Master's Thesis

Autonomous Navigation of a Mobile Forklift

Stefan Kaltner

Institute of Software Technology (IST)
Graz University of Technology

Assessor: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Wotawa
Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Gerald Steinbauer

Graz, January 2015

Abstract

In this thesis, the problem of the autonomous navigation of a mobile forklift is investigated. The setup of the thesis is inspired by an autonomous robot for the automated change of batteries of an electrically driven transporter. In a cooperation with an industrial partner, a mechanical prototype was designed and manufactured. Based on an evaluation of already existing approaches for autonomous navigation, an electrical, algorithmic and software concept was designed and implemented. Because commonly used and freely available approaches primarily rely on the motion model of a differential drive, the approaches had to be adapted for special kinematics of the forklift (Ackermann). The consideration of the particular kinematics of the forklift improved the performance of the autonomous navigation significantly. Finally the functionality and performance of the realization of the overall system were evaluated in extensive and systematical experiments. The results of the evaluation pointed out, that the desired functionality was achieved and a reliable, autonomous navigation in an indoor environment is possible.

Kurzfassung

In dieser Diplomarbeit, wurde das Problem eines autonom navigierenden Gabelhubwagens bearbeitet. Das Setup dieser Arbeit beruht auf einem Roboter für den automatischen Batteriewechsel bei einem elektrisch betriebenen Transporter. In Zusammenarbeit mit einem Partner aus der Industrie wurde dafür ein mechanischer Prototyp entworfen und aufgebaut. Basierend auf der Evaluierung bestehender Ansätze für die autonome Navigation, wurde ein elektrisches, software-technisches und algorithmisches Konzept entworfen und umgesetzt. Da die gängigen und frei verfügbaren Ansätze primär auf dem kinematischen Modell eines Differentialantriebs beruhen, mussten diese für die spezielle Kinematik des Gabelhubwagens (Ackermann) adaptiert werden. Die Berücksichtigung der speziellen Kinematik des Gabelhubwagens verbesserte die Performance des autonomen Navigierens wesentlich. Die Funktionalität und die Performance des realisierten Gesamtsystems wurden abschließend in umfangreichen, systematischen Tests evaluiert. Das Resultat dieser Evaluierung zeigte, dass das System die gewünschten Funktionalitäten erfüllt und dass in einer Indoor-Umgebung eine zuverlässige autonome Navigation möglich ist.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Place/Ort	Date/Datum	Signature/Unterschrift

Acknowledgement

I would like to thank all the people, who supported and encouraged me during my time at the university and especially during the work on my thesis.

First of all I would like to thank all my colleagues at the university, with special thanks to the whole Institute für Software Technologie (IST), my advisor Gerald Steinbauer and all the colleagues at the institute for the great support and the guidance through the work of the thesis. Furthermore I would like to thank the industrial partners, CHANGE, in particular Jürgen Gugler and Manfred Wonisch, for the great opportunity, the support and the financial funding. This thesis was partly funded by the the Austrian Research Promotion Agency (FFG) with the "Innovationsscheck" program and the Science Park Graz [1].

Additionally I thank the Institut für Elektronische Musik und Akustik (IEM), who supported me in conducting the experimental evaluation by providing the Vicon tracking system.

Moreover I would like to thank my family, especially my parents, for offering me the possibility to follow my interests and the never-ending support and backing not least financially.

Additionally i would like to thank VH for enduring the stressful time with me.

Last, but not least my thanks go to MP, PE and CS.

[1] *http://sciencepark.at/*

Contents

List of Figures

List of Tables

1 Introduction

Nowadays the demands on parcel delivery agents are enormous and require the utmost concentration an the additional exchange of heavy batteries would result in an high risk potential. Furthermore the time consumption of the exchange could be optimized and the influence of dirt and contamination of the prone supply system could be minimized.

The scientific field of robotic is evolving pretty fast. State-of-the-art robots are capable of achieving advances concerning navigation, interaction with the environment and much more. Since the field of intervention is constantly broadening, the idea of an autonomous robot, which is changing the battery of an electrical driven transporter was born.

Figure 1.1: The electrically driven transporter designed for parcel delivering fleets, including a scenario of a battery exchange.

An approach targeting these inconveniences consists of the installation of service stations, where a mobile robot is utilized in the form of a forklift (c.f. Figure 1.2). The autonomy of the robot should provide a convenient solution for these problems and reduce the stress of the employees to a minimum. Additionally the time and cost savings could amortize the acquisition and support the deployment of electrical driven, sustainable transporter fleets.

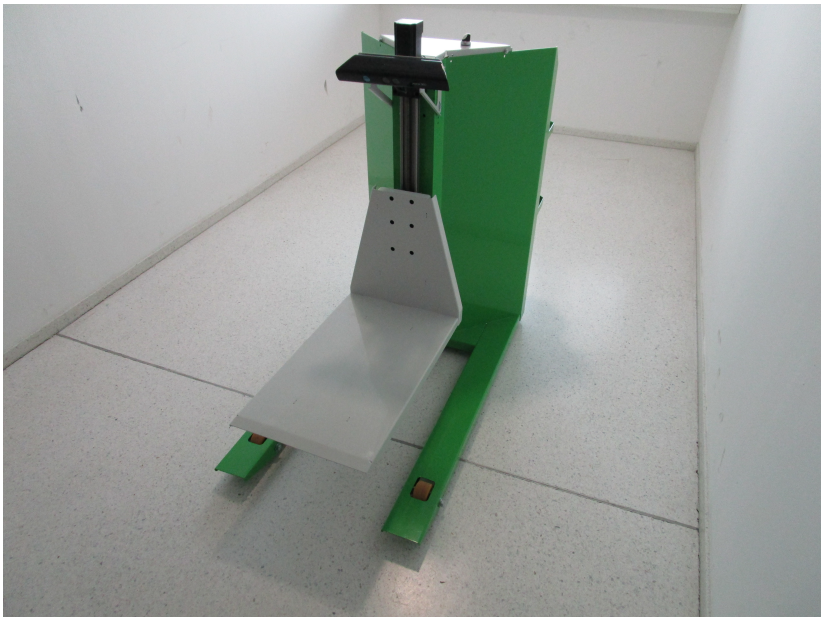

Figure 1.2: The picture shows the realization of the mobile robot.

1.1 Motivation

This idea was developed by a start-up business $CHANGE$ [1] and targets companies with fleets of electric transporters for delivering parcels. As competitiveness in the economical as well as in the technical meaning has to be ensured, the system has to be designed for a low price, but not for the trade off regarding the provided functionality. Since the requirements for a potential, final implementation include the lifting of batteries with about 600 kg the demands are extensive. As a result the main focus of the early phase has to target the design and everything relying on it.

Due to the fact, that there already exists another company which redesigns and rebuilds a mass product forklift, this basic design was taken into immediate consideration. Based on this concept the idea was born to extract and modify the controller of the product in use and develops a fully computer-controlled forklift. Unfortunately a cooperation with the producer of these particular forklift could not be established. Hence a new forklift has to be designed and built. Trough the process of creation, the question regarding the implementation of the interface between hardware and software arises. It demands the evaluation and the usage of communication models and methods, providing access to the functionality of the hardware.

The autonomy of the robot represents the main purpose of the project, for the robot shall be

[1] $http://sciencepark.at/unsere-firmen/556/change$

able to navigate in a known environment (e.g. service station), where even huge obstacles shall be feasible to be located. Navigation denotes the key aspect of autonomy regarding mobile robotics. It combines three dependent parts:

- the *mapping* of an unknown environment based on the perception of the environment und the feedback of the drive
- the *localization* within the mapped environment with respect the current perception and the recorded map.
- the *path planning* and furthermore the path execution depending on a goal and the feedback of the localization.

One of the challenges consists in the variation of the otherwise static environment caused by obstacles or objects of interest.

As dynamic obstacles (e.g. transporters/ cars) have to be integrated into the planning process, the mere collision avoidance is not sufficient and the problem task involves moreover the interaction with obstacles.

The kinematics of a forklift (represented through Ackermann steering) represents a more challenging task as common robot motion models (e.g. differential drive, cf chapter 4.2.3). Most state-of-the-art techniques are optimized for differential drive and omni directional kinematics. Therefore these techniques must either be adopted or new developed.

Further the aimed low budget solution demands the deployment of alternative sensor systems, providing the base for navigation. Already existing and commonly used, standalone systems these days (e.g. laser scanner) are costly due to their efficiency and reliability. The challenge consists in the balancing act of acquisition costs and the performance of the overall systems regarding the sensors.

1.2 Contribution

This thesis contributes answers and approaches for the main objectives of Section 1.1:

Evaluation of Existing Systems First of all different sensor systems, capable of meeting the requirements are explained and discussed, supported by related, already existing projects and approaches.

Construction of the Robot The realization of a robot, built from scratch, was a achieved by the composition of a chassis, two motors (including their controllers) and adequate, electrical wiring concerning energy suppply and communication.

Implementation of the Base Functionality According to the usage of a notebook, providing the processing power for the high level autonomy of the robot, the interface for the communication had to be established. A software structure was developed to grant access to controllers relying on the CAN-Open standard. The main challenge targeted the reusability, hence the disentanglement of the hardware-related interface from the sole usage of fixed controllers and standards.

Implementation of the Autonomy Regarding the Navigation The primary focus was located in the autonomy and the high level control of the mobile forklift. Different approaches had to be inspected, discussed and adopted to fit the demands of the system, regarding kinematic constraints and limited resources (such as time and computational effort). The thesis contributes the realization of an autonomous navigation based on a mobile robot with Ackermann steering, whereas furthermore the environment

Evaluation of the Realization and the Different Navigational Approaches In the end, the thesis evaluates the different approaches of the autonomy in combination with different sensors, including both qualitative and quantitative setups.

1.3 Structure of the Thesis

The thesis is divided into the following chapters:

Problem Description The problem description (c.f. Chapter 2) provides a refined definition of the main objectives. Thus bottom up relevant issues are examined and forwarded to the appropriate chapters and sections.

Related Research Chapter 3 discusses already existing research regarding sensor systems, navigation approaches and robot realizations including Ackermann steering.

Background To fasten an easy understanding of the subjects addressed in the later chapters, Chapter 4 describes and explains encountered technologies, algorithms and software packages.

Implementation In order to maintain a clear line of reasoning, the implementation as well of the hardware as of the software is treated bottom up in Chapter 5. Finally at the end of the chapter, the realization of the mobile forklift is presented.

Evaluation Due to the fact, that the realization has to be as well tested and evaluated concerning the proper base functionality, as well as evaluated with respect to the different resulted approaches of the autonomy, Chapter 6 presents a series of experiments. These experiments are divided into several sections, at first tackling the evaluation of the base functionality of the mobile robot, followed by specific testing concerning the different systems of autonomous navigation. At the end the best evaluated systems are examined regarding the accuracy with respect to a referral system providing ground truth for the tests.

Conclusion In Chapter 7 the experiments and the according results are discussed in respect to the initial goals stated in Chapter 3.

Future Work In this chapter we will present additional approaches not tackled and possible continuations, as well as tasks, which exceeded the resources of this thesis.

2 Problem Description

In order to provide a detailed overview of the problem description it is divided into several subsections marking the major objectives, in a bottom up model starting from the base construction of the hardware. Establishing the autonomy of a navigating, mobile forklift further demands the instrumentation, the software implementation and the high-level control running on a pc, mounted on the forklift Moreover an enlargement of requirements is dictated by the company, we are cooperating with, implying minimal costs and reusability.

In short the goals are:

- planning of the sensors and its proper alignment
- instrumentation of the system
- establishing software framework
- maintaining the navigational autonomy

2.1 Design of the Robot

As mentioned in the introduction (c.f. Chapter 1), the design of the mobile robot represents a very essential part of the thesis, as all other problem definition depend heavily on, or at least are influenced by its details. According to the predetermined (requirement of the partner) base construction of a forklift (which can be seen in Figure 2.1) arises a certain inevitable motion model and kinematics, described by the Ackermann steering geometry.

Due to the fact, that current solutions for robotic navigation mainly address differential drive, commonly used and distributed algorithms could not be integrated directly as will be seen in Section 2.3.

The main focus of the design phase addressed the selection of sensors and moreover their proper alignment and usage. The challenge consists in the decision, which has to be made a priori. As a result all possible problems arising have to be considered. Chapter 3 administers

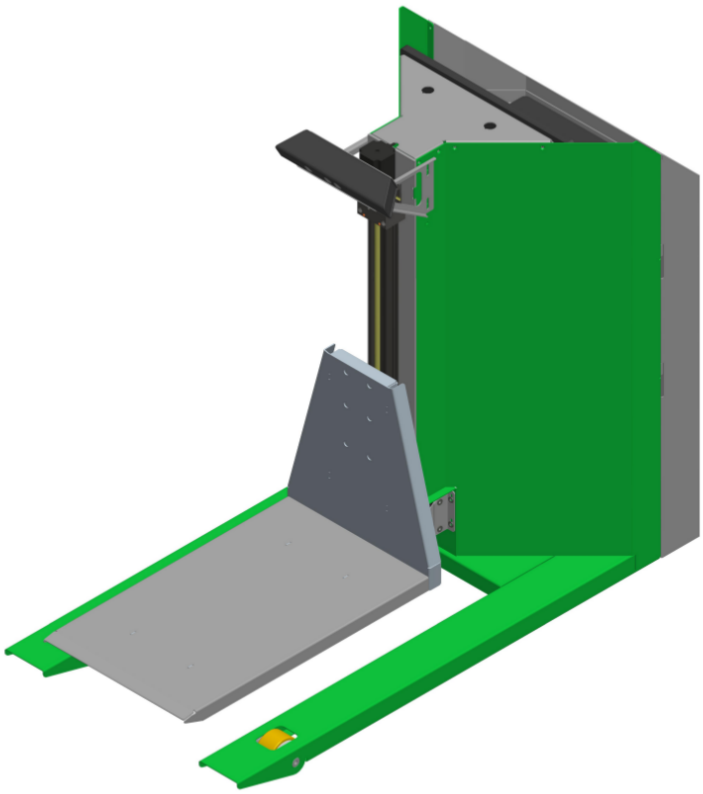

Figure 2.1: In this figure, the rendering of the mobile forklift in the construction phase is shown.

this problem definition by examining state-of-the-art approaches and extracting the pros and cons.

2.2 Hardware Implementation

To achieve the primary goal, consisting of the autonomy of the system, the system has to be instrumented accordingly. Another important milestone is denoted by the fundamental ability to move the robot. Because the company's responsibility consists for the construction of the chassis and the assembly of the actuators, the mounting of the controllers and sensors, as well as the maintaining of the energy supply has to be planned and implemented. Fitting the requirements and tolerances of the devices, different voltages and voltage smoothing have to be supplied. Additionally the proper wiring of motors, controllers and the communication to the processing device has to be installed.

2.3 Software Implementation

The software comprises the infrastructure for the base functionality as well as the high level control, regarding motion planning and obstacle avoidance.
Concerning the base functionality an interface has to be implemented providing the communication between the control pc and the motors.

The autonomy of a robot requires the perception and interaction with its environment. According the requirements, the software is responsible the processing of current senor data perceiving the environment, the planning of a collision free path including the robots kinematics and the reactive behavior concerning obstacles. We decided to benefit from an existing software framework based on Robot Operating System (ROS), although the majority of the algorithms provided are designed for either differential drive or omni-directional platforms. The Ackermann steering geometry, defining the kinematics of the forklift, demands fundamental adoption of the core elements regarding the navigation and further development to fulfill all specifications.

Concerning the initial idea of the forklift delivering task, one of the goals is declared for the reaching accuracy of a target position. The goal accuracy should be about +/-5 cm

3 Related Research

This chapter considers the related research of already existing systems and methods regarding mapping and navigation with robots characterized by Ackermann steering geometry. Furthermore other (partly commercial systems) are presented and described. In the latter part of the section related research with respect to the integration of sensors into systems is discussed.

3.1 Mapping

The mapping task embodies the first essential part of the autonomous navigation as all later planning is based on the generated map. Indoor mapping of a static environment represents a well established research field. In Scott et al. [22], laser scanning algorithms were evaluated and discussed. Current approaches in this field tackle extended tasks such as three dimensional (3D) mapping and outdoor environments [38, 51]. Furthermore new, more inexpensive sensors (e.g. Asus/Kinect) are tested and used [37] and also 3D mapping is accomplished [46, 48]. Chapter 4.2.5 describes the base idea of SLAM as well as improved algorithms.

3.2 Localization

Localization is heavily related to two factors:

- the perception of the environment
- the knowledge of the environment

As the algorithmic of sole localization and SLAM was already targeted in the last century (c.f. Chapter 4.2.5), todays research deals either with the challenges arising from blurry sensors or difficult environments (e.g. monotonic, few landmarks, difficult surfaces).

Other approaches try to achieve the localization via tracking. Therefore a sensor is pointed onto the seen, sending the information of the robot's pose directly to the robot. Such systems

are either realised through camera systems [59, 43], infrared projection [39] or wireless sensor networks [12]

3.3 Path Planning and Navigation

Path planning is part of every navigation system. The planning has to take the kinematics and the shape of the robot into account to be capable of avoiding collisions. This section presents the related research in the fields of Ackermann steering kinematics in the form of forklifts (as is desired for the robot of this thesis) and car-like robots or computer controlled automobiles.

3.3.1 Forklift

As stated before the primary goal is in the autonomy of the robot, which is only feasible, if the planner considers the kinematics. Ackermann steering geometry is rarely used in common robotic research open for public nor it is the use of a forklift. The only scientific contribution with open access which could be found was Correa et al. [8], although this forklift is steered by wire and its main purpose consists in the interaction with humans. Nevertheless industrial [1], [2] or even military [3] approaches can be found. Unfortunately, these appoaches provide solely a glimpse onto the resulting forklift, not revealing the algorithms and methods used.

3.3.2 Automobile

The Ackermann motion model is predominant in the automotive industry and many approaches tackle the problem of an autonomous car. The paper *Carlikerobotnavigationathighspeed* [44] presents an approach using the dynamic window approach (see Chapter 5.2.3). Another system using Ackermann is presented in [56]. This paper presents the optimization of an already existing path for a automobile. At last we present an approaches of a truck coupling with the help of a laser scanner to its trailer [49].

Another rather enormous contribution to this field of research was achieved by the Defense Advanced Research Projects Agency (DARPA) Grand Challenge of 2004 and 2007 and the DARPA Urban Challenge[4]. Teams from all over the world tried to navigate a car

[1] $http://www.rts.uni-hannover.de/index.php/Autonomous_forklift$
[2] $http://logisticsviewpoints.com/2014/04/02/autonomous-fork-lifts-make-warehouse-safer/$
[3] $http://www.popularmechanics.com/technology/engineering/robots/4269583$
[4] $http://archive.darpa.mil/grandchallenge/$

autonomously through the dessert and in the latter challenge through an urban terrain. The paper [9] contributes approaches and results achieved at the DARPA Grand Challange.

3.4 Integrated Systems

In this section we present already existing mobile robots, which are using the sensors presented in Chapter 4. Due to the fact, that the mere alignment and usage of a sensor is not primarily dependent on the kinematics the presented papers are only inspected for the sensor systems itself.

4 Background

This chapter addresses the background to ensure a clear understanding of the following chapters. In order to maintain a clear line of reasoning, it is divided into three sections providing an bottom up description for a mobile robot.

In the first section the technologies integrated in the forklift are presented starting with sensors used in the evaluation part of the consecutive chapter. Afterwards the base technology for the communication is outlined.

The second section targets the algorithms, methods and models used to the location and motion of the robot. Beginning with the basic concepts of a coordinate system and its proper representation, through the description of an object by means of the pose, to the proper motion model of an moving object (or robot). Building up on the motion within a coordinate system, the odometry is discussed. Furthermore, the concept of an autonomous navigation is presented, consisting of mapping, localization and path planning.

Following these algorithms, the third and last section discusses the software implementing the basis for the use of these algorithms (high level control) as well as for the the communication to the hardware interface (low level control).

4.1 Technologies

4.1.1 Laser Detection and Ranging

Laser Detection and Ranging (LaDaR) is a method to optically measure the distance to an object using laser beams. The same procedure is also capable of measuring the velocity, but as the velocity of an object is known due to the motor speed it is seldom used in robotics. To obtain the distance, the sensor emits a laser pulse and detects the reflected pulse irradiated from the object to measure. Following Siegwart et al. [47] three methods are commonly used to measure:

- measure the Time of Flight (ToF c.f. Section 4.1.2)

- measure the beat frequency of a frequency- modulated continuous wave (FMCW) and the reflection.
- measure the phase shift of the reflected light

The latter method is broadly used due to its processing speed and accuracy. In addition there are ways to align multiple sensors or rotate at least one sensor to achieve an array of laser beams or even three dimensional (3D) maps. The connection to this LaDar is either established by an Universal Serial Bus (USB) or in industrial products by Ethernet. Systems with an rotating laser sensor are commonly called laser scanner and are one of the most commonly used sensors in robotics.

For example the SICK LMS100 [1], uses this method to generate an array of measured points and therefor their distance to an obstacle in one plane. It has an aperture angle of 270 degree. The communication is realized via Ethernet. The power consumption is DC based at 24 Volt. With about 2000 $ the SICK LMS 100 is quite expansive, but certified as an industrial product, which is immensely important for the autonomy in environments were humans should be allowed to enter. SICK offers a comprehensive range of LaDaR, but as the LMS 100 remains available to use for this thesis it is chosen in for this comparison.

Besides SICK there exist many other producers of such laser based systems. One of these is Hokuyo [2], which in contrast to the SICK LMS 100 offers a USB connection and also requirering a power supply of 24 Volt. It also provides an aperture angle of 270 degree. Due to the fact that the Hokuyo is not industrial certified it is less expensive.

Another realization of an LaDaR is represented by VALODYNE[3][4]. This type of LaDaR have a set of lasers, which rotate. Therefore this systems can record whole three dimensional maps through rotating the aligned array. In this all-around vision lies the great advantage of these systems. Therefore there is no blind spot, if the sensor is mounted adequately. One of the disadvantages of this system is the high price.

4.1.2 Time of Flight Camera

Time of Flight (ToF) acquires 3D scene information through an active sensor principle. A light impulse is emitted onto the scene and the distance to every point is measured independently and simultaneously. The distance is the result of the time of flight of the emitted light

[1] $http://www.sick.com/group/DE/home/products/product_news/laser_measurement_systems/Seiten/lms100.aspx$
[2] $http://www.hokuyo-aut.jp/02sensor/index.html\#laser$
[3] $http://www.velodynelidar.com/lidar/hdlproducts/products.aspx$
[4] $http://www.velodynelidar.com/lidar/products/brochure/HDL-64E\%20Data\%20Sheet.pdf$

beam. Furthermore the amplitude of the captured light beam is used as a quality sign, as the amplitude is proportional to the amount of light reflected [3].

Depending on manufacturer and model the sensors can have a resolution up to 1280 x 1024 pixels and up to 160 frames per second (FPS). Due to the fact, that this system is relatively new, there are many producers and the development is evolving quite fast. The following Table 4.1 contains some examples of sensors. The informations in the table are extracted from the internet, the regarding webpages are linked in the caption of the table.

In addition there exist few relevant scientific material of their use in navigation [3, 18]. As a result the table relies primarily on information acquired from data sheets of the manufacturer or other sources linked appropriately.

Table 4.1: List of available ToF cameras including actual prices, the field of view denoting the aperture angle and the resolution in pixel. The source of information was the internet: SwissRanger 4000 [5] and CamCube 3.0 [6]

Sensor	Cost [$]	Field of View [Degree horizontal X Degree vertical]	Resolution [Pixel horizontal X Pixel vertical]
SwissRanger 4000	4,295.00	69 x 55	176 x 144
CamCube 3.0	12,000.00	40 x 40	200 x 200

4.1.3 Camera

Commonly known Red-Green-Blue (RGB) cameras can also be used to retrieve environmental information. Primarily this is achieved through object recognition and tracking, which is often quite expensive in respect with the data processing. It is also possible to track motion through the temporal evolution of an image, but state-of-the-art systems consist usually of at least two cameras to be capable of generating a scene of three dimensions. The front of an object is known as 3D but how the rear side looks like or the space occluded are unknown. The variety of camera is huge, important measures are for example the light intensity, to render it less vulnerable against environmental influences concerning light, resolution, to retrieve the maximum of informations and sharpness and therefore be able to distinguish features. A trade off between these measure and the costs for acquisition as well as for the computation, has to be chosen carefully with respect to the requirements of a system.

[5] $http://www.mesa-imaging.ch/products/sr4000/$
[6] $http://www.pmdtec.com/news_media/video/camcube.php$

4.1.4 Kinect 360 and Asus Xtion PRO LIVE

Besides the previously mentioned possibility of computing a three dimensional scene through two (or more) cameras, exists another commonly used realization used by the Kinect 360 and the ASUS Xtion Pro Live. This method includes a combination of a projector and a camera used as a depth sensor. A infrared laser projector projects a predefined invariable pattern onto a scene. The monochrome CMOS camera detects the projected pattern and computes the surface according to the pattern. This technique is called Structure from Light. In addition this type of systems include also a RGB camera.

These two sensors differ little and are therefore considered as similar systems henceforth.

4.1.5 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) measures forces (e.g. inertial or gravitational), which affect the sensor. This is used to determine the orientation and velocity and is realized through the combination of accelerometers and gyroscopes. A gyroscope measures the product of the moment of inertia and the angular velocity, with respect to speed, mass and shape of a certain body. The accelerometers on the other side, take the proper acceleration into account. Therefore they provide information about the actual acceleration (e.g. $9.81 m/s$ on earth surface on the Z-plane).

IMUs can be used for multiple important tasks, e.g. to verify and improve the odometry given from the motors and controllers or to maintain a particular sensor alignment (or other things). In the latter case a LaDaR is always able to scan the same plane being independent of the rotations of the x-y-plane (roll and pitch), through counterbalancing any movement by means of motor alignments in this plane.

The standard IMU is produced by $XSENS$ [5] and is offered in a broad variety of realizations (from low price to high resolution).

4.1.6 Radio Frequency Identification

Radio Frequency IDentification (RFID) denotes the technology of transferring data via electromagnetic waves within a transmitter-receiver-system. Due to the fact, that the communication is established through an electromagnetic field, no line of sight is needed. The reader communicates with the transponders (also called tags) to restore the electronically

[5] *http://www.xsens.de/produkte.php*

concealed information. As a matter of fact the energy consumption of these tags is small enough, that the energy from the magnetic field of the reader or even interrogating radio waves suffice to respond. This leads to very small and economic realizations of tags. RFID is capable of identifying tags. This information can also be used to track the tags.

4.1.7 Radio Detection and Ranging

In order to measure the distance to an obstacle a Radio Detection and Ranging (RaDaR) system deploys electromagnetic waves so called radio waves. Similar the the approaches of LiDaR or Time of Flight, the emitted primary signal is reflected as a secondary signal. This so called echo is detected by the system and based on the amplitude and wavelength conclusions on the detected object can be estimated. Radar is precise in mid- and far-distance. In short-distances the measurement evolve unrelyable [45, 53, 7].

4.1.8 Control Area Network

Control Area Network (CAN) denotes a standard created by the International Organization for Standardization (ISO). It consists of a real time field bus designated for serial data exchange. Regarding the Open System Interconnect (OSI) model [33] all layers above the Data Link layer [57].

4.1.9 CANopen

CANopen denotes a communication protocol based on CAN and the CAN-bus. Primarily it addresses automation engineering and embedded systems.

To fulfill the requirements of CANopen, every device has to transcribe a certain sub-standard. All of them have to cope with the basic demands of a bus-system. It is a multi-master serial bus and therefore connects certain instances (further called nodes). Every node has to have a fixed and unique identification (ID) within a CAN-bus, through which it can be uniquely addressed. These sub-standards all have there own purpose, hence not everyone is capable of sufficing the needs of the designed system.

Ott [6], a well known company in the automation sector, providing among other things solutions for motor-controller-systems, therefore uses the DSP-402 [5] standard relying on the DS-301 [6], which is the adaption of the communication from master (in this design the laptop or

[6] *http://www.ott-antriebe.de/*

computer) and the slave (hence the controller). These two standards are further discussed in Section 5.2.1

4.2 Algorithms

4.2.1 Coordinate System, Representations, Transformations and Frames

One major aspect of navigation concerns the position and orientation relative to objects, frames or coordinate systems. In robotics this is commonly represented using Cartesian coordinate system, which denotes the position and Euler angles to represent the orientation.

The Cartesian coordinate system is defined through coordinate axis which are orthogonal to each other and each represent one dimension. In Figure 4.1 one can see a three dimensional representation of an Cartesian coordinate system using X-, Y- and Z-axis. In this system every point in this system is well-defined and as such unique.

