force-/torque sensor, and gripper API modules. The *Hardware Component Layer* contains the specialised software modules, that implement the functionality of the concrete devices like the Sick LMS 200 laser range finders, Leonardo's motors and encoders, Firewire camera, Mitsubishi Heavy Industries PA10 manipulator, etc. These components communicate through libraries at the *Interface Abstraction Layer* for serial ports, CAN Bus, Arcnet, Firewire, and dedicated I/O via the force-/torque sensor board. The figure also depicts the hardware interfaces and components on the *Interface* and *Sensor/Actuator Layer*, which correspond exactly with the *Interface Abstraction* and the *Hardware Component Layer* on the software side.

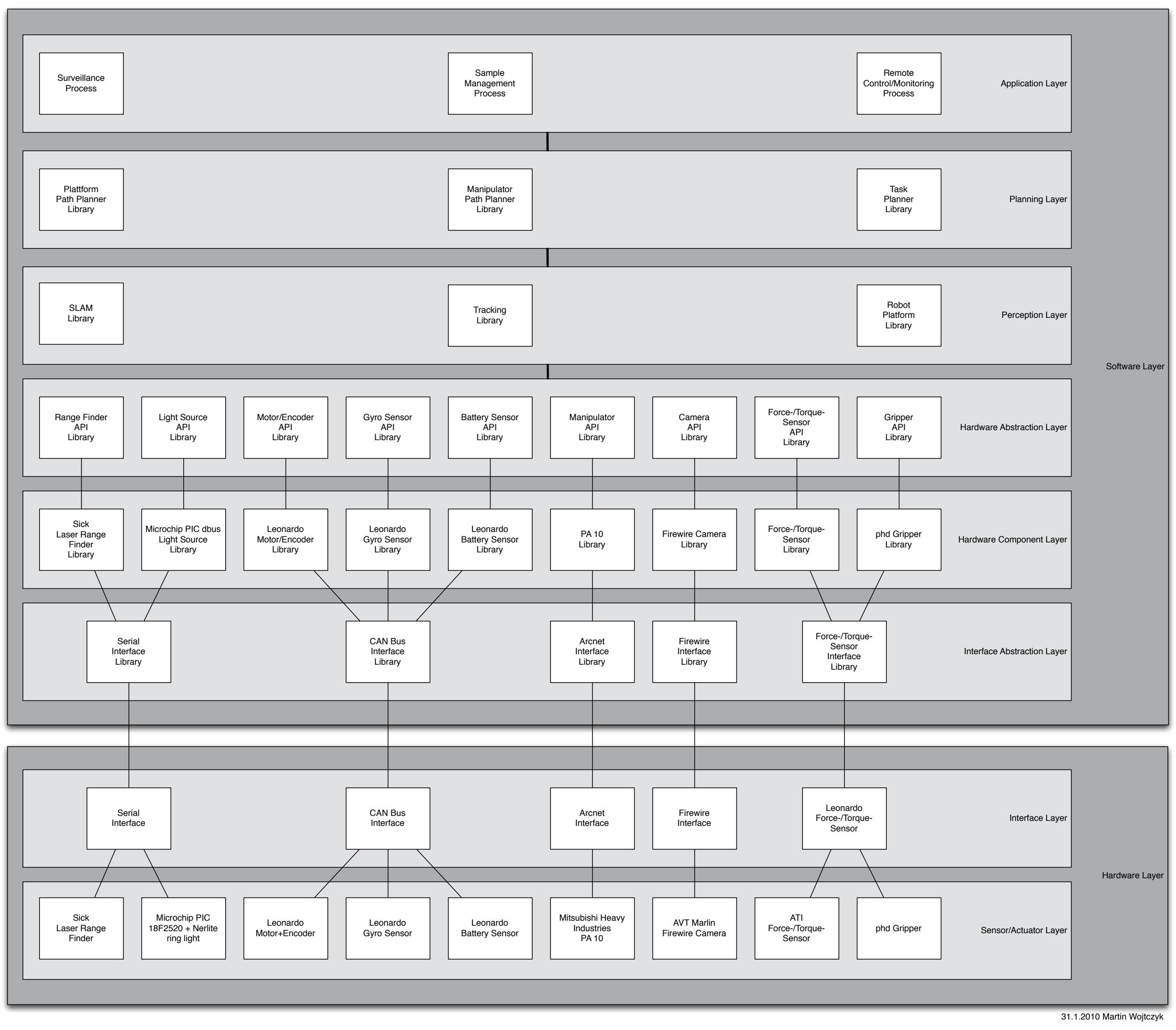

Surveillance Process
Sample Management Process
Remote Control/Monitoring Process
Application Layer
Plattform Path Planner Library
Manipulator Path Planner Library
Task Planner Library
Planning Layer
SLAM Library
Tracking Library
Robot Platform Library
Perception Layer
Software Layer
Range Finder API Library
Light Source API Library
Motor/Encoder API Library
Gyro Sensor API Library
Battery Sensor API Library
Manipulator API Library
Camera API Library
Force-/Torque-Sensor API Library
Gripper API Library
Hardware Abstraction Layer
Sick Laser Range Finder Library
Microchip PIC dbus Light Source Library
Leonardo Motor/Encoder Library
Leonardo Gyro Sensor Library
Leonardo Battery Sensor Library
PA 10 Library
Firewire Camera Library
Force-/Torque-Sensor Library
phd Gripper Library
Hardware Component Layer
Serial Interface Library
CAN Bus Interface Library
Arcnet Interface Library
Firewire Interface Library
Force-/Torque-Sensor Interface Library
Interface Abstraction Layer
Serial Interface
CAN Bus Interface
Arcnet Interface
Firewire Interface
Leonardo Force-/Torque-Sensor
Interface Layer
Hardware Layer
Sick Laser Range Finder
Microchip PIC 18F2520 + Nerlite ring light
Leonardo Motor+Encoder
Leonardo Gyro Sensor
Leonardo Battery Sensor
Mitsubishi Heavy Industries PA 10
AVT Marlin Firewire Camera
ATI Force-/Torque-Sensor
phd Gripper
Sensor/Actuator Layer
31.1.2010 Martin Wojtczyk

Figure 4.5: The reengineered and reimplemented software architecture of the Leonardo robot according to the model proposed in section 4.2.

4.4 F5

The F5 robot is a mobile platform produced by Festo Didactic GmbH & Co. KG in Denkendorf, Germany for the changeable factory scenario as described in section 2.4. It is a wheeled, mobile platform with one laser range finder for localisation, navigation and mapping and is battery-powered. Furthermore two conveyor belts equipped with light barriers at the entry points are mounted on the top of the platform to pickup and unload standard Schaefer Boxes at the changeable factory's stations. All the sensors and actuators are controlled by an on-board computer. Figures 4.6-4.10 show the mobile platform, Table 4.5 summarises the physical dimensions and Table 4.6 the I/O interfaces of the robot.

Length	848 mm
Width	753 mm
Height	ca 700 mm
Overall weight	300 kg
Maximal payload	100 kg

Table 4.5: Dimensions of the F5 mobile robot.

Voltage supply	24 V DC, 110 Ah
Digital inputs	8 + 6
Digital outputs	12 + 6
Analogue inputs	4 (0 – 5 V, resolution 10-bit)

Table 4.6: Summary of the I/O interfaces of the F5 mobile robot.

4.4.1 Robotic Components

The F5 platform drives with two motorised wheels, which are located at the sides of the rotational axis. The motor controllers are connected to the robot's CAN-Bus interface card. One supporting wheel is located in the front, one suspended supporting wheel is located at the back of the platform. Table 4.7 summarises the parameters of the differential drive system.

Drive Type	differential
Motors	2 Maxon motors, 9280 rpm
Gear	156:1
Motorised wheels	2, diameter: 150 mm
Encoder Resolution	2000
Support wheels	1 in the front, 1 in the back

Table 4.7: Drive system parameters of the F5 mobile robot platform.

Attached to the front of the platform is a 180° Leuze laser rangefinder for localisation and mapping, which is connected to the high speed RS-422/485 serial interface card. Two conveyor belts are

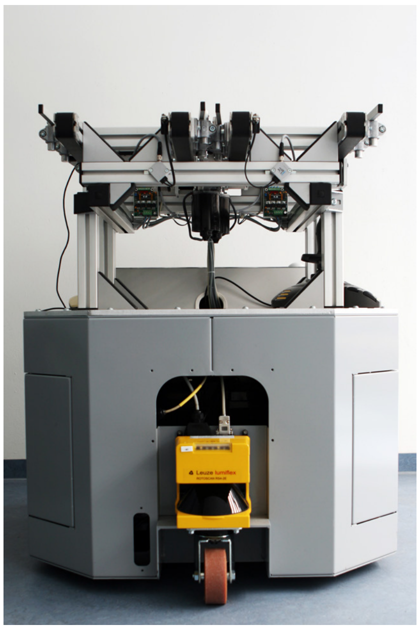


Figure 4.6: Front view of the F5 robot. Noteworthy is the 180° Leuze laser range finder.

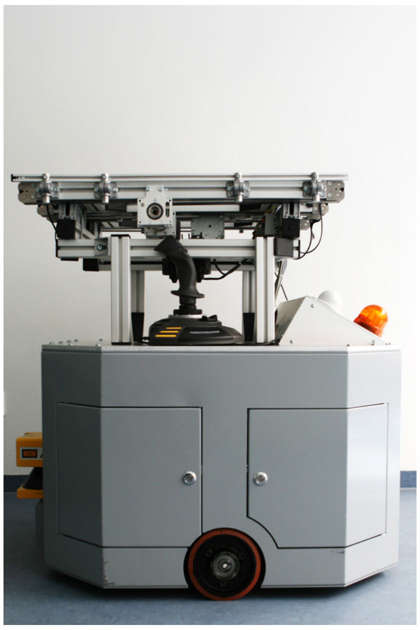


Figure 4.7: Left side of the F5 robot.

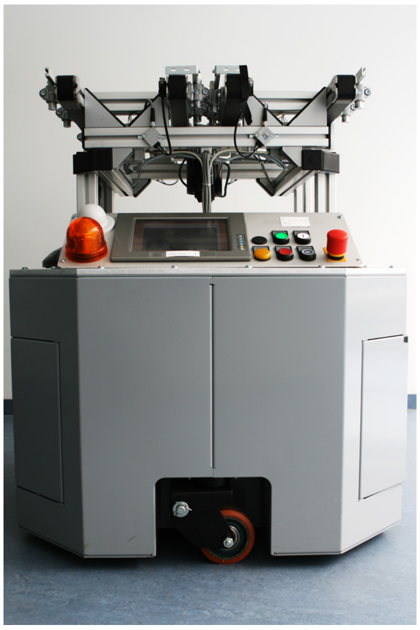


Figure 4.8: Rear view of the F5 robot with the control panel, the touchpad and the warning light.

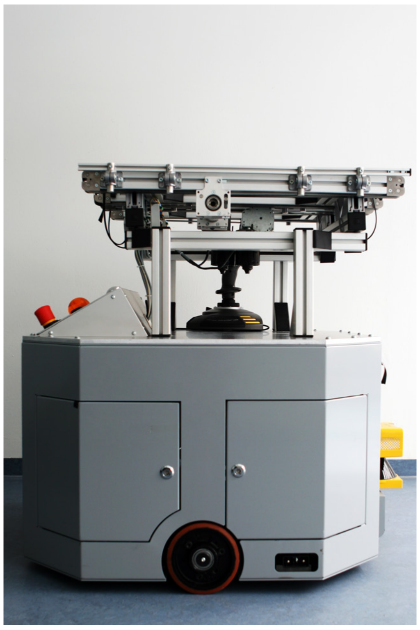


Figure 4.9: Right side of the F5 robot.

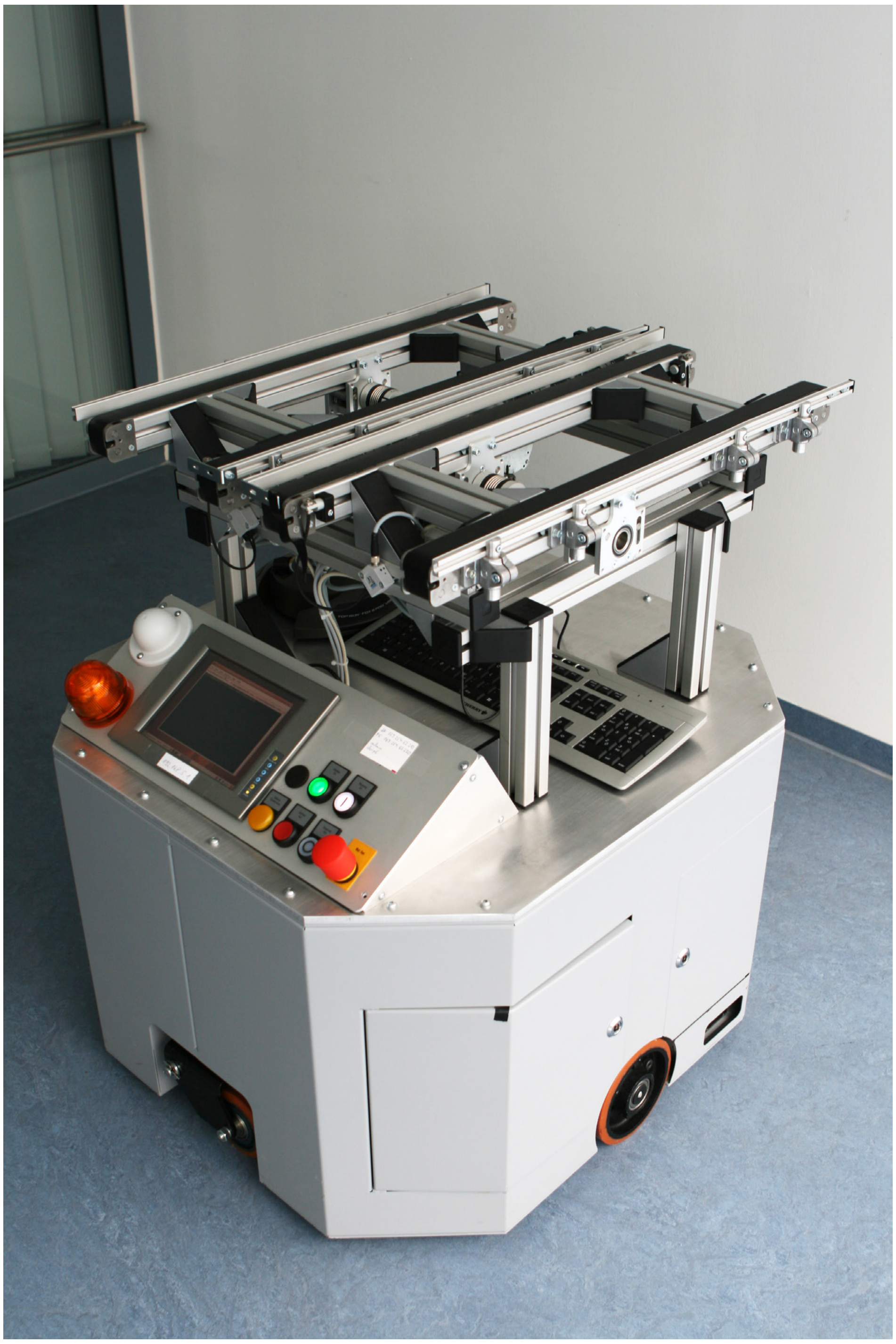


Figure 4.10: The F5 robot with the control panel and touchpad on the rear side and two pairs of conveyor belts mounted on top of the platform to pick up and hand over standardised Schaefer boxes. Light barriers at the entry points of the conveyor belt indicate their occupation.

mounted at the top of the platform, light barriers at the ends of the conveyor belts allow proper loading and unloading of these. A warning flash light, can be enabled, to increase visibility of the platform in a shared space with humans. The conveyor belts, the light barriers and the warning light are connected to the Digital I/Os of a Wago field bus connector, which in turn is controlled via the CAN-Bus interface. Table 4.8 summarises the additional sensors and actuators of the F5 robot.

2	Conveyor belts
4	Light barriers to detect load on the belts
1	Leuze laser rangefinder
1	Warning light

Table 4.8: Summary of the additional sensors and actuators of the F5.

4.4.2 Software Architecture

The software architecture of the F5 robot has been developed from scratch by myself, but with the model from section 4.2 in mind. Since many of the hardware components have not been utilised before, I had to implement the necessary device libraries myself. Nevertheless, I was able to reuse some of the already implemented interfaces and higher level libraries from the Leonardo robot. Figure 4.11 depicts the implemented software architecture.

The *Application Layer* contains the implemented changeable factory and a simple remote operation process. At the *Planning Layer* resides a path planner for collision free motions of the mobile platform. A simultaneous localisation and mapping module and a robot platform library for sensor access form the *Perception Layer*. The *Hardware Abstraction Layer* contains the standardised interfaces to the robot's sensors and actuators. In this case it includes laser range finder, conveyor belt, light barrier, wan warning light API modules. The *Hardware Component Layer* contains the specialised software modules, that implement the functionality of the concrete devices like the Leuze laser range finder, Maxon motors and encoders, Festo conveyor belt, Festo light barrier and the warning light. These components communicate through libraries at the *Interface Abstraction Layer* for serial ports and CAN Bus. The hardware interfaces and components on the *Interface* and *Sensor/Actuator Layer* correspond exactly with the *Interface Abstraction* and the *Hardware Component Layer* on the software side.

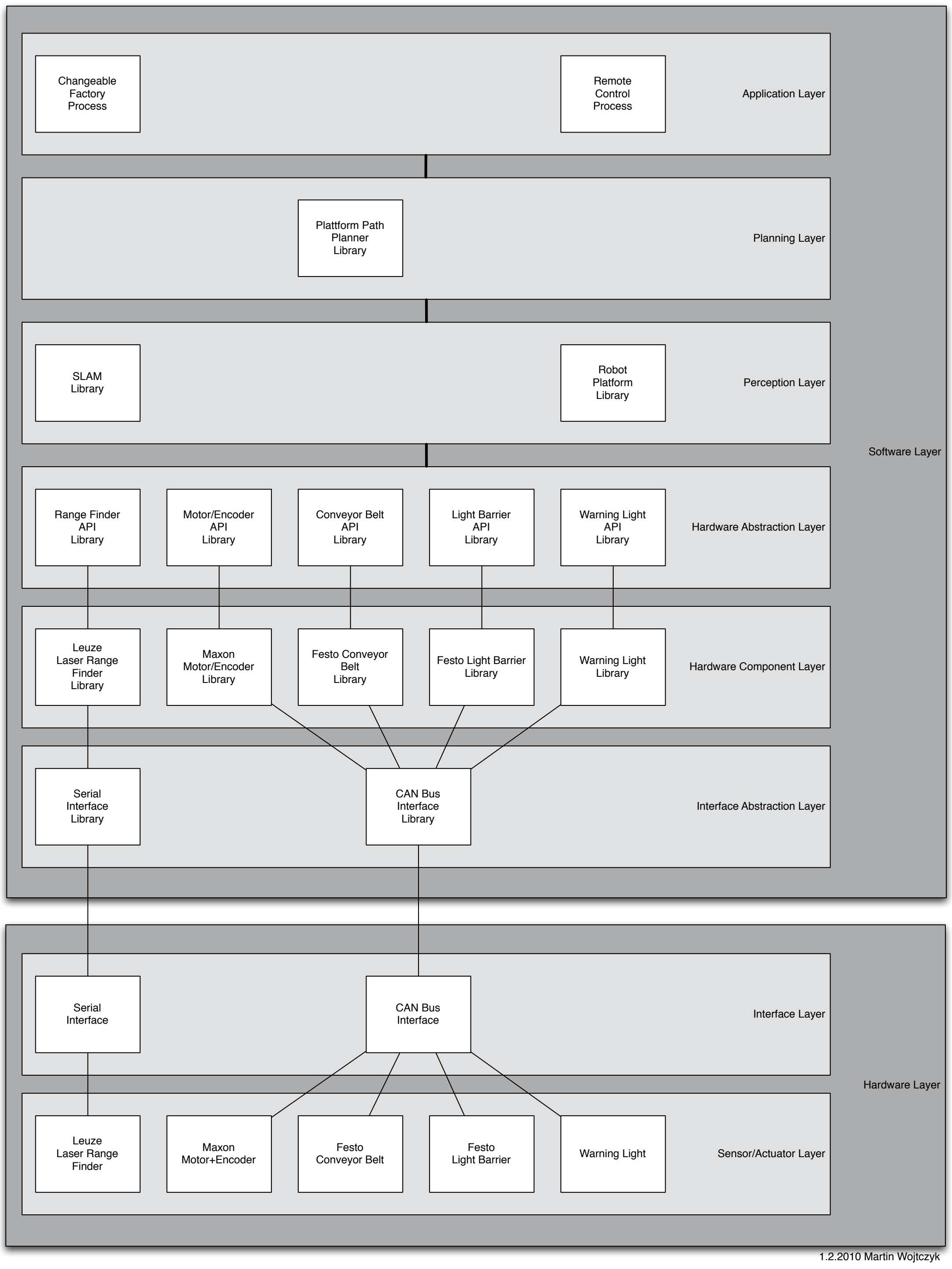


Figure 4.11: The software architecture of the F5 robot according to the layer model of section 4.2. Some of the interface and higher level libraries are identical to the ones of the Leonardo robot, verifying the reusability of these components and the validity of the proposed model.

4.5 F5-S

The F5-S robot is designed to be a successor of the Leonardo robot. Like the F5, it is a mobile platform produced by Festo Didactic GmbH & Co. KG in Denkendorf, Germany for the Life Science Lab, Surveillance and the Housekeeping Scenario. Compared to the F5, the F5-S was designed to be smaller in size to enable its utilisation in domestic environments. It is also a battery-driven, wheeled, mobile platform with one laser range finder for localisation, navigation and mapping and additional distance sensors to detect stairs or difficult floor passages. A robotic arm allows precise manipulation of the environment. Figure 4.12 shows the robot platform, Table 4.9 and Figure 4.13 summarise the physical dimensions and Table 4.10 the I/O interfaces of the robot.

Length	607 mm
Width	630 mm
Height	ca 700 mm
Overall weight	300 kg
Maximal Payload	1.5 kg

Table 4.9: Dimensions of the F5-S mobile robot platform.

Voltage supply	24 V DC, 110 Ah
Digital inputs	6
Digital outputs	6
Analogue inputs	4 (0 – 5 V, resolution 10-bit)

Table 4.10: Summary of the I/O interfaces of the F5-S robot.

4.5.1 Robotic Components

The drive system of the F5-S robot is identical to the one of the F5 robot, except for the distance of the motorised wheels from each other (see Table 4.11).

Drive Type	differential
Motors	2 Maxon motors, 9280 rpm
Gear	156:1
Motorised wheels	2, diameter: 150 mm
Encoder Resolution	2000
Support wheels	1 in the front, 1 in the back

Table 4.11: Drive system parameters of the F5-S mobile robot.

Besides the Leuze laser rangefinder for localisation and mapping and the warning flash light, the F5-S is equipped with four distance sensors connected to the digital inputs. The distance sensors are facing down, hence allow detection of stairs or uneven areas that are difficult to pass. There

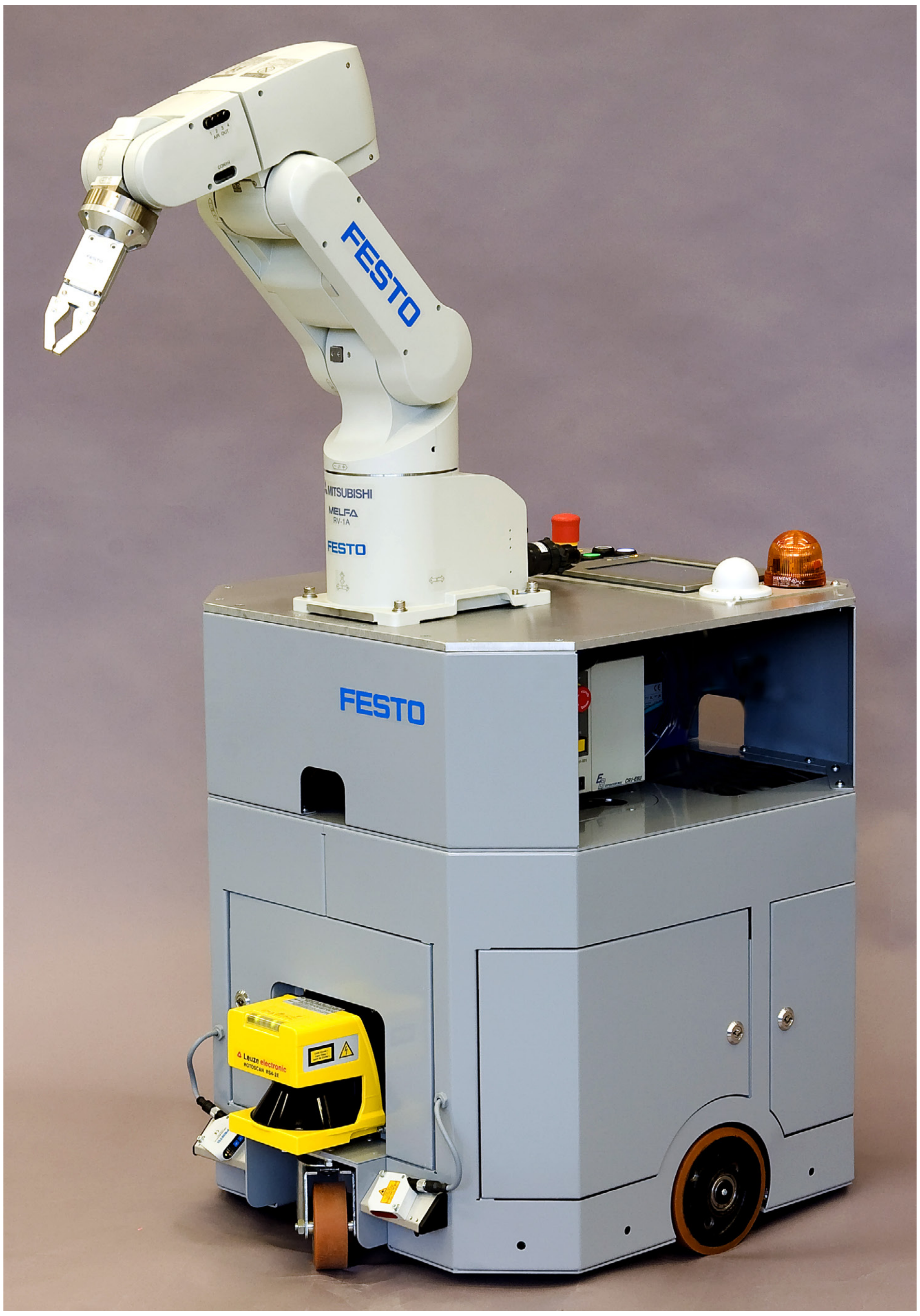


Figure 4.12: The F5-S mobile robot platform. Photo: Festo Didactic.