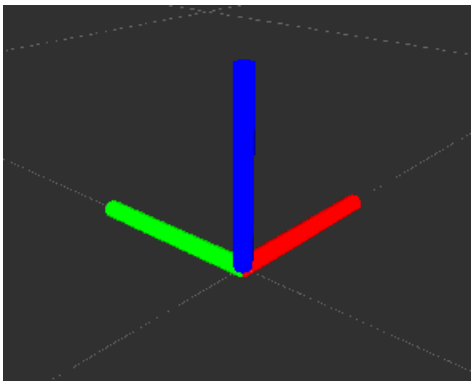

Figure 4.1: This figure displays a standard Cartesian coordinate system. The red arrow denotes the X-axis, the green arrow the Y.axis and the blue arrow the Z-axis

Throughout literature two representations of the Cartesian coordinate system are commonly used. The first one describes a system where an orientation or heading direction with $\theta = 0$ is pointing alongside the X-axis. In this case the direction of positive translational movement causes the mobile robot to solely change its x-value of its location. The Y-axis (θ =90 degree) forms the planar plane in the mentioned 90 degree manor to x. The Z-axis is pointing upward. The other representation is commonly used in computer vision regarding sensors such as cameras (e.g. Kinect/Asus see Section 4.1.4). In this case the Z-axis is pointing forward ($\theta = 0$), such that an two dimensional image of the taken scene would be a X-Y-plane representation regarding the coordinate frame of the sensor system.

Euler angles, which can be seen in Figure 4.2, denote a system of rotations which define the orientation of a rigid body in a coordinate frame. These three parameters are independent of each other and denote the rotation of an object's Cartesian coordinate system, *roll* is rotation about the X-axis, *pitch* is the rotation about the Y-axis and *yaw* the rotation about the Z-axis all regarding the objects coordinate frame. The orientation is represented by using a sequence of these Euler angles, where the order is of great importance. One disadvantage of this representation constitutes the Gimbal Lock [7]. Through the rotation of 90 degree, two rotational planes become parallel, which concludes in the loss of one degree of freedom [40].

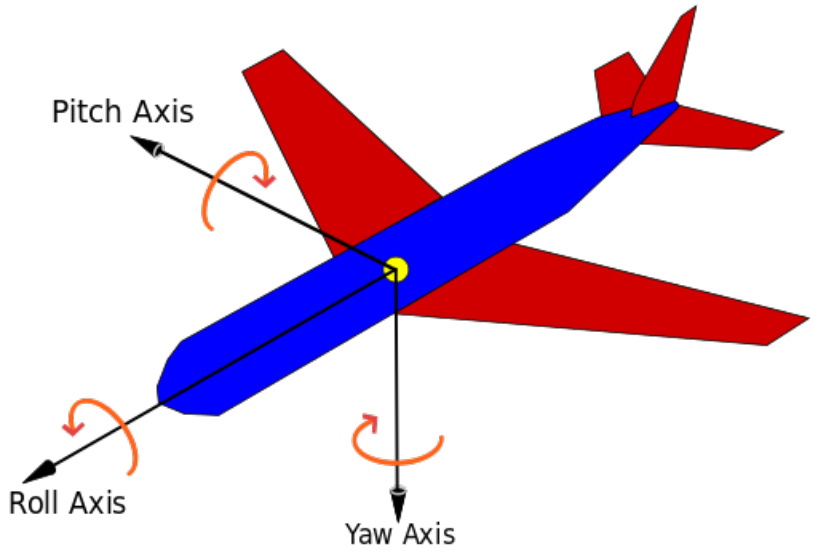

Figure 4.2: This figure depicts the Euler Angles. The arrows denote the rotations about the according axis. Roll about the X-axis, pitch about the Y.axis and yaw about the Z-axis. The picture is taken from $http://en.wikipedia.org/wiki/Euler\_angles$.

Other representations like quaternions are commonly used in state-of-the-art robotic algorithms. They are used, for example, in three dimensional computer graphics or computer vision, alongside with or even as alternative to Euler angles, as they avoid the Gimbal Lock. Quaternions consist of a real value in combination with three values (i, j and k) derived from the imaginary axis. Every quaterion can be expressed uniquely by a linear combination of these values multiplied by a scalar:

$$a + b * i + c * j + d * k \tag{4.1}$$

[7] $http://tetrahedral.blogspot.co.at/2011/03/gimbal-lock.html$

Furthermore is defined:

$$i^2 = j^2 = k^2 = -1 \tag{4.2}$$

Figure 4.3 depicts a graphical representation of quaternions. Moreover it shows the units product of the quaternions in the four dimensionmal space. The 90 degree rotations are as follows:

$$ij = k \tag{4.3}$$

$$ji = -k \tag{4.4}$$

$$ij = -ji \tag{4.5}$$

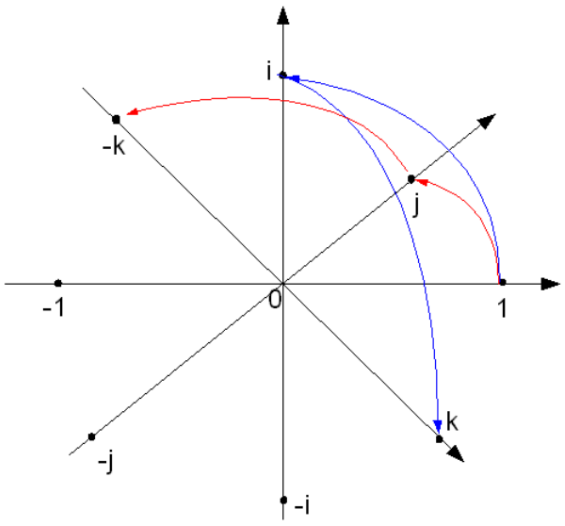

Figure 4.3: This figure depicts the representation of the 4D quaternions. The picture is taken from *http : //en.wikipedia.org/wiki/Quaternion*.

Further informations can be acquired from [29] or [8].

[8] *http : //mathworld.wolfram.com/Quaternion.html*

Throughout literature there are many representations and models used to describe the orientation and localization of an object (i.e. Geographical coordinate system). But as they are not relevant for this thesis they are neither explained nor mentioned further.

Due to the fact, that the knowledge of a robot is often limited regarding its environment, the concept of coordinate frames is induced. A coordinate frame is represented a Cartesian coordinate system as reference point. Each frame has its own origin and the transformation between these origins can be established by the combination of linear transformations according the X-, Y- and Z-Axis in combination with the rotational transformation according the Euler Angles. Often this representation offers a transformation of all used frames to a base frame (e.g. world coordinates or a global frame).

4.2.2 Pose, Orientation, Location and Degrees of Freedom

As Thrun et al. [52] records, in robotics the whereabout of an robot is commonly described regarding his location and his orientation, which is also called heading direction. The location is described with a position inside a coordinate system or any absolute or relative reference to such a system. It is not taking the rotation of the robot into account. If one combines the location with the orientation of the robot one speaks of the pose of a robot. Depending on the properties of a robot, mainly on the degrees of freedom (see Section 4.2.1), the location in our case consists of two dimensions for planar robots and the orientation of an angle θ.

The combination of the three dimensional Cartesian Coordinates and the three Euler angles, a pose can consist of six variable parameters denoted as the degrees of freedom (DoF). The DoF of the space define the maximum DoF a system within this space can have. Therefore the DOF of a system describe its mobility. Depending on the actuators moving the object the number of DOF varies. An aerial vehicle (e.g. helicopter) is able to move in all three dimensions and rotate about all three axis leading to 6 degrees of freedom. On the contrary a car is only capable of controlling the motion in one direction on a plane (e.g. X-Y-plane) and to rotate about the Z-Axis leading to two DOF.

4.2.3 Motion

This section deals with the definition of motion regarding the robot, thus the forklift. In robotics the driving system of a robot varies significantly depending on the designated purpose and the environment. Our main focus lies on the motion on solid and planar environments. Additionally we prospect solely under actuated systems, which means we attest that the input consists only of steering angle and velocity due to the fact, that the robot has two

degrees of freedom (DoF). Further the movement should consist solely of rolling objects, thus should be accomplished by using rolls, wheels and/or spheres, to be able to neglect the more complicate models of sole slipping and sliding objects.

According to Siegwart et al. [47], understanding motion highly correlates with the understanding of the contribution each wheel provides to the motiion of the whole system.

Basically these enumerated rotating objects can be distinguished by their properties. They can either be steerable or not steerable and either driven or non driven. Regarding these properties the three motion models commonly used are:

Differential Drive Differential drive denotes a system which can separately steer two wheels (or group of wheels) opposite to one another (c.f. Figure 4.4). This leads to the ability of rotating in place and afterwards driving in every direction. Systems using differential drive are for example the Forbot platform or the Pioneer DX3.

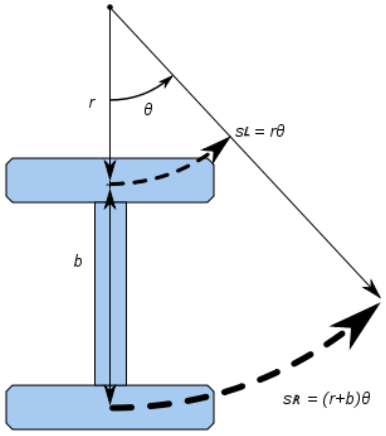

Figure 4.4: The image displays a model of a differential drive. The two wheels can be steered separately. The picture is taken from
http : //en.wikipedia.org/wiki/Differential_wheeled_robot.

Ackermann Steering Ackermann steering is highly common in automotive industry. This model consists of one steered axis and another non-steerable axis (c.f. Figure 4.5). Which of the axis is driven and which is of no importance for the motion model per se.

Omnidirectional As the name suggests, an omnidirectional robot is able to drive in any direction, regarding his steering system (c.f. Figure 4.6). This has to be achieved without previously turning in place. There are different realizations of such systems, for example

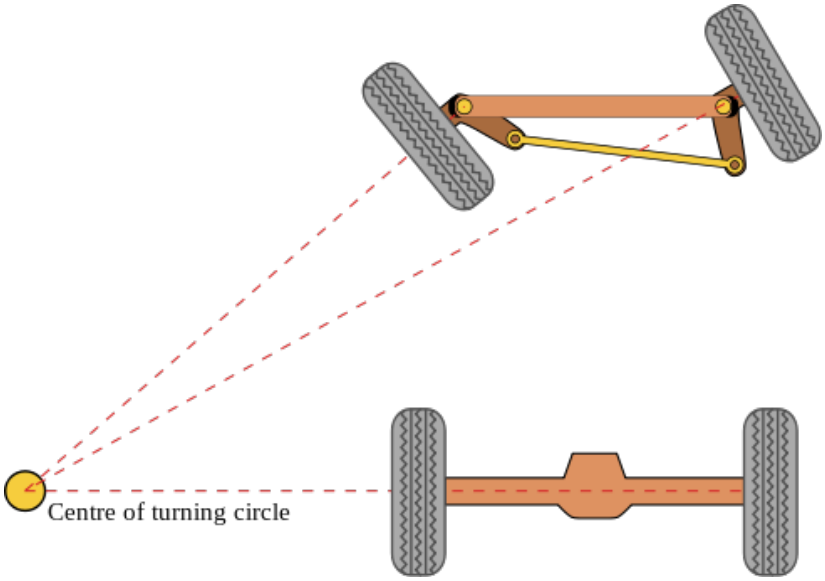

Figure 4.5: The image displays an Ackermann steering geometry, including the center of rotation (centre of turning circle). The picture is taken from *http : //en.wikipedia.org/wiki/Ackermann_steering_geometry*.

through an Mecanum wheel [9], an Omni Wheel [10] or a proper alignment of multiple steerable driven wheels [11].

Synchronous Drive The synchronous drive (synchro drive) is realized with three wheels each driven and steered, controlled by solely two motors [47]. The translational motor controls the velocity of all three wheels and the rotational motor spins all three wheels about their individual steering axis. RWI [12] and further Irobot [13] are the well-known representatives for robots with synchronous drive.

Holonomic and Non-Holonomic According to Siegwart et al. [47], a holonomic robot is represented by a robot with no non-holonomic wheel. Furthermore a non-holonomic wheel requires the derivatives (e.g. $\dot{\phi}$) of a values defining the kinematic contraints (e.g. x, y or Θ). For example the sliding constraint of a robot is a non-holonomic contraints, as it depends on the robot motion $\dot{\xi}$.

[9] *http : //de.wikipedia.org/wiki/Mecanum-Rad*
[10] *http : //en.wikipedia.org/wiki/Omni_wheel*
[11] *http : //www.neobotix − robots.com/mecanum − robot − mpo − 500.html*
[12] *http : //www.openrobots.org/morse/doc/1.2/user/robots/b21.html#*
[13] *http : //www.irobot.de/*

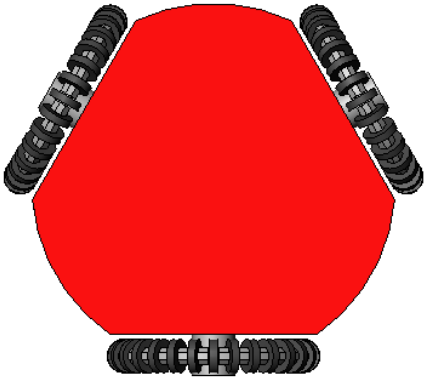

Figure 4.6: The image displays a realization of an omnidirectional robot, which is able to move in every direction on the X-Y-plane. The picture is taken from *http //commons.wikimedia.org/wiki/File Robot_omnidirectional _drive_topview.PNG.*

In the following section we present the principles of the velocity motion model followed by the different motion types. They are therefore discussed with its attention focused on differential drive, because its highly used in state-of-the-art robotics, as well as Ackermann steering. Furthermore, we describe the present special case of Ackermann steering and transfer it to a differential drive model using special assumptions.

4.2.3.1 Principles of the Velocity Motion Model

The velocity motion model as defined in the book Probabilistic Robotics by Sebastian Thrun et al. [52] assumes that one can control the motion of a robot through two velocities: a translational (v) and a rotational (ω) velocity. Nowadays many robotic systems provide these two velocities as controlling interface. Nevertheless the motion itself is constructed out of the superposition of the motion of the individual wheels (independent of controllability of the steering and drive) [47]. A proper movement can only be achieved if the normals of all wheels meet in the center of rotation, as one can see in Figure 4.8.

The kinematics form the basis of the motion model, which is needed for the task of proper collision avoidance and the motion planning task (c.f. Section 3.3) as such implies also fundamental knowledge of masses and inertias of the robot. This motion model therefore is essential for the software control, as all high level software takes into account the kinematic to estimate the robots behavior. The influence of the motion model for the control software will be presented in Section 5.

Due to the influence of uncertainties of the control and of exogenous effects, the estimated model is probabilistic (therefore non-deterministic). To handle the real world's uncertainties, which can hardly be modeled, we add noise parameters to the models of the motion estimation.

Thrun at al. [52] note, that in practical applications it has been proven to be more important to have methods to deal with the influences of uncertainties than to model them accurately.

In the following sections the motion model, inspired by Thrun et al. [52], will be discussed. Henceforth, t denotes the time, v_t the translational velocity, w_t the rotational velocity and u_t a vector consisting of these two velocities at the time t.

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \tag{4.6}$$

Additionally we assume, that a positive value of v_t induce motion following the heading direction and a positive ω_t induce a counterclockwise rotation about the Z-axis.

Basically the motion model influences three major topics:

- *forward kinematics*: motion of the robot according its geometry and velocities of wheels
- *odometry with limited update frequency*: calculating the motion from the actual integrated wheel velocities, depending on the feedback update of the motor encoders
- *fault model of the localization*: the estimated motion of the robot according to the published odometry

4.2.3.2 Motion Model Differential Drive

To properly understand to probabilistic case of the motion model, let us start with exact motion meaning an ideal, noise-free robot. As can be seen in Figure 4.7 taken from [52] the robot moves on a circle with a fixed radius r given by

$$r = |\frac{v}{\omega}| \tag{4.7}$$

Encompassed here are the special cases of $\omega = 0$, where no rotation is added and the robot is moving on a straight line hence a circle with an infinite radius. The other special case is an in-place-rotation, which is described by an circle with radius $r = 0$.

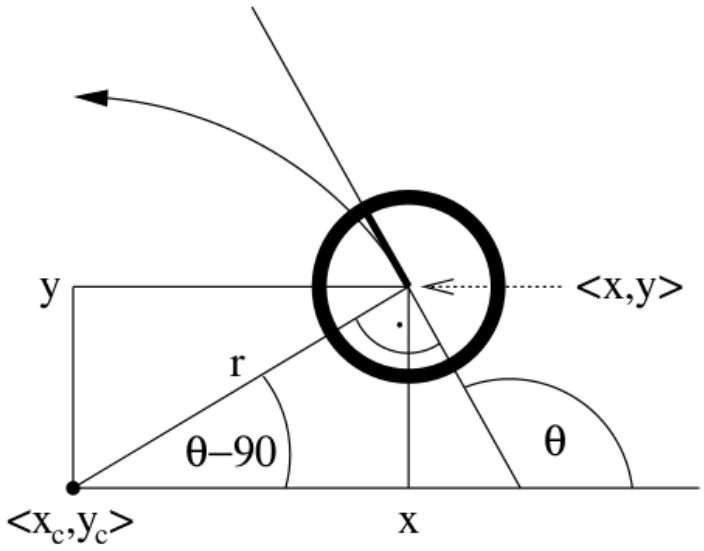

Figure 4.7: Motion model of a differential drive including center of rotation ($< x_c, y_c >$). The robot drives noise free with parameters (v, ω), which lead to x, y and θ. The figure is taken from [52].

Let us assume the initial pose of the robot is described as $x_{t-1} = [x, y, \theta]^T$ and it moves for some time δ_t with a constant velocity u_t. The change of the pose is now derived from

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega} sin\theta + \frac{v}{\omega} sin(\theta + \omega\Delta t) \\ \frac{v}{\omega} cos\theta - \frac{v}{\omega} cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix} \tag{4.8}$$

Through simple trigonometry and the relation (4.7) we can show, that the change of the heading direction is given through $\omega\Delta t$ and the translational advance through $v\Delta t$. The progress regarding the coordinates x and y are caused by the combination of the translational and rotational advances. To be capable of non-constant velocities Δ_t has to chosen adequately small.

As mentioned in the previous Section 4.2.3.1, the motion model influences three major topics of the software control. Due to the fact, that the motion in reality commonly the parameters used in the previous equations of this section are provided with additional noise. The forward

kinematics use the noise in form of uncertainties to avoid collisions. The odometry does not take uncertainties into account, as it provides reliable data, assuming the measurement of the motor sensors, from which the odometry is computed, is reliable, c.f. Section 4.2.4. Due to the fact, that the localization depends on the odometry, measurement uncertainties are added for a fault correction model.

In literature one often sees the parameters of α, which are also used in the later chapters of this thesis regarding ROS package implementations 5.2. These α model the accuracy of the robot, the higher the value, the worse is the accuracy. For now it is sufficient to understand, that the system assumes that every input value imply a certain variance, hence control- and error correction mechanisms are obligate as will be mentioned again by the chapter related to software (c.f. Chapter 5.2).

4.2.3.3 Motion Model Ackermann Steering and Transduction to Differential Drive

s

As discussed in Chapter 5.1.1 the motion model of the mobile forklift is described by Ackermann steering geometry. The basic motion model of Ackermann refers to automobiles, with one steerable and fixed axis both consisting of two connected wheels. Jazar et al. [24] described and proofed the concept of the center of rotation, which is denoted as Ackermann condition. It states that the normal line of the center of each wheel must intersect with the center of rotation. A special case is, alike the special case of the differential drive motion model, sole translational movement without any rotation, where the normal lines intersect at infinity.

Although the steered axis of the forklift consists of only one wheel, the Ackermann condition still applies, as shown in Figure 4.8. The motion concept has to be narrowed down to a reference point. For practical reasons this should either be realized as the back wheel or the point between the two front wheels. For simplicity and in consideration of the planning algorithms used in Chapter 5.2.3, we denote the reference point as a fictitious spot on the front axis and further refer to it as one front wheel of the robot or *baselink*.

With this assumption the robot motion of the front wheel can be considered as differential drive. But to meet the Ackermann condition the back wheel, as it is the steered wheel in our case, has to be moved adequately. As a result motion commands have to be given according to the posture of the steered wheel as

$$u_{ackermann} = \begin{bmatrix} v_{bw} \\ \phi_{bw} \end{bmatrix} \tag{4.9}$$

As can be seen in Figure 4.8 the Ackermann condition dictates the instantaneous center of rotation as intersection of the normal line of the back wheel and the front axis. Furthermore the front wheel as representation of the front axis as well as the back wheel stir along proportional circular paths around the instantaneous center of rotation. Per definition of the motion about the center of rotation both points cope the same ω within the same time. Thus follows that the velocities of these two points has to be the same proportion as the radii r_{bl} for the radius of the front wheel and r_{bw} for the back wheel. Thus follows from simple trigonometry and physics where ω is given as radian:

$$r_{bl} = |\frac{v_{bl}}{\omega}| \tag{4.10}$$

v_{bl} denotes the velocity of the base link and v_{bw} the velocity of the back wheel. ω, as stated before has to be the same for both.
Furthermore as can be again deduced from Figure 4.8 we know that the fixed length of the forklift has an angle of 90 degree to front axis. This leads to a right triangle containing both radii r_{bl} and r_{bw} as well as the length of the forklift l.
As r_{bl} is given through equation (4.10), because of the assumption that the velocity commands refer to the baselink, further one can compute r_{bw} to (4.13):

$$r_{bw}^2 = r_{bl}^2 + l^2 \tag{4.11}$$

$$v_{bw} = v_{bl}\frac{r_{bw}}{r_{bl}} \tag{4.12}$$

$$v_{bw} = v_{bl}\frac{\sqrt{r_{bl}^2 + l^2}}{r_{bl}} = \sqrt{1 + \frac{l^2}{(\frac{v_{bl}}{\omega})^2}} \tag{4.13}$$

With all sides of the triangle known, the angle ϕ of the back wheel arises through:

$$\phi = arctan(\frac{l * \omega}{r_{bw}}) \tag{4.14}$$

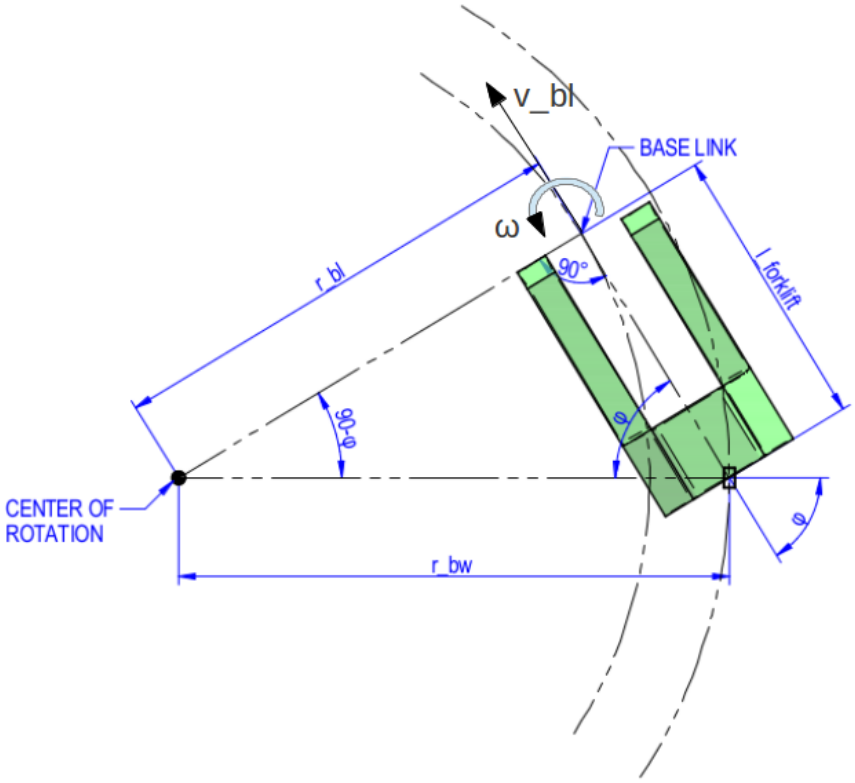

Figure 4.8: Principle of the motion model of the forklift. Back wheel and baselink both moving on a circular path around the instantaneous center of rotation

Again one has to be aware of the special cases for turning in place and sole translation. At the in place rotation the center of rotation corresponds to the baselink and therefore results in $\phi = \pi/2$. The more complicate case arises for the sole translation where the r_{bw} is infinite. Thus this must be specifically treated.
Through equations (4.14) and (4.12) the desired $u_{ackermann}$ can be calculated from u. Obviously the other direction can also be computed.

Thus the odometry package (c.f. Section 4.2.4) is capable of computing the odometry of the front wheel (or baselink) out of the feedback of the motor controllers. Let us further keep in mind the assumption of the baselink as a theoretical differential drive.
Most of the packages and algorithms, used in nowadays navigation algorithms, are either denoted to holonomic robots or to differential drive. Hence if we can transfer the motion model to differential drive we can benefit from already implemented and tested software packages.

4.2.4 Odometry

Odometry is one essential part of navigation systems of autonomous robots. It denotes the integration of measurements directly taken from the motors. Depending on the motion model of the robot, the integration of these measurements leads to a transformation from the world frame (map frame) to the reference point of the robot (e.g. base link). This transformation is presented as a vector, which describes the distance and orientation in an arbitrary coordinate system covered by the robot since the initial start. In 2D navigation the odometry consists of two coordinates (x, y) and the orientation θ of the robot. Although our system is dealing with the real world, the problem can be narrowed down to x, y and θ. This can only be established, if we assume that we are always acting in the Z-plane and/or every information content is checked for its relevance. Hence obstacles which appear only from or up to a certain height a collision check verifies all possible risks and therefore adds or neglect it for the 2D map for navigation. Furthermore it has to be mentioned, that the odometry is often presented with respect to the quaternion representation.

4.2.5 Navigation

Navigation combines the three prerequisites for autonomous navigation:

- the mapping of an environment
- the localization regarding the map and current sensor data
- the path planning and execution to be able to interact with the surrounding

The mapping has to be established relying on a current position, thus a localization. Hence the problem evolves to a chicken-and-egg problem. Due to the limitation regarding quality and reliability of a real robot's perception, the localization has to be probabilistic [4]. Obviously the result of a localization in reality is depending on the previous pose of the robot and the feedback odometry.
As will be discussed later in this section, the uniqueness and the quantity of landmarks are considered as one of the major indication of the quality of the established localization .

In the following paragraphs the basic idea of mapping, localization and path planning and execution are explained.

Simultaneous Localization and Mapping The Simultaneous Localization and Mapping (SLAM) problem consists of an moving object in an unknown environment and denotes one of the fundamental problems of modern robotics both in the conceptual and in the computational meaning. The challenge consists not only in the localization of the object but furthermore in the generation of a map of the unknown environment without any prior knowledge. Based on the online estimation of the trajectory of the object's movement and there spacial relation to the landmarks support is given for the propagation of the localization. How could localization be even possible without being able to elaborate a map from all gathered measurements of the different sensors. Moreover, how could the map building be consistent if the location of the mobile robot, making its observations is not certain [54, 14]. An historical overview concerning the disclosure and the evolution of the problem is given in the paper of Durrant-Whyte and Bailey [14].

The solution of this problem is realized by taking into account both the underlying kinematics (cf. Chapter 4.2.3) of the moving object of interest (mobile robot) as well as the appropriate observation model. In order to be capable of dealing with reality's uncertainty the solution has to be probabilistic [4]. Often this probabilistic approach is realized through a Dynamic Bayesian Network (DBN), where online computation is provided through the Markov assumption [54, 2]. The basic implementation following the paper of Durrant-Whyte and Bailey [14] assumes that the probability distribution

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0) \tag{4.15}$$

can be computed for all times k. This Probability denotes the joint posterior density of the location of landmarks and the the mobile robot regarding the accumulated measurement data of the given sensors.

U_k describes the control vector, which denotes the intended movement of the mobile robot to state x_k between from time $k-1$ to k. Z_k stands for the the set of observations, regarding the position of the mobile robot at time k, m denotes the landmark.

Further the initial pose of the mobile robot has to be known, or guessed (at least deterministic). To achieve the acquisition of map and location recursive two-step model is performed. The prediction step consisting of an time-update taking into account the motion model followed by an measurement-update to correct the prediction with respect to the observation model containing the actual sensor measurements. Hence the map is constructed by fusing the different sensor measurements taken at different known times at (different) known locations. On the other hand the location problem is solved assuming that the landmarks are known at

every time step and the location of the mobile robot can therefore be computed with respect to these landmarks.

There exist many different kind of realizations of the simultaneous localization and mapping problem. The standard version works with an extended Kalman filter (EKF) to deduce the information needed, which is described in Principles of Robot Motion by Choset et al. [4]. Another approach, often used in the RoboCup Rescue league is HECTOR-SLAM [27] [14]. This variation takes additional sensor data into account to improve the localization to be capable of refining the map built in exchange for computational effort.

A special implementation is called GMapping [15], where every particle owns its own individual map of the environment. GMapping uses the Rao-Backwellized Particle Filter [13], which is on the one hand quite computational expensive, but with high variances, the resulting precision is remarkable. The number of particles therefore has to be reduced to a reasonable number to not cause a sharp increase of the computational cost.

Monte Carlo Localization Monte Carlo Localization denotes the principle of determining the pose of an robot, using weighted randomly distributed pose particles [11]. Although its comparably young age it is extensively used and referred to in robotics, especially regarding dynamic environments [58]. It is also capable of solving the "kidnapped robot" problem, which is of highly practical use. The "kidnapped robot" problem describes a special case of localization, where the robot has to resume the navigation after its pose is changed without its knowledge. So a re-localization is inevitably [36]. This "kidnapping" can be caused by some sort of (temporary) software-, hardware failure or collisions as well as through real change of the position and/or the orientation of the autonomous robot.

Like mentioned before the algorithm of the Monte Carlo Localization generates samples, randomly distributed all over the environment. Each of these samples represent a possible location of the robot, which can be seen in Figure 4.9. The peaks in the $bel(x)$-graph of Figure 4.9 denote such particles.

The procedure evolves iteratively and can be split up into three steps per iteration:

- The first step consists of the detection of any kind of landmarks (e.g. walls or obstacles). The algorithm matches these obstacles with the one already known, which can be found in the map. Through the matching, which varies depending on the realization of the MCL (e.g. Monte Carlo Localization with Extended Kalman Filter (MCL-EKF) [58] ,

[14] *http://wiki.ros.org/hector_slam*
[15] *openslam.org*

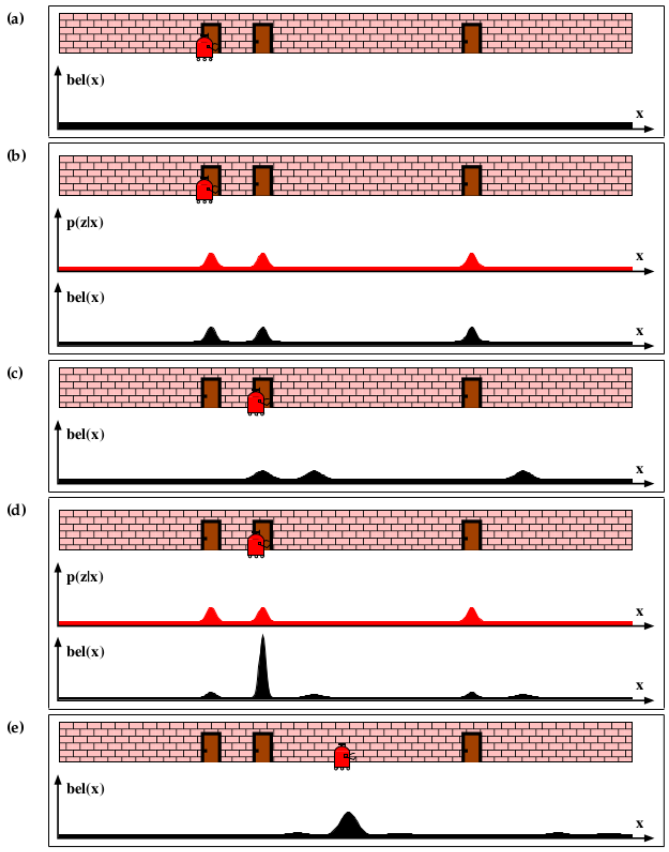

Figure 4.9: Illustration of the localization. The pictures depict the position of the robot. bel(x) denotes the believe of the localization and $p(z|x)$ the probability of observing a doof. The figure The image is taken from [52].

Adaptive Monte Carlo Localization (AMCL [11])), a probability is computed for every generated sample and a weighting proportional to this probability is established.

- The second step consists of a resampling of the original samples according to this new weighting, hence the granularity at the weighting peaks is increased (c.f. Image (d) of Figure 4.9).
- In the third step the robot starts his movement and generates sensor data (e.g. range measurement of a LaDAR as described in Chapter 4.1.1) and odometry data processed according to the motion model (described in Chapter 4.2.4). Assuming there exist sufficient many landmarks in the environment, because obviously the number is again the criterion.

Through a predefined motion model ($p(x_k|x_{k-1}, u_k)$) and a measurement model ($p(x_k|Z^k)$)

the algorithm computes his estimation of the movement of the robot. The variable are the same as in Equation (4.15). Depending on the motion model the samples are adjusted and the steps one to three are again executed and evaluated. In every iteration the granularity of the localization is refined, but as mentioned in [36] in every resampling step the centers of gravity (regions with particles of high weighting) are provided with the new additional samples. Depending on the realization of the algorithm a certain percentage of possible poses are distributed (again uniformly) to ensure robustness against system failures, environmental changes or even the "kidnapping" of the robot [52, 2].

A modification of the MCL Algorithm is called the Adaptive Monte Carlo Algorithm (AMCL). The modification consists, like the name suggests in the adaptability of the sampling. To achieve this, the sampling is realized through a Kullback-Leibler distance(KLD)-sampling. Hence arising problems (e.g. under sampling, enormous computational cost) using varying and unknown distributions can me minimized [15]. A version of this algorithm is available in ROS [16].

Its purpose is limited for localization in 2D environments as is sufficient for our purpose of a system with two DoF. As a result one needs to extract the 2D information out of the 3D environment. This can be achieved by several approaches, for example by using a LaDaR-system. This system provides the required information extracted from a detection plane of the laser, hence the obstacles are projected to a 2D scene. This approach of a projecting 3D depth information onto a 2D environment can also be realized with respect to a depth image. The easiest realization would be to discard the information of the Z-axis.

Path Planning and Path Execution The task of path planning implies the knowledge of the current pose of the robot, a goal pose and the perception of the environment. Choset et al. [4] induces the concept of a *configuration* as the description of a pose. The configuration denotes the complete specification of the position of every point of the system in so called generalized coordinates. Furthermore a *configuration space* (C-space) provides all possible configurations of a system. The dimension of the configuration space is given through the DOF of the the robot as well as the control parameters of the robot. Additionally we introduce the workspace, which includes all collision free configurations of the configuration space. With this in mind we can now assume the path planning task of a continuous mapping as the search for a collision free path to a goal configuration. In Figure 4.10 one can see the variation of the configuration space depending on the robot.

One option of planing consists of the rapidly-exploring random tree (RRT). A random filling

[16] *http://wiki.ros.org/amcl*

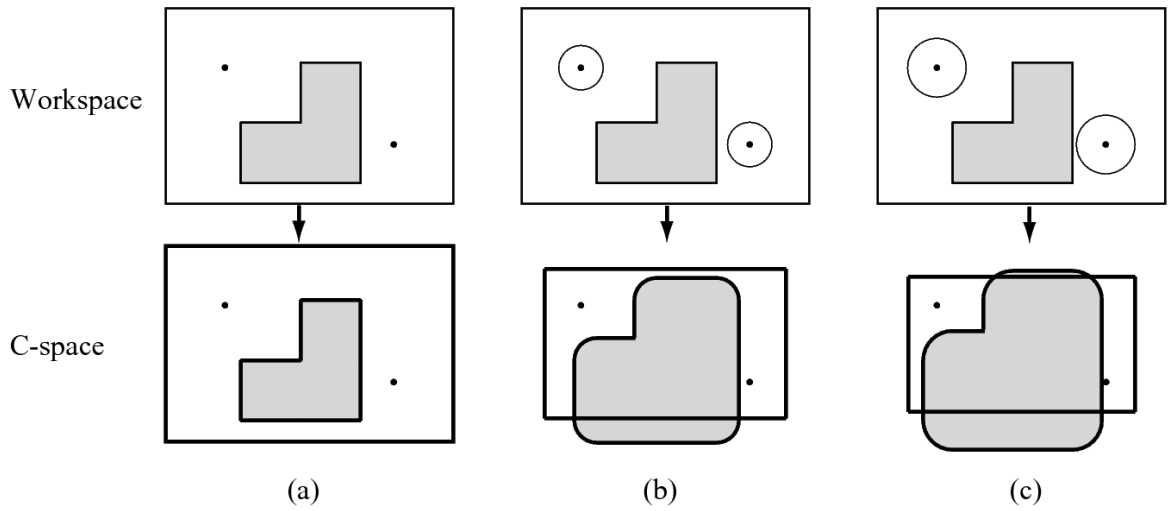

Figure 4.10: The plot is taken from Principles of Robot Motion, by Choset et al. [4]. It describes the relation of configuration space and workspace. The different configurations denote a) a point mobile robot, b) a circular mobile robot and c) a larger form of b).

tree is built in the configuration space consisting of a continuous path form the start pose to the goal pose. The samples are randomly drawn from the search space. The search space consists of collision checked poses of the robot [28, 31].

One method of planning through the configuration space is denoted by the *potential functions*. One example could be explained with respect to positive and negative charges which can be seen in Figure 4.12. Assuming that the robot as well as obstacles are positive charged and the the goal negative, the configuration space can be extended by gradients according to their potential. Based on this scheme, search based algorithm are used to find a valid path (e.g. gradient descent [4, 2]).

Another approach for path planning is denoted by *roadmaps*. In the graph like structure, nodes represent poses in the configuration space and edges paths between two neighboring locations. One has to reconsider, that these roadmaps are relying on a priori knowledge of the environment (e.g. pre-generated maps). On the other hand all paths and are precomputed and as a result the computational effort at runtime can be reduced significantly [4, 26].

Cell decomposition divides the map into several cells depending on the realization. Each cell constitutes a node and neighboring cells are connected through paths (edges). The consequential graph is used to search for a path connecting the start and the goal cell [30, 4, 2]

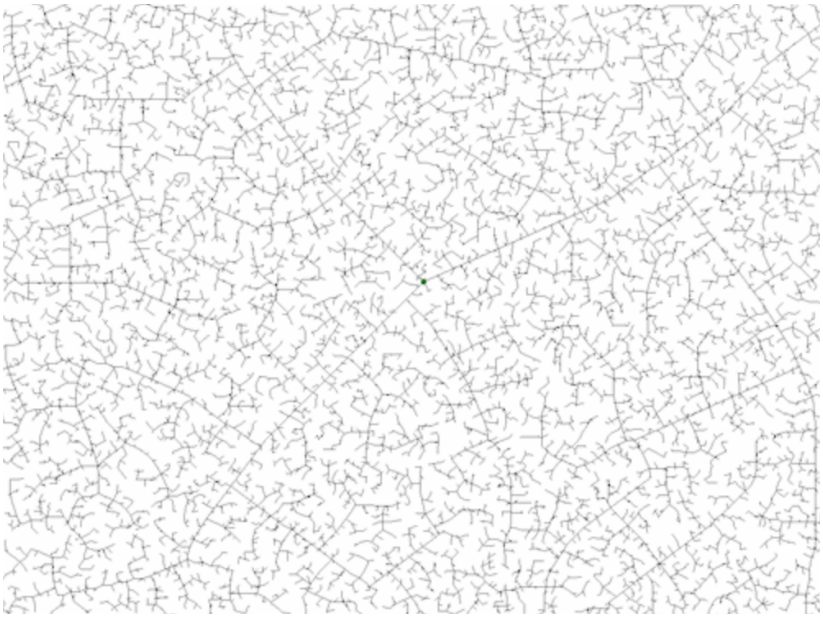

Figure 4.11: The figure depicts the generation of a rapidly-exploring random tree. Random poses are picked and connected to the nearest state. The plot is taken from *http : //en.wikipedia.org/wiki/Rapidly_exploring_random_tree.*

[17].

All these methods can be interpreted as a initial planner creating a path from the current position to the desired goal. This commonly called *global planner* is often used to generate a rough collection of positions between start and goal position. Additionally a concept of the *local planner* takes this collection and computes them to a trajectory consisting of commands, which are propagated to the executional software connected to the hardware. Furthermore the trajectory planning task takes into account the local obstacle avoidance contributed by the current sensor data. The commonly used concepts of local planning is further discussed in Section 5.2.3.

One of the local planner approaches consists in the Dynamic Window Approach [17, 16], which will be discussed more intensely in Section 5.2.3, has to be mentioned here.

[17] *http : //www.cs.cmu.edu/ motionplanning/lecture/Chap*6*-CellDecomp_howie.pdf*

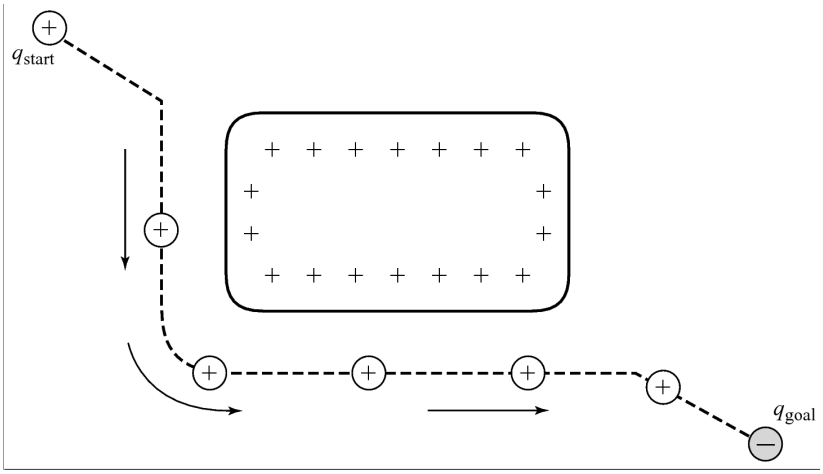

Figure 4.12: The plot is taken from Principles of Robot Motion, by Choset et al. [4]. The figure depicts a realization of a potential field. The robot denotes a positive charge as well as the obstacles, whereas the goal consists of a negative charge. Hence the robot is attracted by the goal and repulsed by the obstacles.

4.3 Software

4.3.1 Robot Operating System

The Robot Operating System (ROS) [42] [18] represents a software-framework which provides the capabilities to control each robot. Hence services such as hardware abstraction message-passing between nodes and processes, node and package management and low-level device control can be obtained.

The framework consists of one core where an arbitrary number of packages can be docked. The core facilitates the base functionality of an operating system such as coordination task and provision of communication. The functional software is implemented in packages which are launched and processed within separated nodes. Each running package consists of at least one node and each node belongs to one package. The same package can be started multiple times, although the node has to have a unique ID consisting of a string. The functionality usually comprises of device driver, hardware control, data processing or high level control. The resulting package-based, modular architecture provide high reusable and an easy opportunity for distributions.

This reusability is also supported through the Berkeley Software Distribution (BSD) license, which grants the base framework (and the majority of the mainly used and distributed

[18] *www.ros.org/*

packages) open source software and therefore free use for both commercial use and research.

The provided communication between those nodes is established through so called Publisher-Subscriber or Service-Listener realizations. On so called ROS topics one node provides data for other nodes subscribing on them. If a topic is published a callback is triggered within the according subscribed node. This model is primarily used to broadcast frequent sources of data, like sensor data or velocity commands which are published with a fixed frequency. Otherwise the service is designed to be called to establish a communication or data transfer between merely two nodes. The procedure is to call a service and wait for the reception of the requested data. A feedback is not mandatory but possible.

The usage of Linux is recommended but not obligatory, for there are as well distributions of other operating systems such as Windows and MacOS and also parts that are platform independent. The main client libraries consist of C++, Python and LISP, but there are already ambitious efforts to integrate also Java, Matlab and many other languages.

As the development emerges, new distributions of the software are provided. This thesis uses ROS Groovy.

ROS includes a visualization-tool called ROS-visualization (RVIZ) as can be seen in Figure 4.13. It is capable of processing and displaying most published topics used within ROS. Its main purpose for this thesis consists in the visualization of the transformations of coordinate frames (e.g. laser scans). Additionally the computed paths of the navigation task and everything related to the global frame or in this case the predefined map can be displayed. Furthermore the tool is used to provide a user-friendly method of initial guessing the current pose of the robot as well as setting an goal again all related to the used map.

4.3.2 Gazebo

Gazebo [19] denotes a simulation environment for all kind of robots. Originally it was developed and deployed within the ROS environment. Gazebo requires a Unified Robot Description Format (URDF) model or a Semantic Description Format (SDF) model of the robot in order to maintain an according simulation. These models include design and texture of the robot as well as physical properties. In Figure 4.14 one can see the simulation of the mobile forklift in a test environment including a transporter.

Unfortunately the support of the Gazebo version included in ROS Groovy was shut down while we started to use the software. Furthermore the later distributions of ROS use Gazebo as an external software. Therefore the latest version of Gazebo can be used.

[19] *http://gazebosim.org/*

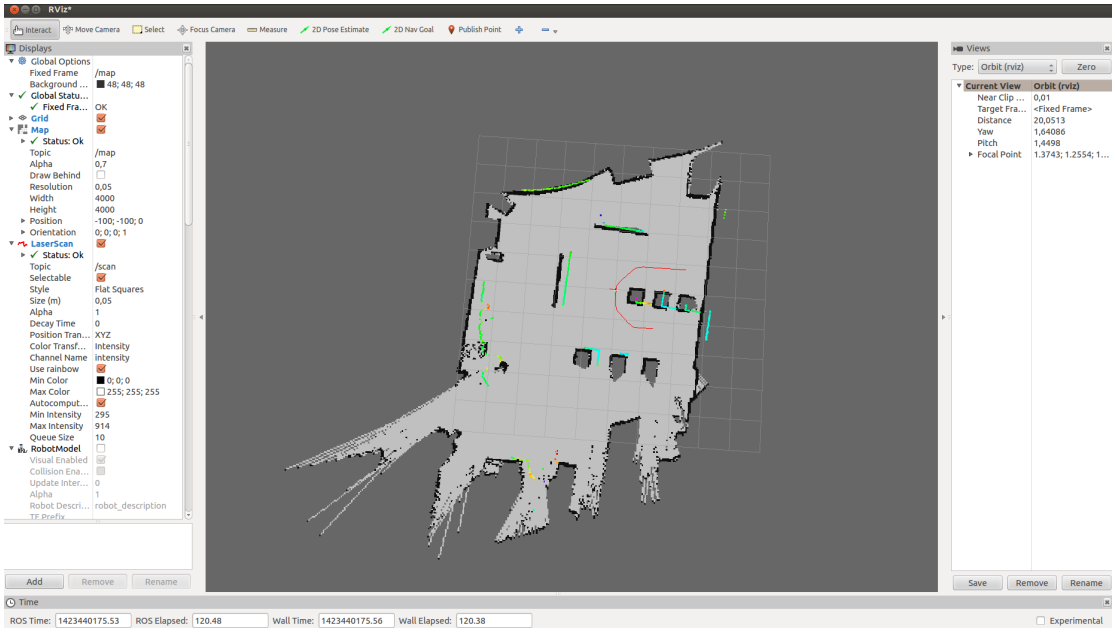

Figure 4.13: The figure shows the visualization of the Robot Operating System (ROS), which is called RVIZ. On the right side the topics, which should be published are chosen adequatly. The visualization of one of the test scenarios of Section 6 can be found in the middle window.

4.3.3 CAN Festival

CAN Festival [20] is an open source project for the communication based on a CAN-bus. This framework comprises the lower level communication including message packaging and handling. The provided open source code is distributed in the pogramming language C. CAN Festival is not dependent on any standard (e.g. DSP402 or DS301) based on CanOpen, but provides an interface. As such it is highly adoptable.
Above the CAN Festival a every CAN Open standard can be implemented according to the requirement of the used device. Beneath this layer a CAN node is needed which represents the interface from software to hardware. This layer can be exchanged according to the device used for establishing the connection between the computer and the CAN-bus.

[20] $http://canfestival.org/$

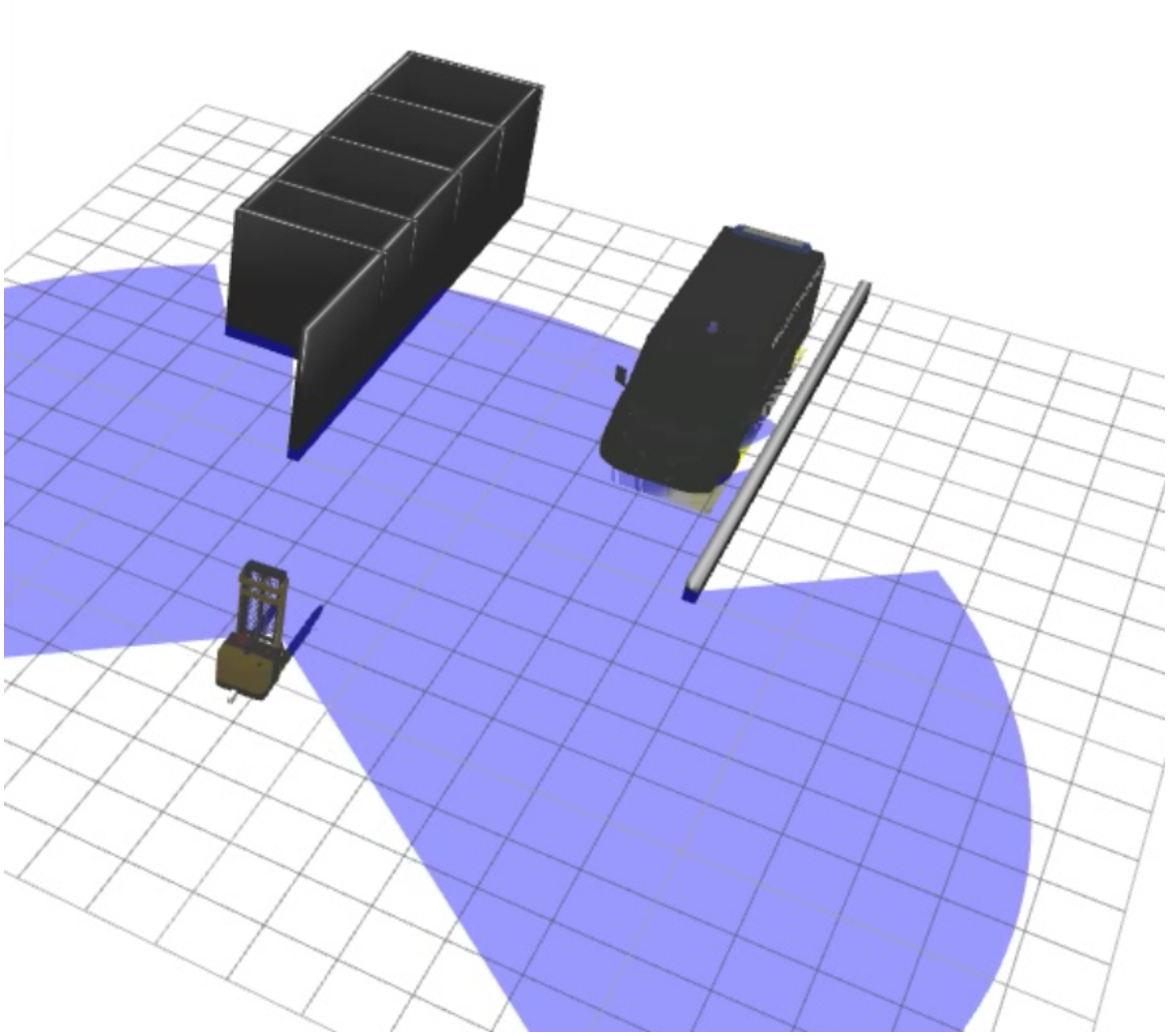

Figure 4.14: The image shows the simulation of the modeled forklift in Gazebo. The forklift is moving in an service station environment including a transporter. The blue rays display the simulated laser scanner mounted on the robot.

5 Implementation

The implementation and system design will be presented again in a bottom up view, addressing the hardware in the first section and the software in the latter. Figure 5.1 presents the overall overview of the system including the main concept. The motors (M), including the control loop with the encoder (E) denote the principal mechanical components. The CAN controller propagate the commands of the software interface, sent over the CAN-bus, to the according motor. Furthermore, the sensors and the end switches (ES) of the rotational motors complete the hardware devices. Including the execution and low level control, the computer (PC) embodies the interface to the hardware . The upper most part consists artificial intelligence (AI) comprising the high level control of the software.

5.1 Hardware Implementation

The system design of the hardware is divided into four major sections. At first the basic design is discussed, illustrating the design decisions regarding the chassis and the mechanics 5.1.1. Thereafter the drive of the robot will be briefly explained 5.1.2, followed a section regarding the controlling. This Section 5.1.3 addresses as well the controller itself as the communication and basic wiring of the system. Last, but not least, the sensors assembled at the robot are presented, including a discussion about there choice in comparison to other sensors already presented in Section 4.1. To ensure a better understanding, the realization of the robot is illustrated in Figure 5.2.

5.1.1 Basic Design

As mentioned before the requirements stated by the industrial partner include the lifting of quite heavy objects. Although this lifting in no concern for this thesis, with a view to further development, the robot should be constructed feasible to fulfill even these demands, which lead to the forklift design. Hence the first constraints are already set. The mobile robot is composed by two axis, a front axis consisting of two rolls and one axis in the back including one wheel. According to a standard forklift the front axis is fixed to rotation, whereas the

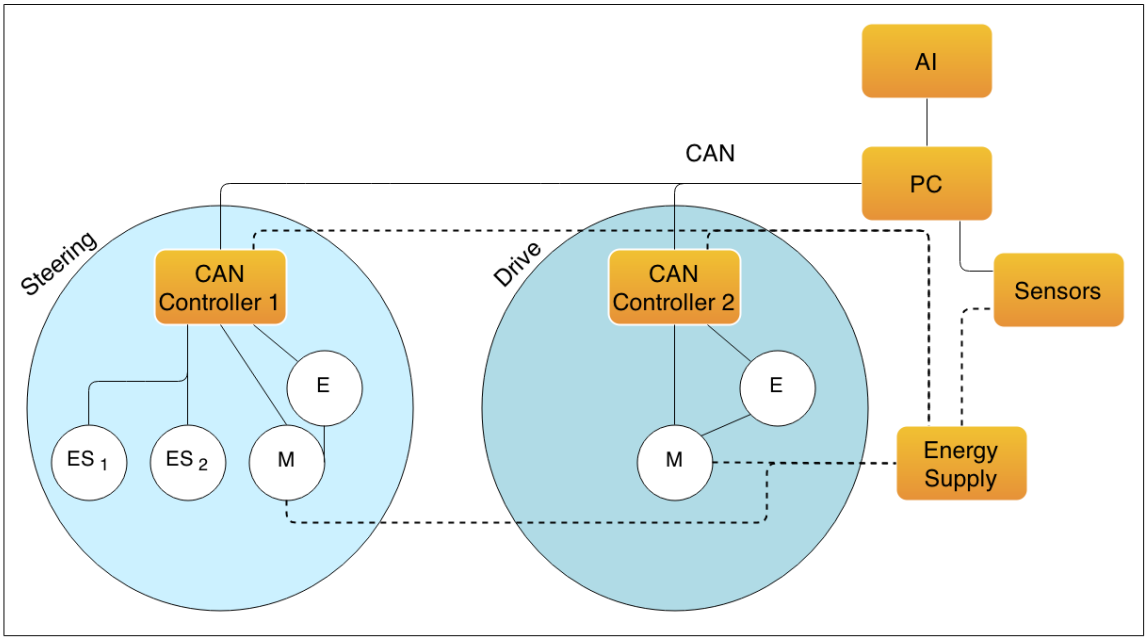

Figure 5.1: The diagram presents the overall overview of the system. AI denotes the artificial intelligence, processed on the PC. The communication to the motors (M) is established via CAN to the CAN Controller. E denotes the encoder of the control loop and ES the two installed end switches. The continuous lines show the communication connections whereas the dotted lines show the basic energy supply.

back wheel is motor driven to provide translational movement. Furthermore a second motor steers a rotational axis on which the first motor and the wheel itself are fixed. As a result this wheel is able to perform translational movement from 90 degrees on the left to 90 degrees on the right. Hence the wheel is able to rotate freely with one additional constraint, that the 105 degrees are not allowed to be exceeded. This has to be guaranteed, because of the design of the chassis, which would result in a collision between the rotating wheel and the chassis. In order to minimize this risk the 15 degrees are taken as a cushion. Furthermore it also benefits the implementation of the steering approach, as this constraint of a maximum angle of 90 degree, redundancy in the sense of motion is diminished.

Induced by the described alignment, the system reassembles to a slight modification of Ackermann steering geometry as motion model which will be analyzed in Section 4.2.3.