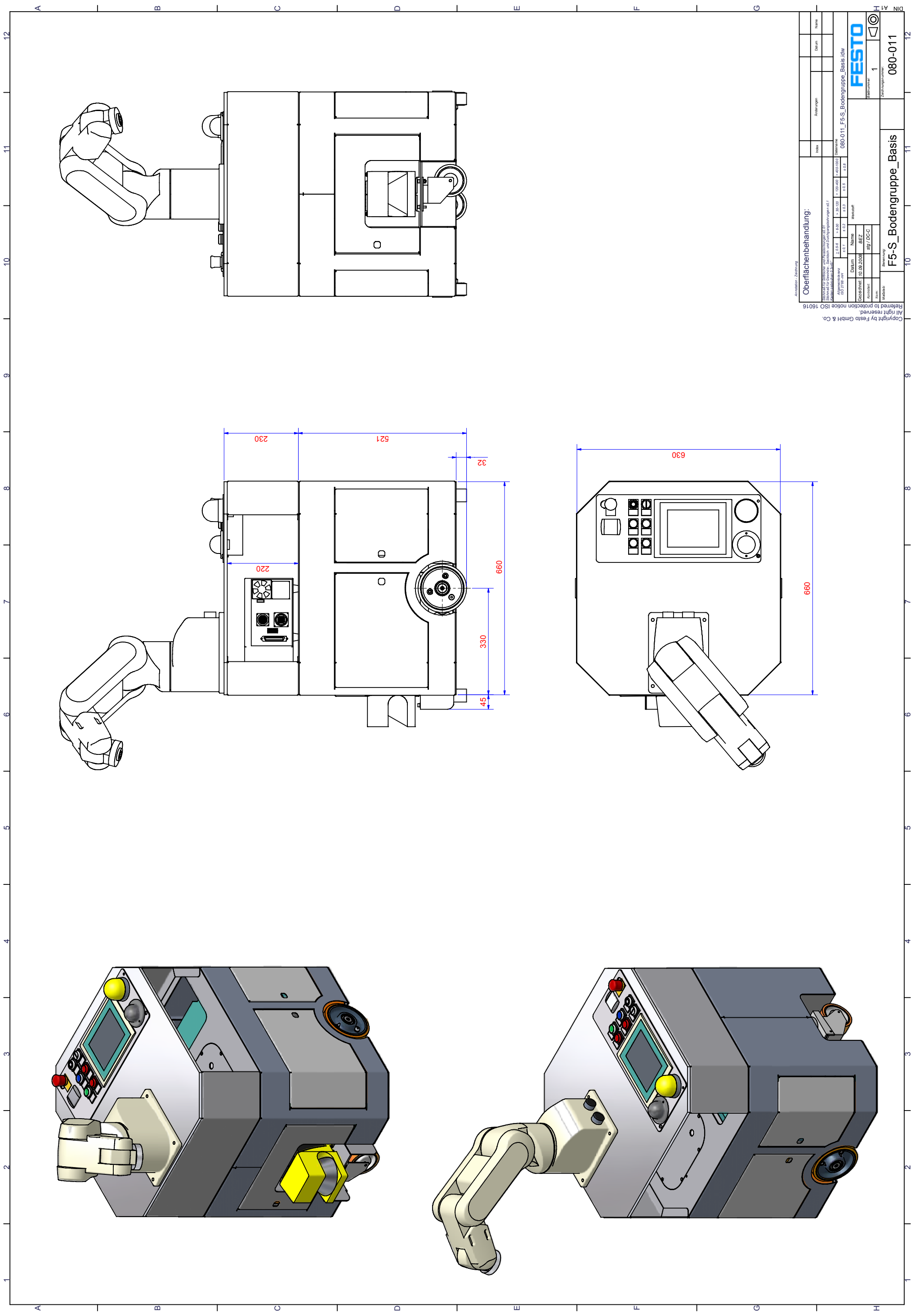


Figure 4.13: Dimensions of the F5-S mobile robot platform. Image: Festo Didactic.

is a second embedded PC in the upper deck of the platform for customised software. Furthermore the F5-S carries a Mitsubishi *MELFA RV-1A* manipulator with six degrees of freedom. A Mitsubishi electrical gripper and an AVT *Guppy F-080C* Firewire camera are attached to the tool the manipulator. Table 4.12 summarises the additional sensors and actuators of the F5-S robot.

1	Mitsubishi MELFA RV-1A manipulator
1	Mitsubishi electrical gripper, range: 30 mm
1	AVT Guppy F-080C Firewire camera
4	Distance sensors
1	Leuze laser rangefinder
1	Warning light

Table 4.12: Summary of the additional sensors and actuators of the F5-S robot.

4.5.2 Software Architecture

The software architecture of the F5-S robot is based on the architecture of the F5 robot, yet extended by the necessary components to interface the manipulator, the gripper and the camera. Nevertheless, most software components could be reused from the Leonardo and F5 platforms. Figure 4.14 depicts the implemented software architecture for the F5-S.

At the *Application Layer* reside the surveillance, sample management, medical lab, personal assistant and remote operation processes. The *Planning Layer* includes a path planner as well as a trajectory generator for the platform and the manipulator respectively. A simultaneous localisation and mapping module and a visual tracking library along with a robot platform library for sensor access form the *Perception Layer*. The *Hardware Abstraction Layer* contains standardised interfaces to classes of devices for the robot's sensors and actuators. Here it includes laser range finder, motor and odometry, dsitance sensor, warning light, manipulator, gripper and camera API modules. The *Hardware Component Layer* contains the specialised software modules, that implement the functionality of the concrete devices which are: Leuze laser range finder, Maxon motors and encoders, Festo distance sensors, warning light, Mitsubishi Electric RV3-SB manipulator, Festo gripper and Firewire Camera. These components communicate through libraries at the *Interface Abstraction Layer* for serial ports, CAN Bus, Ethernet and Firewire. The figure also depicts the hardware interfaces and components on the *Interface* and *Sensor/Actuator Layer*, which correspond exactly with the *Interface Abstraction* and the *Hardware Component Layer* on the software side.

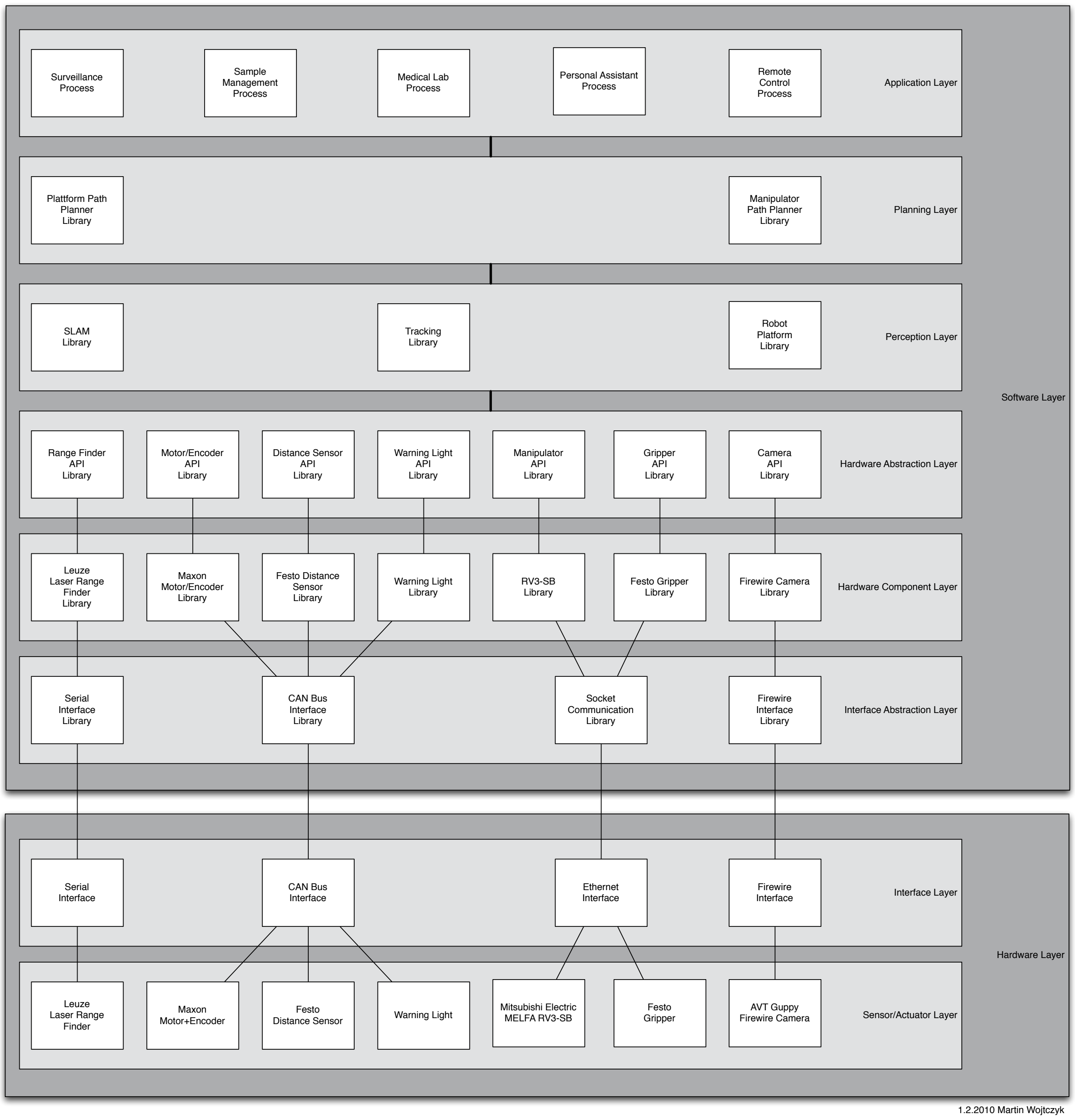


Figure 4.14: The software architecture of the F5-S robot. Since many hardware components are identical to the F5 platform, the appropriate software packages could be reused.

4.6 Robotino

Robotino is the Festo Didactic Learning System for Automation and Technology (see Figure 4.15) to serve the demand for robotic education systems as mentioned in section 2.7. The mobile robot system "is designed to meet a number of different training and vocational requirements" [38]. It is a platform with an open mechanical interface for the integration of additional mechanical devices and an open electrical interface to integrate easily additional sensors or motors of devices. Power is supplied via two 12 V lead gel batteries which permit a running time of up to two hours. Table 4.13 summarises the physical dimensions of Robotino, while table 4.14 lists the available electric interfaces.

The controlling computer of Robotino consists of an embedded PC 104 with a 300 MHz CPU and 128 MB SDRAM, a compact flash card (256 MB) with the operating system (Linux) and a real-time kernel, as well as the C++ API and several demo applications and a Wireless LAN access point.

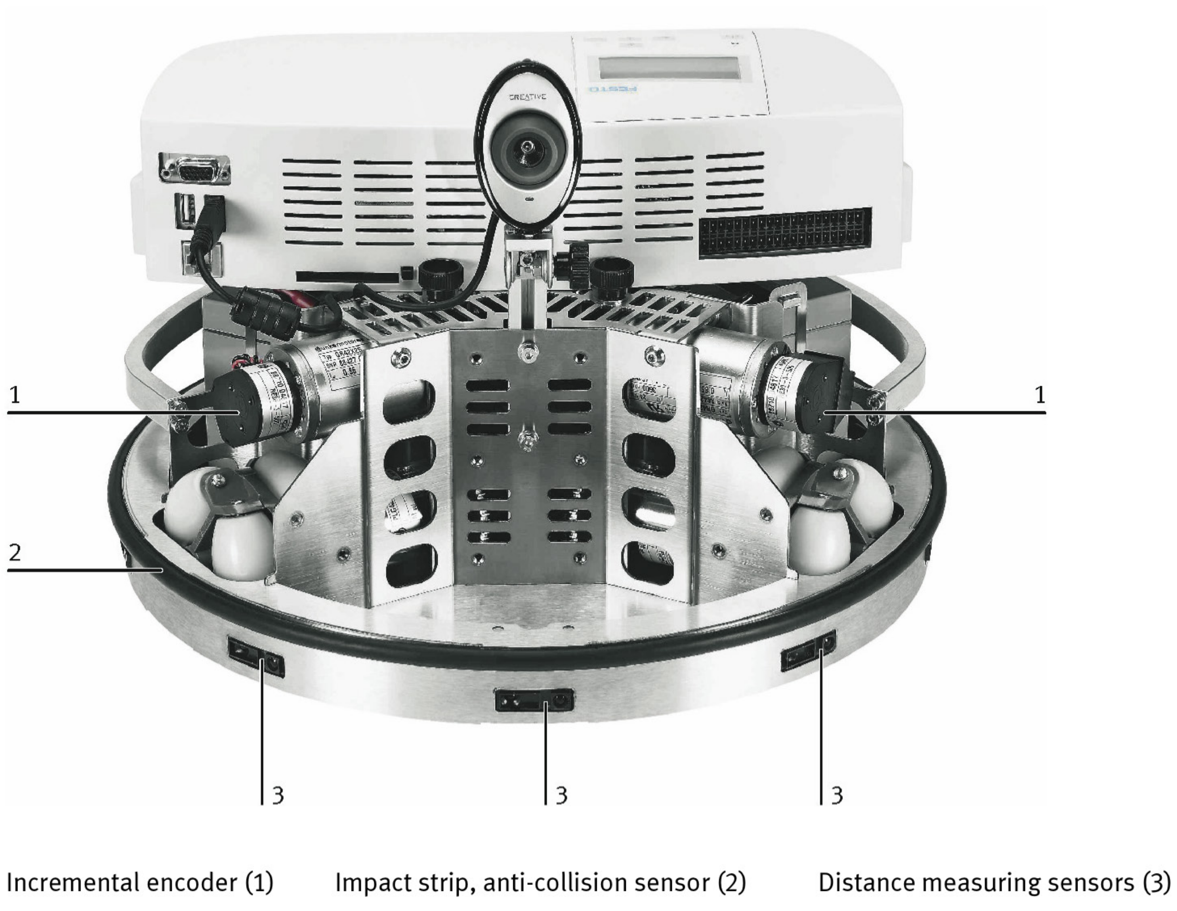

1
1
2
3
3
3
Incremental encoder (1)
Impact strip, anti-collision sensor (2)
Distance measuring sensors (3)

Figure 4.15: Picture of the Robotino platform with the basic configuration [38].

Diameter	370 mm
Height including housing	210 mm
Overall weight	11 kg
Maximal Payload	6 kg

Table 4.13: Dimensions of the Robotino platform [39].

Voltage supply	24 V DC, 4.5 A
Digital inputs	8
Digital outputs	8
Analogue inputs	8 (0 – 10 V)
Relais outputs	2

Table 4.14: Summary of the I/O interfaces of Robotino [38].

4.6.1 Robotic Components

Robotino is driven by three independent drive units with all-way rolling wheels mounted at an angle of 120 ° to each other. The drive units with its characteristic wheels turn Robotino into a holonomic mobile platform. The drive units are integrated in a sturdy, laser welded steel chassis. The chassis is protected by a rubber bumper with an integrated switching sensor. Table 4.15 summarises the attributes of the drive system.

Drive Type	omnidirectional
Motors	3 Dunker motors, 3600 rpm
Gear	16:1
Wheels	3 all-way roller, diameter: 80 mm

Table 4.15: Robotino's drive system [38].

The Robotino platform is equipped with 9 infrared distance measuring sensors which are mounted in the chassis at an angle of 40 ° to one another. Robotino can scrutinise all surrounding areas for objects with these sensors. Each of the sensors can be queried individually via the controller board. Obstacles can thus be avoided, clearances can be maintained and bearings can be taken on a selected target.

The sensors are capable of accurate or relative distance measurements to objects at distances of 4 to 30 cm. The sensor connection is very simple consisting of just one analogue output signal and a power cable. The sensors' evaluation electronics determines the distance and reads it out as an analogue signal. Figure 4.16 depicts the layout of the built-in sensors and the drives.

The anti-collision sensor consists of a switching strip which is attached around the entire circumference of the chassis. A switching chamber is located inside a plastic profile. Two conductive surfaces are situated inside the chamber, between which a given clearance is maintained. These surfaces are short circuited when even minimal pressure is applied to the strip.

The basic configuration comes with two light guide sensors, that can be mounted on the robot and connected to the I/O board. Flexible fibre-optic cables are connected to a fibre-optics unit which works with visible red light. Reflected light is detected. Different surfaces and colours produce different degrees of reflection. However, gradual differences in the reflected light cannot be detected.

An optional inductive proximity sensor is available to detect metallic objects on the floor and is used for continuous-path control. It reads out signals of varying strength depending on whether it is located in the middle or at the edge of the metal strip. Path tracking can thus be implemented in a differentiated way. Table 4.16 summarises the available sensors for the Robotino platform.

9	Analogous distance sensors
1	Bumper with an integrated contact sensor
1	USB Camera
2	Light guide sensors
	Optional: analogue inductive proximity sensor

Table 4.16: Summary of Robotino's sensors and optional accessories [38].

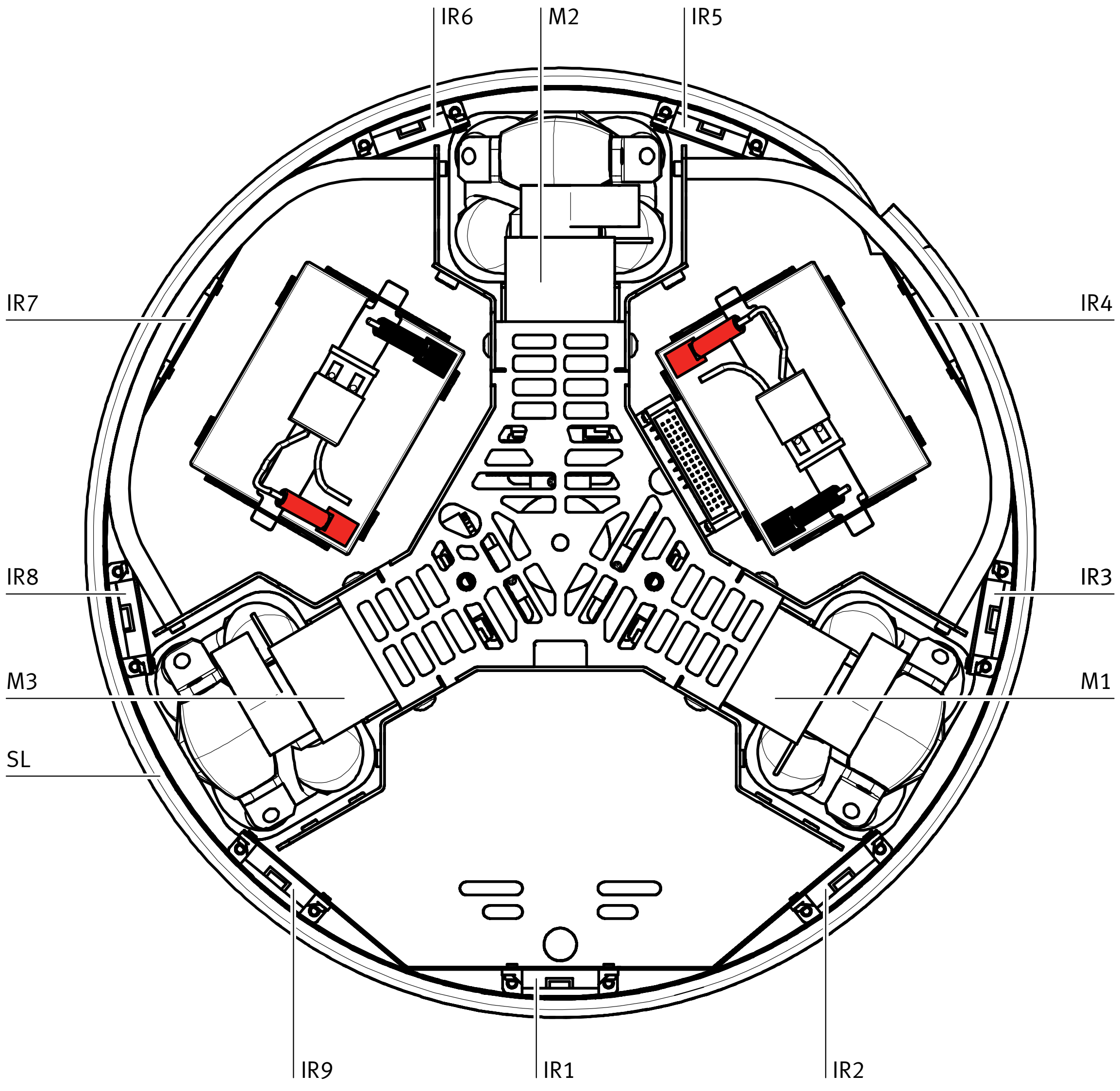
4.6.2 Software Architecture

The basic Robotino package ships with Robotino View (see Figure 4.17), a graphical development environment for robotic applications with numerous prepared function blocks, that have input an output connectors for arbitrary combinations of blocks and to create complex programming sequences following the international standard IEC 61131-3.

Robotino View can be run on an external PC and communicate directly with the Robotino Server running on the embedded PC 104 via W-LAN in order to control the robot system. The function blocks receive a direct feedback from the hardware components in a way so that live interaction with the robot system is enabled. Furthermore Robotino View programs can be transferred to the PC 104 in order to run the applications completely autonomously. Own function blocks can also be implemented in C++ through a well defined interface.

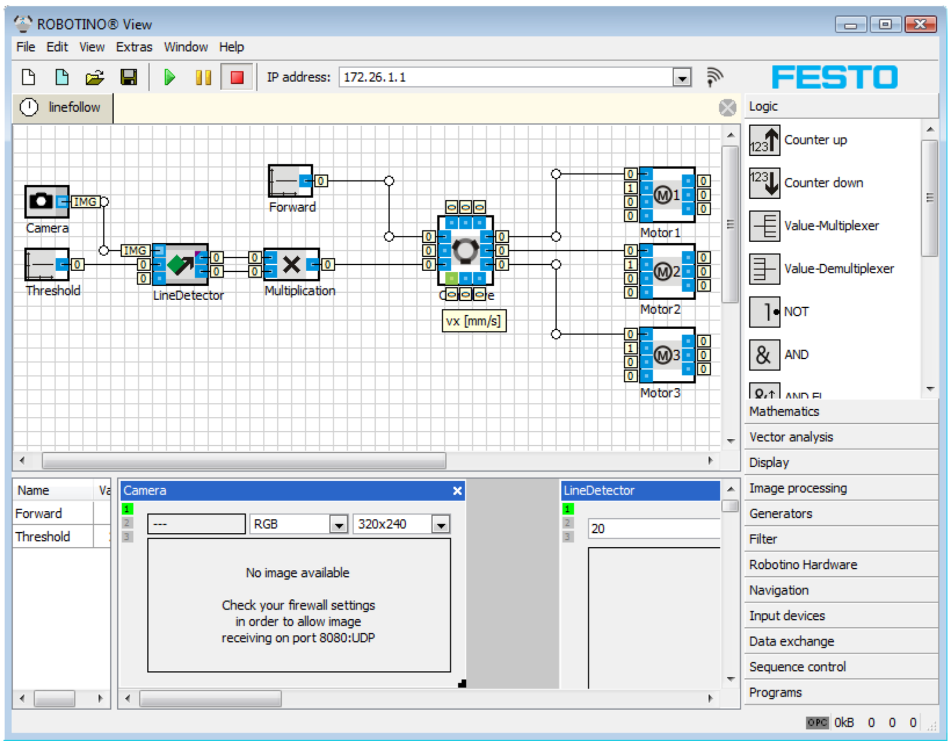
For educational purposes Robotino View introduces the user to:

- Setup of mechatronic systems
- Sensor applications
- Electric motor control
- Electric drives
- Closed loop control of mechatronic systems



Sensor assignments, IR1 to IR9: distance measuring sensors (1 – 9)
M1 to M3: motors (1 – 3) SL = impact strip, anti-collision sensor

Figure 4.16: Overview of the Robotino base plate depicting the sensor and motor positions [38].



No image available

Check your firewall settings
in order to allow image
receiving on port 8080:UDP

Figure 4.17: Robotino View is a graphical development environment providing basic function blocks for visual programming of robotic applications.

- Graphical programming of mobile robot applications
- Introduction to the vision system

Robotino View implements the Robotino specific function blocks based on the Open Robotino API[1]. The API is available as an alternative method to control the Robotino platform without Robotino View.

The API enables the user to:

- Program mobile robots in C, C++, Java, .NET
- Remote control
- Integrate custom vision applications
- Implement autonomous navigation behavior

Figure 4.18 depicts how a software architecture for Robotino can possibly look like, if it is implemented according to the proposed model in section 4.2.

The *Application Layer* would contain the applications such as a surveillance, a changeable factory and a remote operation process. The *Planning Layer* contains a path planner and motion generator for the platform. A simultaneous localisation and mapping module and a visual tracking library along with a robot platform library for sensor access form the *Perception Layer*. The *Hardware Abstraction Layer* contains standardised interfaces to classes of devices for the robot's sensors and actuators. In case of Robotino it would contain range finder, motor and odometry, distance sensors, bumper sensor, and camera API modules. The *Hardware Component Layer* contains the specialised software modules, that implement the functionality of the concrete devices: the PBS Hokuyo range finder, Robotino's motors and encoders, Robotino's distance and bumper sensors and a USB camera. These components communicate through libraries at the *Interface Abstraction Layer* for serial ports, CAN Bus and USB. The figure also depicts the hardware interfaces and components on the *Interface* and *Sensor/Actuator Layer*, which correspond exactly with the *Interface Abstraction* and the *Hardware Component Layer* on the software side.

[1]The Open Robotino API is publicly available from http://svn.openrobotino.org/

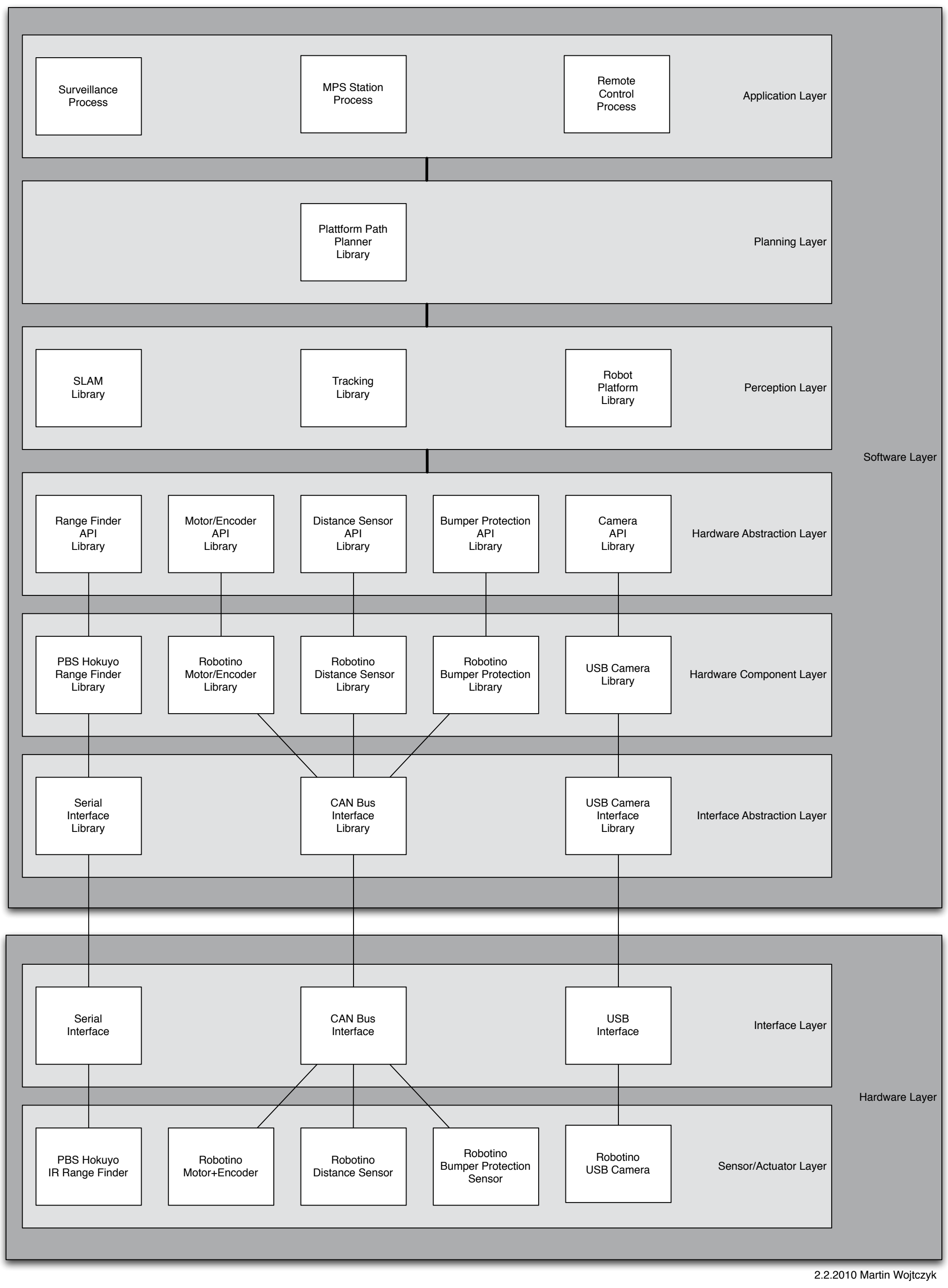


Figure 4.18: A possible description of Robotino's hard- and software components, when applying the proposed model from section 4.2. However this architecture is fictional and has never been implemented this way. This graphic shall only emphasise, that the proposed model is also applicable and valid for already existing robotic platforms on the market and how a software architecture could possibly be implemented for Robotino.

4.7 Summary

In this chapter I proposed a novel layer model to classify soft- and hardware components for service robots. After giving a brief overview of software design principles, that I find important to keep in mind to create sustainable software components, the different layers of this model are explained along with examples. Moreover several different mobile service robots are presented, that I have personally worked on. When I developed the software frameworks for the quite differently equipped Leonardo, F5, and F5-S platforms, I applied the proposed model which enabled me to reuse many software components. These really implemented and a hypothetical software architecture according to the proposed model for the Robotino platform validate the model's versatility and applicability.

Chapter 5

Methodology

This chapter addresses some of the challenges in software engineering for robotic applications and presents the developed solutions.

5.1 Computer Supported Cooperative Work

Computer Supported Cooperative Work (CSCW) is a generic term, which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques [40].

5.1.1 Introduction

CSCW in computer science and software engineering is occurring inherently, once software projects reach a certain size and complexity. The types of CSCW are generally classified by the dimensions of space (same place/different place) and time (same time/different time) as described in the CSCW Matrix (see Figure 5.1).

5.1.2 Application

Collaboration at the same place and at the same time is for example working in a meeting room or two engineers collaborating on one computer screen like in Extreme Programming. Examples for the combination different place/same time are video conferencing and instant messaging systems, screens shared over the network or multi user editors. Collaboration in the same place at different times occurs in team rooms or project specific locations. Collaboration at different places and at

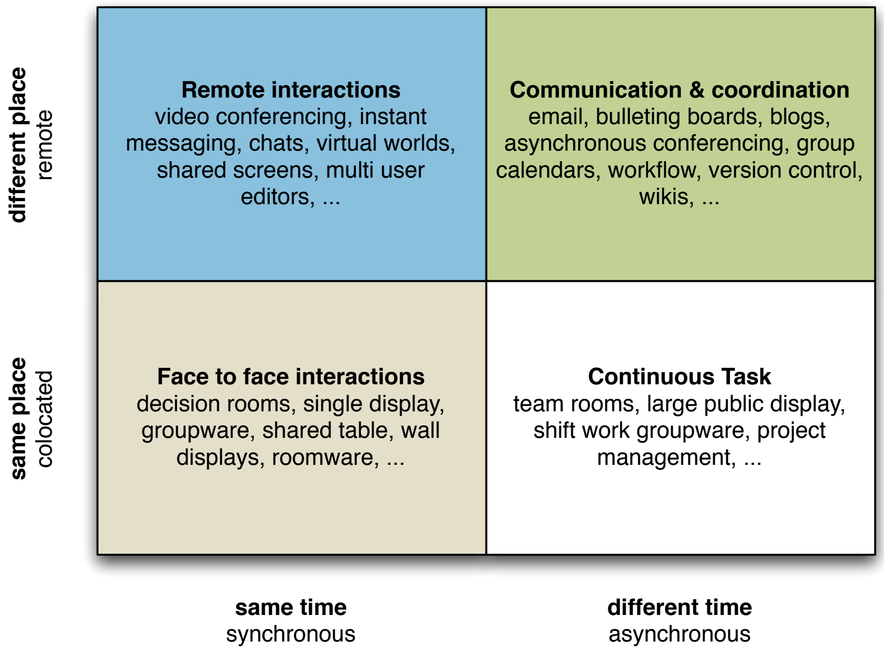

different place
remote
same place
colocated
Remote interactions
video conferencing, instant messaging, chats, virtual worlds, shared screens, multi user editors, ...
Communication & coordination
email, bulleting boards, blogs, asynchronous conferencing, group calendars, workflow, version control, wikis, ...
Face to face interactions
decision rooms, single display, groupware, shared table, wall displays, roomware, ...
Continuous Task
team rooms, large public display, shift work groupware, project management, ...
same time
synchronous
different time
asynchronous

Figure 5.1: The CSCW Matrix, partitioning groupware based on time and space.

different times can be realised with well known communication tools such as email, group calendars and wikis and version control systems which play a very important role in software engineering.

Version Control Systems such as Concurrent Versions System (CVS), Subversion, or Git provide a central- or in case of Git a distributed source code repository for software engineers [41, 42, 43]. They allow managing files and directories and keep track of the changes made to them, over time. This allows developers to recover older versions of their source code or other files and lets them examine the history of how source code or data has changed. The repository inherently also serves as a backup mechanism for the checked in files and allows easy recovery of older states.

5.1.3 Conclusion

The productivity of software engineering teams can be increased dramatically by the application of CSCW. At our department, we utilise a Subversion repository to collaborate on software projects and to distribute a consistent software package to every participating developer [44]. In addition to that, the application of a wiki allows the organic growth of a user generated knowledge base for the distinct projects. Email and instant messaging are used to overcome the boundaries of space and time, if required.

5.2 Cross Platform Software Development

When developing a software tool, one of the first questions arising is: which platform is it targeted on? A platform describes a configuration consisting of *hardware* and *software*. With regard to the hardware, one may want to distinguish between different processor architectures, like x86 or x86 64 bit, PowerPC, ARM, XScale, to mention just a few. Furthermore on embedded systems also the peripheral equipment, like sensors and actuators play an important role. The software part of a configuration is at least determined by the used operating systems, programming languages and their runtime libraries. In robotics and embedded systems, developers are often facing an inhomogeneous variety of operating systems, such as Microsoft Windows, Linux, and real time operating systems, such as QNX, RTLinux or VxWorks. The most commonly used programming languages - at least up to a certain abstraction level - are C and C++.

Even though the programming languages C and C++ have been standardised by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) [45, 46] and - in addition to that - the availability of the C library and the Standard Template Library (STL) [47] enormously simplified development of platform independent applications for the most common operating systems, such a project often already fails at the beginning of the tool chain – the build system or the source code project management.

In my opinion this gap is filled by the open source project CMake in an excellent way. It allows developers to use their favourite development environment on each operating system, yet spares the time intensive synchronisation of platform specific project files, by providing a simple, single source, textual description. With KDE4, CMake was introduced to a very popular project [48]. In this section, I propose a workflow to ease the development of cross platform projects and I show, how I used CMake to create an OpenGL application as a demonstrator for a windowed application running on Windows, Linux and Mac OS X as well as a platform independent camera interface as an example for hardware dependent cross platform applications.

5.2.1 Introduction

Due to the standardisation of C and C++, applications which can be compiled on multiple different platforms, can be easily implemented. On Windows platforms, given a source file like the very simple "Hello World!" application in Listing 5.1 the translation process however requires the manual creation of a Visual Studio project file referencing the source file [49]. On Macintosh computers, people often are used to the Xcode IDE, where the developers need to create the necessary Xcode projects, which reference the source [50]. On Unix/Linux systems developers often use the GNU Autotools or even write Makefiles manually [51, 52].

```
#include <iostream>

using namespace std;

int main(int argc, char** argv)
{
   cout << "Hello␣World!" << endl;
   return 0;
};
```

Listing 5.1: hello.cpp

At a final stage of a cross platform application, the developers may just provide project files for the different platforms, but in most cases a software project is continuously being maintained and as soon as new classes or modules are added to a project or as soon as multiple engineers co-operate on a project, developers desire a tool, that supports synchronising the different Visual Studio- or Xcode projects as well as the Unix Makefiles. The open source project CMake supports developers to manage this kind of projects with simple textual project descriptions out of which generators provide platform specific project files. Table 5.1 summarises the project generators of the current version 2.8.1 for the Windows, Mac OS X and Unix/Linux operating systems. According to the table, a wide variety of development environments is supported on every platform, for example Eclipse with the C/C++ Development Tooling (CDT) extension and even KDevelop 3 or CodeBlocks on each of them in addition to the previously mentioned ones [53, 54, 55].

5.2.2 Application

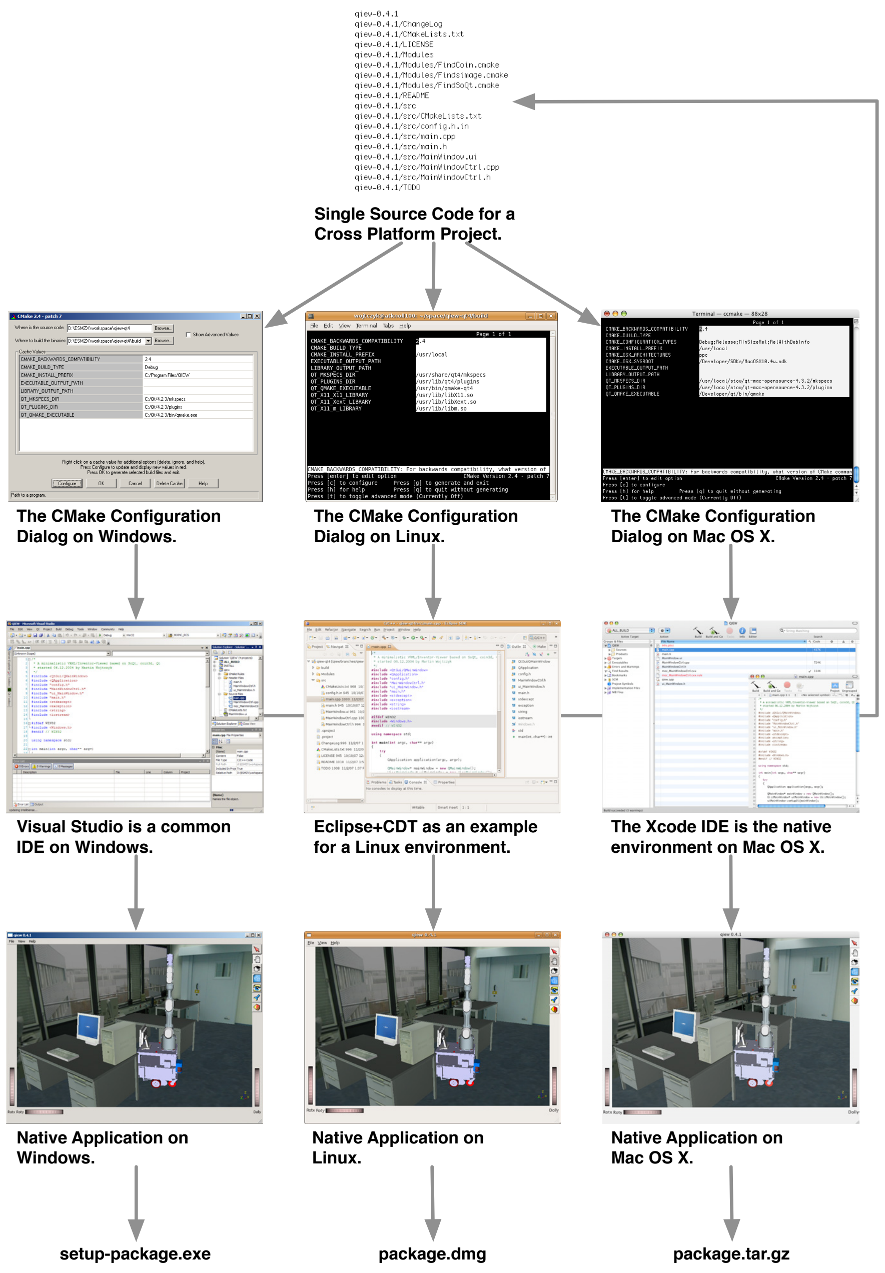
CMake can be downloaded as source code or as installable executable for Windows or as precompiled binaries for Mac OS X, Linux and several Unix systems for free at [56]. Furthermore packages for many Linux distributions, the MacPorts- and the Fink-project for Macintosh users are provided in their repositories for convenient installation and automated updates [57, 58]. By default the precompiled Windows package comes with a Qt based GUI to ease the setup of initial settings for the project under development, while the Unix/Linux and Mac OS X versions of the precompiled package come with a ncurses based console user interface application `ccmake` (see Figure 5.2, 2nd row) [59, 60]. If built from source, a Qt based GUI and/or a ncurses based console user interface will be built on each platform, provided that the necessary libraries Qt and ncurses are installed and accessible by the compiler. Subsequent project updates can be generated by the utilisation of the `cmake` command on each platform.

```
SET(SRCS main.cpp)
ADD_EXECUTABLE(hello ${SRCS})
```

Listing 5.2: CMakeLists.txt

Windows	**Unix/Linux**	**Mac OS X**
Borland Make	Unix Make	Unix Make
MSYS Make	CodeBlocks - Unix Make	Xcode
MinGW Make	Eclipse CDT4 - Unix Make	CodeBlocks - Unix Make
NMake Make	KDevelop3	Eclipse CDT4 - Unix Make
NMake Make JOM	KDevelop3 - Unix Make	KDevelop3
Unix Make		KDevelop3 - Unix Make
Visual Studio 10		
Visual Studio 10 Win64		
Visual Studio 6		
Visual Studio 7		
Visual Studio 7 .NET 2003		
Visual Studio 8 2005		
Visual Studio 8 2005 Win64		
Visual Studio 9 2008		
Visual Studio 9 2008 Win64		
Watcom WMake		
CodeBlocks - MinGW Make		
CodeBlocks - NMake Make		
CodeBlocks - Unix Make		
Eclipse CDT4 - MinGW Make		
Eclipse CDT4 - NMake Make		
Eclipse CDT4 - Unix Make		

Table 5.1: Available project generators for the supported operating systems as of CMake 2.8.1.


```
qiew-0.4.1
qiew-0.4.1/ChangeLog
qiew-0.4.1/CMakeLists.txt
qiew-0.4.1/LICENSE
qiew-0.4.1/Modules
qiew-0.4.1/Modules/FindCoin.cmake
qiew-0.4.1/Modules/Findsimage.cmake
qiew-0.4.1/Modules/FindSoQt.cmake
qiew-0.4.1/README
qiew-0.4.1/src
qiew-0.4.1/src/CMakeLists.txt
qiew-0.4.1/src/config.h.in
qiew-0.4.1/src/main.cpp
qiew-0.4.1/src/main.h
qiew-0.4.1/src/MainWindow.ui
qiew-0.4.1/src/MainWindowCtrl.cpp
qiew-0.4.1/src/MainWindowCtrl.h
qiew-0.4.1/TODO
```

CMake 2.4 - patch 7

Where is the source code: D:\ESMZX\workspace\qiew-qt4

Where to build the binaries: D:\ESMZX\workspace\qiew-qt4\build

Cache Values

CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	Debug
CMAKE_INSTALL_PREFIX	C:/Program Files/QIEW
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
QT_MKSPECS_DIR	C:/Qt/4.2.3/mkspecs
QT_PLUGINS_DIR	C:/Qt/4.2.3/plugins
QT_QMAKE_EXECUTABLE	C:/Qt/4.2.3/bin/qmake.exe

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

Path to a program.

wojtczyk@atknoll100: ~/space/qiew-qt4/build

```
                                                       Page 1 of 1
 CMAKE_BACKWARDS_COMPATIBILITY    2.4
 CMAKE_BUILD_TYPE
 CMAKE_INSTALL_PREFIX             /usr/local
 EXECUTABLE_OUTPUT_PATH
 LIBRARY_OUTPUT_PATH
 QT_MKSPECS_DIR                   /usr/share/qt4/mkspecs
 QT_PLUGINS_DIR                   /usr/lib/qt4/plugins
 QT_QMAKE_EXECUTABLE              /usr/bin/qmake-qt4
 QT_X11_X11_LIBRARY               /usr/lib/libX11.so
 QT_X11_Xext_LIBRARY              /usr/lib/libXext.so
 QT_X11_m_LIBRARY                 /usr/lib/libm.so

CMAKE_BACKWARDS_COMPATIBILITY: For backwards compatibility, what version of
Press [enter] to edit option                          CMake Version 2.4 - patch 7
Press [c] to configure     Press [g] to generate and exit
Press [h] for help         Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Terminal — ccmake — 88x28

```
                                                       Page 1 of 1
 CMAKE_BACKWARDS_COMPATIBILITY    2.4
 CMAKE_BUILD_TYPE
 CMAKE_CONFIGURATION_TYPES        Debug;Release;MinSizeRel;RelWithDebInfo
 CMAKE_INSTALL_PREFIX             /usr/local
 CMAKE_OSX_ARCHITECTURES          ppc
 CMAKE_OSX_SYSROOT                /Developer/SDKs/MacOSX10.4u.sdk
 EXECUTABLE_OUTPUT_PATH
 LIBRARY_OUTPUT_PATH
 QT_MKSPECS_DIR                   /usr/local/stow/qt-mac-opensource-4.3.2/mkspecs
 QT_PLUGINS_DIR                   /usr/local/stow/qt-mac-opensource-4.3.2/plugins
 QT_QMAKE_EXECUTABLE              /Developer/qt/bin/qmake

CMAKE_BACKWARDS_COMPATIBILITY: For backwards compatibility, what version of CMake comman
Press [enter] to edit option                                  CMake Version 2.4 - patch 7
Press [c] to configure
Press [h] for help         Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Single Source Code for a Cross Platform Project.
The CMake Configuration Dialog on Windows.
The CMake Configuration Dialog on Linux.
The CMake Configuration Dialog on Mac OS X.
Visual Studio is a common IDE on Windows.
Eclipse+CDT as an example for a Linux environment.
The Xcode IDE is the native environment on Mac OS X.
Native Application on Windows.
Native Application on Linux.
Native Application on Mac OS X.
setup-package.exe
package.dmg
package.tar.gz

Figure 5.2: Exemplary workflows depicted for the development process on Windows, Linux and Mac OS X platforms. The first row symbolises the single source for the cross platform application. The configuration step in the second row shows the Qt-based user interface of CMake on Windows and the ncurses-based application ccmake on Linux and Mac OS X. In the depicted case, Visual Studio was used to build the native Windows application, eclipse was used for Linux and Xcode for Mac OS X. Changes can be commited directly to the source code repository from within the IDEs. The result of the build process is a native application on each system, which optionally can be packaged automatically for deployment.

Listing 5.2 shows a simple project description for the previously mentioned "Hello World!" application. Invoking CMake with this description will produce a project for the desired development environment on each supported platform. A subsequent build process in the individual developer's favourite environment will then build a native application for this platform.

To build an application out of the two files `hello.cpp` and `CMakeLists.txt`, the path to the project description in the CMakeLists.txt file is passed to `cmake`, an optional parameter `-G` can specify the desired generator. Calling `cmake -help` lists all possible parameters and the supported generators on the running platform.

On Unix/Linux/Mac OS X the build process is performed by the calls in Listing 5.3, since per default Unix Makefiles are generated.

```
hello$ cmake .
hello$ make
```

Listing 5.3: Building the "Hello World!" application on Linux/Unix and Mac OS X.

On Windows the commands in Listing 5.4 perform the same, provided that Visual Studio is installed. If installed, also the GUI application or the ncurses based console user interface application can be used to create the project files. As an alternative to Listings 5.3 and 5.4 you can also generate a Visual Studio Solution, a Xcode project or a KDevelop 3 project as desired and invoke the build process within the IDEs as usual. The only important prerequisite is, that the necessary dependent tools (make, nmake, gcc, link, cl, ...) are available on the command line, thus the environment variables are set properly. A very useful feature is the out of source build, which means that the source code of a project stays separated from the created project files, compilations and possibly generated temporary files to avoid the accidental submission of platform specific files to the source code repository.

```
hello$ cmake . –G "NMake␣Makefiles"
hello$ nmake
```

Listing 5.4: Building the "Hello World!" application on Windows.

5.2.2.1 Project Description

The project description of the "Hello World!" example only defines a variable `${SRCS}`, which references the necessary source code files. The instruction `ADD_EXECUTABLE` notifies CMake to create a `hello` executable target out of the source files passed as an argument. In a similar way static or dynamic libraries can be created by the `ADD_LIBRARY(libname [SHARED | STATIC] sourcelist)` instruction. A complete list of the CMake instructions can be found in the frequently updated online documentation at [61] or in the official book [62]. In contrast to the GNU Autotools - which are only available for Unix environments - complex projects can be described and maintained after a very short learning phase already.

Often software systems need to access external libraries or connected hardware, which is accessed differently on different operating systems. Sections 5.2.2.2 and 5.2.2.3 describe both scenarios.

5.2.2.2 Project Configuration

Since nowadays software projects often rely on external libraries, one of the most necessary features of a build system is the support of the configuration step, hence finding necessary header and library files of external packages. CMake simplifies this step by providing many configuration scripts for the most common libraries. However reusable configuration scripts can also be easily implemented by the user himself as described in the documentation. If, for example, a cross platform application should not be restricted to the command line, you can easily utilise one of the supported GUI toolkits (Qt3, Qt4, GTK+, wxWidgets, FLTK etc.) [63, 64, 65]. This way impressive, windowed cross platform applications can already be implemented just by choosing the right external libraries.

To inspect 3D models on every operating system in the same environment and to establish a software basis for animated simulations for future use, *qiew* a demonstrator and test-bench was implemented. Using the Qt toolkit and Coin3D, a reimplementation of SGI's Open Inventor, a platform independent VRML viewer was created (see Figures 5.2, 4th row) [66, 67]. The source code of the application is available at its website [68]. In the source I also contributed configuration scripts to find the Coin3D libraries on Windows, Linux and Mac OS X.

5.2.2.3 Resolving Platform Characteristics

When it comes to hardware access, software engineers often have to deal with different Application Programming Interfaces (APIs) on each operating system.

As part of a unified, vision based tracking library, I needed to implement a platform independent camera interface, which was put into effect by applying common software engineering methods [19]. In this case I utilised the `AbstractFactory` Design Pattern as described in [34] to encapsulate platform specific code in specialised classes derived from an abstract interface class, which provides the function interfaces for the end user. Figure 5.3 shows the implemented UML class hierarchy with a subset of the implemented methods.

Video sources which we defined as cameras can be recorded movie sequences, USB web-cams out of which many are supported by the open source vision library OpenCV [69] or Firewire cameras. Since most Firewire cameras support the Instrumentation & Industrial Digital Camera (IIDC) Standard [70], they are frequently used in academia. Therefore enhanced support for Firewire cameras

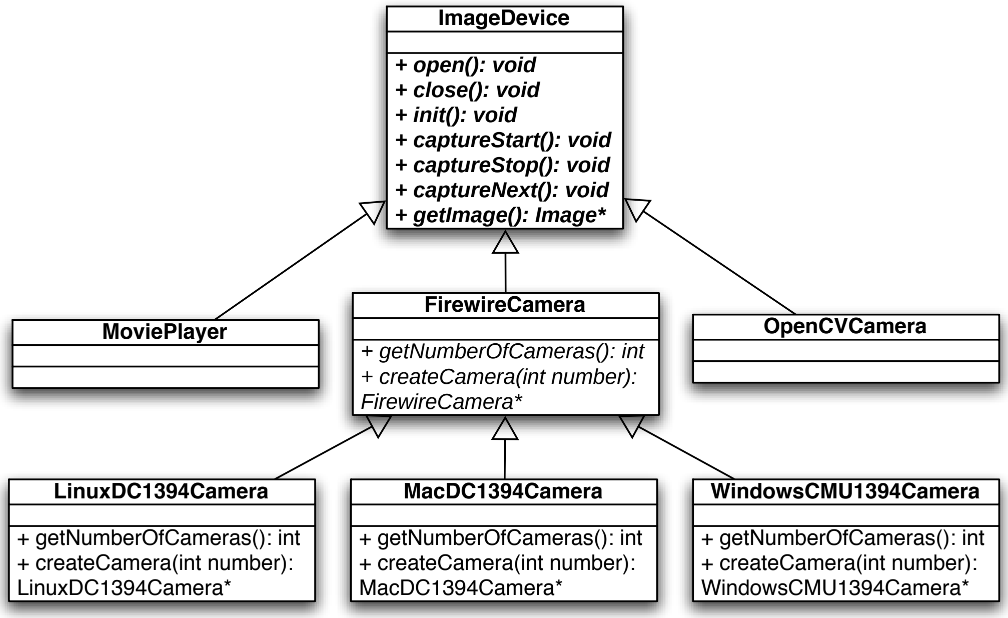

ImageDevice
+ open(): void
+ close(): void
+ init(): void
+ captureStart(): void
+ captureStop(): void
+ captureNext(): void
+ getImage(): Image*
MoviePlayer
FirewireCamera
+ getNumberOfCameras(): int
+ createCamera(int number): FirewireCamera*
OpenCVCamera
LinuxDC1394Camera
+ getNumberOfCameras(): int
+ createCamera(int number): LinuxDC1394Camera*
MacDC1394Camera
+ getNumberOfCameras(): int
+ createCamera(int number): MacDC1394Camera*
WindowsCMU1394Camera
+ getNumberOfCameras(): int
+ createCamera(int number): WindowsCMU1394Camera*

Figure 5.3: A Cross Platform class hierarchy for unified camera access.

was implemented to make use of the standardised access to relevant registers such as white balance, gain or shutter speed.

On Linux and Mac OS X access to Firewire cameras is provided by the commonly used library libdc1394 [71] while the CMU 1394 Digital Camera Driver [72] provides a similar functionality for Windows, yet with a completely different interface. I contributed the necessary configuration scripts to find the libdc1394 library on Linux and Mac OS X and the CMU 1394 Digital Camera Driver on Windows. Furthermore I contributed a software package providing a uniform programming interface for the platform specific APIs.

5.2.2.4 Deployment

Once an application or library is built, it is usually packaged for distribution. While Windows packages mostly come as installable setup files, Unix/Linux packages are often Tarballs or self-extracting shell-scripts and Mac OS X packages usually come as DMG disk images, which directly contain the binaries or an installer package. By the simple use of an `INCLUDE(CPack)` directive in the CMakeLists.txt file, a package target will be generated in the project file and all files, which are tagged with an `INSTALL` command will automatically be added to the appropriate deployment package, when invoked. Table 5.2 summarises all package generators. The generators STGZ, TBZ2, TGZ, TZ and ZIP are available on every platform, provided that the necessary packaging tool is available and create archives with the necessary binaries and/or sources if tagged for installation. The NSIS generator creates an installable windows package based on the Open Source Tool: Nullsoft Scriptable Install System (NSIS) [73]. The generated and installed package will also show up in the system wide list of installed Programs and provides an uninstaller for clean removal. The DEB and RPM generators are available on Linux machines to build commonly used Debian- (DEB) or Red Hat Package Manager (RPM) packages which can be installed and removed easily. The Bundle, DragNDrop, OSXX11 and PackageMaker generators are only available on Macintosh systems and provide native installers for this platform.

Generator	Description	Windows	Unix/Linux	Mac OS X
Bundle	Mac OSX bundle	−	−	+
DEB	Debian packages	−	+	−
DragNDrop	Mac OSX Drag And Drop	−	−	+
NSIS	Null Soft Installer	+	+	+
OSXX11	Mac OSX X11 bundle	−	−	+
PackageMaker	Mac OSX Package Maker installer	−	−	+
RPM	RPM packages	−	+	−
STGZ	Self extracting Tar GZip compression	+	+	+
TBZ2	Tar BZip2 compression	+	+	+
TGZ	Tar GZip compression	+	+	+
TZ	Tar Compress compression	+	+	+
ZIP	ZIP file format	+	+	+

Table 5.2: Available deployment package generators for the supported operating systems.

5.2.3 Workflow

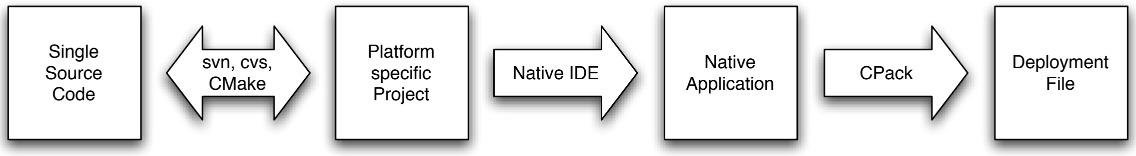


Figure 5.4: The Cross Platform Workflow and its involved Tools.

To sum up, I propose the following workflow for cross platform applications or software components. In the first place, developers get the current source code and project descriptions from a source code and version management system such as Subversion or CVS, for the reasons mentioned in section 5.1.2, no matter which operating system they work on. Afterwards they generate native project files for the development environment, which they prefer to work with by the use of the CMake project generator. From within their favourite IDE they contribute to the project and, if necessary, to the project description files, which are committed back to the source code management system, once a goal is achieved.

For testing, the native build processes are invoked from within the IDE. In addition to that, packaging for deployment can be performed automatically on each supported platform to reduce the effort of application bundling for every new version. Figure 5.4 shows the workflow depicting the project's states in boxes and the utilised transformation tools and their impact directions as arrows. Figure 5.2 depicts the workflow for the previously mentioned cross platform application in section 5.2.2.2 at its different stages which was developed for Windows, Linux and Mac OS X.

5.2.4 Conclusion

It is true, that the proposed workflow requires software engineers developing C/C++ applications to learn an additional tool. On the other hand doing so could make their applications available for a much larger user base running different operating systems. And actually the learning curve with CMake is not as steep as with the GNU Autotools by far. In some software projects the co-operation of users of different platforms is just inevitable. At the chair for Robotics and Embedded Systems at the Technische Universität München a majority of the vision group prefers to implement their algorithms on Windows, while the robotics group usually uses Realtime Linux on their robot platforms, where the vision algorithms are put into effect. This was the initial reason to create a workflow connecting both worlds.