Due to the fact, that the high-level control of the autonomous forklift has to be performed and calculated on a mobile computer, a space for a notebook has to be established in the back. Additionally a space for batteries, as source of energy, the controllers and the proper

(a) Front view of the forklift,

(b) Back view of the forklift,

Figure 5.2: The front and the back view of the forklift are presented in this figure.

wiring is required. In order to maintain short distances, regarding the wiring, both places are arranged in the back, where they would further not interfere with sensor devices.

For safety reasons two switches are installed to separate devices from the energy source. One is realized as plane safety or emergency switch cutting down solely the power supply of the two motors. The duty of the second switch, which is construed as main switch, consists in the separation of batteries from the system.

In the front an additional, electrical driven linear axis was installed to be able to lift certain loads to demonstrate the functionality of an forklift. As the realization of the forklift was planed exclusively for demonstration purposes, the linear axis was small-dimensioned with a maximum payload of 50 kg. For the application purpose, described in section one, this axis should be able to lift up to 650 kg, in the demonstrative realization a scale of about 1:2 was selected.

5.1.2 Motor

As mentioned before the economic aspect was an important part of the component choices. Furthermore the motor should fit the requirements in consideration of torque, acceleration and maximum velocity. As a good trade-off between economic prize and fitting of requirements the SWMV 402340 from Ott [1] is chosen both for translation and rotation. This model consists

[1] http://www.ott-antriebe.de/az/azswmv.html

of an electrical motor based on DC including a gearbox. The transmission ratio is 46/1. The maximum current is limited with 8.5 A.

5.1.3 Control

Controller In order to maintain access to the motor's servo regulator, also called controller, is needed. Again there are some predefined requirements to make sure the system could communicate with the encoders mounted on the motors. First of all the robot system was designed including two identical motors providing rotations and translations. Hence it had to be considered what kind of communication scheme could fit at least two motors at once.

In addition the linear axis of the lifting fork in the front has to be considered too and maybe future work would provide even more controllers. As a result it seems obvious that only a bus-system would fit an unknown, but nevertheless finite amount best regarding in addition simple realization as well as economic costs.

Due to the fact, that the combinations of the Ott motors with there own controllers are well established, the choice of the motors carries weight in the decision of the bus-protocol. Therefore their I-CAN AMI1060-01 [2] servo regulator is a good solution considering the choice of the motor. Because this servo regulator is using the Control Area Network (CAN) (c.f. Section 4.1.8) standard for its communication, this controller fits the all mentioned requirements for the robotic system as well as the electrical demands of the controllers regarding the electrical supply from the system.

Unfortunately Ott does not provide any Linux support for their controllers, only for Windows. As a result, a new plugin must be implemented to provide control of the motor behavior through the Linux-based computer system (c.f. Section 5.2.1). The controller supports, as it is denoted in the documentation, the CAN-Open Standard Proposal DSP-402 [5].

Electrical Wiring To be capable of supply all devices the power source has to provide 24 V direct current (DC) and 12 V also DC. In Table 5.1 all devices are listed, which are used, with their demanded input voltage, maximum currents.

Because of the safety switch two circuits are generated independent of each other. One circuit connects solely and directly the two motors described in Section 5.1.2 with the batteries. The second circuit contains a 24 V voltage stabilizer and 24 V to 12 V converter. Due to the fact that the controllers and the sensors are usually more sensitive to voltage peaks and therefore vulnerable to errors or even destruction.

[2] http://www.ott-antriebe.de/motorsteuerungen.html

Table 5.1: The table presents the required voltages for all used devices.

Device	Voltage demanded [V]
Motor	24
Controller	24
Laser Scanner	24
Kinect	12
Asus	5 (USB)

In order to maintain a proper overview a hat rail is attached. The circuit diagram in Figure 5.3 visualizes both circuits.

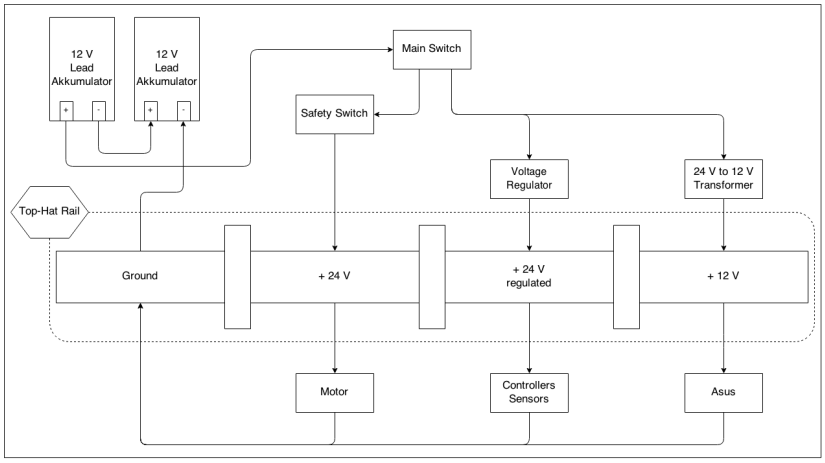

Figure 5.3: Shows the wiring connecting all devices with the batteries including voltage transformers and regulator.

Both circuits are attached to a main switch which separates the supply, via two times 12V lead cell battery connected in series, from the devices. From there two branches fork away.

The first supplies directly the motors trough the slots designated as 24V on the top-hat rail and is provided with a safety switch. As the name suggests this switch is needed to shut down in the case of an emergency.

The second circuit connects the battery on the one side with a voltage regulator and on the other side to a 24 V to 12 V DC transformer. As can be seen in Figure 5.3 the voltage regulator supplies the sensible devices such as the laser scanner or the controllers. Although the accumulators should only vary between 14 V and 11 V, therefore 22V to 28V range of variation, which would be within the designated boundaries of the data sheets of the devices. Nevertheless the possibility exists that voltage peaks or external influences could

cause damage. The transformer transforms the 24 V to 12 V to power the Asus Xtion Pro Live with energy.

All devices are connected using the same ground.

Controller Wiring The Controller, as previously mentioned in Section 5.1.3, has to be provided with the required wiring for the electrical supply and the communication to the computer, motors and the end switches of the back wheel. In the data sheet [3] of the the Ott controller the proper wiring is described including a schematic. Except for the end switches, which are only connected with the rotational motor controller's digital inputs 0 and 1, the wiring of the two controllers is identical (c.f. Figure 5.1.

Communication with the Computer As can be seen in the data sheet of the controller [4], it is sufficient for the CAN communication to solely use ground (denoted as GND), high and low voltage. In the data sheet they are called GND, CAN High and CAN Low.

Communication with the Motors The communication with the motors is realized over an predefined communication bus of Ott. Mounted on the motors are encoder which propagates the commands of the controllers (received through the CAN-bus) to the according motion of the motor.

Communication with the End Switches Due to the fact, that the motor encoders have no memory and therefore solely relative positioning depending on the starting position of the motor, it is urgently needed to handle the absolute position for the rotation of the steered and driven wheel from the software side. Hence two inductive end switches are mounted, one on each side of the robot as can be seen in Image 5.4.

The switches were initially planed to indicate 90 degree if we assume null degree would be the constellation where the heading direction of the mobile forklift is pointing alongside the x-axis. These +/- 90 degree would be sufficient for the mobile robot to be capable of rotating in place.

As the mounting of the end switches results in an angle of 97.2 degree the primordially concept of using these end switches as the detection of the maximum modulation has to be discarded. Alternatively we assume the position on both sides as the absolute value of 97.2 degree and initialize the rotation of the back wheel every time the robot has to be restarted

[3] $http://www.ott-antriebe.de/pdf/deutsch/steuerungen/AMI1060-01\_ICAN\_0509.pdf$
[4] $http://www.ott-antriebe.de/pdf/deutsch/steuerungen/AMI1060-01\_ICAN\_0509.pdf$

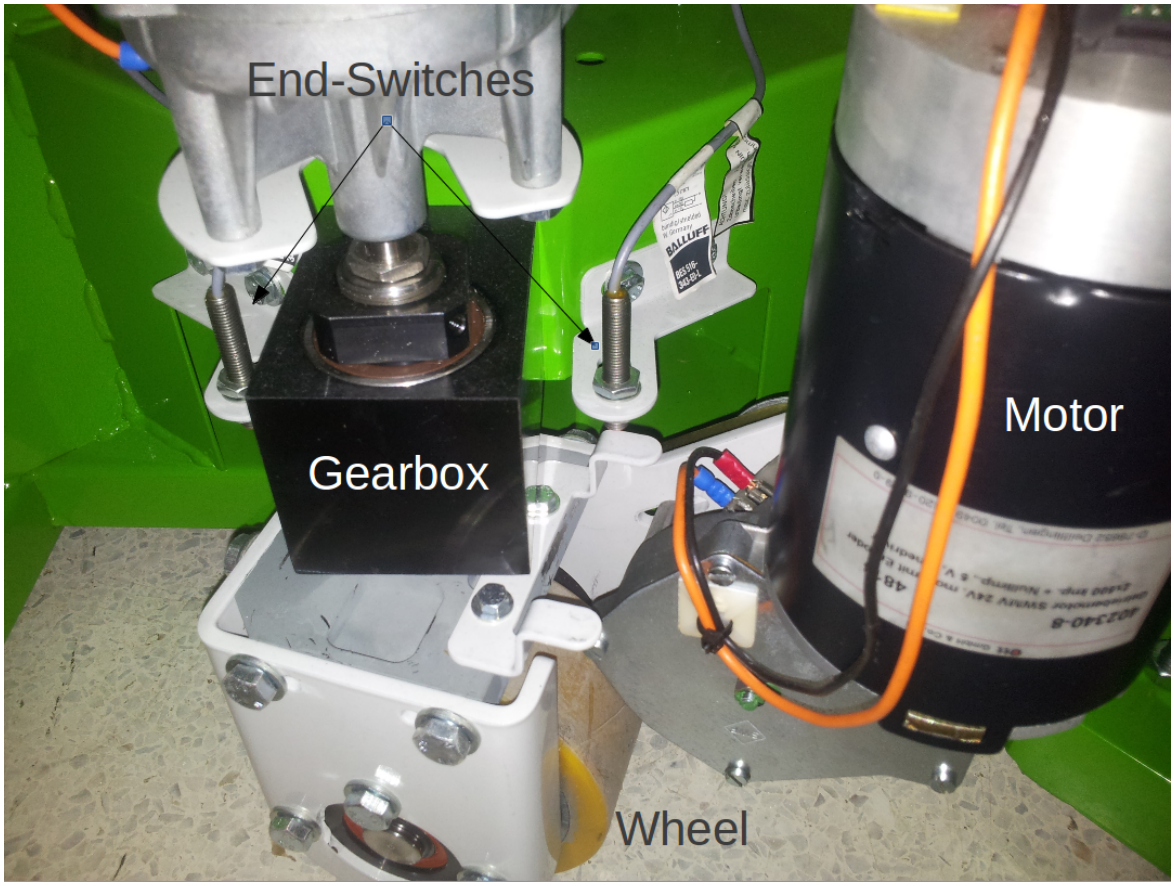

Figure 5.4: The picture displays the assembled end switch (marked with a *Baluff*-flag) pointing down, ready to detect the end switch blade mounted on the rotating backwheel.

or an controller error occurs, which also causes an restart of the controller to ensure reliable performance. This procedure is quite time consuming with an rotational speed both sufficient in accuracy and as fast as possible it takes about 48 seconds, depending on the initial position of the back wheel. Hence this position is not known and one has to attain it through an initializing routine, which will be described further in Section 5.2.1. The Controller offers certain digital input pins, which further be addressed and read out via the CAN-bus (cf. Chapter 5.1.3). The provided inductive end switches respond to the small metallic plates mounted on the rotating back wheel. Unfortunately a lack of stiffness of all the assembled parts result in an uncertainty of less than half a percent, or as it is measured by the encoder in about 150 increments, where 180 degree correspond to 36800. These values are heavily depending of the slip of the floor and the quality of the adhesion of the back wheel.

Communication Bus The description of a bus system has already been given in Section 5.1 and also the more specific realization of the Can-bus. The diagram in Figure 5.5 shows the path from the USB port of the computer connected with the PEAK USB-CAN-BUS adapter to the Can-bus and therefore to the encoders of the motors.

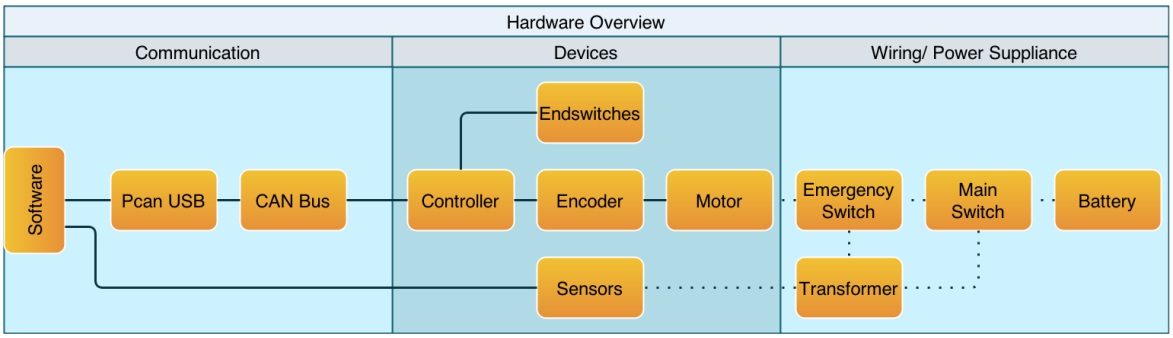

Figure 5.5: The chart depicts information about the dependencies of each device, including the interface to the software and the batteries as power supply.

PCAN USB For the computational unit of the system, currently consisting of a notebook, a communicate with the CAN-bus has to be established. Several commercial realizations are available, nevertheless we chose the PCAN USB by Peak System [5]. The PCAN USB adaptor connects the CAN-bus with the USB port providing hardware driver.

5.1.4 Sensors

The system consists of two types of sensors. First of all the end switches (described in Section 5.1.3) and sensors to perceive the environment. Most of the state-of-the-art systems rely on laser based sensors, because of their reliability considering accuracy, processing speed and robustness. However there are a broad variety of sensors , which are described in Section 4.1 and even more possibilities in combining them or at least aligning them in different ways. The following section focuses on already developed and published systems, which could meet the requirements for the autonomous forklift robot. In addition such systems are described and relying on published papers are compared regarding their performance. Furthermore the matter of the technological and economical feasibility is discussed.

The progress achieved by research and development is significant, hence there is no claim for completeness of the investigation of existing system designs within this thesis. But to start with, all relevant sensor devices are presented and discussed.

[5] $HTTP://www.peak-system.com/PCANUSB.199.0.html$

Sensors can be separated into several different kind of classifications. To keep it simple, the first classification primarily covers the method of measurement:

- sensing remote
- sensing with physical contact

As a matter of fact the sensors are used primarily for localize itself, detecting obstacles and further avoid collisions. Hence sensing with physical contact could not be realized as prime sensors. With respect to the physical properties of the forklift with a weight of more than 75 kg unloaded, the deceleration process would be inevitably to slow as to not collide with an obstacle sensed through a physical contact sensor. As a result the focus is on remote sensing, meaning measure without physical contact and not only the modern usage of the term as aerial sensor technology.

Laser Based Systems As stated above laser-based systems form the majority among the mobile robots in the scientific sector [1, 55, 49, 32]. Due to the high accuracy and the available products including industry standard especially expansive prototypes are equipped with laser scanners. As mentioned in Section 5.1.4, many different products exist as well as several possibilities of aligning one or even more scanners.

- *Valodyne*
  The paper of M. Leingartner et al. [32] focuses on the usage of a Valodyne and denotes as well as paper [38], the quality of this system, although particularly [32] prove, that the best sensor is useless if the appropriate software reaches its limits. Nevertheless there are arguments voting against the utilization of a Valodyne. The main argument is the price but furthermore the rotating laser would require a special mounting not provided.
- *RotatingLaser*
  Another approach initially appointed consists of a rotating or at least swivel-mounted like in the paper of Tatoglu et al. [51]. Again the impressive results accompanied by the potential opposes the computational effort exceeding the needs of system. Although the cost would be reduced compared to the Valodyne the trade off of additional uncertainty caused by the not yet tested and evaluated usage of a moving component has to be considered.
- *2DLaserPlane*
  In order to extract 2D information, commonly, the laser scanner is mounted parallel to the floor. Hence a planar plane is formed which indicate objects and obstacles [1, 55, 49].

Although the problem arises, that obstacles missed if they are above or below that plane, this method is highly efficient. Besides overlooked features the reflection and resulting interferences form the arguments against this system.

Multi-Camera System Many approaches tackle the problem of navigation using at least two cameras. As state-of-the-art algorithms are capable of computing feature tracking in real time, such systems have to be reconsidered too. The computational effort is nevertheless not neglectable and has to be considered. According to Christian Häne et al. [20] the navigation task regarding a stereo depth map is feasible.

Kinect/ Asus The main advantage of these camera systems consists in the maintained sensor fusion and the economic costs as well as the resulting high usage (e.g. turtlebot [6] which is one of the main representatives of ROS) in economic, robotic realizations or systems regarding actuators which interact with the environment. There is already a broad spectrum of implemented software targeting the Kinect. Nevertheless the cheapness has its reason and the sensor system is susceptible against disturbance and interference. Another disadvantage is implied by the aperture angle both vertical and horizontal.

Radar The evaluation of papers targeting systems operating in, interacting or with perceiving their surrounding, also resulted in sensor systems not regarded at first hand. Although not commonly used in state-of-the-art robotic systems the researches of Frederik Sarholz et al. [45] give the occasion to conjecture that the quality of the estimation especially regarding moving objects in far- and mid-distance is remarkable. This is also denoted in the paper published by Damien Vivet et al. [53] where the requirement to the sensor system include high speed movement of a boat. On the other hand, the data acquired beneath a certain threshold of about 1.5 m depending on the sensor, the reliability diminish [7]. Nevertheless the system offers also advantages as stated in the papers, including detection at high speed and in dusty or smoky environments. With that in mind an the costs which range within the reach of a laser scanner this system was rejected.

Time of Flight (ToF) and ToF fused with high resolution cameras Relying on the paper of Gandhi et al. [18] the point cloud constructed from the use of a ToF camera is not sufficient for a high resolution map. Therefore the proposal is made to use a high resolution camera system additionally to add feature tracking to SLAM to improve the behavior. The quoted experiments of [3] reveal poor results using solely the ToF. Nevertheless we must not neglect

[6] *http : //www.turtlebot.com/*

the recent development regarding the sensor. Depending on the description in both papers, the used ToF systems had resolutions about 144 X 177 pixels. As shown in Table 4.1, state-of-the-art sensors are much more potent. Unfortunately we have no information on the real time computation and the quality of the high resolution concerning test conditions.

Single Camera Due to the constant improvement in the processing power of systems maybe even relatively old approaches should be at least reconsidered. For example "Real-Time Simultaneous Localization and Mapping with a Single Camera" by A Davison [10] denotes quite impressive results, although the requirements of this thesis overtop the paper's.

RFID Another interesting approach is obtained by Herianto et al. [21]. The paper denotes a opportunity of a system which is neither be dependent on light conditions nor on reflecting surfaces. Through the establishment of a so called pheromone system consisting of RFID tags embedded into the ground, the robot, provided with an RFID tag reader, is able to navigate. Although the experiments presented seem promising and the interference of natural environments can be diminished, there are two major argument against the system. First of all the system is only capable of navigating in an already tagged environment, hence the obstacle avoidance is aggravated for expected encounters and almost impossible for unexpected collisions. Whereas this could be dissolved by the implementation of a system fusion.

Furthermore, the effort of tagging the whole environment could evolve enormous. In spite of the promising basically idea, regarding the realization with the mobile forklift the arguments against this system prevail.

Overhead Camera Systems Another system design initiated by the industrial partner comprises a multi camera alignment mounted on the celling, surveying the robot and its working environment. Ziaei et al. [59] focus on a similar problem definition. On the one hand the problem definition not include the obstacle avoidance and the planing is limited to fixed way points, on the other hand everything is realized with only one camera mounted. Nevertheless the principle task of navigation should be feasible. Further in [43] Cameras by Rao et al. extend the problem definition and present a solution for navigating in a known environment.

Arguments against the system constitute occlusions evoked by obstacles entering the scene (for example a transporter) and the overhead on communication necessary. Following the company future applications for the forklift could include more than one platform which enlarges the communication overhead even more.

Discussion Due to the fact, that this thesis has limited resources both budgetary and temporally the design decision could not merely focus on the plain technical aspect of the sensor system. Additionally, the costs of the sensor and the existence of reusable software according navigation have an influence on the evaluation for the system. Nevertheless Table 5.2 provides information important for the decision on the used sensor system.

Table 5.2: Table of relevant aspects reconsiders for the choice of the sensor system. ToF denotes Time of Flight and RFID respectively Radio Frequency Identification. The costs and the usage have to be considered with respect to the time of the creation of this thesis. Moreover they are depending on the quality of the product. As this thesis uses ROS Groovy the available code targets the existence within ROS or possibility of integration.

System	Sensor	Cost [$]	Available Code	Usage in Navigation	Availability
Rotating Laser	Laser Scanner	>1500	Not found	uncommon	Yes
Planar Laser	Laser Scanner	>1500	Yes	common	Yes
Single Asus	Asus Xtion Pro Live	120	Yes	common	Yes
ToF Stereo Fusion	ToF and Stereo Cameras	>800 + >250	Yes	uncommon	Yes
Single Camera	Camera	>150	Yes	rare	Yes
RFID Pheromone System	RFID Tags and Reader	25	Not for ROS	rare	Yes
Overhead Camera	Multiple Cameras	depending	Not found	uncommon	No
Radar	Radar	600	Yes	uncommon	Yes

Concerning the overhead camera systems and the RFID tag based system the neglect was caused mainly by the desiccation, that the forklift should be feasible to navigate on its on not relying on extern systems. The radar as primary sensor was rejected as the benefits compared to a laser scanner consist in the velocity and the range. As both components are of no such importance as to counterbalance the near-distance behavior and the rare utilization of this sensor we decided prefer the laser scanner to the radar. Regarding the ToF the acquisition costs as well as the resolution were a clear vote against this sensor compared to the former Kinect 360 or the Asus Xtion Pro Live. The stereo camera system was rejected because of the

fact that the Asus Xtion Pro Live implements a very similar system and moreover is cheaper. Additionally as Jan Smisek 's et al. [48] comparison indicates the quality of geometric errors according feature recognition is pretty similar due to the fact that high resolution cameras where chosen. According to the variation of the laser based systems, the decision was made to reject both the Valodyne as well as a rotating laser. The necessity of efficient navigation and obstacle avoidance could also be granted by a planar laser scanner despite of the previously mentioned obstacles which are solely beneath or above that plane.

In the end the decision is narrowed down to the Asus Xtion Pro Live or the laser scanner. Although the price would favorite the Asus Xtion Pro Live, we at first hand could not guarantee to accomplish the task concerning all requirements. As the reviewed papers for all LiDaR-systems indicate, the tolerances according to the interference of non Lambertian surfaces and additional light sources can not be neglected easily. Furthermore the range in which the two sensors work properly differ. Whereas the Asus Xtion Pro Live is only capable of providing reliable data between 70 cm and 4.5 m, the laser scanner range extends from about 20 cm to 8 m. Additionally, as mentioned in Section 4.1.4, the accuracy of the Asus is far beneath the industrial standard of the laser scanner. Nevertheless we conclude to use both systems independently especially because they could both be mounted without interfering with one another and both sensors are available. So the final decision of the sensor choice was postponed to the empirical evaluation of Chapter 6.

5.2 Software Implementation

The system design of the software comprises three major parts. Constituting the interface to the hardware, the *low-level control* (c.f. Section 5.2.1 for more details) includes the all software regarding the CAN communication and the initialization of the hardware. Above the low-level control the *executive* (c.f. Section 5.2.2 for more details) is responsible for the path execution regarding the adoptions, which have to be made according to the motion model, as stated in Section 4.2.3. Furthermore the computation of the data forwarded from the hardware (as well sensor data as motion feedback) are processed and propagated to the high-level control. The last section, high-level control (see Section 5.2.3 for more details), contains the artificial intelligence for controlling the robot. This section targets the planning task and the control of the path execution.

An overview over this concept is presented in Figure 5.6.

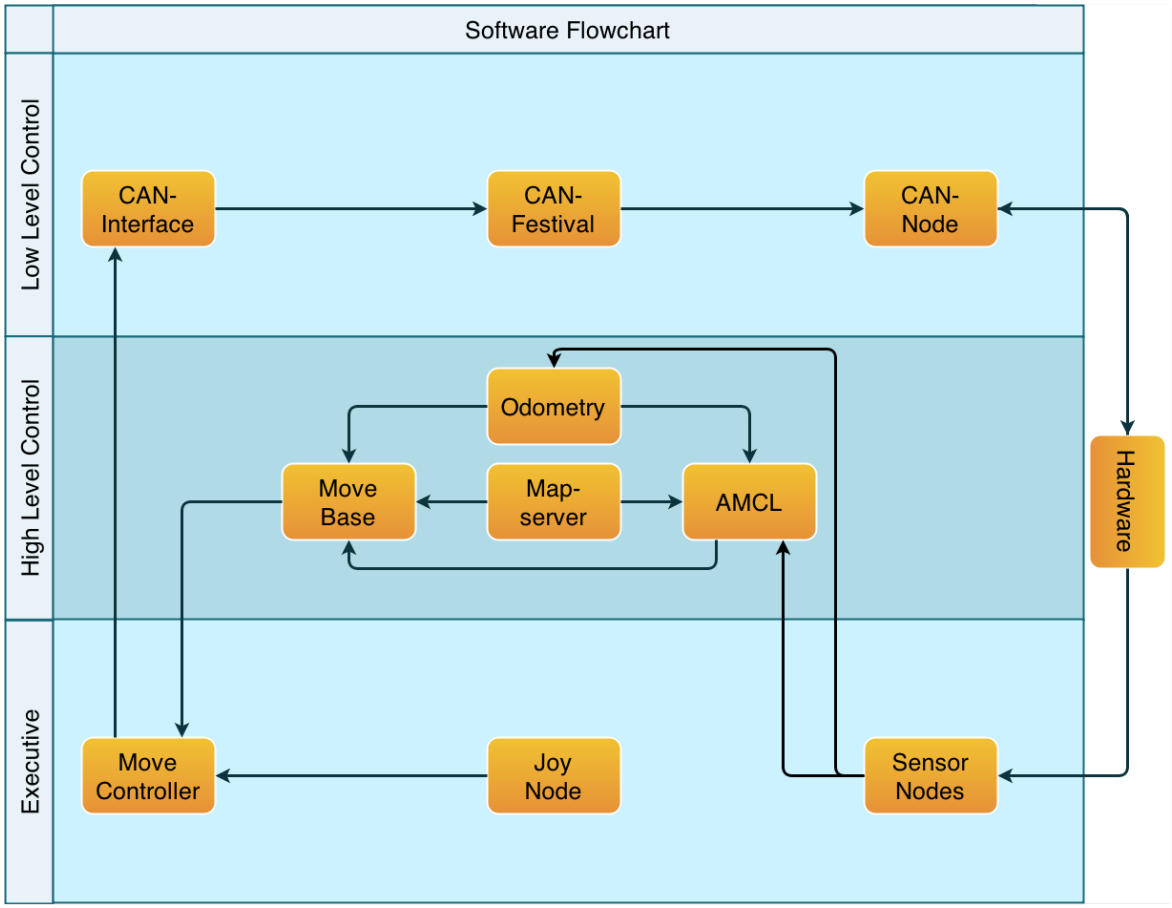

Figure 5.6: In this figure a overview of the software concept is presented including the connections and the categorization.

5.2.1 Low-Level Control

The main purpose of the low-level control consists in the communication of the software with the CAN-bus. Figure 5.7 depicts the structure of the class hierarchy of the CAN-related software. The upper most block CAN Interface, consisting of the $CANinterface$ (c.f. Section 5.2.1) and the two implementations of the standards required by the motion controllers (see Section 5.1.3), is designed modular. It can be substituted by other standards and CAN interfaces addressing other controllers through the CAN Open block. The Can Open block is divided into two classes, the $CANFestival$ (c.f. Section 4.3.3) and the $CANOpenClient$ which is discussed later in this section. It offers the fundamental packaging routine needed for the communication over the CAN-bus. It is designed modular, hence as well the standards defining the specification of the communication, as well as the CAN node forwarding the packages to the CAN-bus replaceable for the use of other realizations.

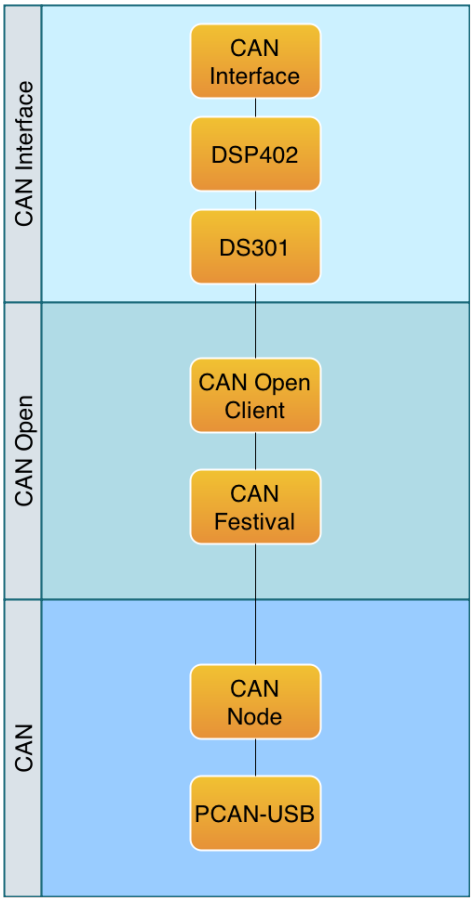

Figure 5.7: This diagram offers information about the class structure of the CAN software used to establish a communication between the software and the controllers of the motors.

CAN node The CAN node driver embodies the interface providing the communication between the software of the computer and the controller. In this project the PCAN-USB is used to transport the packed messages via the USB port of the computer, the PCAN-USB device and the CAN-bus to the motor controllers. The node is solely pipelining the message packages created by the CAN festival and sends it with the appropriate header informations to the PCAN-USB device.

CAN Open Client The CANOpen client provides an interface for the use of the CAN Festival (c.f. Section 4.3.3) for the higher interface of the CAN-Interface. As the provided sole propagation of the packages, sent and received over the CAN-bus does not suffice the requirement of the system, the information has to be processed according to its data type

and content. Furthermore two different routines for sending packages are provided, one following the full handshake routine provided by the CAN Festival and one only waiting for the release of a so called communication line. This communication line is used to establish a communication to one specific object dictionary entry. The line is maintained as long as the process of communication is still active and shut down if all data has been transmitted from both sides. The quantity of the communication lines is limited.

The first sending routine is recommended for settings on the controllers as the system waits for the response of the controller. The second one is recommended for fast transmissions (e.g. velocity commands) as the controller is able to propagate these commands fast enough without changing his own configurations or behavior.

Can Interface Defined through a standard, the communication through the CAN-bus has to obey certain rules and procedures. The range of offered realizations is broad and although they all share the fundamental basics they differ in a considerately large way. Due to this we will solely present the actual used standards, which are predetermined by the programming of the used Ott controllers.

Using the DS301 as communication structure the communication is based on a sender receiver model. It is established either confirmed or unconfirmed. In the literature this is often called communication with handshake or without. According to the standard [6] the interaction with the end device (motor) is called device profile and includes the a communication object, the object dictionary and the application.

The application is denoted as the action performed at the the end device as the communication object is the sole principle of communication overhead caused by the bus. On the other hand, the object dictionary (OD) contains all relevant information concerning what information is delivered and where is it addressed to. Primarily the OD is defined by the DSP 402 standard [5] as the whole functionality of the device (motor) is structured there. DS301 [6] explains the basic communication as well as the fundamental initialization to provide any further transmission.

In our can interface this structure is also implemented using an interface to provide access to the functionality needed to control a robot and to monitor its behavior. The DSP402 class uses the OD to access the demanded parameters and be capable to modify them accordingly. Beneath this class the DS301 class contributes the connection to the CAN-festival supplies the getter and setter methods for the desired parameters.

Implementing the aspect communication three different services can be used according to the desired modality. To access a CAN-node and initiate a communication a network management

(NMT) object is required. After the initialization the communication can either be realized via service data object (SDO) or process data object (PDO).

The SDO provides access to the data object through a service. As the data objects may consist of an arbitrary large data and may contain sensible data, SDO implies a reasonable quantity of communication overhead. Hence the communication via SDO defines every detail of the data object which should be accessed and either be read out or modified.

Basically the PDO resembles the structure of a SDO except for the communication overhead. Therefore it is mapped into another part of the OD when addressing the same object. As a result the PDO can only be used by communications where no flexibility is needed and everything is well defined. Caused by the lack of communication overhead the priority of a PDO surpasses the SDO's. Unfortunately the mapping of manufacture specific OD entries is not defined in the standard and not provided by Ott at first hand. As a result the implementation was realized using SDOs. Due to the fact, that the communication is still manageable at runtime priorities are still neglect able.

Parts of the OD used can be seen in the Table 5.3, whereby the first part denotes the standard and the latter part the manufacture specific part, which was provided by Ott. The index and subindex denote the location within the OD, the size and sign are used for the proper conversion of the packages sent to or received from the entries. Here it should be stated, that the standard comprises very soft requirements as the full functionality of the system does not need to use anything implemented by the standard necessarily. The only obligatory conditions are the defined space of the object dictionary. Obviously the base functionality of the system must be accessible by the defined object dictionary entries, but the manufacturer could implement new entries and solely use them to interact with the controller and the motors, which is contrived by Ott.

The diagram of Figure 5.8 shows the work flow of both rotational and/or translational movement in combination with a read request of the odometry, indluding a mutational wait caused by the entry into an restricted part of the software.

The DSP 402 dictates the implementation of different controlling modes. In principal they are position, velocity and torque mode. The torque mode is used to directly control the force of the motor. The velocity mode is used to define a certain number of rotations per minute relying on the predefined PID controller (software implementation on the controller). When using the position mode, a goal position (either through increments or any definable unit) is aimed using a certain predefined velocity.
These so called modes of operation are again set as configuration within the controller and

Table 5.3: The Table provides examples for the object dictionary (OD) used for the CANOpen communication. The upper part denotes the DSP402-standard (0x6), the latter part the manufacture specific part (0x3). The ID is only a string used for better understanding, instead of the hexdecimal values. The index denotes the location within the OD and the subindex adresses values which are combined in one entry. With help of the size and sign, the software is able read and write within the entries.

OD Entry ID	Index	Subindex	Size	Sign
STM_CONTROLWORD	0x6040	0x00	0x02	UNSIGNED
TARGET_POSITION	0x607A	0x00	0x04	SIGNED
MOTOR_POS_ACTUAL_INCR	0x6063	0x00	0x04	SIGNED
MANUFAC_VEL_DESIREDVALUE	0x3300	0x00	0x04	SIGNED
MANUFAC_MES_VEL	0x3A04	0x00	0x04	SIGNED

again the manufacturer is not strictly binded by the proposal of the standard. hence can implement adoptions of these modes.

Initializing Routine of the Controllers To start up the system obligates the user to a proper initialization of the controllers. Following the DS301 (c.f Section 5.2.1) a series of predefined messages according to a handshake protocol has to be transmitted to ensure duly functionality.

At first the system provides a broadcast over the bus to contact all possible clients or nodes. This is realized through an Network ManagemenT (NMT) message. The NMT objects are solely for the first access to new nodes.

Afterwards the systems state machine has to be triggered into operation enabled, which is the mode, where the full functionality is provided and no error has occurred. The state diagram of the state machine is described in Figure 5.9. The states shown in the figure are denoted as operation modes. As not all state transitions (arrows) are obligate, the optional transitions are marked with dotted instead of continuous arrows. Internally these states are saved into certain registers and as such the mechanism of setting these registers a bitmask called the controlword, which can be directly set trough and SDO message. To read out the current status one can either read out the controlword or the statusword, whereby the statusword consists of a slight variation of bits [5].

As mentioned before, when presenting the underlying standard, it comprises a large possibility to integrate factory specific object dictionaries and furthermore the opportunity to define one's

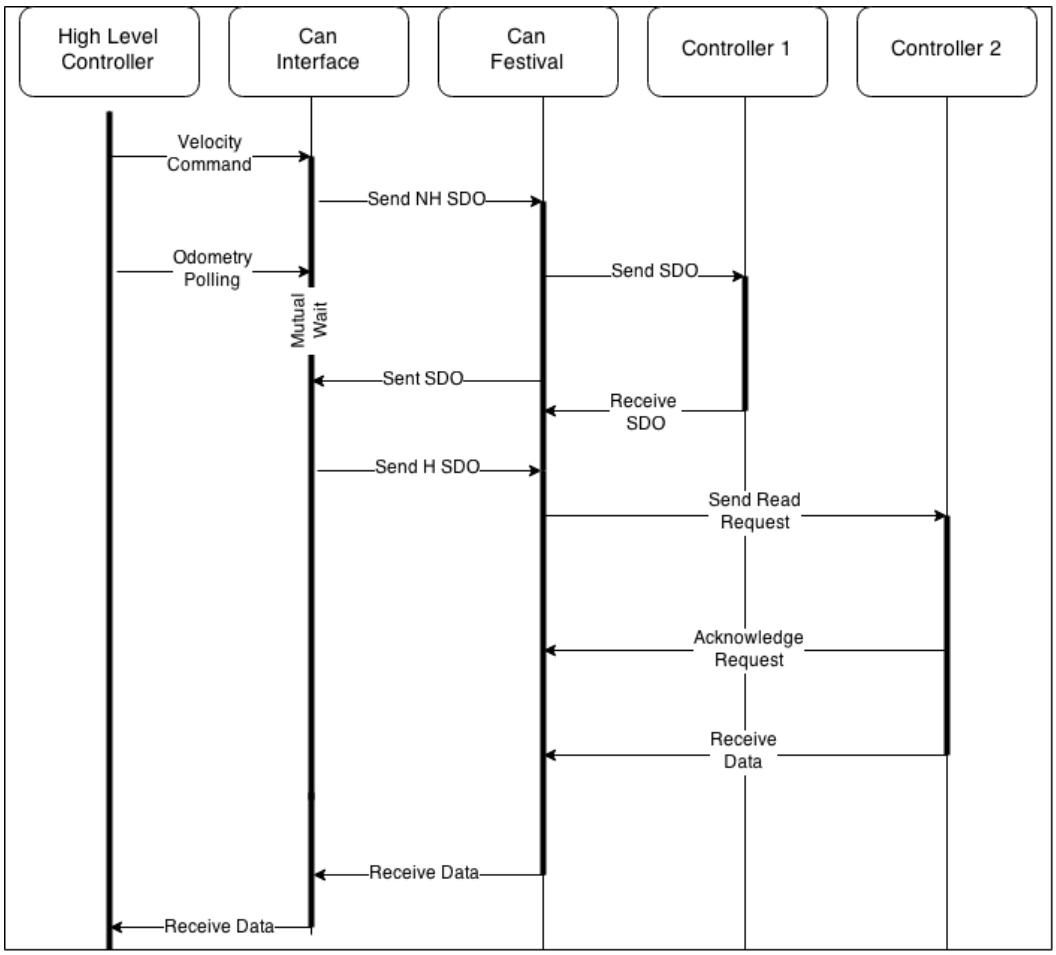

Figure 5.8: The diagram presents the work flow of read and write commands sent by the CAN interface, includind a mutational wait.

own initialization. Furthermore the standard does not insist on the triggering of every single state of the state machine. It is up to the manufacturer to define additional state transition, hence sometimes some steps can be skipped. Nevertheless the fundamental initialization has to be placed at the disposal too. Due to the fact, that this interface should be designed with respect to the re-usability providing an adaptable interface.

Rotational Initialization of the Back Wheel As previously mentioned in Section 5.1.3, the back wheel has to be calibrated adequately with respect to the end switches. At the work flow diagram 5.10 one can see the sequence of the initialization.

At first we have to determine the right end switch position. Due to the fact, that the position mode as such uses the provided positioning mode of the controller, and the absolute position

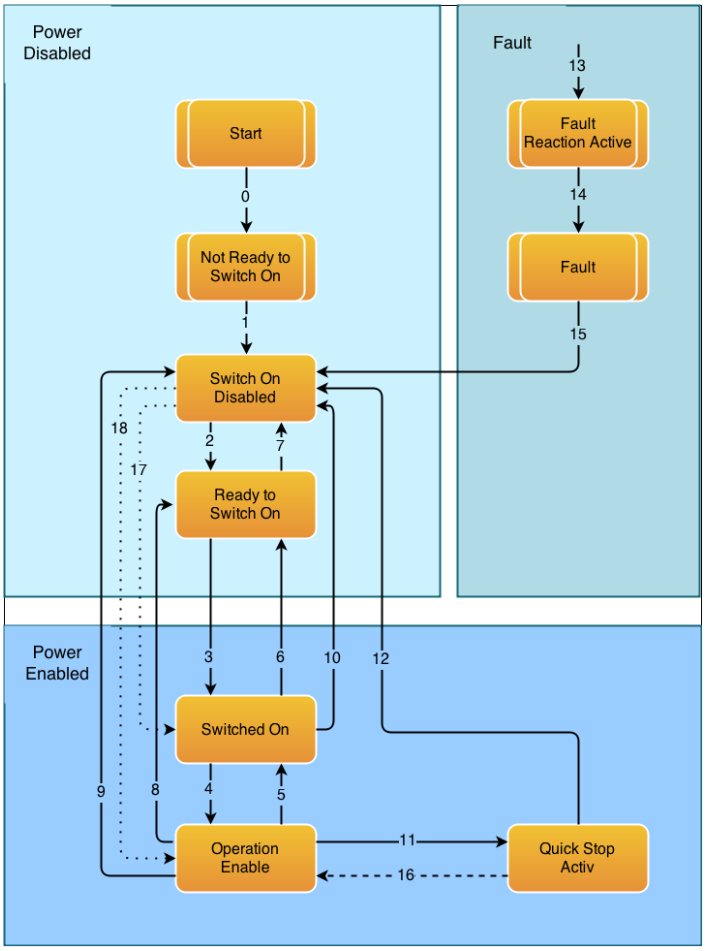

Figure 5.9: This diagram shows the statemachine of the DS301 including all necessary transitions marked with continuous arrows and optional connections marked with dotted arrows.

is unknown we start to rotate with the help of the velocity mode. The end switches are again addressed over the CAN interfaces object dictionary and a change is detected via polling. The controller would provide active interrupts but the proper handling within the Can-Festival is only implemented via blocking wait so far. The reaction time of the polling mechanism works pretty fast so the velocity can be set to a relatively high velocity for the moment. The only upper bound is represented by the assumption made, that the starting position of the back wheel lies between the boundaries of the end switches. Because the probability of an improper switch off of the mobile robot by means an back wheel rotational position outside of the end switches one has to handle even this scenario. If this is the case, the rotation continues until one of the following conditions is met:

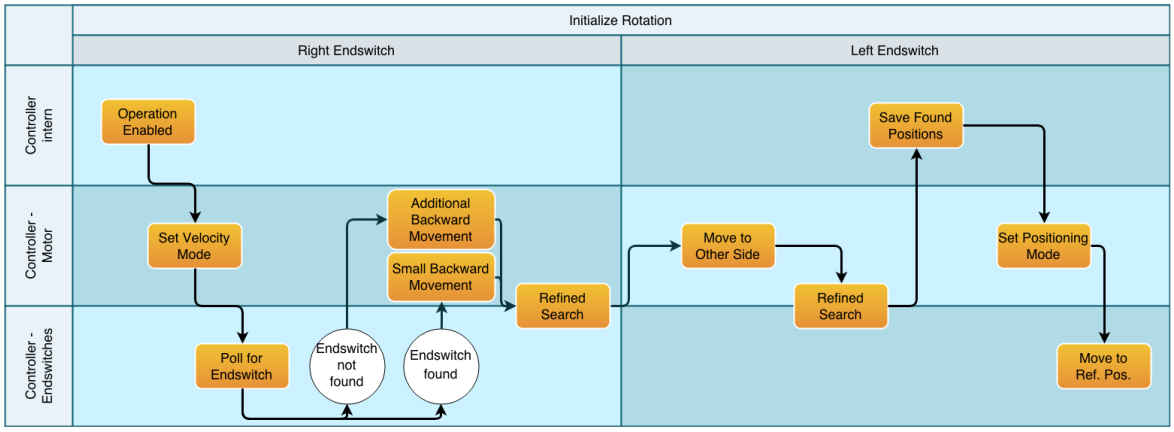

Figure 5.10: The workflow of the initialization of the back wheel controller is represented in this figure. The additional backward movement is caused by the out-of-bounds location of the end-switch.

- there is no incremental change of the actual position of the motor.
- the maximal current defined by the controller itself is reached
- the time is depleted in which the back wheel would rotate more than 210 degrees

Under normal circumstances one of these three would suffice, but the metal plate needed for the detection is the outer most rotational part, which collides first with the chassis. Hence the speed has to be set accordingly so that no physical damage in the form of deformation is caused to this plate. Furthermore the depletion of the rotational time and the maximum current could both cause enough force to that plate to deform it, but due to deformation the detection of the incremental change is also neither trustworthy nor sufficient. The resulting deformation decenter the detected middle position which is assumed to be null degree in relation to the base link, or simplified to the chassis of the robot.

So there are two cases deriving from the initial search for the right end switch. The first one contains the detection of the end switch which leads directly to the refined search. The other case will trigger one of the just mentioned conditions which lead to the knowledge, we are out of bounds and are therefore forced to rotate back to now as well be ready for the refined search.

The refined search has to be implemented on the cause of the time needed for the initialization routine in combination with the accuracy needed for the incremental search of the endpoints. In order to achieve the maximal accuracy many experiments had to be made to detect the best velocity. The accuracy problem arose at the first testings of the setup prototype. Due to

the fact, that the square path test achieved such inaccurate results the incremental search had to be optimized as will be explained in the just mentioned experiment chapter.

Assuming that the proper position is found we can now safe it accordingly within the CAN-interface and set the new velocity to detect the other end switch. As the distance between the two switches should not be fixed, the majority of the distance can be covered fast. And the refined search is started autonomous. If the left end switch can not be found in a reasonable time, the system is in faulty state and the initialization returns an error. But because the robot itself is not capable of handling hardware errors the initialization is aborted and the user has to manually fix the problem and restart the system.

If both positions are found, the mid position between the boundaries is set as reference position (straight heading) and the back wheel is moved onto the according motor increment. Furthermore through the manual measurement of the degrees of the detection point of the switches both the maximum steering angle is related to incremental maxima and increments per degree are calculated as base for the proper positioning mode rotation.

Finally the mode is transformed back to positioning mode, the velocity of the positioning mode is set and the current position is turned to null degree which finalizes the routine.

Motor Communication As mentioned before at the description of the CAN controller, the motors have several different modes of operation. The motor assigned to the translation operates via the velocity mode where a certain velocity is set. This is realized with the help of the SDO without using the handshake, which is sufficient because of the way the command velocities are propagated through the move base.

As described in Section 5.2.1 the reference position is at null degree, hence translational movement. The move controller (c.f. Section 5.2.2) transforms the velocity commands into a steering angle (between +/- 90 degrees). Then the CAN-interface transforms this angle into the appropriate increment and forwards it to the steering motor.

5.2.2 Executive

The executive embodies the bridge between the high level control and the low level control. It comprises three main tasks, all consisting of the processing of data transferred between those two levels. In the direction from the AI to the hardware, the move controller transfers the driving commands. In the other direction the feedback of the motors has to be provided through the odometry and the perception of the environment is preprocessed.

Move Controller In the early stage of the project, we designed and implemented this package for the plugin connecting the system to Gazebo simulation tool. The interface was well prepared to maintain the functionality of processing and further propagating the velocity commands of the high level control to its execution also after the simulator was exchanged with the real hardware. Due to the fact, that most state-of-the-art realizations of algorithms used within the AI is contributed to differential drive and as was stated in Section 4.2.3, the motion model of our robot is Ackermann steering geometry, the motion commands have to be adopted accordingly. The move controller takes the planned values of v_F and ω_F and transfers them to the proper velocity v_b and steering angle ϕ_b of the actuated back wheel. In Figure 5.11 one can see the full propagation and transformation of a velocity command through the AI to the execution of the motors. Starting with an velocity command consisting of a translational velocity (v_F) and a rotational velocity (ω_F) for the reference point or base link, sent from either the high-level control or the joypad. The move controller processes the velocity (v_B) and the steering angle (ϕ_B) for the back wheel according to the derivation of Section 4.2.3. Further the CAN-software transforms the steering angle to the according increments of the motor and the velocity to the rotations per minute. Finally the commands are propagated to the motors and the measured execution returns as a feedback to the controllers.
Furthermore one can see, that the velocity commands do not necessarily come from the AI, but can also be created by the joy stick.

Odometry Naturally the feedback of the motors while executing the proper velocity commands have to be recovered, processed and accordingly propagated to the high level control. As can be seen in Figure 5.11, the controllers get feedback from the executed commands. The rotational motor returns the increments denoting the steering angle, whereas the translational motor returns the measured rotations per minute of the motor. These values are then transformed, according to Section 4.2.3 to the motion model of the mobile forklift, for the base link (reference point). This motion is processed for the time, since the last update of the feedback of the motors and is further integrated over the time, since the start of recording of the odometry. The integration of the motion is then published as a transformation, denoting the relation of the base link in reference to the global frame (e.g. map frame). The odometry is published in quaternions (c.f. Section 4.2.1).

Sensor Node (Laser Scanner, Asus Xtion Pro Live and Kinect 360 Due to the fact, that the sensor data have to be preprocessed in order to suffice the requirements of the high-level control, each sensor is provided by a node including the drivers and algorithms. In our realization, the laser scanner has its one sensor node as well as the Asus Xtion Pro Live and the Kinect 360. All of the used sensors ROS implementations are available.

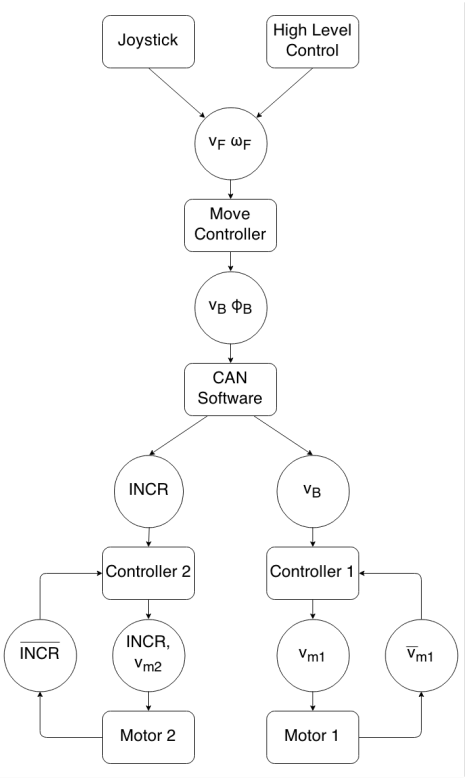

Figure 5.11: The diagram presents the propagation and transformation of a motion command created by either the joy stick or the high level control. INCR denotes the increments used for the positioning of the rotational motor. The variables in the direction from the motor to the controller denote the feedback.

5.2.3 High Level Control

The main purpose of the high-level control consists in the autonomous navigation. The navigation stack provides mapping, localization, path planing and its proper execution. The background has already been discussed in Chapter 4. In this section the software concept behind the individual parts is displayed and their connection. Figure 5.12 displays an overview of realization of the navigation task.

Mapping In the requirements for the system, the generation of the map has to be transformed a priori. As can be seen in Figure 5.12 the packages used for the mapping task are grouped. The grouped part of the navigation is not needed.
Our system relies on the SLAM algorithm Gmapping (c.f. Section 4.2.5), provided by the related ROS package. The parameter default parameter setting was used and the laser scanner

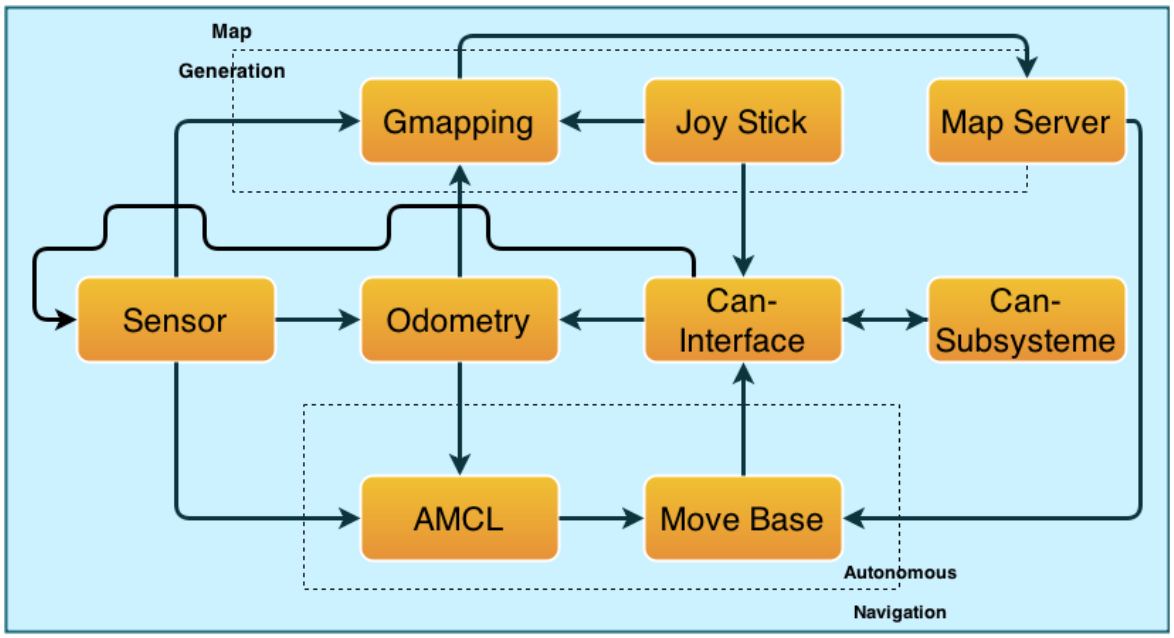

Figure 5.12: This figure presents the overview of the navigation task including the grouping of the mapping and the navigation task.

was chosen as the source for the perception of the environment.

To generate a map, the user has to manually steer the forklift using the game pad. With respect to the integration of the motion feedback of the motors, the algorithm assembles all sensor data and merges them into a map. A demonstration of a map built can be seen in the according bag file.

Figure 5.13 shows a generated map with Gmapping.

Localization The AMCL package implies parameters which are loaded from an configuration file and which are set when the package is launched. These parameters control the behavior of the localization algorithm. ROS provides a default setting for all of them which work out for the most scenarios. For this scenario these default values were adapted regarding the assumption the enhance the performance. In the following paragraph all important parameters which were changed and described including a short justification why they where changed.

Due to the fact that the computation of the AMCL is not exhausting the computational resources of the system the minimum random sampled particles are increased to 500. The distribution of these samples can be influenced by two parameters known as $alpha\_recovery\_fast$

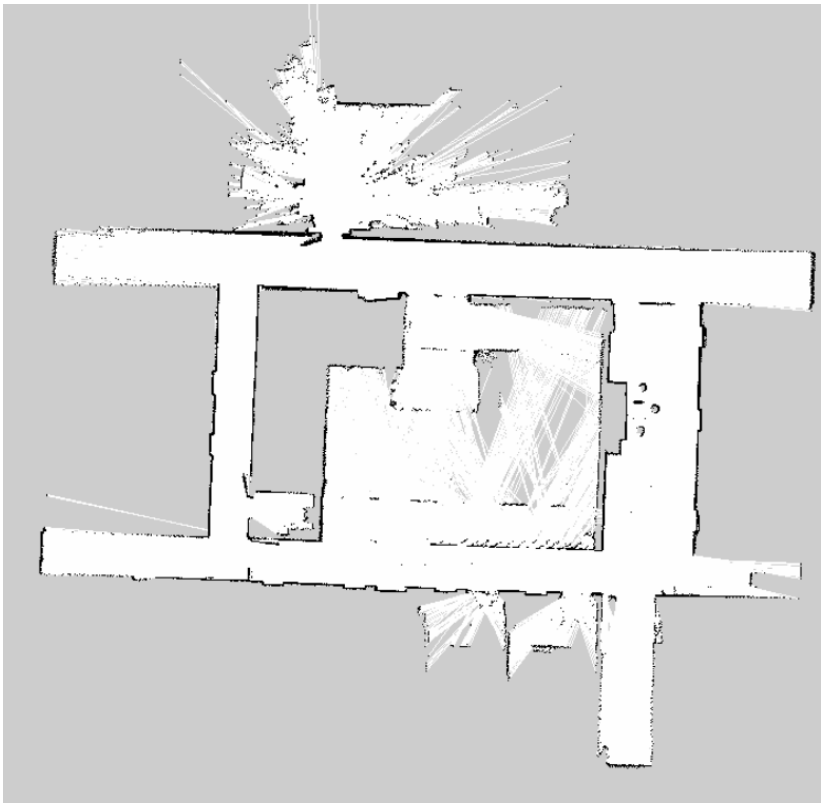

Figure 5.13: This figure shows a generated map of the institute. White denotes free space, grey denotes unknown space and black walls or obstacles.

and *alpha_recovery_slow*. As the name suggests they add more specifically samples to be capable of dealing with the kidnapped robot problem (explained in Section 4.2.5). If the *alpha_recovery_fast* value is set more particles are generated and therefore computed which are not in the immediate vicinity of the current pose of the robot. The more particles are placed outside the vicinity the better a robot can manage unexpected situations regarding the unsupervised change of its position (e.g. map jumps caused by sensor faults). On the other hand too many samples could cause an relocalization to an wrong localization caused by the influence of dynamic obstacles.