During an evaluation of building tools by myself in 2004, CMake was the most promising out of several other tested ones among which were: qmake [59], GNU Autotools, Microsoft Visual Studio, Apple Xcode, smake [74], CONS [75], SCons [76], Apache Ant [77], Jam [78], bjam [79]. Since then, the described workflow with CMake has been used in very many cross platform projects at our department. Furthermore it has been introduced in the two spin-off companies of the department: Robotics Technology Leaders GmbH and Robotics Equipment Corporation GmbH due to the many advantages. From project generation and synchronisation, to configuration and dependency resolving and to deployment packaging, as mentioned above. The utilisation of CMake within well-known projects such as KDE4 or OpenSceneGraph and OpenCV may raise its popularity. I showed, it can be easily integrated in the development process on the most popular operating systems, still letting the developer choose his favourite environment, however more important than that, I showed, it can be used very well as the missing link in managing cross platform projects on the configuration and build level [14].

5.3 Rapid Hardware Prototyping

Development of mechatronic systems is a time consuming process and requires efforts in mechanical, electrical and software engineering which typically are put into effect sequentially in the mentioned order. At the chair for Robotics and Embedded Systems at the Technische Universität München we develop robot and robot part prototypes and the necessary tools to comply with the short timeline requirements. To parallelise the different development steps, we utilise the Rapid Prototyping Toolset EasyKit which consists of a collection of hardware and software modules and our model based, visual programming tool EasyLab.

5.3.1 Introduction

One of the real world projects, where EasyKit was applied to recently, is the autonomous mobile robot platform in a Life Science Lab (see sections 4.3 and 6.1) of a global pharmaceutical company which needed to be equipped with an adaptive light-source to improve illumination conditions for its image processing. The robot is able to carry out a complete sample management process in a pilot plant. Since the robot moves to different stations and analysis devices in a large laboratory, illumination conditions vary according to the time of the day, and according to the distance to the windows and the overhead lights. While the image processing performed perfectly in well illuminated areas, we wanted the robot to be ready also for challenging environments. Several more difficult scenarios are apparent and were encountered: the overhead lights in the lab might just accidentally be switched off by a human, a power outage may occur and in worst case there would be no daylight late at night to illuminate the environment or due to the absence of windows.

This was the driving force to equip the robot with an adaptive, computer-controlled light source to locally illuminate the area of interest according to ambient light conditions. We wanted the light source to be computer-controlled, since different materials have different reflection properties and may cause problematic highlights if the light is just switched on to maximum power [17].

5.3.2 Motivation

Before we started development, we carried out a literature and parts research, not to reinvent the wheel. The requirements for our light-source were the following:

- Computer-controlled brightness
- Low energy consumption
- Easily attachable to the robot's tool

- As cheap as possible

Since the robot is battery driven, we wanted the light source not to consume too much energy. Furthermore it should fit around the camera to illuminate the region which the robot is looking at and its brightness should be controllable on demand by the robot's on-board computer. Finally, also the price is an important factor.

During parts research we found an appropriate LED array, circular in shape, with 24 or alternatively 48 white LEDs, which perfectly fits around machine vision lenses and whose energy consumption is small due to the utilisation of LEDs. Siemens/NERLITE provides these LED arrays for machine vision applications as off-the-shelf components [80, 81, 82]. The power requirements of the selected ring lights are 12 V/160 mA for the 24 LED R-60-1 "V2" and 24 V/160 mA for the 48 LED R-70-2 "V2". The lights are available with different delivery rings, out of which we chose the diffuse ones for a homogenous illumination.

The brightness of LED arrays can be controlled by adjusting the current, which apparently also changes the wavelength of the emitted light or – which is the commonly preferred method – by powering the LEDs with a pulse width modulated signal, where a low frequency will result in low brightness while a high frequency results in a brighter light. During parts research we came across several components, however after intensive investigation we only found one controller which can be directly connected to a computer via RS232: the Gardasoft *Vision PP610* – LED Lighting Controller with RS232 Control [83, 84]. However at a price of 866 EUR, we double-checked, if it was possible to create an appropriate, computer interfaceable controller by ourselves utilising our Rapid Hardware Prototyping Toolset EasyKit, which resulted in this proposal [85].

5.3.3 Hardware Toolbox

Mechatronic applications often utilise many different Integrated Circuits (IC) which need to be arranged and assembled on a circuit board and connected to data interfaces. For rapid prototyping EasyKit includes a modular toolbox based on Match-X with different circuit blocks, which can be easily put together. Figure 5.5 exemplary shows some components out of the construction kit and the connection through the standardised electrical data bus.

The available modules cover a variety of requirements and provide a CPU block with a Microchip PIC 18F2520 Microcontroller [86] as well as a voltage regulator, a serial interface component and sensor connector components among others. The Match-X standard is published by the VDMA and also covers the development transition from Match-X blocks towards small batch and series production [87, 88].

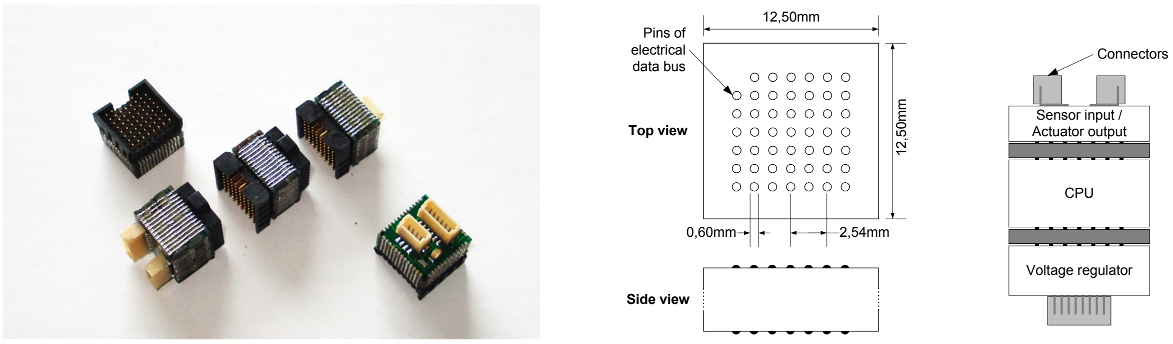


Figure 5.5: Some exemplary modules out of the Match-X construction kit are depicted on the left, while the centre image shows the data bus layout. The right image shows an elementary assembly of three blocks with a CPU, a voltage regulator and sensor connectors.

5.3.4 Software Toolbox

EasyLab is a modular and model based Integrated Development Environment (IDE) for creating software applications focussing on but not limited to embedded systems. It provides two kinds of graphical software models: Structured Flow Charts (SFC) and Synchronous Data Flows (SDF) [89, 90, 91, 92, 93]. SFCs describe the different possible states of a system and the state transitions. The states themselves are references to subprograms implemented in any arbitrary language, while the transitions can be alternative branches conditioned by boolean expressions and parallel branches. A state itself can be another SFC or for example a SDF, which is a directed multigraph, consisting of actors connected by edges, representing the data flow between actors.

The functionality of hardware modules such as described in 5.3.3 is represented by analogous software counterparts with the appropriate input and output connectors for SDFs. In addition to that, the IDE also provides building blocks for generic data types and methods to define constants, variables, arrays and images for example as well as common operations like mathematical functions or image processing filters, just to mention a few of them.

After defining the application logic in EasyLab, the user can invoke the integrated code generator to create customised machine code for the targeted microcontroller, which in turn is usually limited in its resources. Compared to the method to write microcontroller applications from scratch - which still is a common and widely used approach - the utilisation of EasyLab can reduce development time dramatically.

5.3.5 Implementation

Our goal was to develop an adaptive, computer-controlled light-source for the mentioned mobile manipulator to improve illumination conditions for the mounted camera and computer vision system. As

Hardware components we decided to use the Siemens/NERLITE ring illuminators mentioned in 5.3.2 and a controller stack built from Match-X components. The solution for this task was as follows:

Initially the robot takes an image with its onboard camera. Afterwards the brightness of the acquired image is determined by calculating a brightness histogram and its mean in a range from 0 to 255. A controller in the host application gradually increases or decreases the brightness of the ring light by sending brightness values to the serial port until the desired image brightness is achieved or the minimum or maximum possible brightness is reached.

The light's controller-board was designed to have a serial interface to receive brightness commands as plaintext values in a range from 0 to 255, terminated by a linefeed. According to the read value, the microcontroller adjusts the pulse width modulated power signal for the ring-light which is mounted around the camera.

Figure 5.6 shows the controller-board on the left and the test setup for result verification on the right. The controller board basically consists of a Match-X stack with two modules, a power supply connector and connectors for a serial RS485 cable and the ring-light. The Match-X module which is marked black and white contains the voltage regulator for the CPU module and the RS485 serial interface, while the module which is colour-coded in red and orange contains the CPU. The generated pulse width modulated signal is picked up through a wire directly from the data bus and amplified by a MOSFET to match the ring-light's power specification. Figure 5.7 shows the light source with three different brightnesses.

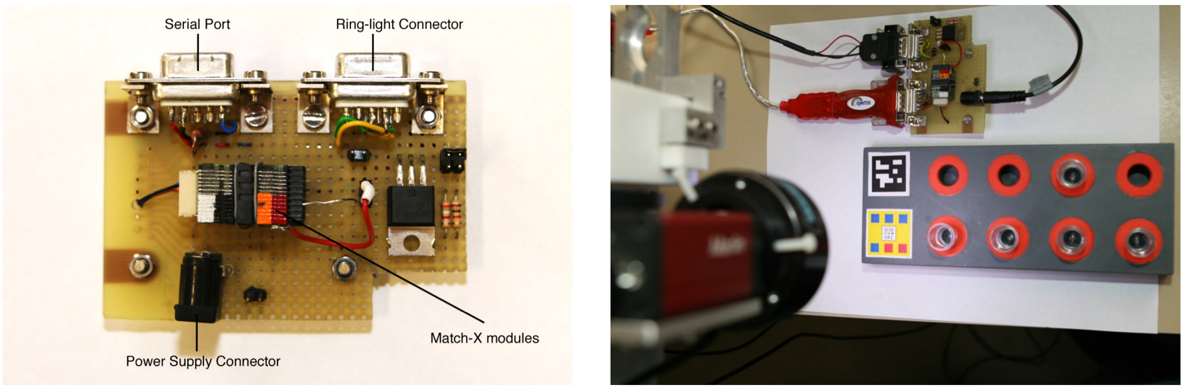


Figure 5.6: The controller board with the serial and the ring-light connector next to the Match-X stack consisting of two modules in this case. The right picture shows the mounted test setup for verification of the results.

5.3.6 Result

To verify that the initial goal of improving illumination conditions for the image processing of the mobile robot was in fact achieved, we set up a series of experiments. The robot was placed in front of a lab

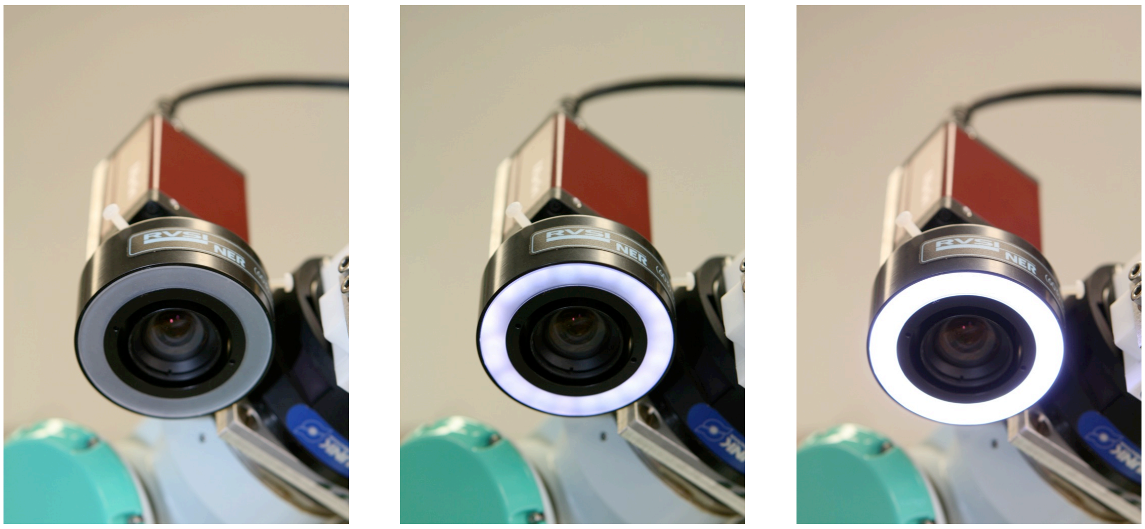
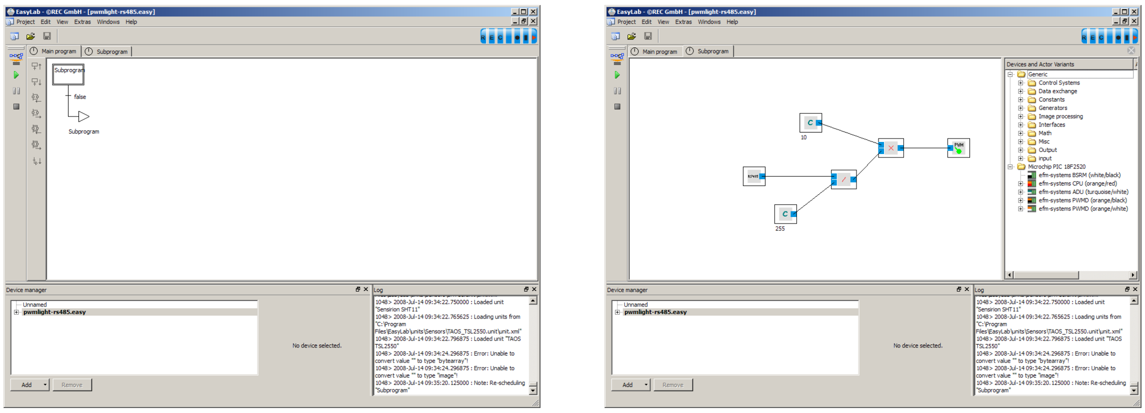


Figure 5.7: The robot's light-source at 0 %, 30 %, and 100 % brightness.


EasyLab - ©REC GmbH - [pwmlight-rs485.easy]
Project Edit View Extras Windows Help
Main program
Subprogram
Subprogram
false
Subprogram
Device manager
Unnamed
pwmlight-rs485.easy
No device selected.
Add
Remove
Log
10
255
RS485
PWM
Devices and Actor Variants
Generic
Control Systems
Data exchange
Constants
Generators
Image processing
Interfaces
Math
Misc
Output
input
Microchip PIC 18F2520
efm-systems BSRM (white/black)
efm-systems CPU (orange/red)
efm-systems ADU (turquoise/white)
efm-systems PWMD (orange/black)
efm-systems PWMD (orange/white)

```
1048> 2008-Jul-14 09:34:22.750000 : Loaded unit
"Sensirion SHT11"
1048> 2008-Jul-14 09:34:22.765625 : Loading units from
"C:\Program
Files\EasyLab\units\Sensors\TAOS_TSL2550.unit\unit.xml"
1048> 2008-Jul-14 09:34:22.796875 : Loaded unit "TAOS
TSL2550"
1048> 2008-Jul-14 09:34:24.296875 : Error: Unable to
convert value "" to type "bytearray"!
1048> 2008-Jul-14 09:34:24.296875 : Error: Unable to
convert value "" to type "image"!
1048> 2008-Jul-14 09:35:20.125000 : Note: Re-scheduling
"Subprogram"
```

Figure 5.8: The software implementation of the adaptive light-source with EasyLab. The main program - on the left - consists of a simple subroutine - shown on the right - with the necessary functional blocks, which can be easily arranged and connected by the user. The program reads a value in the range of 0 to 255 from the serial port, this value is converted into a floating point number through the division by 255 and is scaled up afterwards to a range of 0 to 10 which determines the frequency for the pulse width modulated power signal of the LED array.

workbench and the arm was positioned above a rack with vials. The rack has a coloured patch for calibrating the arm's position by calculating the centroids of the colour segments.

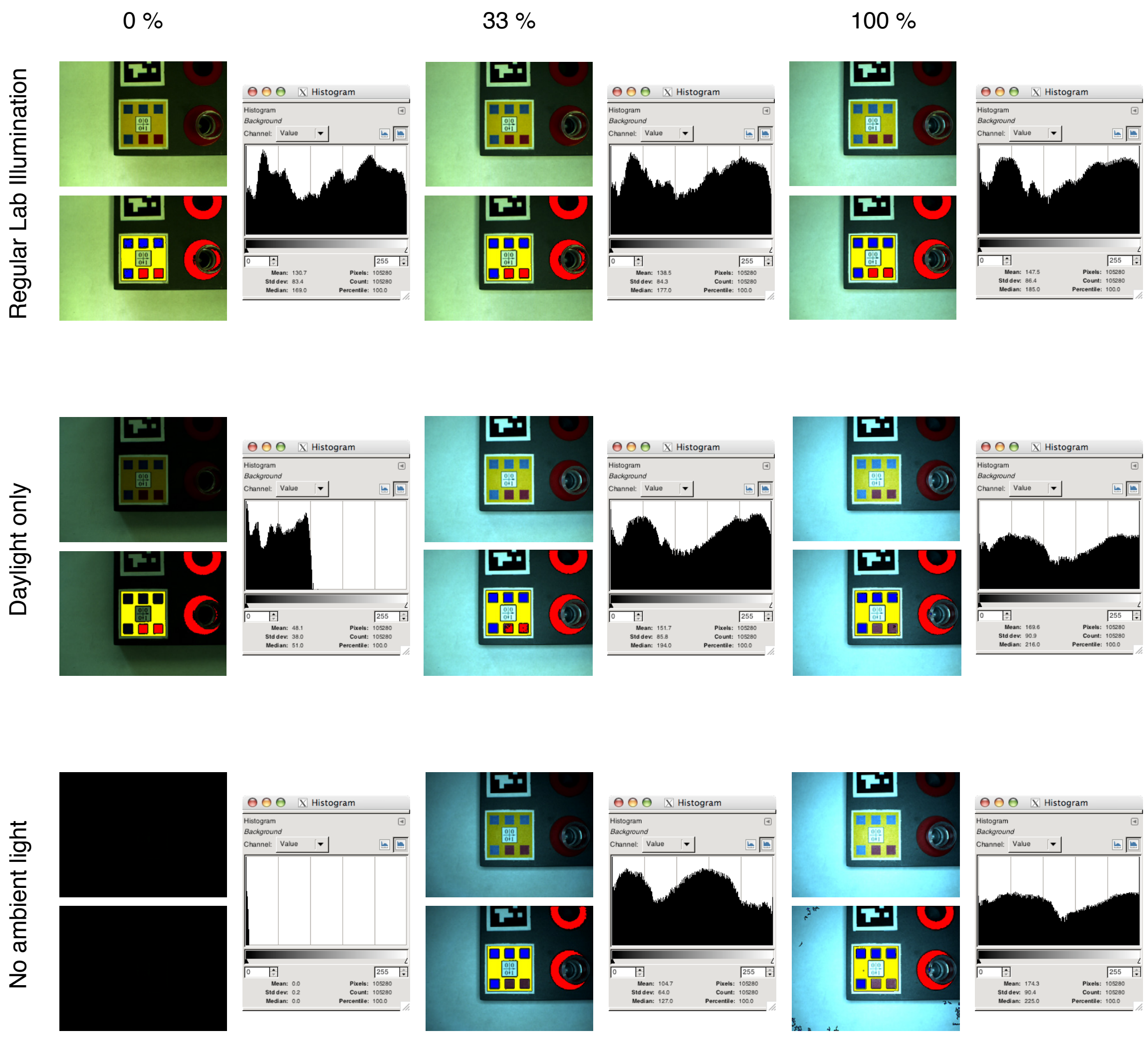
Figure 5.9 depicts the experiment and the results. The first row shows pictures taken at regular lab illumination with the ceiling lights switched on during daytime, while the second row shows pictures which were taken when the ceiling lights were switched off and the scene was illuminated by daylight only which entered through the lab windows. The third row of pictures were taken in darkness with almost no ambient illumination. The three columns in each figure distinguish between different brightness levels of the LED array, which were set to off, 30 %, and 100 % brightness.

Every item in the figure consists of three pictures: the source image, which was taken with the robot's camera, the processed, colour segmented picture below and the brightness histogram with a linear x-axis and a logarithmic y-axis, representing dark values on the left and bright values on the right.

Taking a closer look at the results, and comparing the 0 % column of the figures with the 33 % and 100 % ring light brightness columns, it is apparent that the colour segmentation works significantly better in the more challenging scenarios, where the ceiling lights in the lab are off. These were just three fixed illumination values for better comparability, however the robot's host application is actually implemented in an adaptive way, gradually changing the intensity of the light-source with respect to the mean of the brightness histogram to get optimal results. Furthermore it is possible to adjust the light appropriately, if highlights on shiny surfaces are detected.

5.3.7 Conclusion

The mobile robot's object recognition is carried out by the utilisation of a mounted machine vision camera. The challenge however is the different illumination conditions in a large life science laboratory. Thus the development of an adaptive light source for the visual servoed mobile manipulator was indispensable. NERLITE/Siemens build LED based ring illuminators suitable for many purposes, yet, in our case, the brightness needs to be computer controllable depending on the currently observed object and the ambient light to prohibit highlights on shiny surfaces and to allow pattern matching and vial inspection even in dark areas of the lab. We are aware that computer interfaceable LED controllers are available off the shelf, but we want to point out, that developing custom mechatronic applications utilising the hardware toolbox EasyKit based on Match-X in conjunction with our software toolbox EasyLab can reduce development resource requirements dramatically in terms of money, man-power and time. In the presented application one single business day was sufficient to implement the necessary LED controller, which receives brightness commands via the serial port of a computer, depending on the visual feedback received from the connected camera. The developed computer-controllable light-source contributes significantly to the autonomy and versatility of the robot.



```
Mean: 130.7      Pixels: 105280
Std dev: 83.4    Count: 105280
Median: 169.0    Percentile: 100.0
```

```
Mean: 138.5      Pixels: 105280
Std dev: 84.3    Count: 105280
Median: 177.0    Percentile: 100.0
```

```
Mean: 147.5      Pixels: 105280
Std dev: 86.4    Count: 105280
Median: 185.0    Percentile: 100.0
```

```
Mean: 48.1       Pixels: 105280
Std dev: 38.0    Count: 105280
Median: 51.0     Percentile: 100.0
```

```
Mean: 151.7      Pixels: 105280
Std dev: 85.8    Count: 105280
Median: 194.0    Percentile: 100.0
```

```
Mean: 169.6      Pixels: 105280
Std dev: 90.9    Count: 105280
Median: 216.0    Percentile: 100.0
```

```
Mean: 0.0        Pixels: 105280
Std dev: 0.2     Count: 105280
Median: 0.0      Percentile: 100.0
```

```
Mean: 104.7      Pixels: 105280
Std dev: 64.0    Count: 105280
Median: 127.0    Percentile: 100.0
```

```
Mean: 174.3      Pixels: 105280
Std dev: 90.4    Count: 105280
Median: 225.0    Percentile: 100.0
```

Figure 5.9: Colour segmentation at the Cedex vial rack. While the colour segmentation shows very good results in all cases when the ceiling lights are switched on in the lab, it becomes much harder to examine the colours in a lab when the ceiling lights are off, be it day or night. The LED array significantly improves colour detection in the latter cases.

5.4 Human Robot Interface

The user interface is where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

5.4.1 Introduction

Prices for commercial service robots grow disproportionate with increasing complexity, which asks for intuitive user controls to avoid any damage and to motivate the robot's utilisation. This section addresses briefly how human robot interaction can be implemented in both traditional and unconventional ways.

5.4.2 Graphical User Interface

Graphical user interfaces (GUIs) accept input via devices such as a computer keyboard and a mouse and provide articulated graphical output on the computer monitor. Several Theses have been published, that deal with the construction, usability and implementation of user interfaces in detail [94, 95, 96]. GUIs are realised by the utilisation of the native developer tools of an operating system or with a GUI Toolkit, which are often cross platform capable as opposed to the native developer tools.

Due to the diverse variety of installed operating systems on robotic platforms and client computers, the utilisation of a GUI Toolkit is preferable for robotic applications in my point of view. Ideally, this allows the same code to be compiled for different operating systems as showed in section 5.2. Table 5.3 summarises the attributes and features of the most commonly used C/C++ GUI Toolkits: Qt [59], the GIMP Toolkit (GTK+) [63], Fast Light Toolkit (FLTK) [65] and wxWidgets [64]. A detailed overview about the mentioned and further Toolkits is given in Leslie Polzer's article "GUI Toolkits for The X Window System" [97].

With regard to the feature set, the mentioned GUI Toolkits are pretty similar nowadays as opposed to some of the earlier versions. Although Qt requires the most resources in terms of disk space and ram for an implemented graphical user interface application, it is my personal Toolkit of choice, due to the big user and developer community as well as third party add-ons and due to its use in a variety of well-known projects such as KDE, Google Earth, Skype, Adobe Photoshop Elements and others [98].

	Qt	Gtk+2	FLTK	wxWidgets
Windows	+	o	+	+
Unix/Linux	+	+	+	+
Mac OS X	+	o	+	+
OpenGL	+	+	+	+
GUI Designer	+	+	+	+
Native Widgets	+	+	–	+
Resources	–	o	+	+
Implementation	C++	C	C++	C++

Table 5.3: A summary of features of common Graphical User Interface Toolkits for C/C++.

5.4.3 Simulation

Robotics Simulators are often used to create embedded applications for robots without physically depending on the actual machine. Simulators can help getting familiar with a robotic device without the need to buy hardware. However adaptive simulators that are connected to real robots and that would update the virtual reality environment based on the perceived world of the connected robot, are powerful human robot interfaces for remote operations.

Based on my Diploma Thesis I developed a simulation framework with a physics engine for mobile service robots that displays highly detailed 3D models of robots in their environments (see Figures 5.10 and 5.11) [99]. The simulation can be used for training and testing purposes or to connect to real robots for remote operation.

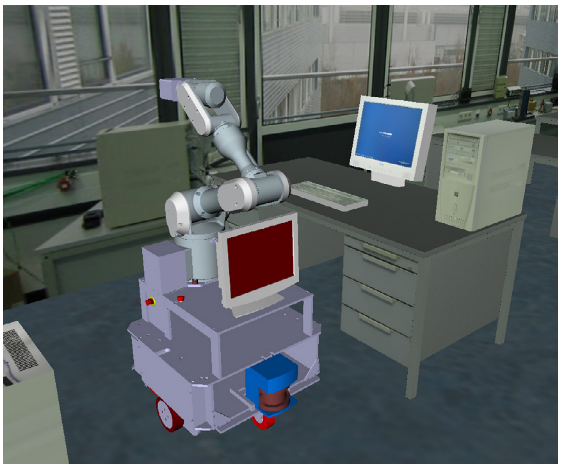


Figure 5.10: Simulation of a Leonardo robot.

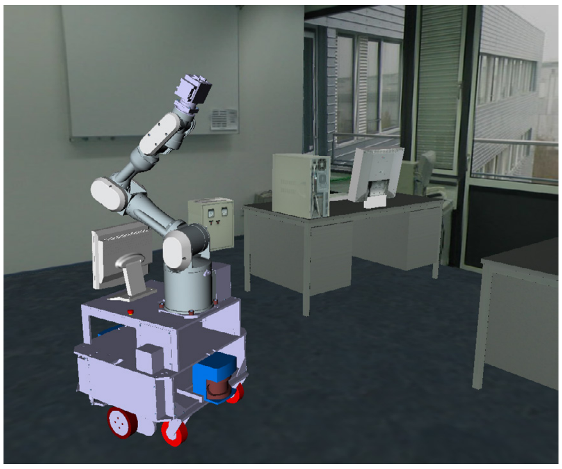


Figure 5.11: Leonardo simulation.

5.4.4 Voice User Interface

A Voice User Interface (VUI) makes human interaction with computers and machines possible through voice or speech recognition and synthesis and serves as a platform in order to initiate a service or process. The VUI is the interface to any speech application. Controlling a machine by simply talking to it was science fiction only a short time ago, but was introduced into more and more recent consumer devices. However people have very little patience for a "machine that doesn't understand".

Colleagues at the department of Robotics and Embedded Systems have demonstrated a successful implementation of speech recognition and synthesis in the EU funded project Joint Action Science and Technology (JAST) [100].

5.4.5 Remote Controls

The goal for autonomous service robots is ultimately to gather enough context information through speech, vision, and gesture recognition to trigger desired robot tasks just as if humans interacted with each other. However as long as speech recognition still misses every tenth word and as long as instantaneous 3D reconstruction from images and semantic interpretation of acquired models remains prototypical and limited to small controlled areas, the user benefits from intuitive remote controls for mobile service robots. While most robots have USB ports to connect a mouse and a keyboard, users tend to avoid controlling robots like computers as most industrial robots have a complicated user interface. In contrast to that, when giving users remote controls at hand, that they are familiar with from other applications in every day life – such as the WiiMote or the iPhone (see Figures 5.12 and 5.13) – they are more willing to look into and experiment with robots.