The initial pose has to be set including the covariances to facilitate the proper pose estimation at the beginning. This estimation is outsourced to the RVIZ tool where it is provided manually via an graphical user interface (GUI). Obviously a better initial estimation results in better localization results.

As mentioned before the AMCL's particle sampling is influenced by the odometry of the robot. This odometry is integrated using a motion model (c.f. Section 4.2.3) and although

the final motion model of the robot consists of Ackermann it can be narrowed down through the proper calculations to an differential drive. This is needed because AMCL as well as the move base are processing the odometry data regarding certain motion models. AMCL is only supporting differential drive and omnidirectional drive.

The full set of parameters can be obtained from the documentation in the ROS Wiki [7].

To sum up, the configuration of the AMCL is very important for a rather good and fast localization. As the localization in this case depends on the motion model regarding the odometry, no specific adoptions had to be made according to the kinematics of the robot. Thus solely the parameters, which define the weighting the reliability of the sensor and the reliability of the odometry, (alpha_recovery_fast and alpha_recovery_slow) had to be adjusted. Additionally the number of samples was increased and the intervals regarding the resolved motion has been reduced.

Move Base The move base is again a ROS package. Its main purpose consists in the task of generating a valid path from the current position and orientation to a user defined (or predefined) target pose and the proper execution of this plan. This process of collision avoidance can be roughly divided into two separate stages: a global and a local planner [16].

Both planners use so called costmaps. A costmap denotes the configuration space of the environment (c.f. Section 4.2.5) based on the current sensor data. Especially the local planner take rely heavily on such costmaps to avoid lethal collisions with obstacles. Figure 5.14 shows a vizualization of the generated map overlayed with the inflated obstacles of the costmap. These inflated areas can be gone through as the added value to the path is not lethal. Nevertheless, the algorithm tries to stay on free space.

The move base package includes default planners but also an interface to integrate new developments or adaption of the existing ones. The following description refers only to the planners used in the experiments and evaluations or planners which were discarded for particular reasons.

Global Planner The move base uses a global planner to generate a global plan based on the map. Depending on the realization the path is built up by the usage of search algorithms. As it is mentioned in the related sections some of these collision checks include motion models.

[7] *http://wiki.ros.org/amcl*

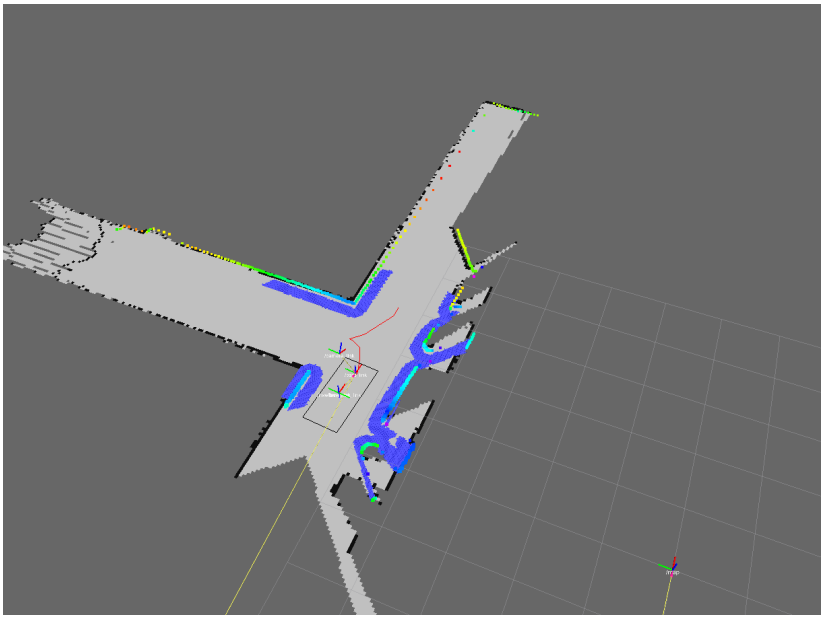

Figure 5.14: This figure shows a generated map overlayed with the inflated obstacles of the costmap. The laser scans (yellow to green) are inflated by a blue colour.

NAVFn Navigation function (Navfn) is the default planner used by the move base. To simplify the process of motion planing this algorithm assumes the robot to be circular. Furthermore the costmap is modified in the manner that every obstacle is extended by the radius of the robot. As a result, the robot can be further reduced to a single point (c.f. Section 4.2.5. With respect to these assumptions and modifications the planing phase is reduced to a minimum [19]. This global planner uses the Dijstra's shortest path algorithm [23] including a breadth-search first. Navfn is based on a potential field method (c.f. Section 4.2.5).

The extension of the obstacles is used as a compensation of not taking into account any other kinematics of the mobile robot than differential drive or omni directional. To meet the requirements of a differential drive robot, which constitutes the basis for the used realization, the radius is defined by the distance between the base link and the outer most part of the robot. Thus the costmap is extended by more than the full length of the robot, hence at least 91 cm (without safety buffer). As a result this setup is not feasible to navigate within narrow passages, or at least the radius has to be reduced accordingly, risking collisions.

Figure 5.15 shows the blocked path, caused by a higher inflation radius, which would guarantee a collision free path planning, without consideration of the kinematics of the forklift.

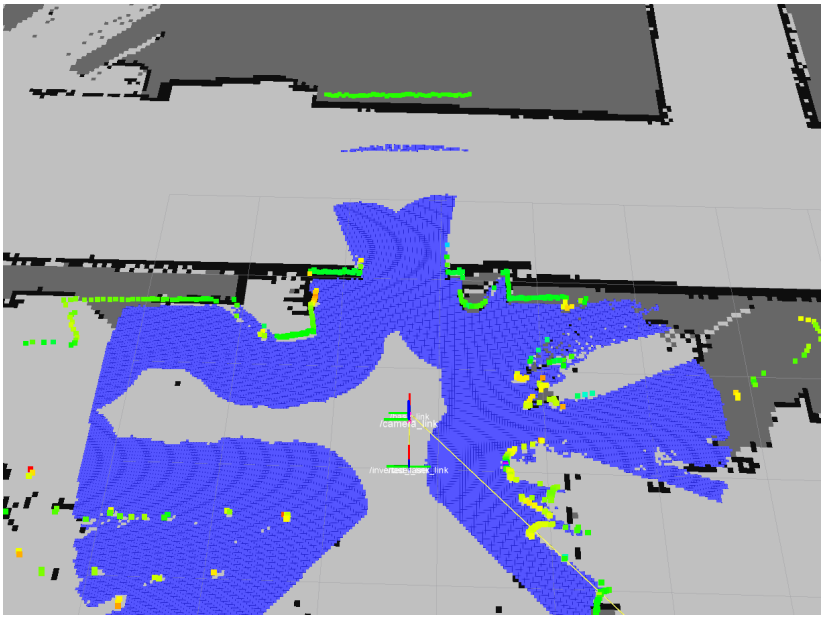

Figure 5.15: This figure shows a generated map overlayed with the inflated obstacles (blue) of the costmap. Due to the high inflation radius the path is blocked.

SBPL Lattice The search-based planner lab (SBPL) Lattice algorithm [34] [8] is a concept of building a path from the current pose to a desired goal by putting together predefined motion primitives (MP). Motion primitives are the trajectories which can be generated out of the motion models described in the previous Section 4.2.3. These MP are the principal component of the global trajectory planner Search-Based Planning (SBPL) (c.f. Section 5.2.3 where it is discussed further). Using search-algorithms to find a solution, the algorithm is taking the actual footprint of the robot into account which is a huge advantage considering the motion model and the appearance of the mobile robot we are using. The SBPL Lattice planner focuses on A Star (A*) based algorithms extended with special heuristics to improve them for motion planing. This planner is already integrated into ROS and is linked to an external website [9]. Following the informations posted on this site there are existing papers which will become public soon. The underlying concept is described in the journal [34, 35], but the search algorithms are not the same as in the package we are using.

The search algorithms can be varied and the package also provides an interface to add new or adapted implementations. By default Anytime Repair A Star (ARA*) and Anytime Dynamic A Star (AD*) are included which are both explained in the journal [35].

[8] *http://sbpl.net/*
[9] *http://sbpl.net/*

The motion primitives can be generated with the help of an Matlab script, which is available on the already mentioned homepage. This page also includes a manual how to modify the script accordingly to fit a motion model.

Figure 5.16 shows a plan generated by the SBPL Lattice Planner. The red line denotes a combination of the predefined motion primitives according to the kinematics of the robot. This figure is also used later in Chapter 6.

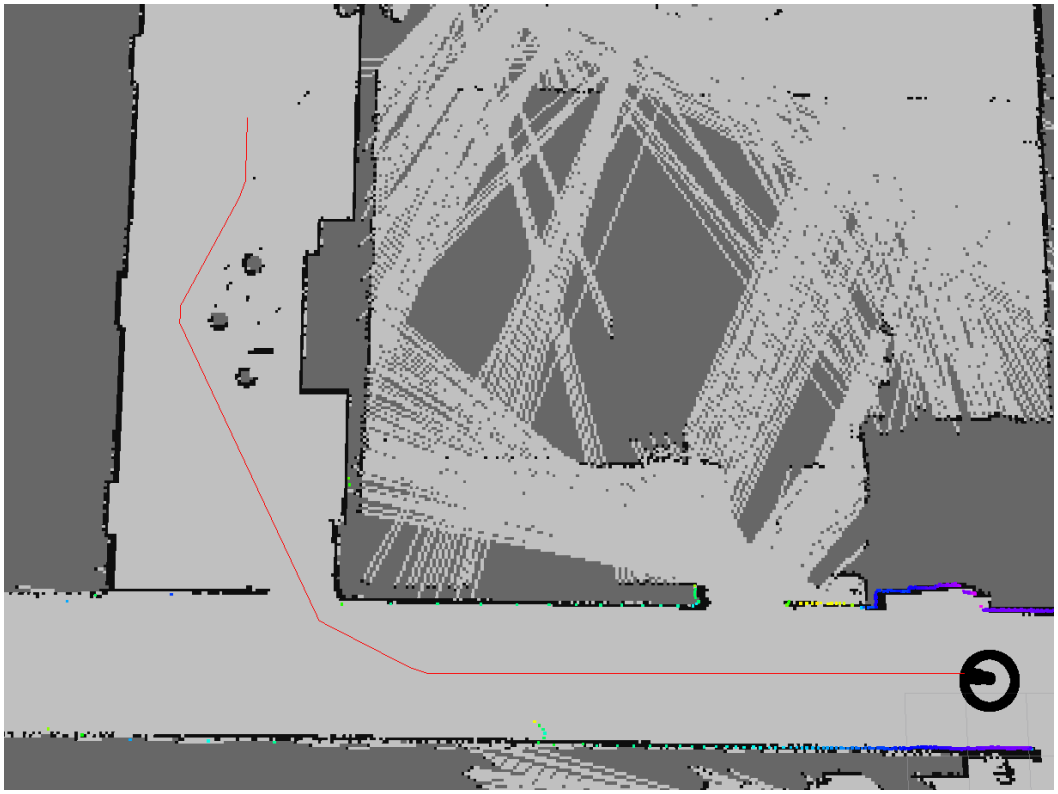

Figure 5.16: The figure presents a planned path pf the SBPL Lattice Planner. The red line shows the combination of motion primitives building a trojectory from start pose to goal pose, according to the motion model of the robot.

Carrot Planner Carrot Planner denotes a very simple global planner specializes on the adaption of the user defined goal configuration or goal position. It takes the goal and tries to find a feasible goal as close as possible to the desired goal, by solely propagating back the direct vector from staring pose to end pose. Due to this simplicity neither the kinematics nor the shape of the robot is taken into account.

As a result, this planner is discarded, because of the simplicity, as no kinematics are taken into account at all.

OMPL The Open Motion Planing Library (OMPL) [50] is an open source project. It is a Library for sampling based motion planing algorithm without restriction to particular collision checker or visualization tool. As a result it can be included into ROS. Unfortunately the standard framework is not yet included into ROS only into the MOVIT which is a motion planing tool designed for arm navigation [10]. Trough the provided ompl-app [11] one can easily try out different search-algorithms even with predefined motion models. Ackermann steering is as well included, although the search-algorithms for this kind of kinematics are limited to solely two versions. Both of them are complete with the trade off of being either costly regarding the processing time or delivering non optimized paths. The latter one include a large number of loops, which is hard to follow due to the kinematics of our robot.

The tool was used in the early stage of the experimental phase. Unfortunately, the arising path consists in the mean of too much slopes and turnings. The less movable kinematics of the Ackermann steering geometry and the inertia of the forklift result in marbled performance.

Local Planner Relying on the global plan, the task of the local planner consists of avoiding obstacles which block the existing global path. The path may be blocked because of a change of the environment or even a malfunction of the global planner itself. The local planner further generates the velocity commands in order to follow the path. As the local planner's main task is the obstacle avoidance of the nearby surrounding, the costmap, generated from the current sensor data (e.g. laser scans), denotes the main instrument. Due to the fact, that the avoidance include fast and accurate planning the range is limited as is the range of the costmap. Nevertheless the costmap is the a good memory source for obstacles not inserted into the global map, if for example the robot has to move backward without proper perception point in this direction.

Dynamic Window Approach The dynamic window approach (DWA) is taken into account the dynamics of a mobile robot [17] and uses therefore even the inertia of it [16]. Computing the collision for an dynamic window prevents the system of overtaxing the robot. An example would be a robot at full speed and a planner which sends sudden turning command. The inertia would cause the robot to over steer and a potential collision could be the result. A similar example is presented in the section experiments of [16] and [17]. Another great advantage is the adaptable window size, hence the window size can be adjusted according to the velocity and or even to the processing power available (although consideration of the power is not yet integrated).

[10] *http : //moveit.ros.org/*
[11] *http : //ompl.kavrakilab.org/gui.html*

To guarantee optimal performance the broad set of parameters has to be set properly. The full list of parameters can be found in the appendix. We will briefly enlist all varied parameters compared to the default values.

TrajectoriePlannerROS The TrajectoriePlannerROS denotes the default local planner used by move base. According to the ROS Wiki page [12] the planner takes into account the global plan. Based on the kinematics of the robot the velocity commands for the robot are calculated. The computed costs rely on both the collision avoidance and the proximity to the global path, the full algorithm of the used parameters is presented on the homepage. The parameter settings has to be changed according to the kinematics presented in Section 6.2.2.

[12] $http://wiki.ros.org/base_local_planner$

6 Evaluation

In this chapter, we present results of an empirical evaluation of the system. First of all we target the base functionality of the forklift. Moreover we test and analyze the accuracy of motion execution and the proper integration of its feedback.

First we present the stage of getting the whole system started as well as the treatment of problems and errors regarding the hardware and the software. Including occurred errors as well as methods of fixing them or at least minimize either the occurrence or their impact.

The latter part consists of the high level testing of planners and systems and is denoted as advanced experiments.

6.1 Basic Experiments

After assembling the mechanical and electrical parts of the autonomous forklift, the system had to be checked accordingly. Therefore a set of commonly used methods and test setups were applied to the robot. To ensure proper functionality every single part had to be tested. The traceability of occurring errors had a reasonable role within the stepwise arranged process of testing. As a result the initial experiments were addressed to the basic components of the system. Afterwards the system parts were step by step integrated into the experiments and each experiment was repeated several times until a adequately working system was established.

The first stage of the examination concerned to the proper command executions given to the CAN interface. Although the motor and controller provided a well designed, clean interface to control the motion of the motor, the realization mounted on the robot and supply of motion through the back wheel could have induced uncertainties and errors. Therefore a test program was executed which sent a translational velocity command of one m/s for the duration of two seconds and the covered distance was measured.
As the test setup does not depend on the surroundings it is neglected here. The results seemed to be reasonable but as they were measured manually it was not accurate enough to

be sufficient as reliable proof. Therefore, the first experiment was denoted as the evidence of the rough accuracy.

Next the odometry was added which granted a feedback of the system relying on the feedback of the motor controllers. Hence an adapted experiment based on the idea of a square-path-test also known as UMBmark-test [25] was composed. Usually a square-path-test, as described in [25], consists of a robot, which is commanded to move along a square with fixed length. The distance, concerning a Cartesian coordinate frame x and y between the desired and the achieved goal, is measured. The results permit conclusions regarding the quality of the execution of the velocity commands and the odometry.

As the space at the laboratory was limited and the space-consuming motion primitives of the mobile forklift at most at rotational movement the corridor outside the lab was used. The floor plan, including the corridor, generated by using Gmapping, can be seen in Figure 6.1. Due to the fact, that the corridor is not squared, the corridor as a whole could not be used for a squre path test. Furthermore the width of the corrider does not either suffice the requirements. Hence the robot was manually commanded via a game pad to drive along the corridor until he reaches exactly the starting spot marked accordingly at the beginning. Then the deviations of the odometry in x, y and Θ were analyzed.

The results were at the first look devastating, although several attempts were made, as can be seen in Table 6.1.

Table 6.1: This table states the results of the first modified squared path test, denoting the derivations in directions x and y and the rotational derivation of orientation ϕ.

Test Number	Deviation x [m]	Deviation y [m]	Deviation ϕ [rad]
1.	+3.460	-15.558	-0.811
2.	+2.544	-13.209	-0.795
3.	+2.392	-14.813	-0.640

Afterwards the robot was placed in the corridor heading to a wall in the distance of 8 meters. Then translational velocity commands were sent to the robot and the difference between the odometry (starting at (0,0)) and the 8 m minus the distance indicated by the laser scanner. If the odometry would be working appropriately the difference should vary around 0, but it wasn't. As we ensured, that the controller is reporting the same velocities as it was commanded to execute, the error had to be within the CAN interface. After double checking all set values the error was found to be the hard coded radius of the back wheel, which was measured manually to 6 cm instead of 6.25 cm. Due to the fact, that the corridor was that large the error accumulated and could be found.

The second experiment was repeated resulting in much better deviations as can be seen in Table 6.2. Additionally a bag-file of the last square-path-test was made which can be found (link or reference to appendix with the link there).

Table 6.2: This table states the results of the second squared path test, denoting the derivations in directions x and y and the rotational derivation of orientation ϕ.

Test Number	Deviation x [m]	Deviation y [m]	Deviation ϕ [rad]
1.	-1.498	-0.731	-0.077
2.	-1.367	-1.025	-0.065
3.	-1.054	-0.533	-0.078

Though the deviation of more than one meter in x direction could seem high, in relation to the covered distance it is adequately an regarding the first square path test (c.f. Table 6.1 the difference is enormous (difference in x: 1,492333334 meter, difference in y: 13,763666667 meter and difference of Θ: 0,675333334 rad).

Knowing that the odometry was working accordingly the next challenge consisted in getting the different planner setups working. To be capable of starting the planners and the move base, a map had to generated using the chosen Gmapping package. As this package is solely relying on the already tested parts odometry and laser scanner the first generated map, alike shown in Figure 6.1 of Section 6.2, was made. Although the next issue was detected, as the map included several small, black dots, consisting of one pixel and indicating an obstacle, alongside the covered path. In Figure 6.1 they are already been erased.

Due to the fact, that initially the map is generated with no obstacle, but only unknown space, these obstacles had to be inscribed by Gmapping according to the laser scans. Double checking the laser scans in RVIZ lead to the conclusion, that the laser scanner was detecting an obstacle in a small distance outside the borders of the footprint. As reflections were eliminated as source of disturbance the lens of the laser scanner was examined. A small scratch on the covering of the laser caused certain reflections. So a filter was implemented into the software interpolating around the pixels of the scratch which rectified the data gained by the laser scanner.

The parameter tuning of each local planner (DWA and TrajectoryPlannerROS) and global planner (Navfn and SBPL Lattice planner) combination was established manually through empirical testing without special setup until either ten positive attempts (no collisions) were made or at least the best local maximum of a system according to the positive attempts was found. Therefore no further collected data can be presented except for the already mentioned reasonable parameter changes in Section 5.2.3.

6.2 Advanced Experiments

Primarily the focus of the advanced experiments was on two objectives. First of all, the different systems and planners should be tested in different environments pointing out their strengths and weaknesses. Therefore, the first section is mainly appointing the qualitative comparison. In the second section the quantitative results will be presented, which were produced with the help of a tracking system described in Section 6.2.2.

In the end of every of section a discussion will be presented targeting the choice of the single testing setups and the best parameter sets of the planners and system which approved as the best choice so far.

6.2.1 Qualitative Experiments

Concerning the planners, both local and global, presented in Section 5.2.3, the first series of test tend to evaluate the success. Due to the fact that the setup itself contains no ground truth about its whereabout, solely the probabilistic guess of the AMCL and the integration of the motor motion feedback were considered. Both methods do not claim to be optimal, hence the results can neither lodge a claim on the accuracy of an planned and executed path nor on the precision a goal was found regarding the real world. Nevertheless, we define a successful and a non-successful outcome of an experiment by the reaching of a pose in the vicinity of the aimed goal without colliding with any kind of obstacles. Additional a test is negative, if the robot cannot plan a collision-free plan to the aimed goal (e.g. caused by planner's specifications, or external interferences).

This section presents test cases to solely compare and carve out the differences. Discussing and understanding of emerged problems and errors constitutes the main focus of it.

Regarding the experiments itself some important facts have to mentioned. At first every combination of global and local planner was tested at least 10 times to achieve at least a minimum of significance. The data collected from the experiments were written down, additionally the ROS provided recording was made for several test series mainly including successful trials. Although these, so called, bag-files provide the most important parts of the communication between the running ROS packages, the reproduction may be difficult due to the following aspects:

- Depending on the quality of the localization the mobile forklift had, the starting pose, although it was marked on the floor as well as photographed to claim reproducibility, could alter because of the probabilistic localization.

- The concussions arising from the unevenness of the floor are not to be neglected concerning their partly drastically influence on the systems hardware itself as well as on the laptop computing. Due to the fact that the system holds no suspension at all, the stiffness of the forklift propagate all vibrations and concussions to the hardware and could for example produce loose contacts and/or modifications on the sensor mounting angles.
- Although the system claims to have achieved a goal properly, in reality the mobile robot may have collided with walls or other obstacles. In most cases a small jump in the localization is caused by collision, if the impact was strong enough. Nevertheless the data collected by the observer have to be considered important to ensure more significant results.

The recorded map is provided for every test case although the location of the test was confined to two locations (plus a third one used at the qualitative experiments of Section 6.2.1):

- the floor of the laboratory
- the ground floor of the building

At the begin of every test series the map is presented. Although it consists most of the time of the same floor (c.f. Figure 6.1), in every section only the according parts are represented including a planned trajectory.

The results are outlined by a table enlisting the tested system and the trial number as well as the according occurred errors and the success. The table shows the results of the first test, including the sensor, local and global planner. The column "successful" denotes the attempts resulting in a "goal reached" prompt of the planner, taking into account for negative attempts as well abortions by the planner as collisions. Furthermore, the table shows, which condition caused the failed attempt.

6.2.1.1 Basic Movement

The first test tested the trivial navigation without changes of the environment in comparison to generation of the map. Figure 6.2 shows the part of the Gmapping-generated map of the laboratory's floor used for this test.

For every test sequence (different combinations of planners and systems) the starting spot was the same and marked on the floor. Thus the ground truth of the starting spot was provided manually. Otherwise the goal pose was not transmitted hardcoded, but also manually through RVIZ. Nevertheless this goal position had a small variance.

Figure 6.1: The figure represents the floor plan of the institute. Most of the base experiments took place there. It is generated through the usage of Gmapping, while exploring the floor with the robot steered through a game pad. The black pixel state obstacles, whereas light grey states free space. The dark grey parts are unexplored space but should never be tackled within our experiments.

Table 6.3 depicts the results of the first test series. The first test setup resulted in quite surprising outcome. Due to the fact, that the door opening denoted the only obvious challenge we expected good results for all combinations. Mandatory retrospectively the design was not easy at all for the Navfn. Not only that parameter had to be reduced concerning the inflation radius to even be able to move in the corridor. As a result to this the global planner was unable to avoid the collisions. Furthermore the acute angle made it substantially impossible for the local planners to not collide with the back of the less moveable forklift, regarding the kinematics.

6.2.1.2 Door Challenge

To refine the results and to seperate the wheat from the chaff, the goal was placed within the laboratory itself. Figure 6.3 shows the part of the map tested, including as well goal pose as starting pose. The unoccupied space of the door consists of 90cm, which is comparably small to the width of the robot with respect to its kinematics.

Again the starting spot was marked on the ground and the goal pose was estimated through

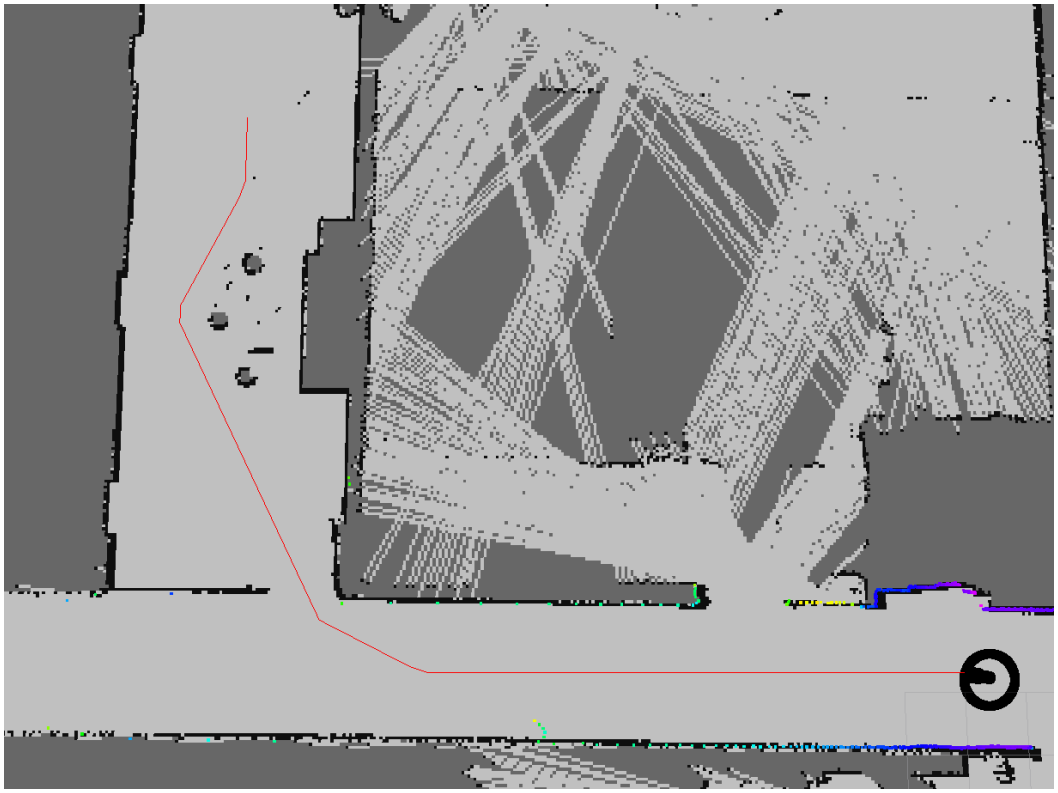

Figure 6.2: The figure represents the part of the institute's floor plan, with about 11m height and 15 m broadth. The first test series of the quantitative experiments took place there. The red line marks the global plan, the colored lines denote the laser scans ranging from yellow to violet, representing the different distance measurements. The black pixel state obstacles, whereas light grey states free space. The dark grey parts are unexplored space but should never be tackled within our experiments. The start is placed on the lower right end of the global path, marked by the black circle. The line within the circle denotes the orientation.

RVIZ.

Table 6.4 includes all results, positive as well as the conditions for the negative attempts.