To overcome the complexity of robot control, I implemented a remote control service utilising a WiiMote for the Leonardo robot. The WiiMote is an inexpensive Bluetooth device, that many people are familiar with and which was well appreciated by colleagues in the biotechnology lab. The iPhone and iPad are also well suitable devices for remote robot control, as they can display current camera images and a live map on their large touch screens.

5.4.6 Image Processing

Image and video processing are powerful tools for human robot interaction. Their application can vary from object and person detection and tracking to the creation of a complete 3D world model. Object and person tracking for initial teaching phases for lab walkthroughs is described more in detail in the next section.

Figure 5.12: The WiiMote is a cheap and easily interfaceable Bluetooth device, that people intuitively know how to use.

Figure 5.13: The iPhone is a multi-touch device with a large screen, that many people are familiar with.

5.4.7 Conclusion

This section presented different methods for human robot interaction which can be realised in many different ways via graphical or voice user interfaces, with simulators and intuitive remote controls or image processing, just to mention a few. The presented applications summarise what I personally experimented with. The right tools can tremendously increase the comfort for end users when controlling mobile service robots.

5.5 Visual Tracking

Visual object tracking is the process of locating moving objects in video sequences in real-time, by exploiting redundant frame-to-frame context information through model-based computer vision techniques. Recent surveys cover many state-of-the-art tracking methodologies [101, 102]. The Open Tracking Library (OpenTL) [103] is an approach of our department to unify many of the state-of-the-art model based visual tracking methodologies in a single software framework [19].

5.5.1 Introduction

In all of the discussed application scenarios (see chapter 2) visual object tracking became a key technology. In the Lab Automation scenarios, a person and object tracker were implemented based on OpenTL to realise a walkthrough teaching process. The same algorithm was used in a TV Studio to locate a moderator and control a manipulator with a camera mounted on it. This section presents the underlying algorithm for the person and object tracker.

5.5.2 Architecture

The software architecture of OpenTL is based on the elementary concepts of every visual tracking system. A scene with more or less prior information about it is recorded by one or multiple image sensors. The prior information enables creation of a model, which at first needs to be detected and in subsequent images needs to be tracked. Figure 5.14 depicts the generic flow diagram for a model-based tracking application.

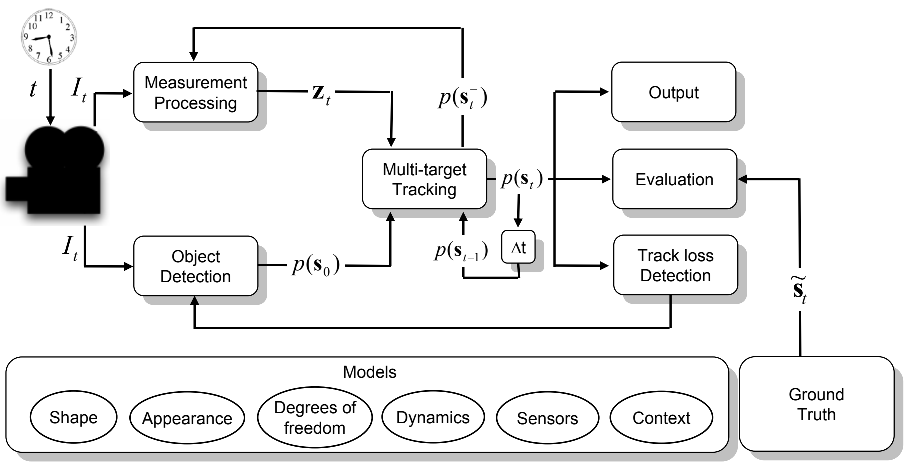


Figure 5.14: Generic flow diagram for a model-based visual tracking application.

The main classes that our library provides are building blocks for this kind of applications. They have been organised in functional layers representing an increasing level of abstraction: from base utilities, image acquisition devices and data structures, up to application-related functionalities (see Figure 5.15).

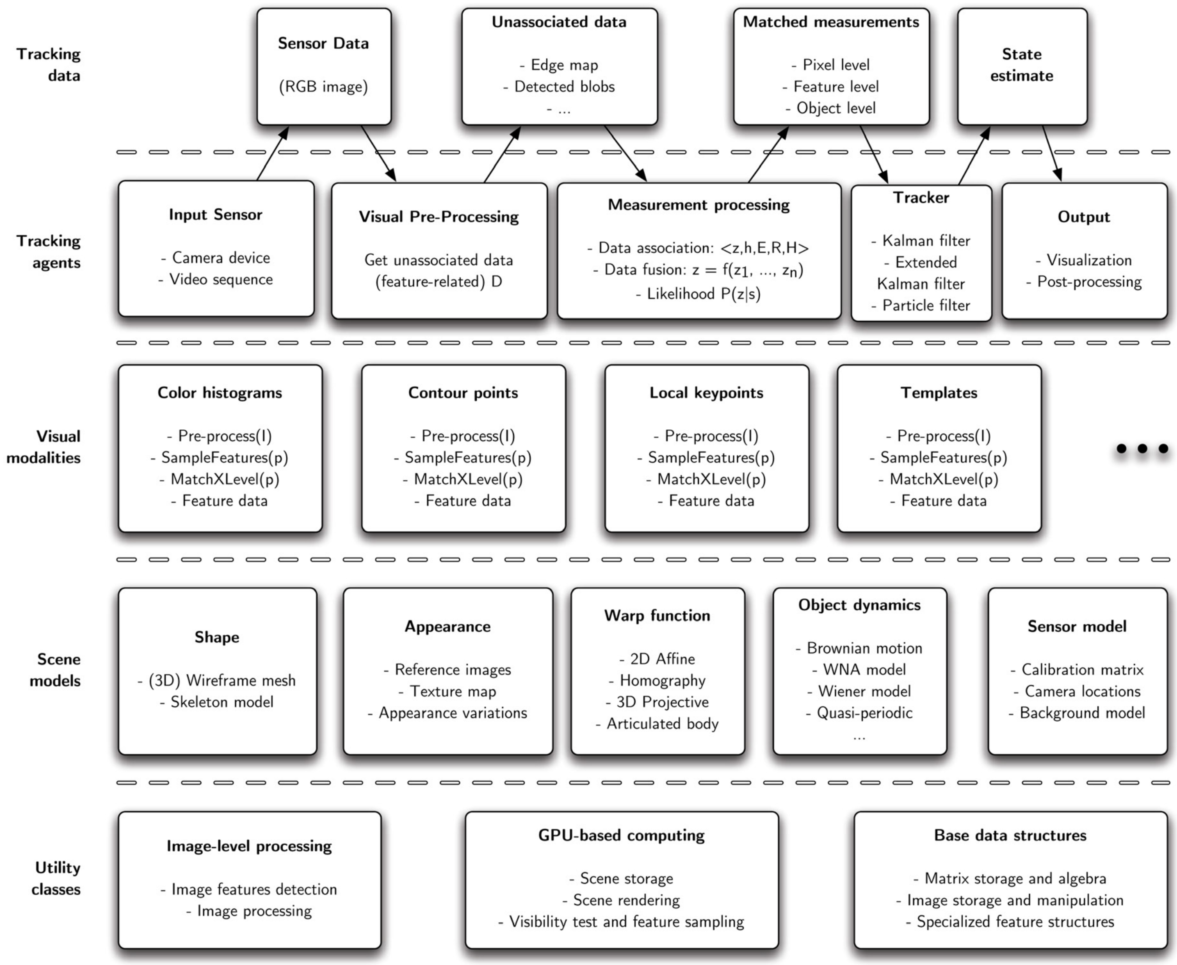

Tracking data
Sensor Data (RGB image)
Unassociated data - Edge map - Detected blobs - ...
Matched measurements - Pixel level - Feature level - Object level
State estimate
Tracking agents
Input Sensor - Camera device - Video sequence
Visual Pre-Processing Get unassociated data (feature-related) D
Measurement processing - Data association: <z,h,E,R,H> - Data fusion: z = f(z1, ..., zn) - Likelihood P(z|s)
Tracker - Kalman filter - Extended Kalman filter - Particle filter
Output - Visualization - Post-processing
Visual modalities
Color histograms - Pre-process(I) - SampleFeatures(p) - MatchXLevel(p) - Feature data
Contour points - Pre-process(I) - SampleFeatures(p) - MatchXLevel(p) - Feature data
Local keypoints - Pre-process(I) - SampleFeatures(p) - MatchXLevel(p) - Feature data
Templates - Pre-process(I) - SampleFeatures(p) - MatchXLevel(p) - Feature data
Scene models
Shape - (3D) Wireframe mesh - Skeleton model
Appearance - Reference images - Texture map - Appearance variations
Warp function - 2D Affine - Homography - 3D Projective - Articulated body
Object dynamics - Brownian motion - WNA model - Wiener model - Quasi-periodic ...
Sensor model - Calibration matrix - Camera locations - Background model
Utility classes
Image-level processing - Image features detection - Image processing
GPU-based computing - Scene storage - Scene rendering - Visibility test and feature sampling
Base data structures - Matrix storage and algebra - Image storage and manipulation - Specialized feature structures

Figure 5.15: The layered software architecture of OpenTL with the currently implemented components.

The OpenTL building blocks can be arbitrarily combined to implement a variety of different visual tracking applications. The functional layers serve the following purposes:

1. *Tracking Data*: the raw sensor signal I_t; any pre-processed data related to a given visual modality; measurement variables and residuals associated to a given object; multi-target posterior state distribution s_t.

2. *Tracking Agents*: input devices, visual pre-processing, feature processing (sampling, match-

ing, data fusion) and likelihood computation, Bayesian tracker, output visualisation and post-processing.

3. *Visual Modalities*: common abstraction for visual features such as image pre-processing, sampling from the model, warp in image space, data association, residual and covariance computation; within each class, storage for off-line and on-line selected model features, and intermediate processing results.

4. *Scene Models*: off-line available models for each object class; base shape and appearance; degrees of freedom for pose and appearance parameters, and related warp functions with Jacobian computations; object dynamics; sensor and context models.

5. *Utility Classes*: low-level, model-free image processing and computer vision tools; base data storage for matrices, vectors and images; basic algebra and image manipulation functions; GPU-assisted scene rendering tools and visibility testing of geometric primitives under a given camera view.

5.5.3 Implementation

The visual tracking software component for the person and object tracker were designed and implemented following the layered architecture of Figure 5.15.

Image input The scene is recorded by an AVT Marlin Firewire camera, that provides RGB colour images. The models in this case consist of single sub-images containing the face, the hand and the upper part of the body as well as additional reference images of known lab equipment.

Image pre-processing The recorded images as well as the reference sub-images are transformed into the HSV colour space to create two dimensional colour histograms z_{col} for the colour-based likelihood.

Tracker Each tracker holds a state-space representation of the 2D model pose, given by a planar translation (x, y) and scale h of the respective rectangular model in the image plane. Considering a single person tracker consisting of a hand, a head and an upper body tracker, the three particle filters provide the sequential prediction and update of the respective 2D states $s_1 = (x_1, y_1, h_1)$, $s_2 = (x_2, y_2, h_2)$ and $s_3 = (x_3, y_3, h_3)$.

Every particle filter generates several prior state hypotheses s_t^i from the previous particle distribution $(s^i, w^i)_{t-1}$ through a Brownian motion model.

$$s_t^i = s_{t-1}^i + v_t^i \tag{5.1}$$

with v a white Gaussian noise of pre-defined covariance in the (x, y, h) state variables. A deterministic resampling strategy [104] over the previous weights w_{t-1}^i is employed every time in order to keep a good distribution of the particle set.

For each generated hypothesis, the tracker asks for a computation of the likelihood values $P(z_{col}|s_{head}^i)$ for the head, $P(z_{col}|s_{hand}^i)$ for the hand and $P(z_{col}|s_{body}^i)$ for the upper body tracker.

Colour likelihood The rectangular reference image boxes defining the teacher's shape model are warped onto the HSV image at the predicted particle hypotheses s_t^i; for each patch p, underlying H and S color pixels are collected in the respective 2D histogram $q_p\left(s_t^i\right)$, that is compared with the reference one q_p^* through the Bhattacharyya coefficient [105]

$$B_p\left(q_p\left(s\right), q_p^*\right) = \left[1 - \sum_n \sqrt{q_p^*\left(n\right) q_p\left(s, n\right)}\right]^{\frac{1}{2}} \tag{5.2}$$

where the sum is performed over the $(N \times N)$ histogram bins (currently implemented with $N = 10$).

The color likelihood is then evaluated under a Gaussian model in the overall residual

$$P(z_{col}|s_t^i) = exp(-\sum_p B_p^2/\lambda_{col}) \tag{5.3}$$

with given covariance λ_{col}

Estimated state Afterwards the average state $\overline{s}_t$

$$\overline{s}_t = \sum_i w_t^i s_t^i \tag{5.4}$$

is computed for each tracker.

In order to ensure jitter-free operations the trackers' outputs are finally filtered by a temporal low-pass filter with higher cut-off frequencies with respect to the expected covariance of motion parameters.

Loss detection An important feature of our system is the loss-detection, as a face or a hand disappears when the teacher turns around to lead the robot and to re-initialise the trackers as the teacher faces the robot and points at locations of interest.

In principle, target losses can be detected by checking the likelihood and covariance for the particle set. A covariance test is independent of the actual likelihood value, but it may fail to detect a loss when the particle set concentrates on a false positive which has a low covariance as well. On the other hand, the likelihood test is dependent on the likelihood value, which can vary under changing illumination conditions, however the changes would occur relatively slow considering the large amount of frames in a certain period of time.

Therefore we implemented a likelihood test on the estimated state $\overline{s}_t$ of each tracker and declare a loss whenever $P(z_{col}|\overline{s}_t)$ decreases below a minimum threshold value P_{min}. This threshold is set as a percentage (e.g. $\leq 10\%$) of a reference value P_{ref}, initially set to the maximum likelihood value.

In order to provide adaptivity to variable postures as well as light or shading variations, P_{ref} is being slowly adapted if the last likelihood $P(z_{col}|\overline{s}_{t-1})$ is comparable to P_{ref} (e.g. $\geq 60\%$). When a track loss occurs, the affected particle filter are re-initialised with the diffuse prior, until the target becomes visible again and the likelihood increases above the threshold.

5.5.4 Conclusion

Based on the described method we implemented a multi target tracking system for a Teaching Process to guide a mobile robot through an unknown laboratory and to teach positions of interest for subsequent, repetitive walkthroughs [5]. The same algorithm was previously used in a TV Studio to autonomously control a camera robot based on the person tracker's feedback while tracking the moderator of a TV production [15], which demonstrates the versatile application domains of the Tracking framework.

5.6 Dead Reckoning and Odometry

In traditional aviation or nautical navigation, the term "dead reckoning" means to estimate position without external references. The position of a vehicle is dead reckoned using knowledge about its course and speed over a period of time. In robotics, the necessary information is often acquired by measuring wheel revolutions. Encoders are coupled to a robot's drive wheels and act like digital odometers. They generally provide a good estimate of displacements s_L and s_R for the left and right wheels. However accuracy depends on factors such as friction and slip on the surface tat the robot moves along.

5.6.1 Introduction

Many wheeled mobile robot platforms have a differential steering system. So do the Leonardo, F5, and F5-S platforms. If the wheels turn at equal speed, the robot moves along a straight line. If both wheels turn in opposite directions at the same speed, the robot pivots. Two constant but different speeds applied to the wheels will make the robot move along a circular path.

5.6.2 Application

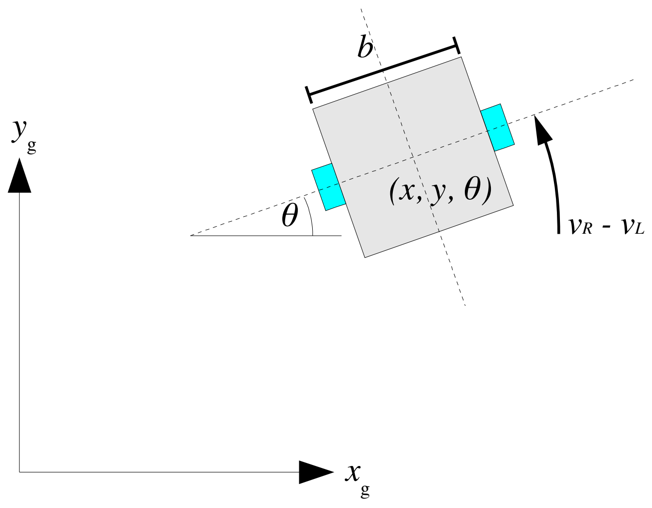


Figure 5.16: Differential drive system.

Given a robot platform with a differential drive system and wheel encoders, a user wants to control motion executions and keep track of where the robot is. In the first case the desired forward velocity

v and angular velocity θ is given and the speeds v_L and v_R of the left and right motors are to be determined.

According to figure 5.16:

$$v_L = r\theta \tag{5.5}$$

$$v_R = (r+b)\theta \tag{5.6}$$

$$v = (r+\frac{b}{2})\theta \tag{5.7}$$

From equation 5.7 r can be resolved:

$$r = \frac{2v-b\theta}{2\theta} \tag{5.8}$$

$$v_L = \frac{2v-b\theta}{2} \tag{5.9}$$

$$v_R = \frac{2v+b\theta}{2} \tag{5.10}$$

with r being the radius of the circular path, b the distance between the two motorised wheels and v the speed at the pivot point of the platform. Be r_W the radius of the wheels and g the gear reduction factor, the desired speeds for the wheels become:

$$s_L = \frac{2v-b\theta}{4r_W\pi}g \tag{5.11}$$

$$s_R = -\frac{2v+b\theta}{4r_W\pi}g \tag{5.12}$$

In the second case the position and orientation are to be computed from the known velocities of both motorised wheels. In this case the following common formulas are applied [106]:

$$\bar{v} = \frac{(v_R+v_L)}{2} \tag{5.13}$$

$$\theta = \frac{(v_R-v_L)}{b}+\theta_0 \tag{5.14}$$

$$x = \bar{v}\cos(\theta)+x_0 \tag{5.15}$$

$$y = \bar{v}\sin(\theta)+y_0 \tag{5.16}$$

5.6.3 Conclusion

This section showed how to compute the desired speeds for the wheels of a robot with a differential drive system and how to keep track of the robot's pose by the utilisation of wheel encoders. Although these formulas are just approximations, they are well applicable as long as the sampling frequency is high enough. With this implementation and an update frequency of 100 Hz the Leonardo and F5 robots are able to manoeuvre with a precision of a few mm in medium sized rooms. Nevertheless an accumulating error is inevitable. For this reason the odometry readings often serve as measurements for the motion model of laser range finder based localisation and mapping approaches.

5.7 Simultaneous Localisation and Mapping

Perception plays an essential role in Service Robotics. Mobile service robots have to know where they currently are, where they find objects and places, which they are providing a service for and they have to be aware of their environment to prohibit injuries or damages caused by collisions. This section describes how laser rangefinder data is used for simultaneous localisation and mapping.

5.7.1 Introduction

Particle filter based, probabilistic approaches have shown very promising results for simultaneous localisation and mapping (SLAM). Based on DP-SLAM, an improved software package has been developed which was utilised successfully on the Robotino and Leonardo platforms utilising SICK LMS 200 laser range finders. "SLAM addresses the problem of constructing an accurate map in real time despite imperfect information about the robot's trajectory through the environment". Unlike other approaches that assume predetermined landmarks, the presented algorithm is purely laser range finder based. It uses a particle filter to represent both robot poses and possible map configurations. DP-SLAM is able to maintain and update hundreds of candidate maps and robot poses efficiently by using a novel map representation called distributed particle (DP) mapping [107, 108]. "Even with an accurate laser range finder, map-making presents a difficult challenge: A precise position estimate is required to make consistent updates to the the map, but a good map is required for reliable localisation. The challenge of SLAM is that of producing accurate maps in real time, based on a single pass over the sensor data, without an off line correction phase. Straightforward approaches that localise the robot based upon a partial map and then update the map based upon the maximum likelihood position of the robot tend to produce maps with errors that accumulate over time. When the robot closes a physical loop in the environment, serious misalignment errors can result." This is avoided by implementing a particle filter that also creates map hypotheses.

5.7.2 Implementation

A particle filter is a simulation-based method of tracking a system with partially observable state. It maintains a weighted set of sampled states $S = \{s_1...s_m\}$, called particles. Upon observing an observation o the particle filter:

1. Samples m new states $S' = \{s'_1...s'_m\}$ from S with replacement.
2. Resamples each new state through a Markovian transition model: $P(s''|s')$.
3. Weighs each new state according to a Markovian observation: $P(o|s'')$.
4. Normalises weights for new set of states.

For localisation a particle represents a robot's pose (x_i, y_i, θ_i). The motion model differs across robot types and terrains but it is generally based on odometry values and takes a linear shift into account to address systematic errors and gaussian noise. For Odometer changes of x, y, and θ a particle i results in:

$$\begin{aligned} x_i &= a_x * x + b_x + N(0, \sigma_x) \\ y_i &= a_y * y + b_y + N(0, \sigma_y) \\ \theta_i &= a_\theta * \theta + b_\theta + N(0, \sigma_\theta) \end{aligned}$$

with linear correction factors a and b and gaussian noise $N(0, \sigma)$ with mean 0 and standard deviation σ. Assuming a standard deviation in laser readings and $\delta = d' - d$ where d' is the reported distance of a laser measurement and d the calculated laser cast within the generated map the total posterior for particle i is

$$P_i = \Pi_k P(\delta_{ik} | s_i, m),$$

where δ_{ik} is the difference between the expected and measured distances for sensor cast k and particle i.

The key contribution of DP-SLAM is distributed particle mapping, which allows to create a particle filter over robot poses and maps. Its core data structure is an ancestry tree and an occupancy grid which enable an efficient and fast mapping process. The environment is subdivided into an array of rectangular cells. The resolution of the environment mapping depends on the size of the cells. Additionally, a probabilistic measure of occupancy is associated with each cell. This measure marks the cell as occupied or unoccupied. A grid square consists of an ancestry tree which stores all particles which have updated this grid square and illustrates an internal storage of a map.

5.7.3 Conclusion

While most particle filter based SLAM methods require substantial resources in terms of memory and computing power, our solution has been adapted to run in soft realtime on currently available off the shelf PCs or Laptops and for easily adjusting the necessary resource parameters to provide an appropriate solution for different mobile platforms. This software package has been installed on the Leonardo robot for its first industrial application. Figures 5.17 and 5.18 show first results of automatically acquired maps in the Life Science Laboratory, where the Leonardo robot was installed. In comparison to other approaches, DP-SLAM does not need predetermined landmarks and is accurate enough to close loops without any additional off-line techniques. The data association problem was eliminated through the abandonment of landmarks. Finally it is not necessary to predetermine the environment.

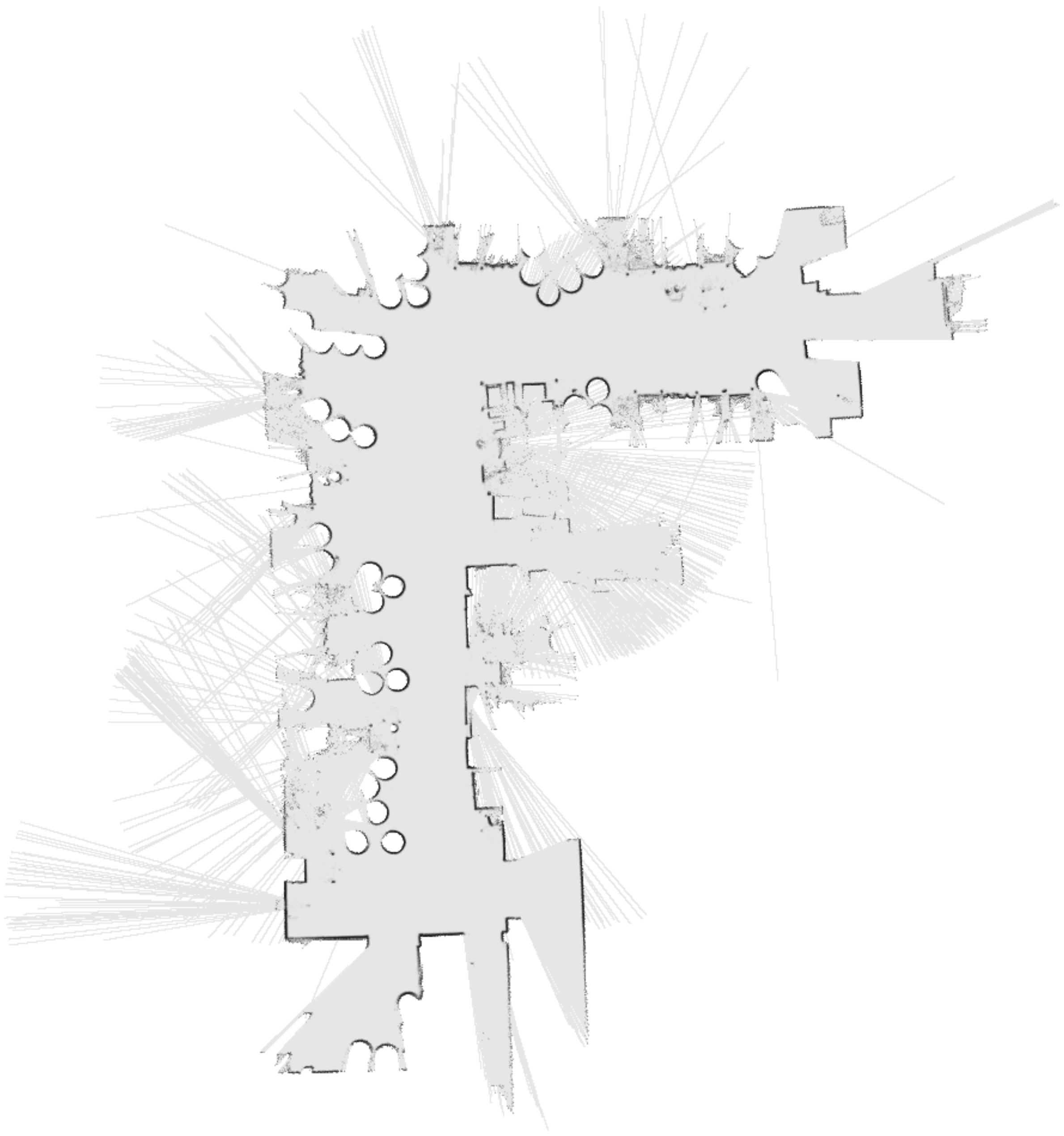


Figure 5.17: An automatically generated map, created by the Leonardo robot in a Cell Culture Development laboratory.

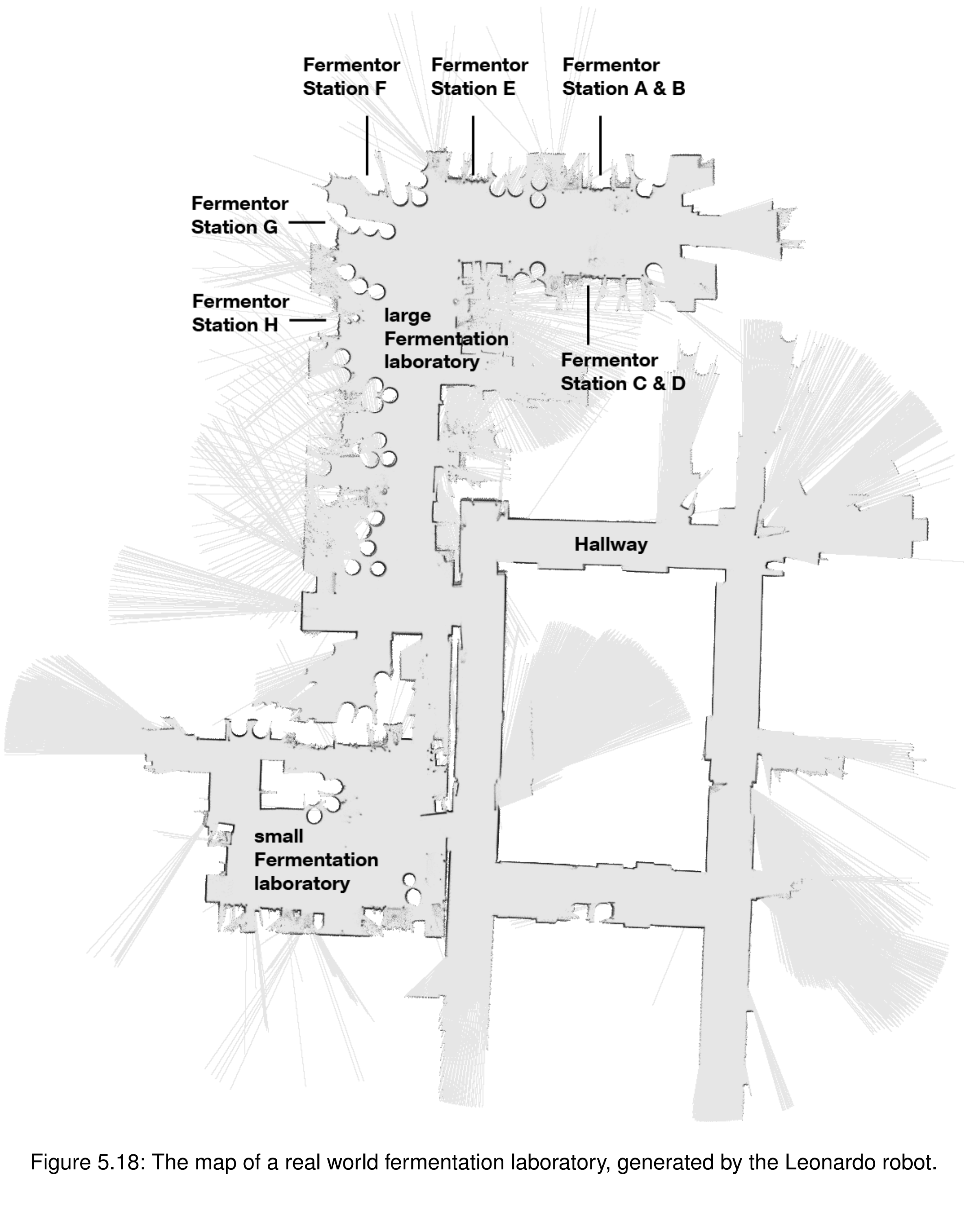


Figure 5.18: The map of a real world fermentation laboratory, generated by the Leonardo robot.