The second test was therefore tested without the combination of Navfn and TrajectoryPlannerROS. It has to be mentioned, that the first attempts using the laser scanner combined with SBPL-Lattice and DWA had to be aborted, due to the fact that every single attempt ended half a meter in front of the door. Due to the fact, that the door was tackled with a quite acute angle, the varnish of the door reflected in such an inconvenient way, that the laser scanner detected an obstacle in the door passage. Hence the planner capitulated. Taking advantage of the fully opened door, building an obstacle with help of the open spread door to

Table 6.3: The table displays the first test series including all combinations of planners with the laser scanner. Furthermore the most promising local planner was also tested with the Asus and both global planners. The three right side columns visualize the results of the test, denoting positive attempts and the cause of the negative attempts. Navfn is the abbreviation for Navigation Function and DWA for Dynamic Window Approach. SBPL means Search-Based Planner Lab and TPROS stands for the TrajectoryPlannerROS.

Sensor	Global Planner	Local Planner	Positive Attempts	Collision	Goal not Reached
Laser Scanner	Navfn	TPROS	1	6	3
Laser Scanner	Navfn	DWA	1	6	3
Laser Scanner	SBPL-Lattice	TPROS	3	7	0
Laser Scanner	SBPL-Lattice	DWA	10	0	0
Asus	Navfn	DWA	1	7	2
Asus	SBPL-Lattice	DWA	2	6	2

Table 6.4: The table depicts the first test series including all combinations of planners with the laser scanner. Furthermore the most promising local planner was also tested with the Asus and both global planners. The three right side columns visualize the results of the test, denoting positive attempts and the cause of the negative attempts. Navfn is the abbreviation for Navigation Function and DWA for Dynamic Window Approach. SBPL means Search-Based Planner Lab and TPROS stands for the TPlannerROS.

Sensor	Global Planner	Local Planner	Positive Attempts	Collision	Goal not Reached
Laser Scanner	Navfn	DWA	1	4	2
Laser Scanner	SBPL-Lattice	TPROS	2	2	3
Laser Scanner	SBPL-Lattice	DWA	6	1	1
Asus	Navfn	DWA	0	3	5
Asus	SBPL-Lattice	DWA	0	0	8

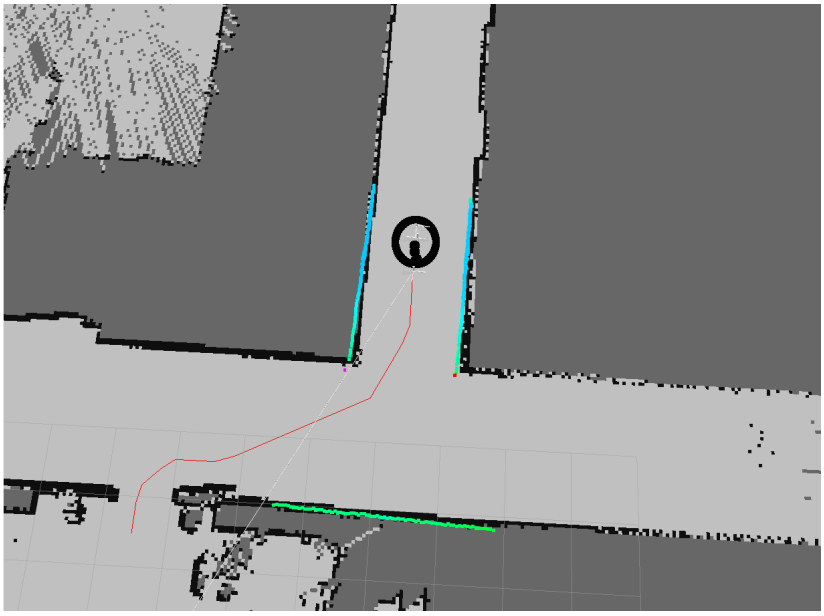

Figure 6.3: The figure represents the part of the institute's floor plan. The second test series of the qualitative experiments took place there. The red line marks the global plan, the colored lines denote the laser scans ranging from yellow to purple, representing the different distance measurements. The black pixel state obstacles, whereas light grey states free space. The dark grey parts are unexplored space but should never be tackled within our experiments.The start is placed on the upper right end of the global path, marked by the black circle. The line within the circle denotes the orientation.

force the local planner to tackle the free space with a more obtuse angle, the results showed in Table 6.4 were produced. The results of the Asus could not be considered representative because of the vibrations and strokes caused by the bottom seal the system crashed at most attempts either leaving a non movable robot behind or worse a driving robot with a dead control system. The crash of the system could be narrowed down to the USB bus of the computer. As the laser scanner is connected via Ethernet port the problem is solely affecting the Asus and the game pad. In virtue of these vastly outcomes, the Asus in combination with Navfn was rejected too.

As concluded from the results of the first two tests the combination of laser scanner, SBPL-Lattice and TrajectoryPlannerROS was also discarded. The DWA planner outperformed the TrajectoryPlannerROS in every single challenge. Due to the fact, that the latter planner per se included no advantage, neither visible nor included in the description which would justify a further use, no further tests were made.
Although the setup was expanded with obstacles both tested combination could improve their

performance, although not significantly. Nevertheless both systems were able to avoid the obstacles easily, in spite of the fact that they were placed accordingly to force the systems to adapt their initial plans.

Remark on the Asus: The problem denoted of the lack of robustness of the whole system with the processing of the extensive load of data forwarded by the Asus, in combination with the previous mentioned strokes and vibrations, which caused the odometry to crash. Additionally one has to say, that due to the reflecting glass walls even the navigation in a straight corridor evolved to a quite challenging task.

6.2.1.3 Basic Movement including Door Challenge

To derive more significant data according to the efficiency of the planner regarding the ability to plan over longer distances another experiment was conducted. To avoid redundancy, goal and staring pose were exchanged. One can see in Figure 6.4, the extract of the floor map. The door was tackled from the other side. The results of this experiment are presented in Table 6.5. As in the previous experiments SBPL-Lattice Planner in combination with DWA and the laser scanner achieved the best performance with 6 positive attempts out of 8, without colliding with an obstacle.

Table 6.5: The table reports the second test series including all combinations of planners with the laser scanner. Furthermore the most promising local planner was also tested with the Asus and both global planners. The three right side columns visualize the results of the test, denoting positive attempts and the cause of the negative attempts. Navfn is the abbreviation for Navigation Function and DWA for Dynamic Window Approach. SBPL means Search-Based Planner Lab and TPROS stands for the TPlannerROS.

Sensor	Global Planner	Local Planner	Positive Attempts	Collision	Goal not Reached
Laser Scanner	Navfn	DWA	1	4	2
Laser Scanner	SBPL-Lattice	TPROS	2	2	3
Laser Scanner	SBPL-Lattice	DWA	6	1	1
Asus	SBPL-Lattice	DWA	0	0	8

6.2.1.4 Basic Movement including Door Challenge and Obstacle Avoidance

To further extend the testing setup in the test of Section 6.2.1.3 obstacles were added. Due to the fact, that these obstacles were not recorded, the global planner could not consider

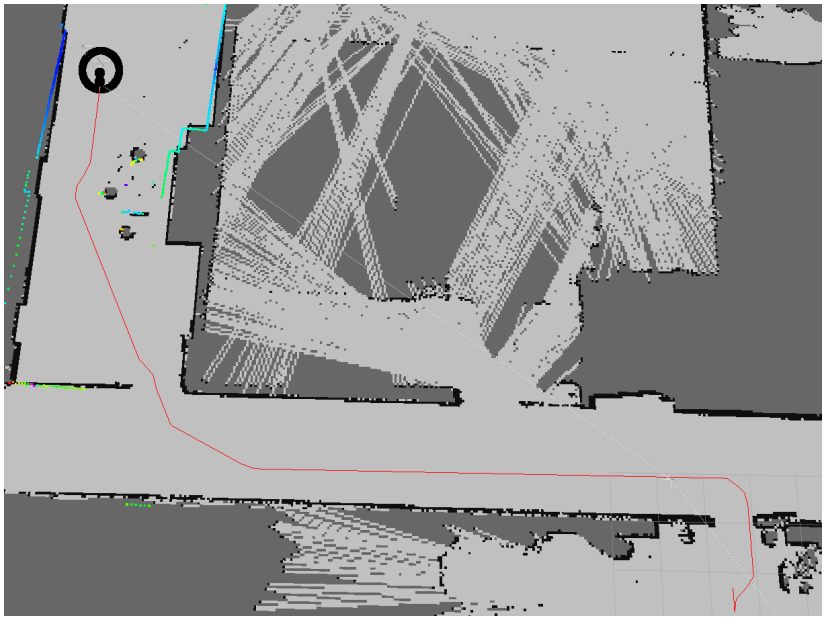

Figure 6.4: The figure represents the part of the institute's floor plan. The third test series of the qualitative experiments took place there. The red line marks the global plan, the colored lines denote the laser scans ranging from yellow to violet, representing the different distance measurements. The black pixel state obstacles, whereas light grey states free space. The dark grey parts are unexplored space but should never be tackled within our experiments.The start is placed on the upper left end of the global path, marked by the black circle. The line within the circle denotes the orientation.

them. The plan had to be changed accordingly at the moment the obstacles were detected by the local planner which had to avoid them.

The Table 6.6 represents the results of the test sequences.

In the last qualitative test series on the laboratory's floor, only the laser scanner was tested. It has to be mentioned, that the parameter setting was slightly changed, due to the fact, that at all other scenarios the backward movement was forbidden or at least was punished at the planning stage, due to the absence of sensors pointing backward. Without backward movement the fourth task would be impossible. Furthermore the goal was actually not exactly reached by the SBPL-Lattice setup, but only because to test observer caused an interference which was not noticed until the evaluation of the experiment. As a result a obstacle was detected and inscribed into the costmap before starting to move backward. Afterwards the laser scanner was not able anymore to clear the unoccupied space and therefore the system avoided the goal. Nevertheless, the turning manoeuvre itself was accomplished several times,

Table 6.6: The table displays the third test series including a selection of combinations of planners and sensors. The three right side columns visualize the results of the test, denoting positive attempts and the cause of the negative attempts. Navfn is the abbreviation for Navigation Function and DWA for Dynamic Window Approach. SBPL means Search-Based Planner Lab and TPROS stands for the TPlannerROS.

Sensor	Global Planner	Local Planner	Positive Attempts	Collision	Goal not Reached
Laser Scanner	Navfn	DWA	2	3	2
Laser Scanner	SBPL-Lattice	DWA	7	0	1
Asus	SBPL-Lattice	DWA	0	0	8

which can be seen in 6.7.

6.2.1.5 Turning Manoeuvre

One of the requirements revealed due the course of development was the challenging task of turning especially concerning the small environments of the laboratory's floor. Hence we tried to present a challenging course, where a simple in place rotation can not be established, because of the kinematics of the forklift. Considering the sensor alignment, one had to be aware, that the robot could only perceive the environment limited, as no sensors point backward. Every test series was repeated seven times.

Figure 6.5 represents one expected solution for the design of the challenge. The environment of this test was placed on the ground floor of the laboratory's building.

Due to results of the preceding tests only the two most promising setups were chosen for this run. The results of the tested systems are presented in table 6.7.

Table 6.7: The table displays the fourth test series including a selection of combinations of planners and sensors. The three right side columns visualize the results of the test, denoting positive attempts and the cause of the negative attempts. Navfn is the abbreviation for Navigation Function and DWA for Dynamic Window Approach. SBPL means Search-Based Planner Lab.

Sensor	Global Planner	Local Planner	Positive Attempts	Collision	Goal not Reached
Laser Scanner	Navfn	DWA	1	6	0
Laser Scanner	SBPL-Lattice	DWA	5	1	1

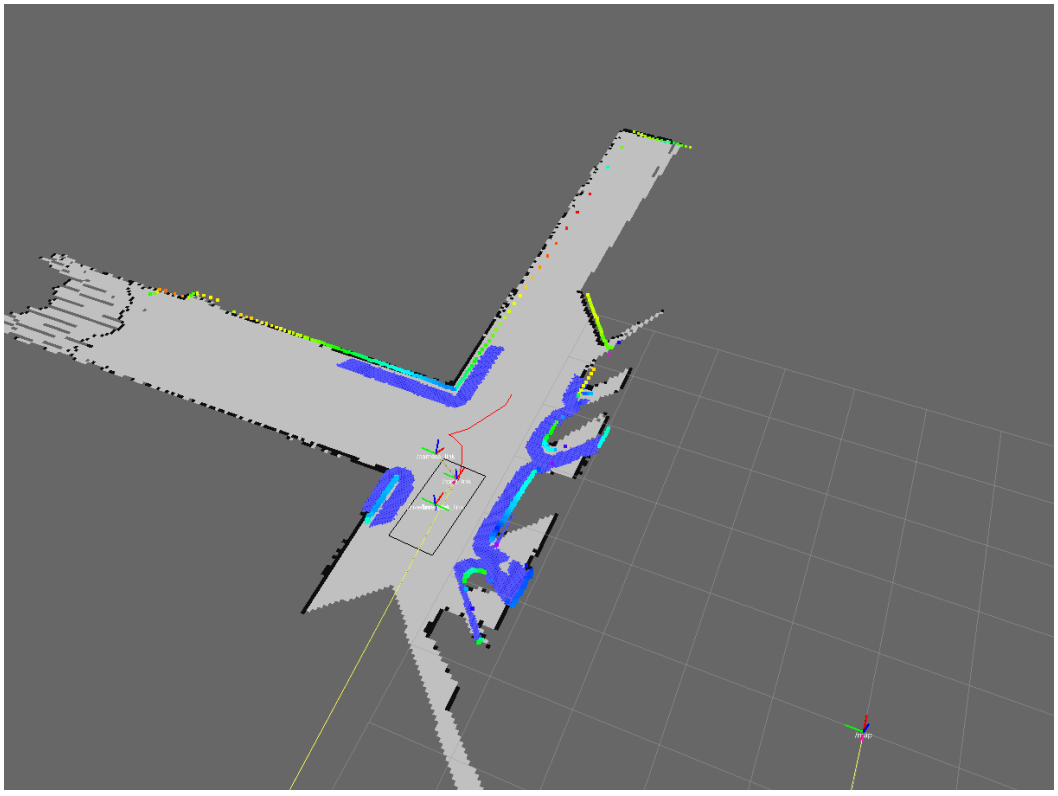

Figure 6.5: The figure represents the part of the institute's floor plan. The fourth test series of the qualitative experiments took place there. The red line marks the global plan, the colored lines denote the laser scans ranging from yellow to violet, representing the different distance measurements. The black pixel state obstacles, whereas light grey states free space. The dark grey parts are unexplored space but should never be tackled within our experiments. The start is placed on the upper left end of the global path, marked by the black circle. The blue pixels denote the inflated obstacles according to the inflation radius of the robot.

6.2.2 Quantitative Experiments

One of the requirements defined, targeted the accuracy of the navigation, as the docking station for the batteries has to be reached with a maximum offset of 10 cm. Thus the last series of experiments tackles the quantitative evaluation of the system.

With respect to the results of the already performed experiments the aim of the qualitative tests were evaluated in detail to three combinations of planners and sensor setups. Getting the opportunity of using a tracking system added the new potential of the ground truth. The reliable feedback of a fixed tracking system, could now be compared to the estimated pose of the system maintained by AMCL (c.f. Sections 4.2.5 and 5.2.3). This tracking system

consisted of the Vicon system *Bonita* [1] (henceforth called solely Vicon system), which consists of an alignment of cameras pointing from various directions onto a scene. Equipped with reflecting spheres, an object can be tracked based on an infrared projector and detector system. The higher the density of these cameras the more precise the setup works out [39].

Due to the fact that the space at the Vicon tracked room was even more limited than the laboratory's the test setup had to be carefully chosen. To achieve a variance of solely 0.2 mm the Vicon cameras therefore monitored a space of four times four meters. So the opportunity of variation was not present and the only difference between first (Section 6.2.2.1) and the second setup (Section 6.2.2.2) comprises an additional obstacle added to the scene after generating the map.

The alignment of the used markers fixed on the robot is presented in Figure 6.6. The arrangement of markers was chosen according the experience of the tracking expert. He suggested to use spots as high as possible, to ensure they were never obscured by the robot itself or obstacles. Furthermore the number of three was chosen to be capable of triangulate them to improve the calculation of a reference point. This reference point was selected as the point in the middle between the two front roles, hence similar to the baselink of our system (cf. Section 4.2.3).

Figure 6.7 comprises the floor plan of the Vicon System's room including the setup for the first two tests.

At last the map (c.f. Figure 6.8) was recorded using the reference point marked in the floor of the room as origin of the Vicon system as starting point of the robot. Thus the origin of the recorded map denotes also the origin of the Vicon system.

To all performed tests ROS bag-files had been recorded.

6.2.2.1 Basic Movement

Although the ground truth would only provide information targeting the localization, hence as long as the only source for localization in the system consisted of the AMCL alone for every constellation of planners and system tested, the decision was made to nevertheless test more extensively. As stated before the scenario is shown in Figure 6.7 representing the alignment of obstacles. Due to the results and the discussion of Section 6.2.2 solely the following combinations of sensors and planners are tested:

- Laser Scanner - DWA - SBPL-LATTICE

[1] *http://www.vicon.com/system/bonita*

Figure 6.6: The picture presents the alignment of the marker needed for the tracking system.

- Laser Scanner - DWA - NAVFN
- Asus - DWA - SBPL - LATTICE

Every system is tested with about 10 test run to achieve a sound basis for a comparison.

6.2.2.2 Basic Movement including Obstacles

As mentioned before the environment was basically the same as in Setup one (previous section) except for an additional placed obstacle after the turn, which can be seen in Figure 6.9. To challenge the system even more, it was aligned such that the robot was not able to detect it at the beginning of the test, hence the local planner had to report dynamically that the way is blocked and a new plan had to be provided.

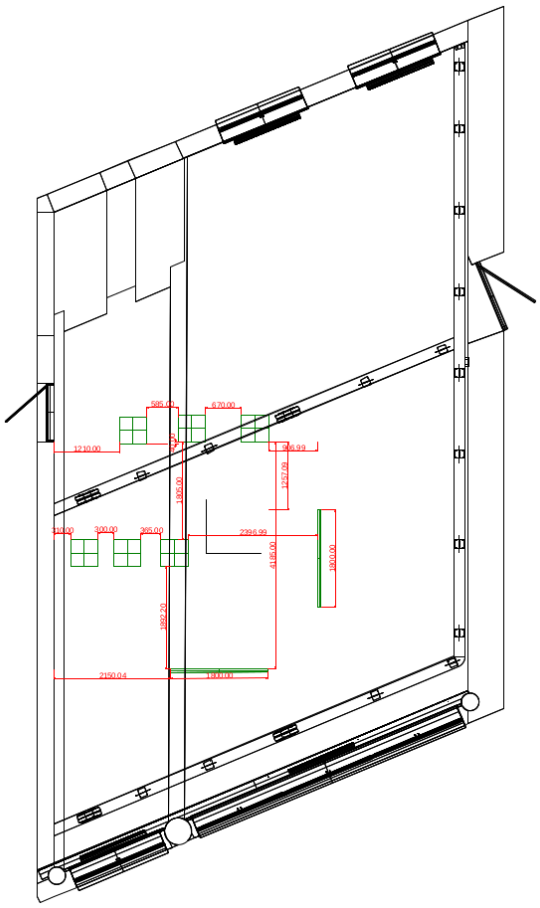

Figure 6.7: The figure presents the top-view of the room where the Vicon tests took place. The green objects represent the obstacles placed for the scenario. The red lines denote the according linear measures.

6.2.2.3 Results and Discussion

Although the setup was not as extended as it was desired, the test cases could point out the quality of the systems. All test runs were recorded in bag files which are provided on request. Unfortunately not the whole ROS communication could be recorded, due to internal ROS error, which caused a corruption of the full file. Notwithstanding this, a second bag file was recorded including the most important topics to reproduce the motion and the localization of the robot.

Concerning the first setup (c.f. Section 6.2.2.1) it has to be mentioned, that the Asus Xtion Pro Live could not be tested accordingly. In the single case tested with the Asus, the interferences in the small room including other infrared projectors disturbed the sensor to

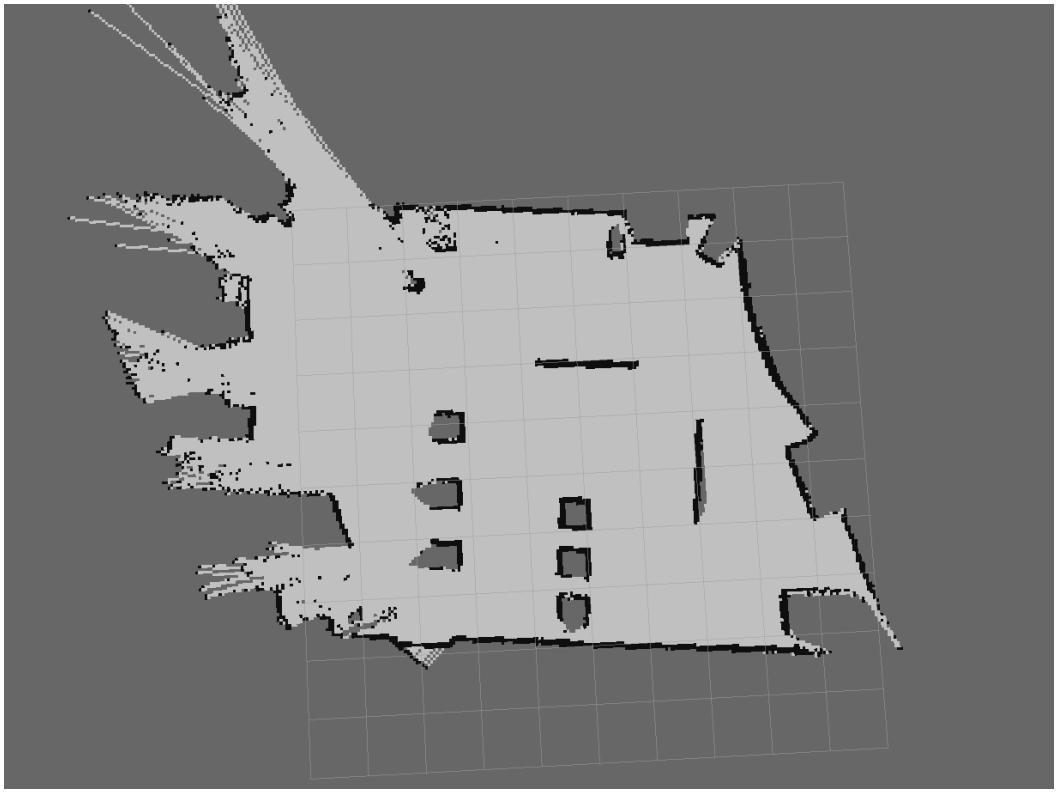

Figure 6.8: The recorded map of the scenario developed for the Vicon test. It is taken from RVIZ and the black lines denote the walls and obstacles. Furthermore dark gray represents unknown environment and light gray the free space.

extensive. Furthermore the transformation of the structure of the environment, perceived by the sensor, into a laser scan were too blurry for a proper navigation. The problem mainly consisted in the appearing of non-existent obstacles. In particular, because these artifacts were projected into the proximity of the footprint. Hence the robot almost immediately entered the recovery mode, which lead to collisions due to the limited space.

Both other planners, as well SBPL-Lattice planner as Navfn, reached the goal. Analyzing the bag files revealed some interesting information. First of all in both setups the DWA local planner tried to optimize the path to the goal. As in the case of the SBPL-Lattice planner the global path is not very smooth, due to the fact, that the task of planning is highly dependent on the number of MP (c.f. Section 5.2.3). A higher value of MP would cause latencies in the planning state. Nevertheless the SBPL-Lattice setup could achieve very good results regarding the accuracy for the goal pose, which will be discussed later in this section at the description of Equation 6.1. The covariances of the main diagonal denote the variance and are therefore a measure for the quality of the accuracy of the localization compared to the

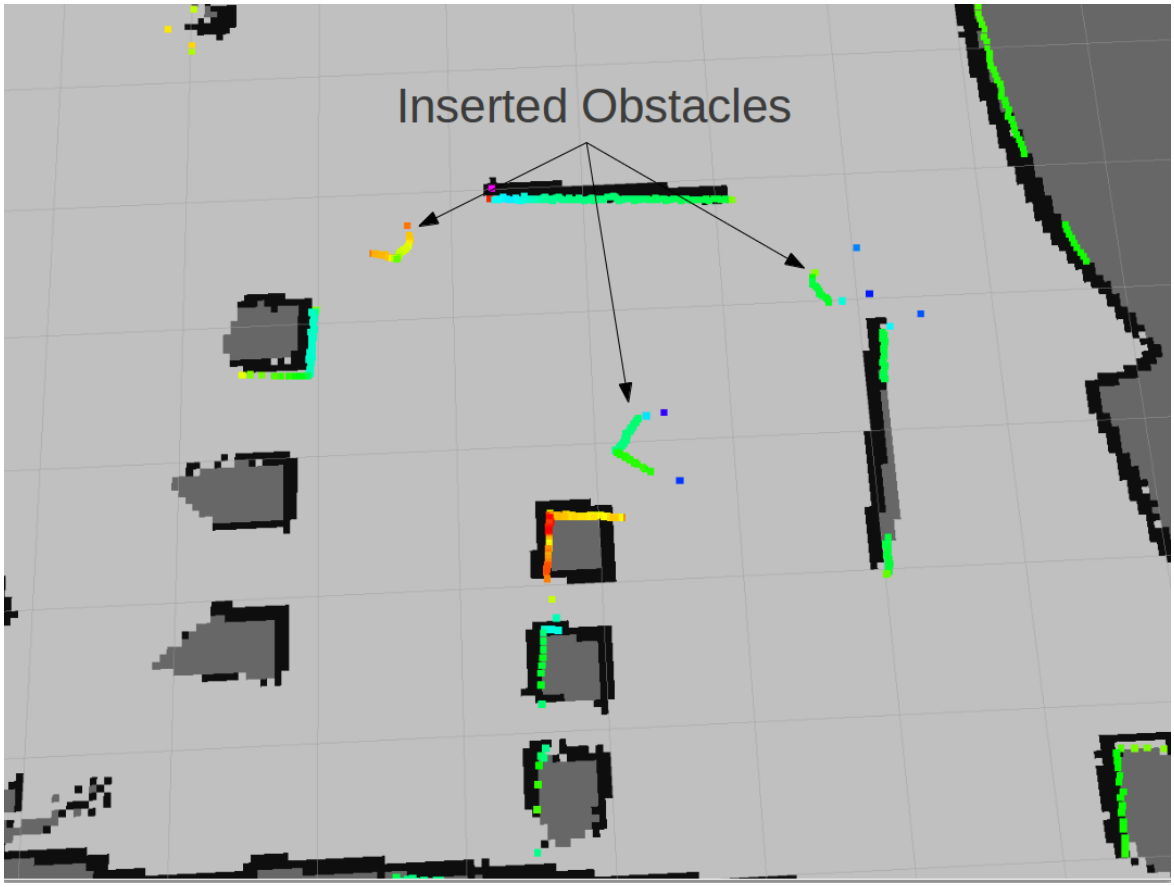

Figure 6.9: This plot shows the unknown obstacles of the last scenario indicated by the colored laser scans. It is taken from RVIZ and the black lines denote the walls and obstacles. Furthermore dark gray represents unknown environment and light gray the free space.

ground truth, provided by the Vicon system. The off-diagonal values denote the monotonic relation of the two variables.
One can notice a slight drift of the goal and start positions in all experiments. This was caused through the markers of the Vicon system. Although the alignment should have been transformed into the base link of the robot, unfortunately the acquired data featured no proper transformation. We considered transforming the points afterwards, but due to the fact, that the manually recorded correct alignment transformation only worsened the results, we maintained the plots adding the information, that the recorded point is about 10 cm behind the baselink.

According the data collected for Navfn, two major points of criticism emerged. First of all the inflation radius had to be reduced, so a path could even be found. Caused by this, the planner

arranged the way points quite near on the brink of feasibility, such that the DWA had to reduce the speed. Secondly, the goal pose was exceeded although the parameters demanded a maximum before the end point. Furthermore, the end orientation was achieved through a final in place rotation. In a setup more narrow than in the scenario this could have proven critical, but all in all caused no collisions. Nevertheless, no collision was caused and though the accuracy was slightly worse than the results (c.f. Figure 6.10 of SBPL-Lattice, this could not be stated as significant, like in the results of Section 6.2.2. This can be explained by the fact, that the assembled course did not include rotation within narrow passages (like the door(s) at the quantitative experiments 6.2.2. Hence the footprint and the overall kinematic of the robot played no important role.