5.8 Summary

This chapter described commonly utilised methods for the development of service robots. Section 5.1 gave a brief overview about collaboration methods, section 5.2 demonstrated how to develop cross platform software components for heterogenous applications, such as robots are. In section 5.3 a method is shown for rapid hardware prototyping that speeds up the development process for robotic hardware components. The later sections 5.5 and 5.7 focus on perception, to dramatically improve the awareness of robot's environments.

Chapter 6

Results

The results section gives a brief overview, of how the initially stated application scenarios for mobile service robots have been addressed with the solutions that we developed in collaboration with our industrial partners.

6.1 Lab Automation

We addressed the Lab Automation Scenario (see sections 2.1 and 2.2) with a feasibility study about implementing the complete sample management process in a real industrial Life Science Pilot Plant at Bayer HealthCare LLC in Berkeley with an autonomous mobile manipulator Leonardo (see section 4.3). Sample management is an inevitable and time-consuming part during the development and production of biopharmaceuticals to keep track of growth parameters and to adjust these as it becomes necessary. The robot has been developed in a close cooperation between the University of Bielefeld, the Technische Universität München, Germany, and Bayer HealthCare in Berkeley, California [109]. After transferring the robot to the customer, it has been upgraded and customised to match local conditions and to serve the company's specific lab devices. Reliability and robustness have been demonstrated in 101 error free sampling cycles.

Precise device interaction is carried out as follows: to move to workstations, the robot positions itself using the laser range finders' feedback which is matched against the acquired map of the lab. Furthermore, to overcome precision issues of the platform, all relevant devices are tagged with colour markers as shown in Figure 6.1, which the robot approaches with the arm and its mounted tool. The camera is constantly giving visual feedback for its actual position (see Figure 6.2). Finally, after reaching known positions, scripted force controlled movements are triggered to prohibit damage due to physical contact.

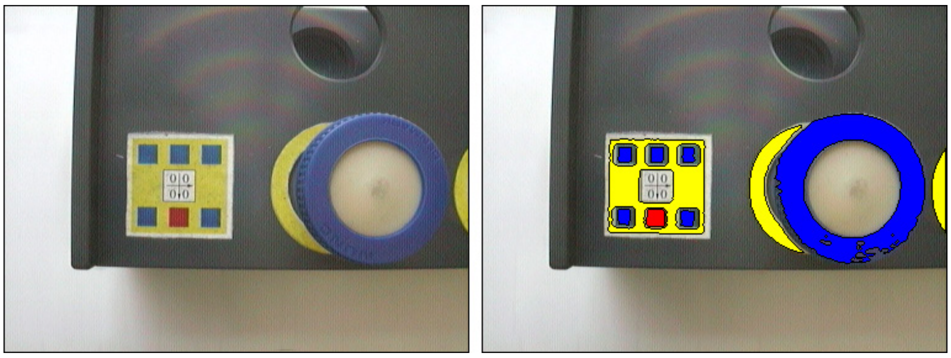


Figure 6.1: Image Processing using colour features.

To serve the local requirements, the robot's tool has been replaced in a way that it can handle 10 ml Vacutainer sample tubes as well as common 50 ml Falcon[1] sample tubes and Cedex[2] cups. The analog camera was replaced by a firewire camera to improve image quality, furthermore a Siemens/NERLITE[3] ring illuminator was attached to the lens to improve the illumination in critical areas (see also section **??**). Figure 6.3 shows the new tool of the robot arm with a sample.

The lab itself has a bioreactor and a Cedex Cell Counter. Depending on the cell sample a dilution step was necessary for the Cedex. Humans usually perform the dilution by hand utilising common manual pipettes. To enable the robot to carry out the dilution, a sample preparation station was constructed consisting of a Hamilton[4] PSD/2 electrical syringe connected to a Hamilton modular valve positioner serving the necessary liquids. The sampling in turn is performed by a dedicated pneumatic valve, which was constructed to take the sample vials. All lab devices were connected to controllers with serial or ethernet network interfaces, which enabled the robot to trigger specific actions, when required. Figure 6.4 shows the complete setup in the pilot plant, while figure 6.5 depicts the sample management process as an UML sequence diagram [110].

First, the robot invokes the sampling of the bioreactor. Once done, it takes the vial from the sampling valve. In the next step it places the vial at the sample preparation station and triggers the dilution process. In the meantime the robot picks up a new Cedex cup and places it at the dispenser needle of the preparation station, where 1 ml of the sample is dispensed after complete preparation. Afterwards the robot places the Cedex cup at the Analyser and triggers the measurement process. Meanwhile the sampling vial is removed from the preparation station by the robot and placed in the waste basket, triggering an automated cleaning program of the preparation station. Furthermore the robot also removes the Cedex cup after it's analysation and places it in the biohazard waste basket. Finally the

[1]http://www.bd.com/
[2]http://www.innovatis.com/products_cedex
[3]http://www.nerlite.com/
[4]http://www.hamiltoncompany.com/

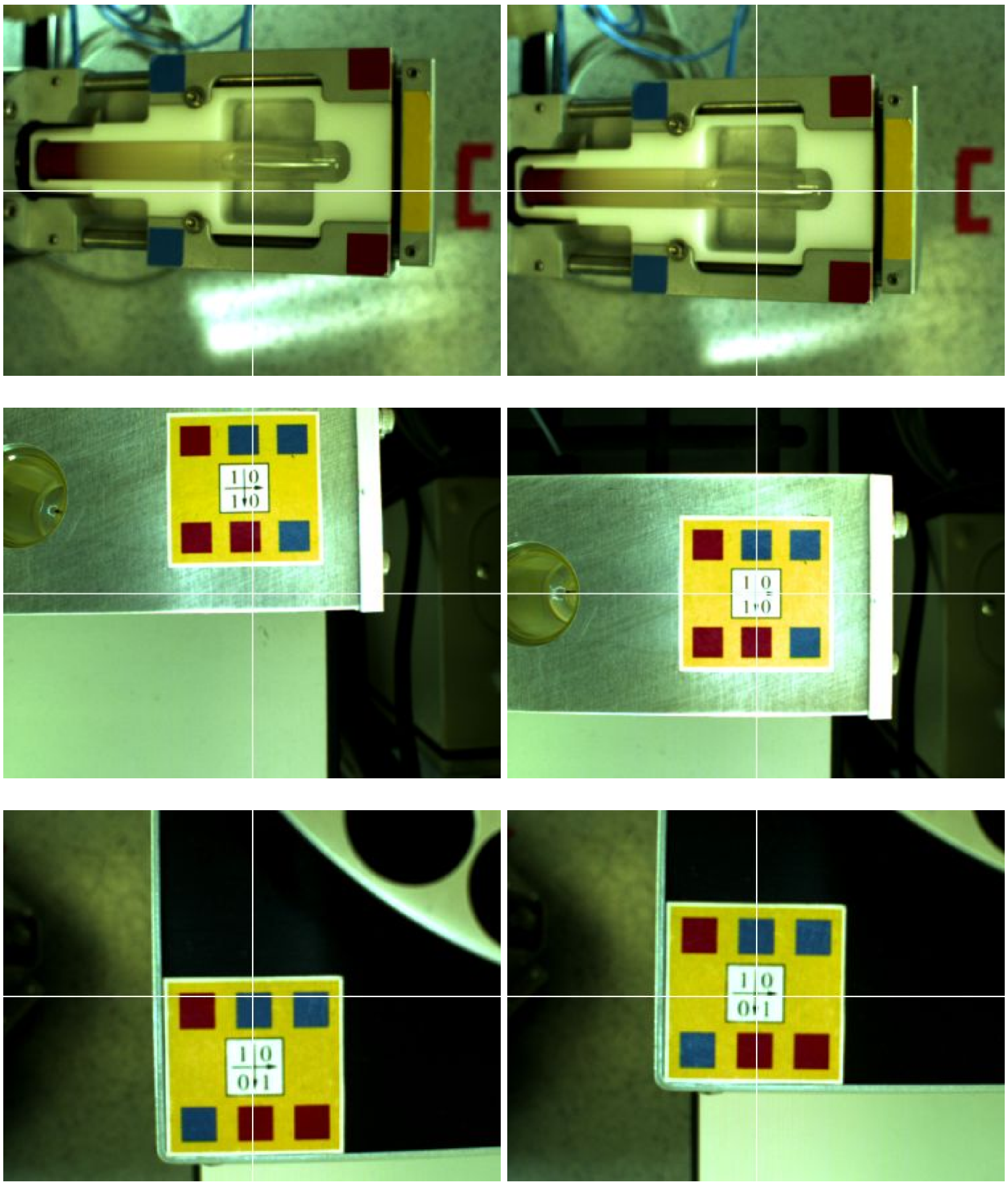


Figure 6.2: Three examples for approaching different lab devices: the sample valve (top), the sample preparation station (middle) and the Cedex cell counter (bottom). Iteratively the robot arm's position converges over a defined position on top of the colour marker.

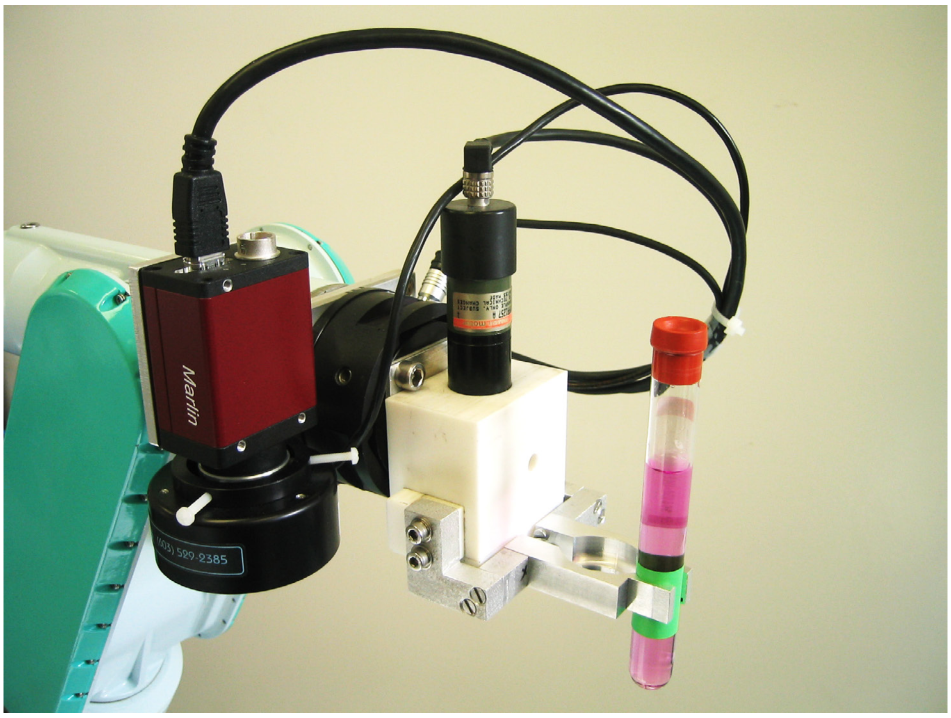


Figure 6.3: The robot's new tool with a Firewire camera and a ring illuminator surrounding the lens. The adjusted gripper carries one of the new 10 ml sample vials.

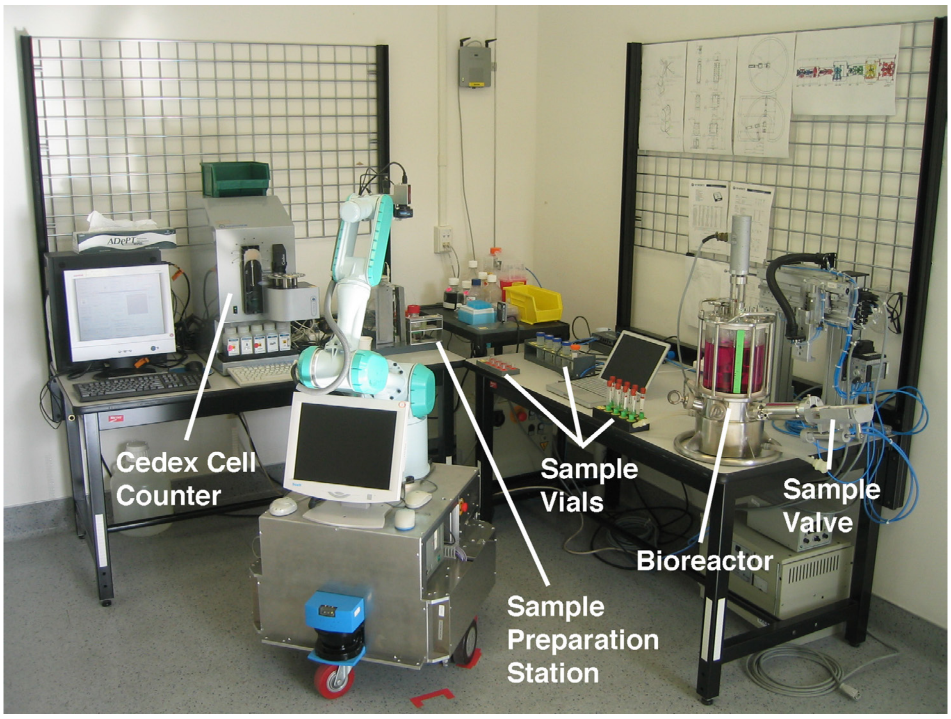


Figure 6.4: The robot lab for the cell counting sample management process with all necessary devices.

robot places a new 10 ml sample vial in the cradle of the sampling valve, which leaves the setup ready for the next cycle [22].

To ensure repeatability, the implemented process was tested for its robustness. Therefore the robot was required to carry out 100 error free cycles. The sampling valve was driven pneumatically and equipped with steam connectors for sterilisation. It has been tested for approximately 2000 cycles without any mechanical issues, while the entire sample management process was carried out successfully for 101 times. Out of that number, 75 cycles were executed with real cell samples to measure deviations and the physical stress caused by the automated sampling and sample preparation. Figures 6.6 and 6.7 show eight viability and viable cell density (VCD) measurements per sample comparing the robot's results with two manual results side by side – one manual result with the sample taken out of the original 50 ml sample vial and one out of the Vacutainer vial, which the robot had just used. While the graphs show good overall comparability, especially with the cells from cell line 2, which seem to be less sensitive, the robot's results in general appear to be slightly lower. The reason for this probably resides in the higher number of syringe strokes at the sample preparation station utilised by the robot [21, 16, 11, 9].

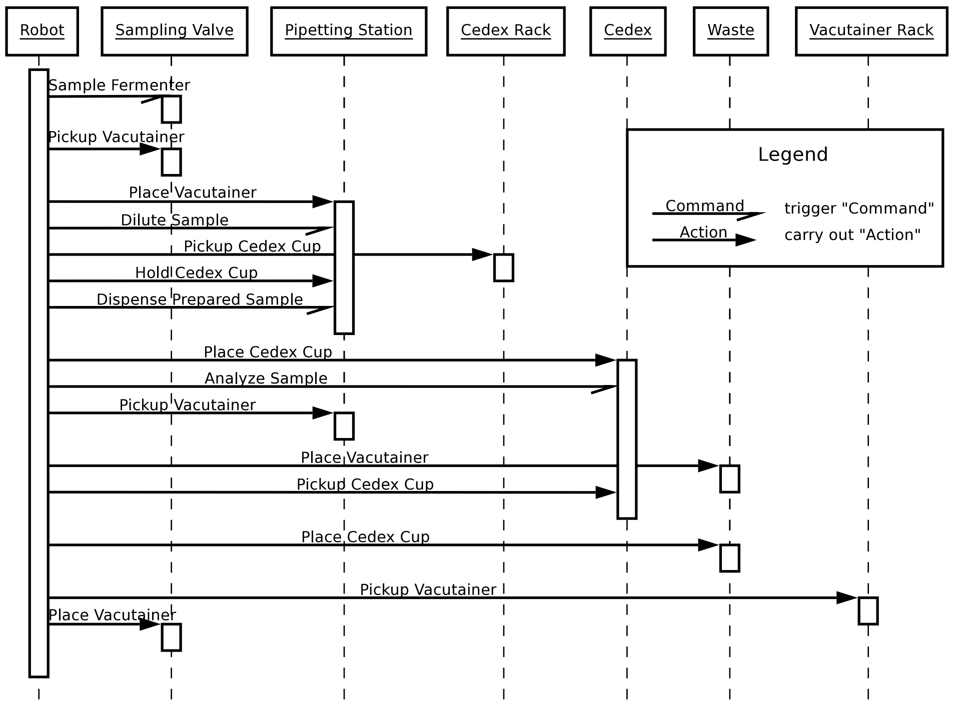
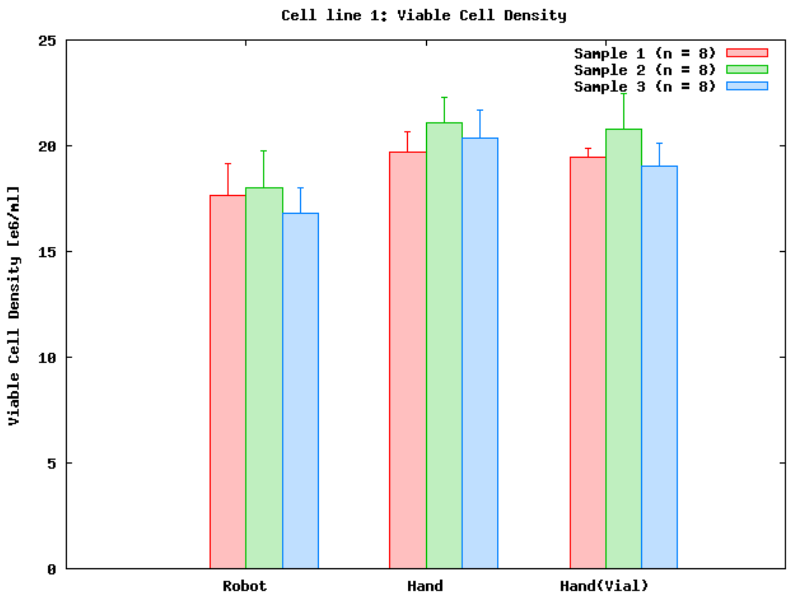


Figure 6.5: The implemented sample management process as an UML sequence diagram. The boxes on the top represent the devices, which the robot needs to operate. The two different kind of arrows symbolise commands, which are triggered by the robot and actions which the robot carries out itself.



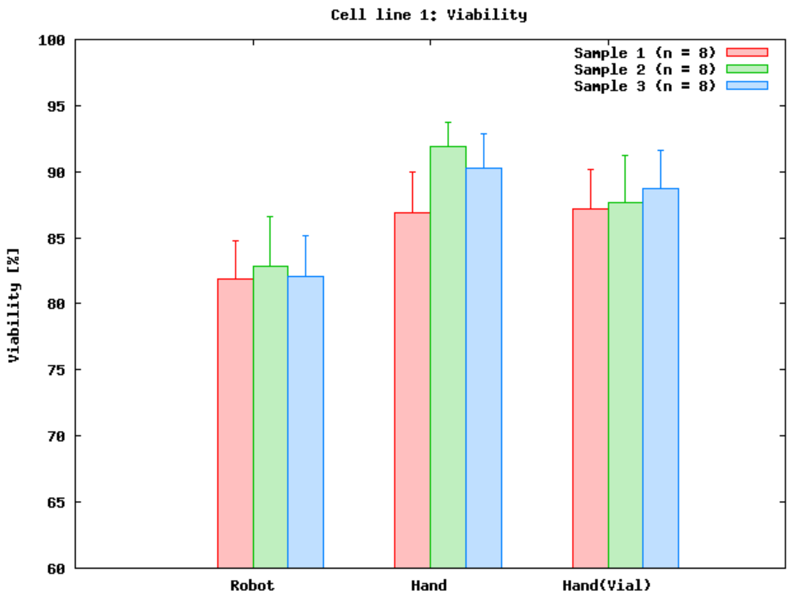
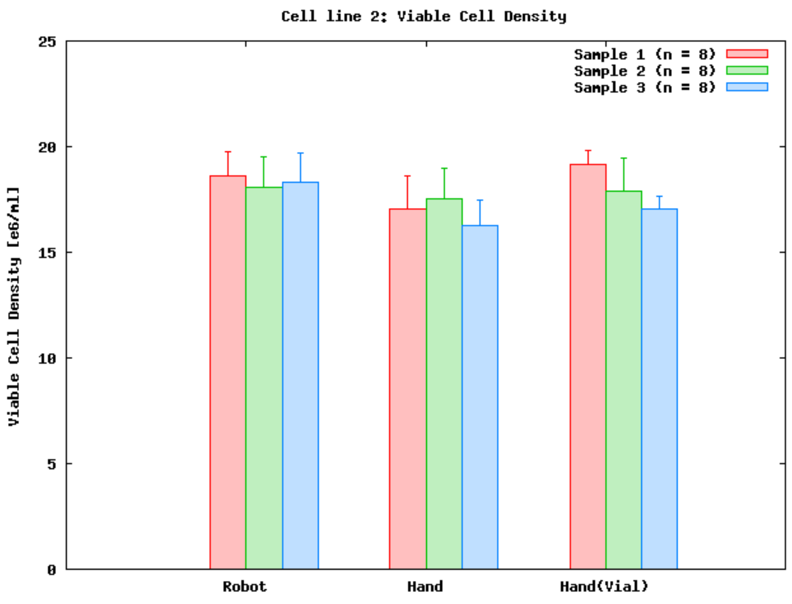


Figure 6.6: Comparison of the Viable Cell Density and Viability from three different samples of Cell line 1. Each bar represents the average and the standard deviation of eight measurements with the same cell sample. Furthermore the measurements are split into three sections: measured by the robot (Robot), measured manually out of the original cell source at almost the same time (Hand) and measured manually out of the 10 ml vial, that the robot had just used (Hand(Vial)). The results show slightly lower viable cell density and viability values when measured by the robot. This may be a result of the increased stress and higher number of syringe strokes at the sample preparation station compared to manual pipetting, which may have an impact on this cell line in particular.



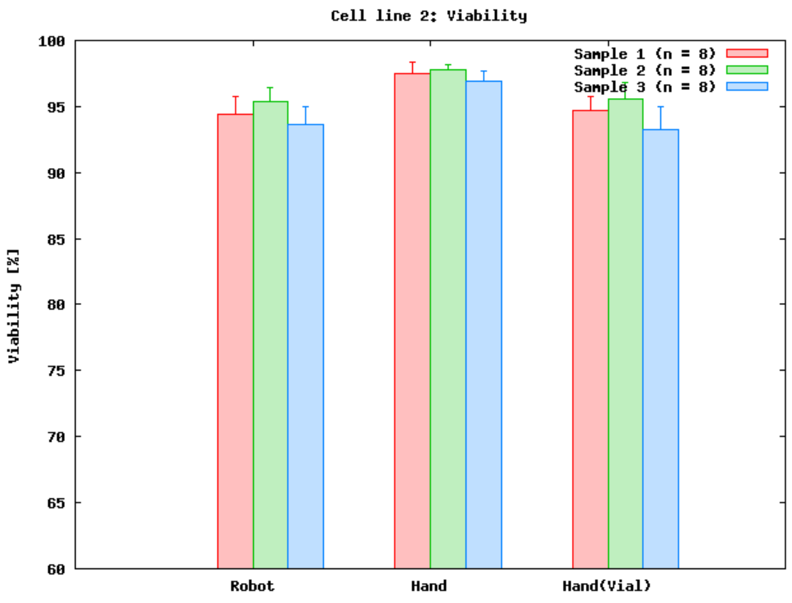


Figure 6.7: Comparison of the Viable Cell Density and Viability from three different samples of Cell line 2. Each bar represents the average and the standard deviation of eight measurements with the same cell sample. Furthermore the measurements are split into three sections: measured by the robot (Robot), measured manually out of the original cell source at almost the same time (Hand) and measured manually out of the 10 ml vial, that the robot had just used (Hand(Vial)). The results show no significant difference between the robot's measurements and the manual measurements.

6.2 Surveillance

After carrying out the feasibility study for complete Sample Management the mobile robot Leonardo has been installed in a real Cell Culture Development laboratory on site at Bayer HealthChare LLC in Berkeley for monitoring multiple high cell density perfusion reactors. As the project progressed new challenges were encountered. The lab setup was not completely static. Therefore the robot should be easily adjustable to a changing environment, without the demand for a dedicated programmer. A novel simultaneous localisation and mapping (SLAM) algorithm based on the laser range finder measurements (see section 5.7) and a person tracker based on the high resolution images from the camera (see section 5.5) enable the robot to discover an unknown laboratory and to monitor Fermentor stations (see Figure 6.8).

During the initial teaching phase the robot follows the teacher and continuously generates a map (see Figure 6.9). Constraint rules and the continuous evaluation of the laser range finders' data makes sure, that collisions with the possibly unknown environment are prohibited. At points of interest the teacher indicates with a gesture to record the location and to search the location for known lab devices, such as fermentors and pumps (see Figure 6.10). These are checked by the robot autonomously on subsequently scheduled walkthroughs [10, 5, 4, 2].

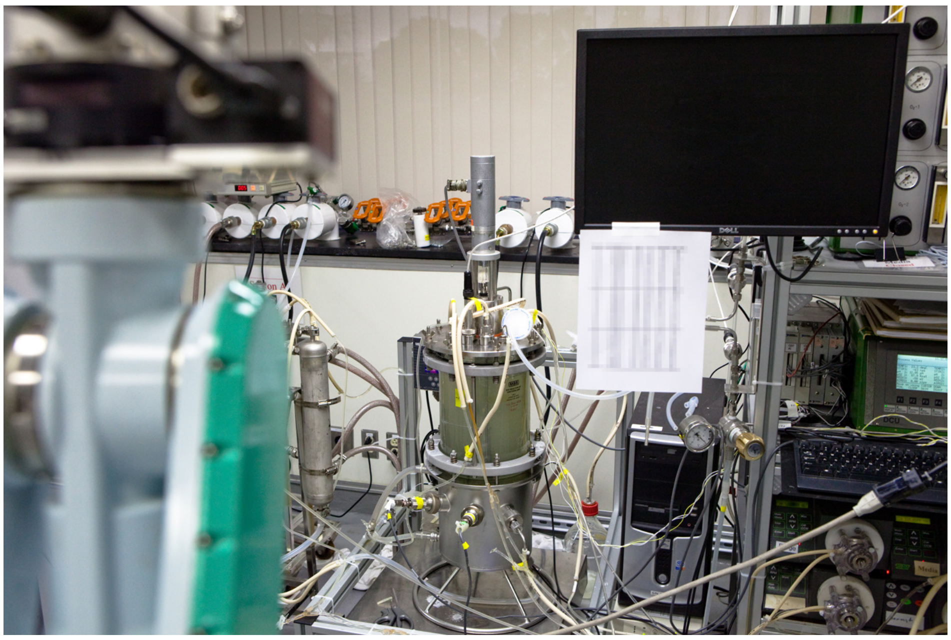


Figure 6.8: The mobile robot Leonardo checking a fermentor station.

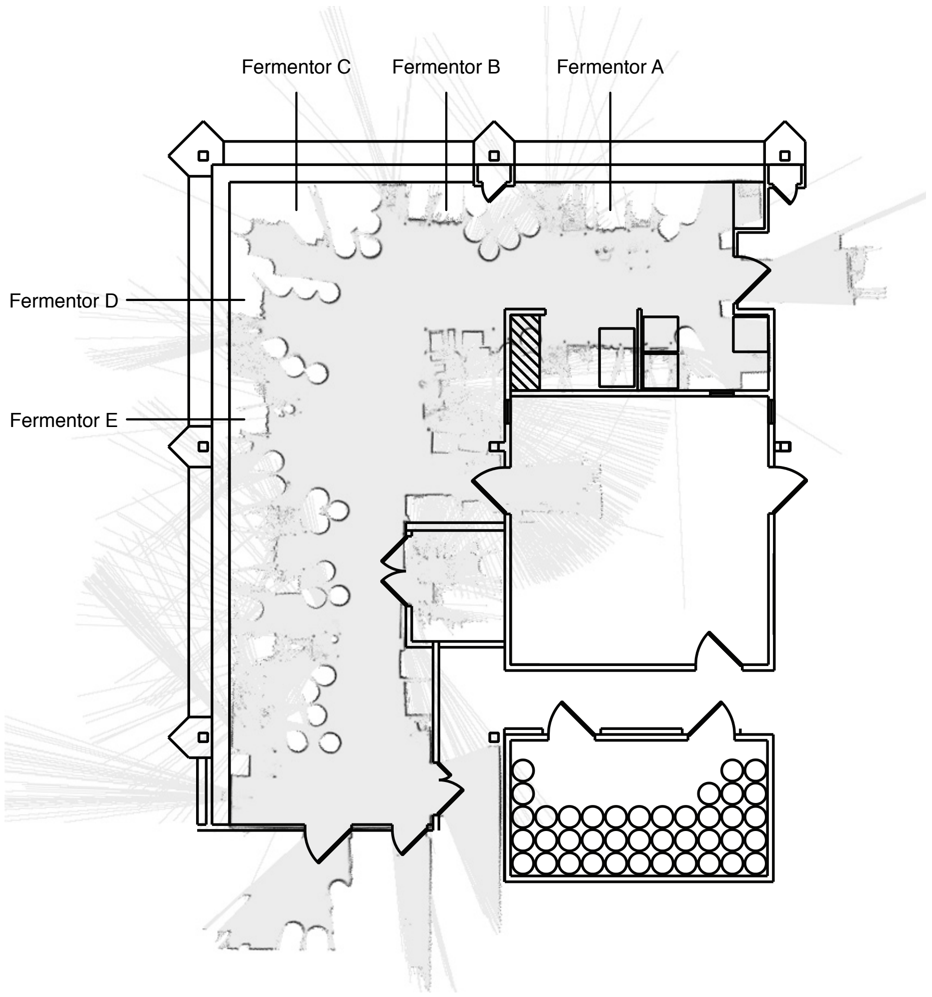


Figure 6.9: Automatically created map of the laboratory with the stations of interest indicated. The real floor plan overlays and shows how precisely the map was generated.

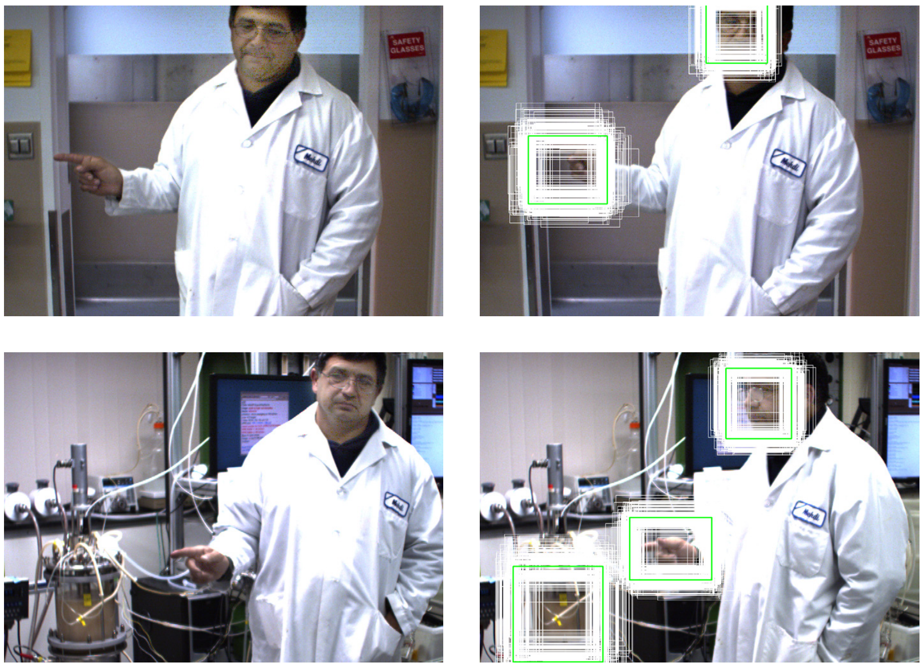


Figure 6.10: In the shown pictures, the scientist's face and hand (upper row) are detected and tracked by the robot until he arrives at a fermentor station (lower row) where the fermentor is detected as well. This way and in combination with the automated mapping and path planning a human guide can teach a lab walkthrough and indicate positions of interest for a repetitive fermentor monitoring routine, without preprogramming any knowledge about the environment.

6.3 Changeable Factory

The F5 platform has been developed to address the Changeable Factory scenario. I implemented the software architecture based on the proposed layer model. Therefore I was able to reuse many software components, that I previously developed for the Leonardo platform.

The developed system was shown at the *Hannover Messe 2009* (Hannover fair 2009) where it was navigating autonomously and docking to a changeable factory made of Festo MPS modules. Figure 6.11 shows a changeable factory and figures 6.12 and 6.13 show the F5 robot at the Hannover Messe 2009. In the meantime the platform has been shipped to its final destination: the IFF at the university of Stuttgart.

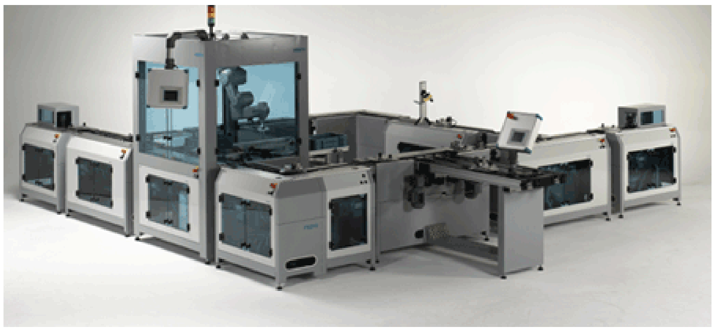


Figure 6.11: Changeable Factory made of Festo MPS modules. Photograph: Festo Didactic.

6.4 Housekeeping

The F5-S platform is the successor of the Leonardo platform and is intended for the Housekeeping as well as future Lab Automation scenarios. Despite the different actuators and manipulator compared to the F5 platform, the fine grained software architecture, which I developed according to the proposed layer model from section 4.2 allowed to reuse almost the entire code base of the larger F5 platform.

The first F5-S was shipped to the Technische Universität Berlin in 2009 and successfully installed at the demo apartment of the DAI Labor.

6.5 TV Studio

RoboKam is a well established product for Virtual TV production studios. Several RoboKams have been installed at ZDF's most recent news studio N1 in Mainz and at RTL's n-tv news center in Cologne, Germany. While most of the software components have been developed by RTLeaders, our visual tracking framework *OpenTL* became an essential key component for autonomous camera moves during live productions [15].

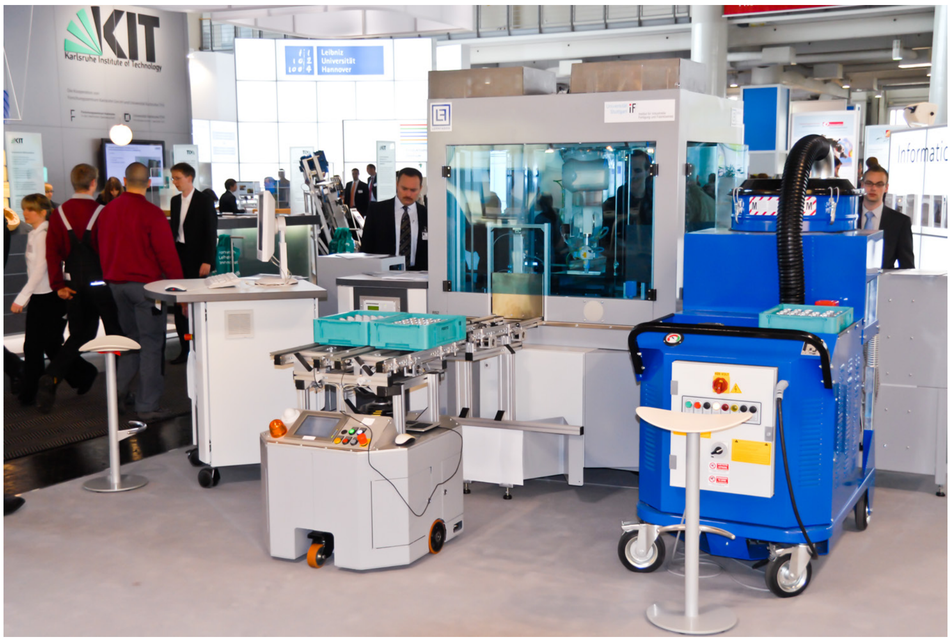

KIT
Karlsruhe Institute of Technology
Leibniz Universität Hannover

Figure 6.12: Closeup of the F5 robot in front of a changeable factory.

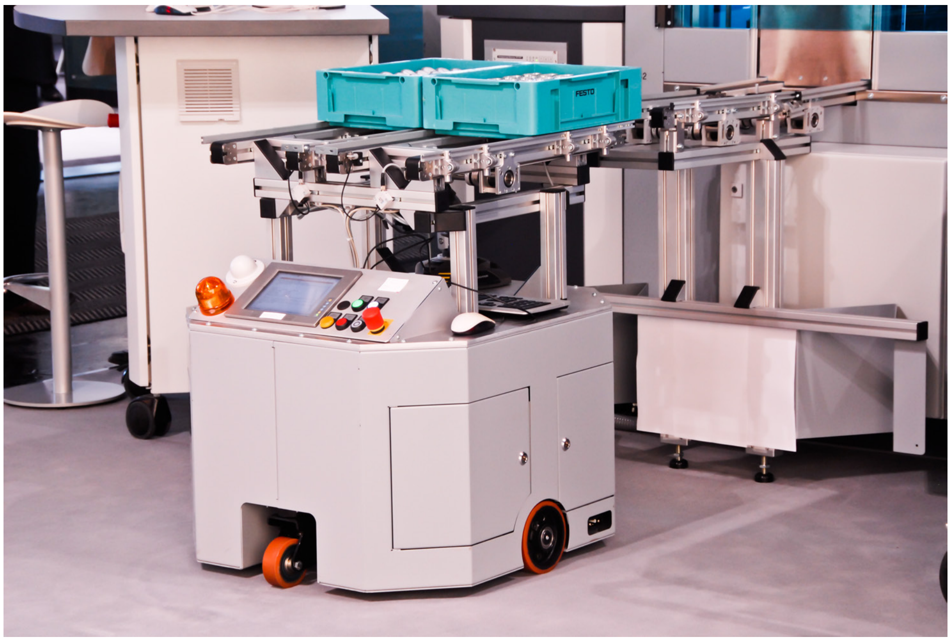


Figure 6.13: F5 docking to a changeable factory at the *Hannover Messe 2009*.

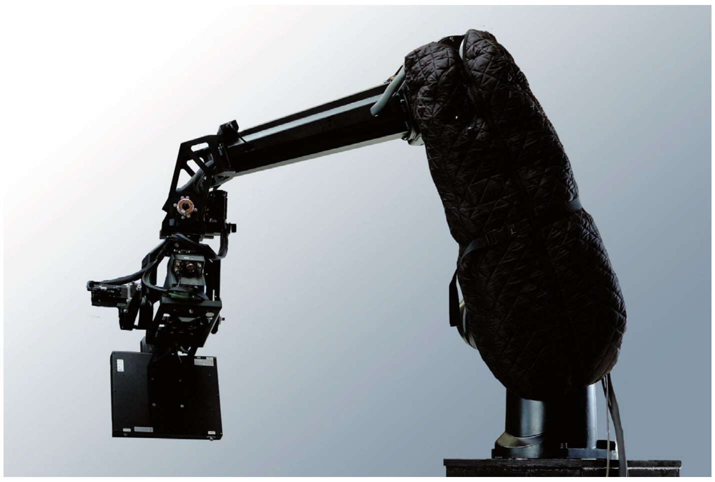


Figure 6.14: RoboKam. Photographs: RTLeaders.

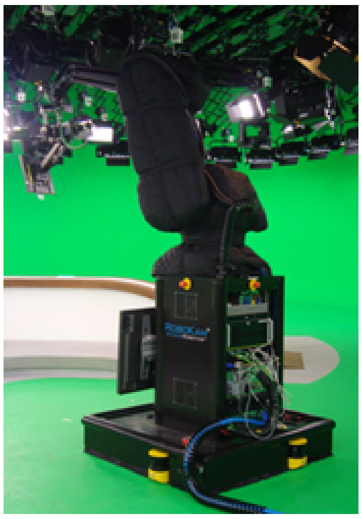


Figure 6.15: RoboKam in a TV Studio.

6.6 Robot Education

Essential parts for the control, localisation and navigation as well as for the Robotino simulator have been developed at our department for Robotics and Embedded Systems [7]. In the meantime Robotino has been sold over 1000 times world wide and became a great and affordable product for robot education purposes.

6.7 Summary

This chapter presented how we approached the different application scenarios for mobile service robots with the various robot platforms that were developed in collaboration with strong industrial partners. The Lab Automation scenarios as well as Surveillance and Housekeeping with a versatile personal assistant were addressed with the Leonardo and the F5-S mobile robots. The larger platform F5 with a different actuation setup was utilised for the Changeable Factory. TV Studio productions were implemented with RoboKam and Robot Education has been addressed with Robotino. The Software architectures presented in chapter 4 have been implemented according to the proposed classification model from section 4.2 hence validated the model's purpose. The number of reusable software components speak for themselves.

Chapter 7

Conclusion

In this thesis I presented a model to classify and describe the hard- and software-architecture of mobile service robots and to ease the development of their components. A total of seventeen authored and five co-authored publications have emerged up to now addressing individual aspects of this thesis. I applied the proposed model to several service robot platforms developed at our department. It is validated by the utilisation of the created software components on a variety of robots in real world applications. My work on the Biotechnology robot has been recognised by the independent author Harald Zähringer and appeared in an article in the 4/2009 issue of *Lab Times*, a well-known european biotechnology journal [109]. I addressed certain solutions in service robotics, including automated mapping and localisation as well as visual tracking to improve the perception and usability of current mobile service robots.

After several years of independent work on mobile service robots by individuals or groups, standardisation efforts for robotic hard- and software components become noticeable in projects like Robot Operating System (ROS) and Open Architecture Humanoid Robotics Platform (OpenHRP) to overcome their big variety and complexity and to avoid reinventing the wheel over and over again. Ultimately the development of service robots will benefit from standardised software components and interfaces, since only reusable software architectures will sustain.

Automatic lawn mowers and vacuum cleaners are just the beginning of the growing field of service robotics. To make classical robots – as we know them from industrial automation – smart, they need to be equipped with sensors to become aware of their environment and sufficient computing power to be able to process and interpret continuously the acquired sensor data. To make and keep service robots versatile, their software architecture should be modular and scalable. Classes of modules need well specified interfaces to make components exchangeable; dependencies between components should be kept to a minimum. As more and more research teams are collaborating on standardisation efforts and exchangeable soft- and hardware components, we may see more and more mobile service robots entering our homes.

List of Figures

List of Tables

List of Publications

[1] Martin Wojtczyk, Rüdiger Heidemann, Chetan Goudar, and Alois Knoll. Parallel computing for autonomous mobile robots. In *Intel Developer Forum 2010*, San Francisco, USA, September 2010.

[2] Martin Wojtczyk, Rüdiger Heidemann, Chetan Goudar, and Alois Knoll. Monitoring bioreactors in a cell culture development laboratory with an autonomous mobile manipulator. In *Cell Culture Engineering XII*, Banff, Alberta, Canada, April 2010.

[3] Martin Wojtczyk, Mark A. Nanny, and Chetan T. Goudar. Aquatic chemistry on the iphone: Activity coefficient, ionization fraction and equilibrium constant determination. In *239th American Chemical Society National Meeting & Exposition*, San Francisco, USA, March 2010.

[4] Martin Wojtczyk, Rüdiger Heidemann, Chetan Goudar, and Alois Knoll. Bioreactor monitoring in a cell culture development laboratory with an autonomous mobile manipulator. In *239th American Chemical Society National Meeting & Exposition*, San Francisco, USA, March 2010.

[5] Martin Wojtczyk, Giorgio Panin, Thorsten Röder, Claus Lenz, Suraj Nair, Rüdiger Heidemann, Chetan Goudar, and Alois Knoll. Teaching and implementing autonomous robotic lab walkthroughs in a biotech laboratory through model-based visual tracking. In *IS&T / SPIE Electronic Imaging, Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques: Autonomous Robotic Systems and Applications*, San Jose, CA, USA, January 2010.

[6] Martin Wojtczyk, Rüdiger Heidemann, Chetan Goudar, and Alois Knoll. Task and data parallelism in robotics for autonomous mobile manipulators. In *Intel Developer Forum 2009*, San Francisco, USA, September 2009.

[7] Michael Geisinger, Simon Barner, Martin Wojtczyk, and Alois Knoll. A software architecture for model-based programming of robot systems. In Torsten Kröger and Friedrich M. Wahl, editors, *Advances in Robotics Research – Theory, Implementation, Application*, pages 135–146, Braunschweig, Germany, June 2009. Springer-Verlag Berlin Heidelberg.

[8] Claus Lenz, Giorgio Panin, Thorsten Röder, Martin Wojtczyk, and Alois Knoll. Hardware-assisted multiple object tracking for human-robot-interaction. In François Michaud, Matthias Scheutz, Pamela Hinds, and Brian Scassellati, editors, *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 283–284, La Jolla, CA, USA, March 2009. ACM.

[9] Martin Wojtczyk and Alois Knoll. Utilization of a mobile manipulator for automating the complete sample management in a biotech laboratory. a real world application for service robotics. In *Proceedings of The Sixth International Symposium on Mechatronics & its Applications ISMA09*, American University of Sharjah, Sharjah, United Arab Emirates, March 2009. IEEE.

[10] Martin Wojtczyk, Giorgio Panin, Claus Lenz, Thorsten Röder, Suraj Nair, Erwin Roth, Rüdiger Heidemann, Klaus Joeris, Chun Zhang, Mark Burnett, Tom Monica, and Alois Knoll. A vision based human robot interface for robotic walkthroughs in a biotech laboratory. In François Michaud, Matthias Scheutz, Pamela Hinds, and Brian Scassellati, editors, *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human Robot Interaction*, pages 309–310, La Jolla, CA, USA, March 2009. ACM.

[11] Martin Wojtczyk, Rüdiger Heidemann, Klaus Joeris, Chun Zhang, Mark Burnett, Tom Monica, and Alois Knoll. Automating the complete sample management in a biotech laboratory with a mobile manipulator. a real world application for service robotics. In Leon Trilling, D. Perkins, Dionysios (Dion) D. Dionysiou, Leonid Perlovsky, Kent Davey, David Landgrebe, Miguel A. Marino, D. L. Russell, Steven H. Collicott, Marco Ceccarelli, and John W. Lund, editors, *Proceedings of the 8th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION*, pages 347–356, Cambridge, UK, 2009. World Scientific and Engineering Academy and Society, WSEAS Press.

[12] Martin Wojtczyk and Alois Knoll. Real world applications of service robots and safety issues. In *International Expert Days Service Robotics*, Hausen, Germany, February 2009. Schunk.

[13] Giorgio Panin, Erwin Roth, Thorsten Röder, Suraj Nair, Claus Lenz, Martin Wojtczyk, Thomas Friedlhuber, and Alois Knoll. ITrackU: An integrated framework for image-based tracking and understanding. In *Proceedings of the International Workshop on Cognition for Technical Systems*, Munich, Germany, October 2008.

[14] Martin Wojtczyk and Alois Knoll. A cross platform development workflow for C/C++ applications. In Herwig Mannaert, Tadashi Ohta, Cosmin Dini, and Robert Pellerin, editors, *Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on*, pages 224–229, Sliema, Malta, October 2008. IEEE Computer Society.

[15] Suraj Nair, Giorgio Panin, Martin Wojtczyk, Claus Lenz, Thomas Friedelhuber, and Alois Knoll. A multi-camera person tracking system for robotic applications in virtual reality tv studio. In

Proceedings of the 17th IEEE/RSJ International Conference on Intelligent Robots and Systems 2008. IEEE, September 2008.

[16] Martin Wojtczyk, Michael Marszalek, Rüdiger Heidemann, Klaus Joeris, Chun Zhang, Mark Burnett, Tom Monica, and Alois Knoll. Automation of the complete sample management in a biotech laboratory. In Samson Abramsky, Erol Gelenbe, and Vladimiro Sassone, editors, *Proceedings of Visions of Computer Science, BCS International Academic Conference*, pages 87–97, Imperial College, London, UK, September 2008. The British Computer Society.

[17] Martin Wojtczyk, Simon Barner, Michael Geisinger, and Alois Knoll. Rapid Prototyping of an Adaptive Light-source for Mobile Manipulators with EasyKit and EasyLab. In *SPIE Optics and Photonics 2008, Illumination Engineering, Eighth International Conference on Solid State Lighting: Applications*, San Diego, CA, USA, August 2008.

[18] Martin Wojtczyk, Kushal Abhyankar, Suraj Nair, and Alois Knoll. A computer vision based approach for cell tracking to increase throughput in visual drug discovery. In *Natural Product Discovery and Production II*, Whistler B.C., Canada, June 2008.

[19] Giorgio Panin, Claus Lenz, Martin Wojtczyk, Suraj Nair, Erwin Roth, Thomas Friedelhuber, and Alois Knoll. A unifying software architecture for model-based visual tracking. In *Image Processing: Machine Vision Applications. Edited by Niel, Kurt S.; Fofi, David. Proceedings of the SPIE, Volume 6813, pp. 681303-681303-14 (2008).*, volume 6813 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, March 2008.

[20] Martin Wojtczyk and Alois Knoll. Parallelisierung der automatischen Zellzählung von In-situ-Mikroskopen mittels GRID-Technologie. In *21. PARS-Workshop*, Hamburg, Germany, May 2007.

[21] Martin Wojtczyk and Rüdiger Heidemann. Status report: Installation and modification of a mobile robot platform to perform a complete automated sample management process. Bayer internal report, Bayer HealthCare Pharmaceuticals, May 2006.

[22] Martin Wojtczyk, Rüdiger Heidemann, Klaus Joeris, Chun Zhang, Mark Burnett, Alois Knoll, and Konstantin Konstantinov. The Use of a Mobile Robot for Complete Sample Management in a Cell Culture Pilot Plant. In Rodney Smith, editor, *Cell Technology for Cell Products, Proceedings of the 19th ESACT Meeting, Harrogate, UK, June 5-8, 2005*, pages 543–547. Springer Netherlands, 2007.

Bibliography

[23] *IFF Jahresbericht*. Institut für industrielle Fertigung und Fabrikbetrieb der Universität Stuttgart, 2006.

[24] Egon Müller. Vernetzt planen und produzieren VPP 2006. Tagungsband, Technische Universität Chemnitz, Chemnitz, September 2006.

[25] Bill Gates. A Robot in Every Home. *Scientific American Reports*, 18(1):4–11, February 2008.

[26] International Standards Organization. *ISO 10218-1:2006 Robots for industrial environments – Safety requirements – Part 1: Robot*.

[27] Dirk Lütkemeyer, Iris Poggendorf, Torsten Scherer, Jianwei Zhang, Alois Knoll, and Jürgen Lehmann. First steps in robot automation of sampling and sample management during cultivation of mammalian cells in pilot scale. *Biotechnology Progress*, 16(5):822–828, 2000.

[28] Torsten Scherer, Jianwei Zhang, Alois Knoll, Iris Poggendorf, Dirk Lütkemeyer, and Jürgen Lehmann. Roboterisierung des Probenmanagements in der Biotechnologie. In *Proc. of GVC/DECHEMA Tagung Prozessmesstechnik in der Biotechnologie*, Bamberg, Germany, 2000.

[29] Alois Knoll, Torsten Scherer, Iris Poggendorf, Dirk Lütkemeyer, and Jürgen Lehmann. Flexible Automation of Cell Culture and Tissue Engineering Tasks. *Biotechnology Progress*, 20(6):1825–1835, 2004.

[30] Torsten Scherer. *A Mobile Service Robot for Automisation of Sample Taking and Sample Management in a Biotechnological Pilot Laboratory*. Dissertation, University of Bielefeld, October 2004.

[31] Erik Schulenburg, Norbert Elkmann, Markus Fritzsche, Angelika Girstl, Stefan Stiene, and Christian Teutsch. Lisa: A robot assistant for life sciences. 2007.

[32] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, May 2008.

[33] Edward Yourdon and Larry L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and System Design.* Prentice Hall, 1979.

[34] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, Reading, 19. edition, 2000.

[35] Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering*. Prentice Hall, Upper Saddle River, 2000.

[36] Michele Marchesi, Giancarlo Succi, Don Wells, Laurie Williams, and James Donovan Wells. *Extreme Programming Perspectives*. Pearson Education, 1st edition, August 2002.

[37] Axel Schneider and Daniel Westhoff. Autonomous Navigation and Control of a Mobile Robot in a Cell Culture Laboratory. Diplomarbeit, University of Bielefeld, June 2002.

[38] Ralph-Christoph Weber. *Robotino Manual*. Festo Didactic GmbH & Co. KG, Denkendorf, Germany, 2007.

[39] Ulrich Karras. Robotino – an open learning mobile robot system for robocup. April 2009.

[40] P. Wilson. *Computer Supported Cooperative Work: An Introduction.* Kluwer Academic Pub., 1991.

[41] *CVS – Concurrent Versions System*. http://www.nongnu.org/cvs/.

[42] *Apache Subversion*. http://subversion.apache.org/.

[43] *Git – Fast Version Control System*. http://git-scm.com/.

[44] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly Media, 2008.

[45] American National Standards Institute. *ANSI X3.159-1989 "Programming Language C"*.

[46] International Standards Organization. *ISO/IEC International Standard 14882, Programming Languages – C++*.

[47] Silicon Graphics, Inc. *Standard Template Library Programmer's Guide*. http://www.sgi.com/tech/stl/.

[48] Troy Unrau. *The Road to KDE 4: CMake, a New Build System for KDE*, 2007. http://dot.kde.org/1172083974/.

[49] Microsoft Corporation. *Visual Studio Developer Center*. http://msdn.microsoft.com/en-us/vstudio/default.aspx.

[50] Apple Inc. *Xcode*. http://developer.apple.com/tools/xcode/.

[51] Gary V. Vaughan, Ben Elliston, Tom Tromey, and Ian Lance Taylor. *GNU Autoconf, Automake, and Libtool*. Sams Publishing, 2000. http://sources.redhat.com/autobook/.

[52] Free Software Foundation, Inc. *GNU Make*. http://www.gnu.org/software/make/.

[53] The Eclipse Foundation. *Eclipse – an open development platform*. http://www.eclipse.org/.

[54] KDevelop Team. *KDevelop – an Integrated Development Environment*. http://www.kdevelop.org/.

[55] The Code::Blocks Team. *Code::Blocks – The open source, cross platform, free C++ IDE.* http://www.codeblocks.org/.

[56] Kitware Inc. *CMake - Cross-platform Make*. http://www.cmake.org.

[57] The MacPorts Project Team. *The MacPorts Project*. http://www.macports.org/.

[58] The Fink Team. *Fink*. http://www.finkproject.org/.

[59] Nokia. *Qt — A cross-platform application and UI framework*. http://qt.nokia.com/.

[60] Free Software Foundation, Inc. *ncurses*. http://www.gnu.org/software/ncurses/.

[61] Kitware Inc. *CMake documentation*. http://www.cmake.org/cmake/help/documentation.html.

[62] Ken Martin and Bill Hoffman. *Mastering CMake*. Kitware Inc., Clifton Park, New York, 2006.

[63] GTK+ Team. *GTK+*. http://www.gtk.org/.

[64] Julian Smart et al. *wxWidgets*. http://www.wxwidgets.org/.

[65] Bill Spitzak et al. *Fast Light Toolkit (FLTK)*. http://www.fltk.org/.

[66] Kongsberg SIM AS. *Coin3D – 3D Graphics Developer Kit*. http://www.coin3d.org/.

[67] Silicon Graphics, Inc. *Open Inventor*. http://oss.sgi.com/projects/inventor/.

[68] Martin Wojtczyk. *qiew – a minimalistic and portable VRML/Inventor Viewer.* http://www.qiew.org.

[69] Intel Corporation. *OpenCV – Open Source Computer Vision*. http://opencvlibrary.sourceforge.net/.

[70] 1394 Trade Association. *IIDC 1394-based Digital Camera Specification*, February 2004.

[71] Damien Douxchamps. *libdc1394: The API for IEEE1394 / Firewire cameras*. http://damien.douxchamps.net/ieee1394/libdc1394/.

[72] Christopher Baker. *CMU 1394 Digital Camera Driver*. http://www.cs.cmu.edu/~iwan/1394/.

[73] NSIS Team. *Nullsoft Scriptable Install System (NSIS)*. http://nsis.sourceforge.net/.

[74] Jörg Schilling. *smake*. http://ftp://ftp.berlios.de/pub/smake/alpha/smake-1.2a23.tar.gz.

[75] *CONS: A Make replacement*. http://www.baldmt.com/cons/.

[76] *SCons: A software construction tool*. http://www.scons.org/.

[77] Apache Software Foundation. *Apache Ant*. http://ant.apache.org/.

[78] Perforce Software. *Jam*. http://www.perforce.com/jam/jam.html.

[79] Rene Rivera, David Abrahams, and Vladimir Prus. *Boost.Jam*. http://www.boost.org/doc/html/jam.html.

[80] Siemens AG. *NERLITE Machine Vision Illumination, Machine Vision Lighting and Machine Vision Imaging Products from Siemens*. http://www.nerlite.com.

[81] Siemens AG. *NERLITE R-60-1 "V2" Series Ring*. http://www.nerlite.com/PDFs/r-60-1-v2-series.pdf.

[82] Siemens AG. *NERLITE R-70-2 "V2" Series Ring*. http://www.nerlite.com/PDFs/r-70-2-v2-series.pdf.

[83] Gardasoft Vision. *Lighting Controllers for Machine Vision – Gardasoft*. http://www.gardasoft.com/.

[84] Gardasoft Vision. *PP610 - LED Lighting Controller with RS232 Control*. http://www.gardasoft.com/pdf/610Bro.pdf.

[85] Technische Universität München. *EasyKit: Innovative Entwicklungs- und Didaktikwerkzeuge für mechatronische Systeme*. http://easykit.informatik.tu-muenchen.de/.

[86] Microchip Technology Inc. *PIC18F2520*. http://www.microchip.com.

[87] Verband Deutscher Maschinen- und Anlagenbau e.V. – Arbeitsgemeinschaft Match-X. *VDMA – Einheitsblatt 66305: Bausteine und Schnittstellen der Mikrotechnik*, July 2005. http://www.match-x.org/.

[88] VDMA. *VDMA Homepage*. http://www.vdma.org/.

[89] International Electrotechnical Commission. *Norm EN 61131*, 2003.

[90] Edward Ashford Lee and David G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Trans. Comput.*, 36(1):24–35, 1987.

[91] Shuvra S. Bhattacharyya, Edward A. Lee, and Praveen K. Murthy. *Software Synthesis from Dataflow Graphs*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[92] Shuvra S. Bhattacharyya, Joseph T. Buck, Soonhoi Ha, and Edward A. Lee. Generating compact code from dataflow specifications of multirate signal processing algorithms. pages 452–464, 2002.

[93] Hyunok Oh, Nikil Dutt, and Soonhoi Ha. Memory optimal single appearance schedule with dynamic loop count for synchronous dataflow graphs. In *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific design automation*, pages 497–502, Piscataway, NJ, USA, 2006. IEEE Press.

[94] Robert Neuss. *Usability Engineering als Ansatz zum Multimodalen Mensch-Maschine-Dialog*. PhD thesis, Technische Universität München, 2001. http://tumb1.biblio.tu-muenchen.de/publ/diss/ei/2001/neuss.pdf.

[95] Elke Siemon. *Über den Entwurf von Benutzungsschnittstellen technischer Anwendungen mit visuellen Spezifikationsmethoden und Werkzeugen*. PhD thesis, TU Darmstadt, 2001. http://elib.tu-darmstadt.de/diss/000119/siemon.pdf.

[96] Yongmei Wu. *Design Patterns and Frameworks for Developing WIMP+ User Interfaces*. PhD thesis, TU Darmstadt, 2001. http://elib.tu-darmstadt.de/diss/000189/DissWu-1.pdf, http://elib.tu-darmstadt.de/diss/000189/DissWu-2.pdf.

[97] Leslie Polzer. Gui toolkits for the x window system., 2003. http://freshmeat.net/articles/gui-toolkits-for-the-x-window-system.

[98] Qt in industries. http://qt.nokia.com/qt-in-use/.

[99] Martin Wojtczyk. Entwicklung eines Simulationssystems für Roboter in virtuellen Welten unter Berücksichtigung physikalischer Gesetzmäßigkeiten. Diplomarbeit, Technische Universität München, May 2004.

[100] Markus Rickert, Mary Ellen Foster, Manuel Giuliani, Tomas By, Giorgio Panin, and Alois Knoll. Integrating language, vision and action for human robot dialog systems. In *Proceedings of the International Conference on Universal Access in Human-Computer Interaction, HCI International*, volume 4555 of *Lecture Notes in Computer Science*, pages 987–995, Beijing, China, July 2007. Springer.

[101] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006.

[102] Vincent Lepetit and Pascal Fua. *Monocular Model-based 3d Tracking of Rigid Objects (Foundations and Trends in Computer Graphics and Vision(R))*. Now Publishers Inc, 2005.

[103] Robotics and Embedded Systems, Faculty of Informatics, Technische Universität München. *Open Tracking Library (OpenTL) – A general-purpose tracking library*. http://www.opentl.org/.

[104] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998.

[105] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 661–675, London, UK, 2002. Springer-Verlag.

[106] J. Borenstein, H. R. Everett, and L. Feng. *"Where am I?" Sensors and Methods for Mobile Robot Positioning*. The University of Michigan, March 1996.

[107] Austin I. Eliazar and Ronald Parr. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 03).*, 2003. http://www.cs.duke.edu/~parr/ijcai03.ps.gz.

[108] Michael Lukas Marszalek. Simultane lokalisierung und kartenerstellung in kombination mit verfahren des maschinellen lernens - simultaneous localization and mapping methods in combination with machine learning methods. Diplomarbeit, Technische Universität München, May 2008.

[109] Harald Zähringer. Product survey: Cell culture devices – latest toys for cell cultivators. *Lab Times*, 4:53–56, 2009.

[110] Scott W. Ambler. *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press, 2004.

[111] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[112] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255, 2004.

[113] Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[114] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.

[115] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation.* John Wiley and Sons, Inc., New York, NY, USA, 2002.

[116] Robert Hanek and Michael Beetz. The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *Int. J. Comput. Vision*, 59(3):233–258, 2004.

[117] Y. Benezeth, B. Emile, H. Laurent, and C. Rosenberger. Vision-based system for human detection and tracking in indoor environment. *International Journal of Social Robotics*, 2(1):41–52, March 2010.

[118] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *Int. J. Rob. Res.*, 26(7):661–676, 2007.

[119] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[120] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.

[121] Samuel S. Blackman and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library, 1999.

[122] Andrew Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[123] Gary R. Bradski and James W. Davis. Motion segmentation and pose recognition with motion history gradients. *Mach. Vision Appl.*, 13(3):174–184, 2002.

[124] Carsten G. Bräutigam, Jan-Olof Eklundh, and Henrik I. Christensen. A model-free voting approach for integrating multiple cues. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 734–750, London, UK, 1998. Springer-Verlag.

[125] M. Bray, E. Koller-Meier, P. Mueller, L. Van Gool, and N. N. Schraudolph. 3d hand tracking by rapid stochastic gradient descent using a skinning model. In A. Chambers and A. Hilton, editors, *1st European Conference on Visual Media Production (CVMP)*, pages 59–68. IEEE, March 2004.

[126] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[127] Alexander Camek. Entwicklung einer Roboter-Architektur für mobile, verteilte Systeme und Einführung einer Selbstlokalisation mittles Markov-Modellen. Diplomarbeit, Technische Universität München, May 2006.

[128] SangHyun Chang, Rangoli Sharan, Michael Wolf, Naoki Mitsumoto, and Joel W. Burdick. People tracking with uwb radar using a multiple-hypothesis tracking of clusters (mhtc) method. *International Journal of Social Robotics*, 2(1):3–18, March 2010.

[129] Norman Chin, Chris Frazier, Paul Ho, Zicheng Liu, and Kevin P. Smith. *The OpenGL Graphics System Utility Library (Version 1.3)*. Silicon Graphics, Inc., 1998. http://www.opengl.org/documentation/specs/glu/glu1_3.pdf.

[130] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.

[131] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models—their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59, 1995.

[132] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Lecture Notes in Computer Science*, 1407:484–498, 1998.

[133] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.

[134] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(2):138–150, February 1996.

[135] Daniel DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 335–343, London, UK, 1992. Springer-Verlag.

[136] T. Drummond and R. Cipolla. Visual tracking and control using lie algebras. *cvpr*, 02:2652, 1999.

[137] Tom Drummond and Roberto Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 20–36, London, UK, 2000. Springer-Verlag.

[138] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):932–946, 2002.

[139] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[140] Austin I. Eliazar and Ronald Parr. DP-SLAM 2.0. In *IEEE 2004 International Conference on Robotics and Automation (ICRA 2004)*., 2004. http://www.cs.duke.edu/~parr/dpslam2.pdf.

[141] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[142] Amalia F. Foka and Panos E. Trahanias. Probabilistic autonomous robot navigation in dynamic environments with human motion prediction. *International Journal of Social Robotics*, 2(1):79–94, March 2010.

[143] Thomas Friedlhuber. Effiziente Generierung optimaler Roboterbahnen in 3D modellierten Umgebungen unter Berücksichtigung von verschiedenen Nebenbedingungen. Diplomarbeit, Technische Universität München, June 2007.

[144] Simone Frintrop, Achim Königs, Frank Hoeller, and Dirk Schulz. A component-based approach to visual person tracking from a mobile platform. *International Journal of Social Robotics*, 2(1):53–62, March 2010.

[145] Christian Frueh and Avideh Zakhor. 3d model generation for cities using aerial photographs and ground level laser scans. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2.2, pages II–31–38, Kauai, USA, 2001. IEEE.

[146] Christian Frueh and Avideh Zakhor. Reconstructing 3d city models by merging ground-based and airborne views. *Lecture Notes in Computer Science*, 2849(1):306–313, 2003.

[147] Christian Frueh, Siddharth Jain, and Avideh Zakhor. Processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision*, 61(2):159–184, February 2005.

[148] Bill Gates. A Robot in Every Home. *Scientific American*, 296(1):58–65, January 2007.

[149] W. Wayt Gibbs. A New Race of Robots. *Scientific American*, 290(1):58–67, March 2004.

[150] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[151] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[152] D. O. Gorodnichy, S. Malik, and G. Roth. Affordable 3d face tracking using projective vision. In *International Conference on Vision Interfaces*, pages 383–390, 2002.

[153] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[154] Gregory D. Hager and Kentaro Toyama. X vision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding: CVIU*, 69(1):023–037, 1998.

[155] David L. Hall and James Llinas. *Handbook of Multisensor Data Fusion*. CRC Press, June 2001.

[156] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Fast image-based object localization in natural scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, Lausanne, pages 116–122, 2002.

[157] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: A real time system for detecting and tracking people. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 962, Washington, DC, USA, 1998. IEEE Computer Society.

[158] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[159] Chris Harris. Tracking with rigid models. pages 59–73, 1993.

[160] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2. edition, 2003.

[161] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[162] Michael Isard and Andrew Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 893–908, London, UK, 1998. Springer-Verlag.

[163] Michael Isard and John MacCormick. Bramble: A bayesian multiple-blob tracker. In *ICCV*, pages 34–41, 2001.

[164] Tobias Jahn. Inbetriebnahme des Neobotix MobileArm unter Berücksichtigung lokaler Umgebungsbedingungen und Entwicklung einer universellen Programmierschnittstelle. Diplomarbeit, Technische Universität München, January 2004.

[165] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[166] Boyoon Jung and Gaurav S. Sukhatme. Real-time motion tracking from a mobile robot. *International Journal of Social Robotics*, 2(1):63–78, March 2010.

[167] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[168] John Kessenich. *The OpenGL Shading Language - Language Version: 1.20*. 3Dlabs, Inc. Ltd., 2006. http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.20.8.pdf.

[169] Zia Khan, Tucker Balch, and Frank Dellaert. Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model. Las Vegas, 2003.

[170] Zia Khan. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(11):1805–1918, 2005. Member-Tucker Balch and Member-Frank Dellaert.

[171] Mark J. Kilgard. *The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3*. Silicon Graphics, Inc., 1996. http://www.opengl.org/documentation/specs/glut/glut-3.spec.pdf.

[172] N. Kiryati, Y. Eldar, and A. M. Bruckstein. A probabilistic hough transform. *Pattern Recogn.*, 24(4):303–316, 1991.

[173] Marco La Cascia, Stan Sclaroff, and Vassilis Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):322–336, 2000.

[174] Boris Lau, Kai O. Arras, and Wolfram Burgard. Multi-model hypothesis group tracking and group size estimation. *International Journal of Social Robotics*, 2(1):19–30, March 2010.

[175] Joon-Woo Lee and Jeong-Jung Kim. Improved ant colony optimization algorithm by path crossover for optimal path planning. *IEEE International Symposium on Industrial Electronics (ISIE 2009)*, July 2009.

[176] Silicon Graphics, Inc. *OpenGL Graphics with the X Window System (Version 1.4)*, 2005. http://www.opengl.org/documentation/specs/glx/glx1.4.pdf.

[177] Claus Lenz, Giorgio Panin, and Alois Knoll. A GPU-accelerated particle filter with pixel-level likelihood. In *International Workshop on Vision, Modeling and Visualization (VMV)*, Konstanz, Germany, October 2008.

[178] J. P. Lewis. Fast normalized cross-correlation. In *Vision Interface*, pages 120–123. Canadian Image Processing and Pattern Recognition Society, 1995.

[179] Tony Lindeberg. *Scale-Space Theory In Computer Vision*. Kluwer Academic Publishers, 1994.

[180] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):441–450, 1991.

[181] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[182] Le Lu, Xiang-Tian Dai, and Gregory Hager. A particle filter without dynamics for robust 3d face tracking. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 5*, page 70, Washington, DC, USA, 2004. IEEE Computer Society.

[183] Dirk Lütkemeyer, Iris Poggendorf, Torsten Scherer, Jianwei Zhang, Alois Knoll, and Jürgen Lehmann. Robot automation of sampling and sample management during cultivation of mammalian cells on pilot scale. In *Proc. UEF Cell Culture Engineering VII*, Santa Fe, NM, 2000.

[184] Dirk Lütkemeyer, Iris Poggendorf, Torsten Scherer, Jianwei Zhang, Alois Knoll, and Jürgen Lehmann. Let's start moving – a mobile robot has the potential of complete automation in monitoring mammalian cell cultures. In *Proc. UEF Cell Culture Engineering VIII*, Snowmass, CO, USA and GVC/Dechema Jahrestagung, Wiesbaden, Germany, 2002.

[185] John MacCormick and Michael Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 3–19, London, UK, 2000. Springer-Verlag.

[186] Frederik Maes, André Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *IEEE Trans. Med. Imaging*, 16(2):187–198, 1997.

[187] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. Cg: A system for programming graphics hardware in a c-like language. *ACM Transactions on Graphics*, 22(3):896–907, July 2003.

[188] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[189] Iain Matthews and Simon Baker. Active appearance models revisited. Technical Report CMU-RI-TR-03-02, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2003.

[190] R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16(3):233–246, 1989.

[191] Hans Moravec. Rise of the Robots. *Scientific American*, 281(1):124–135, December 1999.

[192] Hans Moravec. Rise of the Robots. *Scientific American Reports*, 18(1):12–19, February 2008.

[193] Oscar Martinez Mozos, Ryo Kurazume, and Tsutomu Hasegawa. Multi-part people detection using 2d range data. *International Journal of Social Robotics*, 2(1):31–40, March 2010.

[194] Richard M. Murray, Zexiang Li, and Shankar S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC, March 1994.

[195] Jackie Neider and Tom Davis. *The Official Guide to Learning OpenGL, Version 1.1*. Addison Wesley, 2. edition, 1997.

[196] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.

[197] Richard A. Newcombe and Andrew J. Davison. Live dense reconstruction with a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, USA, June 2010. IEEE.

[198] Katja Nummiaro, Esther Koller-Meier, and Luc J. Van Gool. An adaptive color-based particle filter. *Image Vision Comput.*, 21(1):99–110, 2003.

[199] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, pages 21–51, August 2005.

[200] Anand Panangadan, Maja Matarić, and Gaurav S. Sukhatme. Tracking and modeling of human activity using laser rangefinders. *International Journal of Social Robotics*, 2(1):95–107, March 2010.

[201] Giorgio Panin, Alexandor Ladikos, and Alois Knoll. An efficient and robust real-time contour tracking system. In *Proc. IEEE International Conference on Visual Systems (ICVS)*, page 44, January 2006.

[202] Giorgio Panin and Alois Knoll. Mutual information-based 3d object tracking. *International Journal of Computer Vision*, 78(1):107–118, January 2008.

[203] Giorgio Panin and Alois Knoll. Real-time 3d face tracking with mutual information and active contours. In *International Symposium on Visual Computing (ISVC)*, Lake Tahoe, Nevada, USA, December 2007.

[204] Giorgio Panin, Erwin Roth, and Alois Knoll. Robust contour-based object tracking integrating color and edge likelihoods. In *International Workshop on Vision, Modeling and Visualization (VMV)*, Konstanz, Germany, October 2008.

[205] Giorgio Panin, Thorsten Röder, and Alois Knoll. Integrating robust likelihoods with monte-carlo filters for multi-target tracking. In *International Workshop on Vision, Modeling and Visualization (VMV)*, Konstanz, Germany, October 2008.

[206] In Kyu Park, Hui Zhang, Vladimir Vezhnevets, and Heui-Keun Choh. Image-based photorealistic 3-d face modeling. In *FGR*, pages 49–56, 2004.

[207] M. S. Paterson and F. F. Yao. Binary partitions with applications to hidden surface removal and solid modelling. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 23–32, New York, NY, USA, 1989. ACM Press.

[208] Edward Pervin and Jon A. Webb. Quaternions for computer vision and robotics. *Computer Vision and Pattern Recognition*, pages 382–383, 1983.

[209] John Peterson, Paul Hudak, Alastair Reid, and Gregory D. Hager. Fvision: A declarative language for visual tracking. In *PADL '01: Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages*, pages 304–321, London, UK, 2001. Springer-Verlag.

[210] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual-information-based registration of medical images: a survey. *Medical Imaging, IEEE Transactions on*, 22(8):986–1004, 2003.

[211] Iris Poggendorf, Dirk Lütkemeyer, Torsten Scherer, Jianwei Zhang, Alois Knoll, and Jürgen Lehmann. Using an industrial robot arm for monitoring cultivations of mammalian cells in pilot scale. In *Computer Applications in Biotechnology 2001: modelling, monitoring and control of biotechnological processes, Proc. 8th IFAC international conference*, Quebec City, Canada, June 2001.

[212] Iris Poggendorf. *Einsatz eines Serviceroboters zur Automatisierung der Probenentnahme und des Probenmangements während Kultivierungen tierischer Zellen in einer Technikumsumgebung*. Dissertation, Universität Bielefeld, June 2004.

[213] J. Principe, D. Xu, and J. Fisher. Information theoretic learning. In Simon Haykin, editor, *Unsupervised Adaptive Filtering*. John Wiley & Sons, New York, 2000.

[214] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

[215] David Reynard, Andrew Wildenberg, Andrew Blake, and John A. Marchant. Learning dynamics of complex motions from image sequences. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 357–368, London, UK, 1996. Springer-Verlag.

[216] Markus Rickert. Entwicklung eines Systems zur Bahnplanung für mobile Roboter. Diplomarbeit, Technische Universität München, June 2004.

[217] Stefan Riesner. *RoboKam*. Robotics Technology Leaders GmbH, München, Germany, 2007. http://www.studiorobotics.com/mediapool/56/560506/data/RoboKamNAB.pdf.

[218] Randi J. Rost. *OpenGL Shading Language*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[219] Randi J. Rost. *OpenGL Shading Language*. Addison-Wesley Professional, 2nd edition, January 2006.

[220] Stefan Roth, Leonid Sigal, and Michael J. Black. Gibbs likelihoods for bayesian tracking. *cvpr*, 01:886–893, 2004.

[221] Erwin Roth, Giorgio Panin, and Alois Knoll. Sampling feature points for contour tracking with graphics hardware. In *International Workshop on Vision, Modeling and Visualization (VMV)*, Konstanz, Germany, October 2008.

[222] Multiple UAVs path planning algorithms: a comparative study. B. moses sathyaraj and l. c. jain and a. finn and s. drake. *Fuzzy Optimization and Decision Making*, 7(3):257–267, September 2008.

[223] Torsten Scherer, Iris Poggendorf, Axel Schneider, Daniel Westhoff, Jianwei Zhang, Dirk Lütkemeyer, Jürgen Lehmann, and Alois Knoll. A service robot for automating the sample management in biotechnological cell cultivations. In *Proc. 9th IEEE Conf. on Emerging Technologies and Factory Automation, Vol. 2*, pages 383–390, Lisbon, Portugal, 2003. IEEE.

[224] Torsten Scherer, Iris Poggendorf, Axel Schneider, Daniel Westhoff, Jianwei Zhang, Dirk Lütkemeyer, Jürgen Lehmann, and Alois Knoll. A service robot for automating the sample management in biotechnological cell cultivations. In *Emerging Technologies and Factory Automation, 2003. Proceedings. (ETFA '03)*, volume 2, pages 383–390. IEEE, September 2003.

[225] Mark Segal and Kurt Akeley. *The OpenGL Graphics System: A Specification (Version 2.1)*. Silicon Graphics, Inc., 2006. http://www.opengl.org/documentation/specs/version2.1/glspec21.pdf.

[226] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[227] Nils T. Siebel and Stephen J. Maybank. Fusion of multiple tracking algorithms for robust people tracking. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 373–387, London, UK, 2002. Springer-Verlag.

[228] Hans-Jürgen Siegert and Siegfried Bocionek. *Robotik: Programmierung intelligenter Roboter*. Springer-Verlag, Berlin, 1996.

[229] G. Silveira and E. Malis. Real-time visual tracking under arbitrary illumination changes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, USA, June 2007.

[230] Iryna Skrypnyk and David G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR '04: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 110–119, Washington, DC, USA, 2004. IEEE Computer Society.

[231] Ross Cunniff, Matt Craighead, Daniel Ginsburg, Kevin Lefebvre, Bill Licea-Kane, and Nick Triantos. ARB_occlusion_query (OpenGL Extension Registry).

[232] Pat Brown and Michael Gold. EXT_texture_integer (OpenGL Extension Registry).

[233] L. D. Stone, T. L. Corwin, and C. A. Barlow. *Bayesian Multiple Target Tracking*. 1st. Artech House, Inc., 1999.

[234] P. Thevenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, July 2000.

[235] P. Thevenaz and M. Unser. Optimization of mutual information for multiresolution image registration. *IEEE Transactions on Image Processing*, 9(12):2083–2099, 2000.

[236] K. Toyama. Look, ma — no hands!' hands-free cursor control with real-time 3d face tracking. In *Proc. Workshop on Perceptual User Interfaces (PUI'98)*, pages 49–54, San Francisco, 1998.

[237] Kentaro Toyama and Gregory D. Hager. Incremental focus of attention for robust vision-based tracking. *Int. J. Comput. Vision*, 35(1):45–63, 1999.

[238] A. Treptow, G. Cielniak, and T. Duckett. Active people recognition using thermal and grey images on a mobile security robot. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, 2005.

[239] Jochen Triesch and Christoph Von Der Malsburg. Democratic integration: Self-organized integration of adaptive cues. *Neural Comput.*, 13(9):2049–2074, 2001.

[240] M. Unser, A. Aldroubi, and M. Eden. B-Spline signal processing: Part I - Theory. *IEEE Trans. Signal Process.*, 41(2):821–833, 1993.

[241] M. Unser, A. Aldroubi, and M. Eden. B-Spline signal processing: Part II—Efficient design and applications. *IEEE Transactions on Signal Processing*, 41(2):834–848, 1993.

[242] Michael Unser, Akram Aldroubi, and Murray Eden. The L_2-polynomial spline pyramid. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):364–379, April 1993.

[243] Michael Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999. IEEE Signal Processing Society's 2000 magazine award.

[244] Microsoft Corporation. *nmake Reference*. http://http://msdn.microsoft.com/en-us/library/dd9y37ha(VS.71).aspx.

[245] Luca Vacchetti and Vincent Lepetit. Stable real-time 3d tracking using online and offline information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1385–1391, 2004.

[246] Ngo Anh Vien, Nguyen Hoang Viet, SeungGwan Lee, and TaeChoong Chung. Obstacle avoidance path planning for mobile robot based on ant-q reinforcement learning algorithm. *Lecture Notes in Computer Science, Advances in Neural Networks – ISNN 2007, Part I*, pages 704–713, 2007.

[247] Nguyen Hoang Viet, Ngo Anh Vien, SeungGwan Lee, and TaeChoong Chung. Obstacle avoidance path planning for mobile robot based on multi colony ant algorithm. *First International Conference on Advances in Computer-Human Interaction*, 2008.

[248] Paul A. Viola. *Alignment by Maximization of Mutual Information*. PhD thesis, Massachusetts Institute of Technology – Artificial Intelligence Laboratory, June 1995.

[249] Paul Viola and III William M. Wells. Alignment by maximization of mutual information. *Int. J. Comput. Vision*, 24(2):137–154, 1997.

[250] Paul A. Viola and Michael J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001.

[251] Thomas Weich. Die Kamera Festo SBOx. Diplomarbeit, Technische Universität München, September 2007.

[252] Wolfgang Wein, Barbara Roeper, and Nassir Navab. 2d/3d registration based on volume gradients. In *SPIE Medical Imaging 2005: Image Processing*, volume 5747, pages 144–150, Washington, DC, USA, 2005. J. Michael Fitzpatrick, Joseph M. Reinhardt; Eds.

[253] Greg Welch and Gary Bishop. Scaat: incremental tracking with incomplete information. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 333–344, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[254] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, 2004.

[255] W. Wells, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis. Multi-modal volume registration by maximization of mutual information. *Medical Image Analysis*, 1(1):35–51, 1996.

[256] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[257] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2d+3d active appearance models. In *CVPR (2)*, pages 535–542, 2004.

[258] James Joseph Clark and Alan L. Yuille. *Data Fusion for Sensory Information Processing Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1990.

[259] X. Yun, M. Lizarraga, E. R. Bachmann, and R. B. McGhee. An improved quaternion-based kalman filter for real-time tracking of rigid body orientation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 03), Las Vegas, Nevada*, pages 1074–1079, 2003.

[260] X. Yun, C. Aparicio, E. R. Bachmann, and R. B. McGhee. Implementation and experimental results of a quaternion-based kalman filter for human body motion tracking. In *IEEE International Conference on Robotics and Automation, ICRA 2005, Barcelona, Spain*, pages 318–322, 2005.

[261] Hans-Joachim Bungartz, Michael Griebel, and Christoph Zenger. *Einführung in die Computergraphik*. Vieweg Verlag, Braunschweig, 1996.

[262] Nanning Zheng, Xiaoyi Jiang, and Xuguang Lan, editors. *Advances in Machine Vision, Image Processing, and Pattern Analysis.*, volume 4153/2006 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, August 2006.

[263] Nanning Zheng and Jianru Xue. *Statistical Learning and Pattern Analysis for Image and Video Processing.* Springer-Verlag London Limited, London, 2009.

[264] Gernot Ziegler, Art Tevs, Christian Theobalt, and Hans-Peter Seidel. Gpu point list generation through histogram pyramids. Research Report MPI-I-2006-4-002, Max-Planck-Institut für Informatik, Saarbrücken (Germany), June 2006.