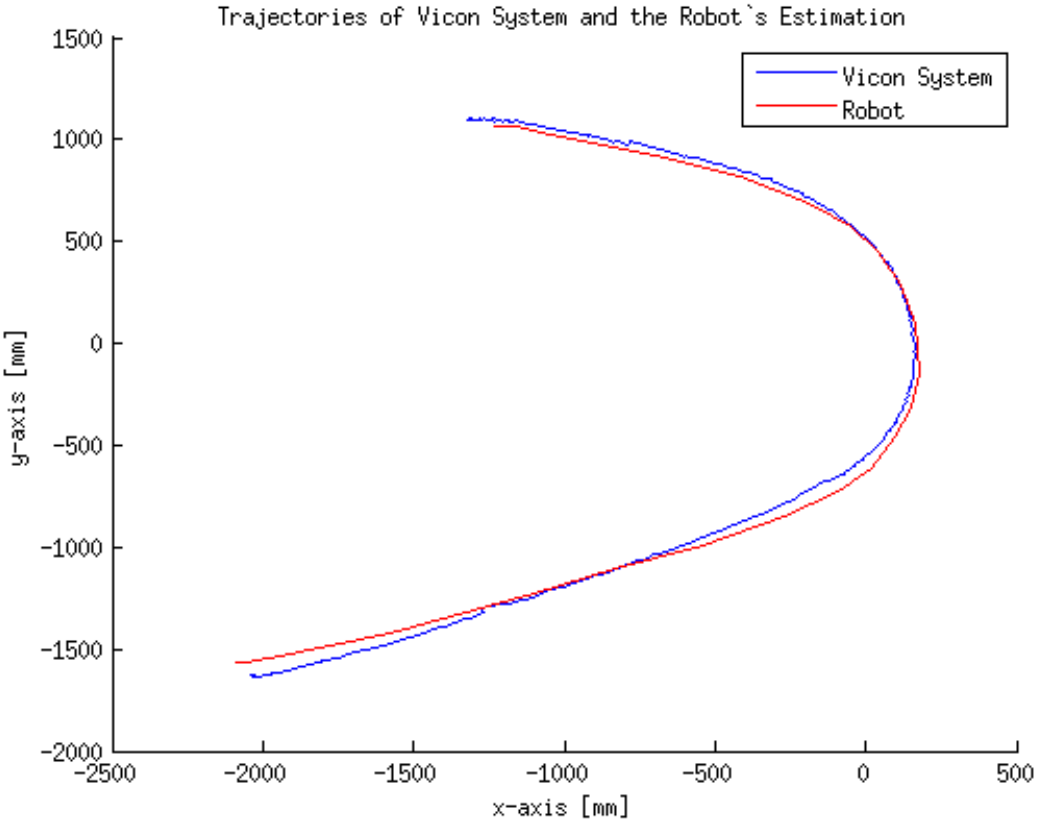

Figure 6.10: The plot depicts the trajectory of the ground truth provided by the Vicon system and the robot's estimation of his current localization. As the legend states, the red trajectory visualizes the ground truth as the blue line denotes the estimated motion of the system using Navfn as global planner.

To evaluate the accuracy we compared both the recorded values of the trajectory from the

Vicon system and the transformation from map to baselink recorded in the tf-topic of the bag-file to each other. Figure 6.11 presents the two according trajectories from the first test setup for the SBPL-Lattice system. As stated before one has to keep in mind, that the divergence of the start and end position is caused by the error of recording. This error nonetheless has only a negligible influence on the remaining parts as the small difference induce no significant change of the trajectory as one could deduce from derivations of Section 4.2.3.3.

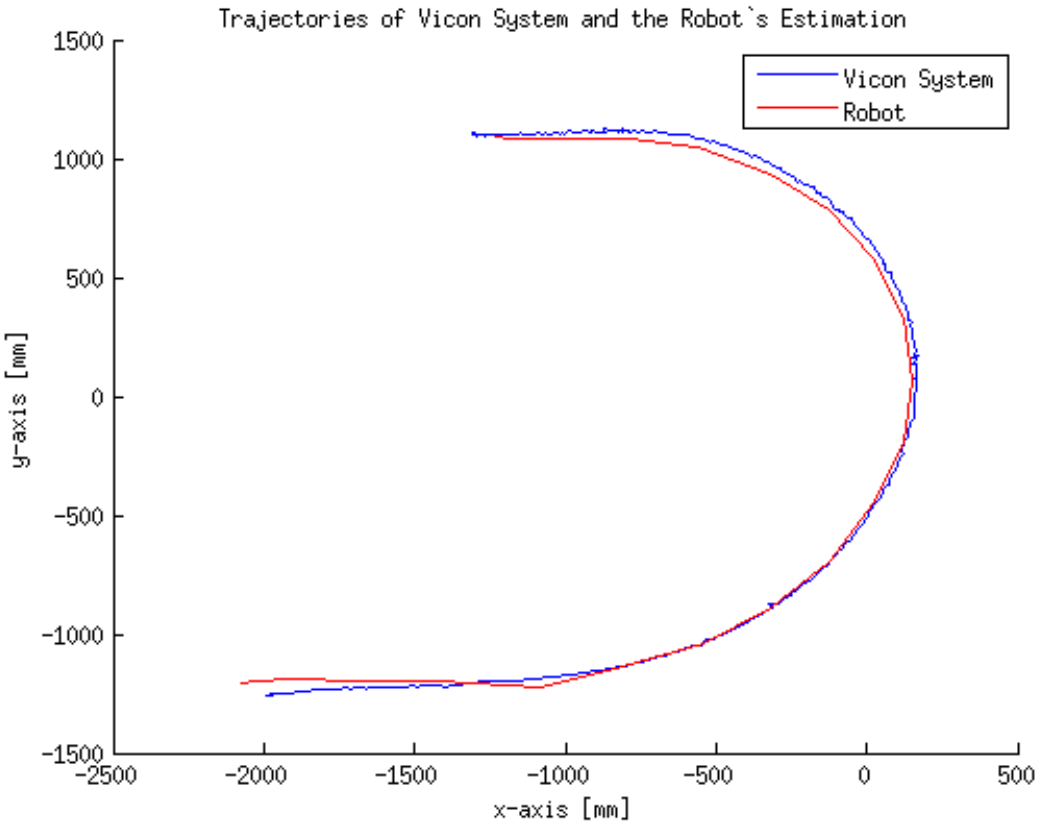

Figure 6.11: The plot depicts the trajectory of the ground truth provided by the Vicon system and the robot's estimation of his current localization. As the legend states, the red trajectory visualizes the ground truth as the blue line denotes the estimated motion of the system using SBPL-Lattice as global planner.

At the end we present the results of the last experiment including the obstacles which were added after the recoding of the map. This scenario was due to limitation of time only performed by the SPBL-Lattice system. As one can see as well in Figure 6.12 as in the covariance matrix (6.3) the uncertainty about the localization of the robot increased. This

can be explained by the fact that AMCL tries to match the detected laser scans to the pre-generated map. Thus the new obstacles could not be found, the uncertainty obviously rises. When observing the odometry transformation frame, one can see in all 5 runs, that the moment the unknown obstacles appear, the transformation begins to alter with small jumps. These are caused by the matching of the AMCL and although the odometry would be consistent the pose of the baselink is modified according to the matching of the laser scans. As a result, the localization worsened and only improved a bit at the reaching of the goal pose, due to the fact, that the unknown obstacles are no longer within the field of view of the laser scanner.

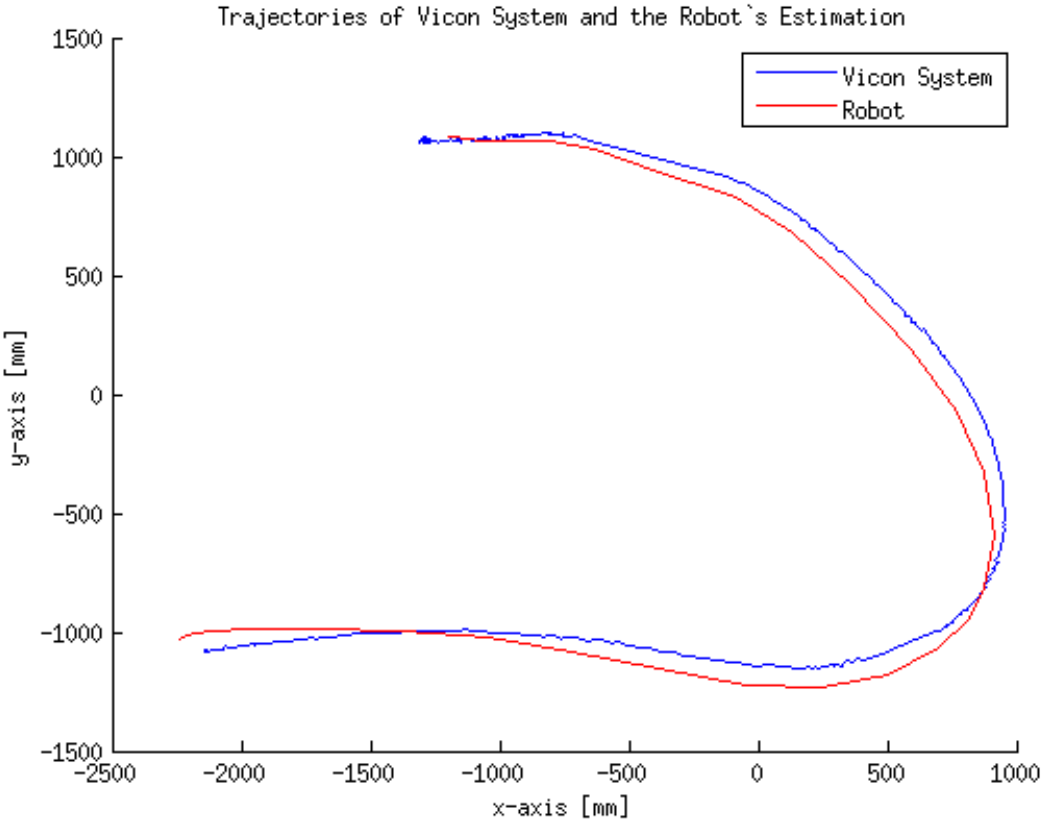

Figure 6.12: The plot depicts the trajectory of the ground truth provided by the Vicon system and the robot's estimation of his current localization. The setup is extended with obstacles to force the robot to avoid a collision the initial plan would cause. As the legend states, the red trajectory visualizes the ground truth as the blue line denotes the estimated motion of the system using SBPL-Lattice as global planner.

Figure 6.13 and Figure 6.14 show the modification through the SBPL-Lattice planner when

the unknown obstacles were detected.

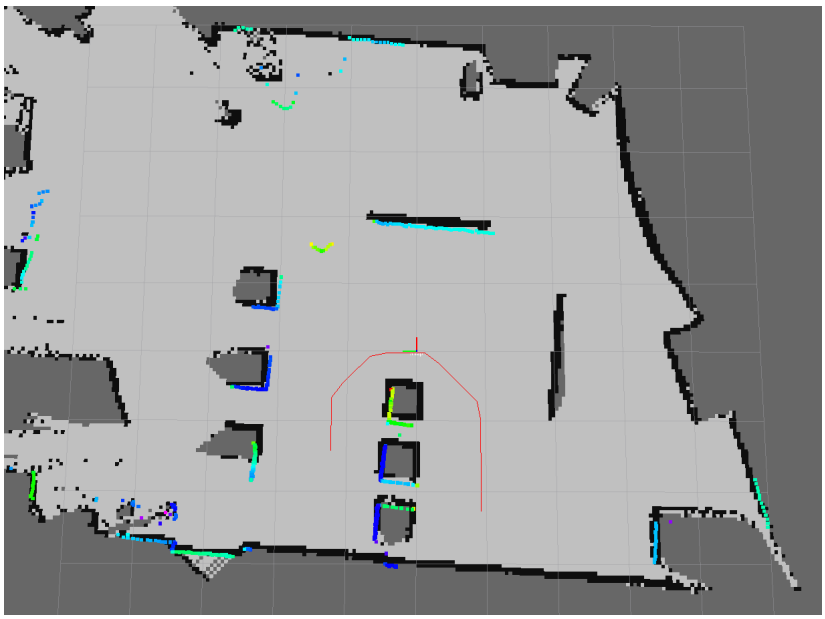

Figure 6.13: The figure displays the planned trajectory of the SBPL-Lattice planner before detecting the before unknown obstacles. The shot is taken from Rviz replaying the recorded bag-files. The black lines and pixels denote walls and obstacles. Furthermore dark gray represents unknown environment and light gray the free space. The red line indicates the planned trajectory provided by the global planner.

To sum up the results, the covariance matrix of all presented test cases are displayed. For this purpose we chose not the best solutions but representative ones, as one can see when inspecting the bag files. As can be concluded from the figure captions:

- SBPL-Lattice: test case 3
- Navfn: test case 7
- SBPL-Lattice including obstacles: test case 4

The covariance matrices in Equation (6.1) are measured in centimeters.

$$\begin{bmatrix} 0.006261 & -0.002243 \\ -0.002243 & 0.005128 \end{bmatrix} \tag{6.1}$$

$$\begin{bmatrix} 0.007661 & -0.004004 \\ -0.004004 & 0.006905 \end{bmatrix} \tag{6.2}$$

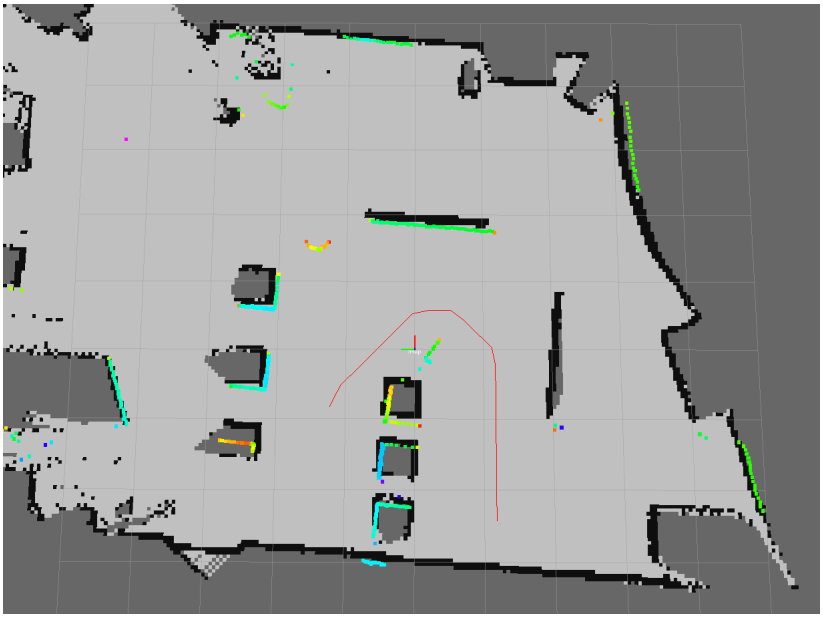

Figure 6.14: The figure displays the planned trajectory of the SBPL-Lattice planner after detecting the before unknown obstacles. The shot is taken from Rviz replaying the recorded bag-files. The black lines and pixels denote walls and obstacles. Furthermore dark gray represents unknown environment and light gray the free space. The red line indicates the planned trajectory provided by the global planner.

$$\begin{bmatrix} 0.019500 & -0.001306 \\ -0.001306 & 0.019194 \end{bmatrix} \tag{6.3}$$

Concerning the covariance matrix, it has to be mentioned, that we had to process the data in order to achieve meaningful results. On the one hand the mentioned transformation problem caused an offset of about 10 cm. Furthermore, the timestamps of the samples are not synchronized, hence one has to interpolate between the points. Additionally the relational long period of ordering a goal after the system is all started up until the first move triggers to many points, which would improve the results as the localization is rather good at the start. The covariances again prove, that the system is working out quite well (neglecting the offset), proven by the values of Equation (6.1). Concerning the creation of the covariances, x and y of the robot's estimation of its whereabouts were taken and compared against the Vicon system's ground truth. Hence this ground truth was taken as μ (e.g. expected value) and the according covariances were calculated over the whole trajectory of one single experiment. The provided data also indicate the already stated significant, qualitative influence of the

unknown obstacles in equation (6.2) compared to equation (6.2) or equation (6.2)

7 Conclusion

We now want to present the conclusion of this thesis reflecting upon the defined goals stated in Chapter 2.

Regarding the aim for instrumentation of the system, the design was mostly dictated by the industrial partner. The comparison of sensors offered a wide spectrum on possibilities and a clear decision on the usage of the sensors could be made.

In contrast to the initial, planned procedure, the implementation of a mobile robot turn out to be much more challenging. Whereas the the expected main focus lay on the design of new algorithms and advanced sensor fusion within the high level control, majority of the time was consumed by the fixing of the proper communication with the motors accompanied by the challenges of the real world.
On the one hand the decision to build the whole system from scratch implied the use and adaptation of open source software lacking of documentation. Furthermore the fact, that the implementation of a communication standard evolves redundant, if the manufacturer is allowed to base the full functionality on appropriate, manufacture-specific areas of the standard.

The complexity of reality and the system itself has an even greater impact onto the system, due to the fact that open source software is not as reliable as commercial software. Thus especially the base components should be chosen with the greatest care. Further the selection should also reconsider the importance of the community in particular regarding open source software.

The implementation of the base functionality including the adoptions, which had to be made to be capable of moving the forklift according to it's kinematics, worked out well and contributes an interface for commonly used differential drive software. Hence the software framework of the forklift provides a standalone interface for high level control with state-of-the-art software and new technologies.

We further conclude, that the adaptions made to the software implementation to cope with the kinematics of the forklift accordingly, far exceed the standard ROS move base. The

empirical experiments showed that the standard software is not capable of taking special motion models into account. Furthermore these experiments demonstrate the autonomy of the mobile forklift even in narrow passages like doors and corridors and the presence of unknown obstacles.

Last, but not least, the generated software package provides a well working, autonomous navigating, mobile forklift. As the results of experiments in point out, relying on an working robot could unleash the full potential of the field of research.
Considering the outcome of Chapter 6 both the qualitative as well as the quantitative test series point out, the amazing capability of an inertial, massive robot of navigating autonomous through an environment including obstacles. The experiences made through every step of the development are worth their gold, respecting that an autonomous robot was created from scratch.

8 Future Work

Due to the extent of the thesis not all ideas, approaches and enhancements could be examined and developed. In this section all these are mentioned to point out tasks for future investigation.

8.1 Can Interface

As mentioned in Section 8.1 the PDO is not yet implemented. Due to the standard suggests it would enhance the performance of the communication. Whereas at the moment the communication is more than sufficient, future usage could require it of at last could use it as enhancement.

8.2 Lift - RS 485

Regarding the bus system, the forklift already has assembled a lift on it. Although it was an integral part of the seminar project by Bernhard Puchinger [41] and the implementation followed the guiding instruction, the appropriate software could not move the lift. As later detected the replaced bus adapter supported another communication technique (full-duplex/ half-duplex).

8.3 Sensors

Regarding the sensors the greatest open issue is denoted of the usage of the Asus Xtion Pro Live. Due to reflections, which could maybe diminished or even erased somehow by properly mounting the camera and add suspension to the forklift, the achieved results could be way better. Furthermore the idea was developed to use multiple devices to enhance the aperture angle.

The mounting of the Asus causes another problem concerning the software packages provided by ROS. As the pointcloud transformations are quite expansive, therefore the packages use primarily the depth frame, which can not be transformed with packages available in ROS Groovy. Hence either another alignment has to be chosen or a transformer has to be developed.

Another aspect found in the course of the writing stage is the stated (c.f. Section 5.1.4) improvement in the field of ToF. At the time the related research took place, a sensor costed about 600 $ and had a relatively small field of view. Due to the availability of the Kinect One a new, potent competitor in the field of economic sensors has been established.

Additionally a second sensor could be mounted on the back of the robot. Currently all knowledge of the robot regarding the space behind him consists of the map and the not yet decayed laser scans of the costmap.

8.4 Design of the Forklift

As previously stated (c.f. Section 5.1) the forklift does not include any suspension. This leads to quite extensive vibrations, which cause severe system failures as well from the laptop as from the sensors and forklift hardware.

8.5 Planner

Last but not least (according to the attempted approaches and ideas) the planner contain a large potential field of ongoing research. Although the results are quite remarkable, both planner could further be enhanced. For example, the SBPL Lattice planner is consisting of quite a number of motion primitives. Nevertheless the path could be improved concerning both the smoothness and the problem mentioned in section 6.2.1.3 when tackling doors.

One possible approach would be the further investigation of the ompl-app to access a broad variation of planning algorithms. Even the development of an all new planner focusing on special kinematic challenges should be considered.. The idea of designing a roadmap with respect to the kinematic (e.g. utilization of motion primitives) seems promising, not only for the forklift, but for all non-holonomic, non-differential-drive robots.

Furthermore, the local planner can be extended by the three dimensional costmaps inducing not only obstacles above or beneath the plane used in the current version, but also to be able to for example classify objects. For example could the motion of an approaching human

or transporter be detected and an avoidance manoeuvour initiated, taking into account the estimated course of the transporter.
Using three dimensional navigation, more challenging non-planar environments, such as scenarios including ramps, could be managed.

One last idea tackles the scenario of exchanging the battery of a transporter using additional actuators. Most of state-of-the-art planner are specialized on either path planning of a mobile platform, or motion planning of actuators such as arms. Of course the assignment of specialized tools has its benefits, but a planning task including one single planner could provide a huge advantages addressing the simplicity of developing advanced technologies.

Bibliography

[1] I Baldwin and P Newman. Laser-only road-vehicle localization with dual 2d push-broom lidars and 3d priors. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[2] CM Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag London Limited, 2006.

[3] V Castanda, D Mateus, and N Navab. Slam combining tof and high-resolution cameras. *IEEE Workshop on Applications of Computer Vision (WACV)*, 2010.

[4] H Choset, KM Lynch, S Hutchinson, G Kantor, W Burgard, LE Kavraki, and S Thrun. *Principles of Robot Motion, Theory, Algorithms and Implementations*. A Bradford Book, The MIT Press, 2005.

[5] CiA. Canopen device profile for drives and motion control, cia draft standard proposal dsp-402. Technical report, CAN in Automation, 1998.

[6] CiA. Canopen application layer and communication profile cia draft standard 301. Technical report, CAN in Automation, 2002.

[7] B Clarke, S Worrall, G Brooker, and E Nebot. Sensor modelling for radar-based occupancy mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[8] A Correa, MR Walter, L Fletcher, J Glass, S Teller, and R Davis. Multimodal interaction with an autonomous forklift. *5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010.

[9] B Davies and R Lienhart. Using cart to segment road images. Technical report, University of Augsburg, 2005.

[10] AJ Davison. Real-time simultaneous localisation and mapping with a single camera. *IEEE International Conference on Computer Vision*, 2003.

[11] F Dellaert, D Fox, W Burgard, and S Thrun. Monte carlo localization for mobile robots. *International Conference on Robotics and Automation*.

[12] N Deshpande, E Grant, and TC Henderson. Target-directed navigation using wireless sensor networks and implicit surface interpolation. *IEEE International Conference on Robotics and Automation*, 2012.

[13] A Doucet, N de Freitas, K Murphy, and S Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. *TU Darmstadt*, 2013.

[14] H Durrant-Whyte and T Bailey. Simultaneous localization and mapping: Part i. *Robotics and Automation Magazine, IEEE*, 2006.

[15] D Fox. Kld-sampling: Adaptive particle filters. *Department of Computer Science and Engineering, University of Washington*, 1999.

[16] D Fox, W Burgard, and S Thrun. Controlling synchro-drive robots with the dynamic window approach to collision avoidance. *Intelligent Robots and Systems '96, IROS, IEEE*, 1996.

[17] D Fox, W Burgard, and S Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 1997.

[18] V Gandhi, J Cech, and R Horaud. High-resolution depth maps based on tof-stereo fusion. *IEEE International Conference on Robotics and Automation*, 2012.

[19] BP Gerkey and K Konolige. Planning and control in unstructured terrain. *ICRA Workshop on Path Planning*, 2008.

[20] C Häne, C Zach, J Lim, A Ranganathan, and M Pollefeys. Stereo depth map fusion for robot navigation. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[21] Herianto and D Kurabayashi. Realization of an artificial pheromone system in random data carriers using rfid tags for autonomous navigation. *IEEE International Conference on Robotics and Automation*, 2009.

[22] T Hervier, S Bonnabel, and F Goulette. Accurate 3d maps from depth images and motion sensors via nonlinear kalman filtering. *IEEE International Conference on Robotics and Automation*, 2000.

[23] N Jasika, N Alispahic, A Elma, K Ilvana, L Elma, and N Nosovic. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. *MIPRO*, 2012.

[24] Reza N. Jazar. *Vehicle Dynamic Theory and Application.* Springer Science and Business Media, 19.03.2008.

[25] C Jung and W Chung. Design of test tracks for odometry calibration of wheeled mobile robots. *Intech - Open Access Publisher*, 2011.

[26] L E Kavraki, P Svestka, J-C Latombe, and H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 12, NO. 4*, 1996.

[27] S Kohlbrecher, J Meyer, T Graber, K Petersen, O von Stryk, and U Klingauf. Hector open source modules for autonomous mapping and navigation with rescue robots. *UNCERTAINTY IN ARTIFICIAL INTELLIGENCE PROCEEDINGS*, 2000.

[28] J J Kuffner and S M LaValle. Rrt-connect : An efficient approach to single-query path planning. *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, 2000.

[29] J B Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality.* Princeton University Press, 2002.

[30] Latombe, Lazanas, and Shekhar. *Robot Motion Planning with Uncertainty in Control and Sensing.* Department of Computer Science Stanford University, 2006.

[31] S M LaValle. Rapidly-exploring random trees: A new tool for path planning. *Department of Computer Science, Iowa State University*, 1998.

[32] M Leingartner, J Maurer, G Steinbauer, and A Ferrein. Evaluation of sensors and mapping approaches for disasters in tunnels. *Safety, Security, and Rescue Robotics (SSRR), 2013*, 2014.

[33] Y Li, W Cui, D Li, and R Zhang. Research based on osi model. *Communication Software and Networks (ICCSN)*, 2011.

[34] M Likhachev and D Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *Robotics:Science and Systems IV*, 2009.

[35] M Likhachev, G Gordon, and S Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. *School of Computer Science, Carnegie Mellon, University Pittsburgh*, 2003.

[36] RD Luo, KC Yeh, and KH Huang. Resume navigation and re-localization of an autonomous mobile robot after being kidnapped. *Robotic and Sensors Environments (ROSE), 2013*, 2013.

[37] HW Keat LS Ming. An investigation of the use of kinect sensor for indoor navigation. *Second Brazilian Conference on Critical Embedded Systems*, 2012.

[38] F Moosmann and c Stiller. Velodyneslam. *Intelligent Vehicles Symposium (IV)*, 2011.

[39] T Niemüller, A Ferrein, G Eckel, D Pirro, P Podbregar, T Kellner, C Rath, and G Steinbauer. Providing ground-truth data for the nao robot platform. *RoboCup 2010: Robot Soccer World Cup XIV*, 2011.

[40] L Perumal. Quaternion and its application in rotation using sets of regions. *International Journal of Engineering and Technology Innovation, vol.1, no.1*, 2011.

[41] B Puchinger. Kombot - an autonomous mobile order picking robot. Technical report, Institute for Technical Informatics, Technical University Graz, 2012.

[42] M Quigley, B Gerkey, K Conley, J Faust, T Foote, J Leibs, E Berger, R Wheeler, and A Ng. Ros: an open-source robot operating system. *icraoss09*, 2009.

[43] RS Rao, V Kumar, and CJ Taylor. Planning and control of mobile robots in image space from overhead cameras. *International Conference on Robotics and Automation*, 2005.

[44] K Rebai, O Azouaoui, M Benmami, and A Larabi. Car-like robot navigation at high speed. *IEEE International Conference on Robotics and Biomimetics*, 2007.

[45] F Sarholz, J Mehnert, J Klappstein, J Dickmann, and B Radig. Evaluation of different approaches for road course estimation using imaging radar. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[46] A Scott, LE Parker, and C Touzet. Quantitative and qualitative comparison of three laser-range mapping algorithms using two types of laser scanner data. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[47] R Siegwart and R Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[48] J Smisek, M Jancosek, and T Pajdla. 3d with kinect. *Computer Vision Workshops (ICCV Workshops), 2011*, 2005.

[49] R Stahn, G Heiserich, and A Stopp. Laser scanner-based navigation for commercial vehicles. *IEEE Intelligent Vehicles Symposiu*, 2007.

[50] I A Sucan, M Moll, and L E Kavraki. The open motion planning library. *Robotics & Automation Magazine, IEEE*, 2012.

[51] A Tatoglu and K Pochiraju. Point cloud segmentation with lidar reflection intensity behavior. *Robotics and Automation (ICRA), 2012*, 2012.

[52] S Thrun, W Burgard, and D Fox. *Probabilistic Robotic*. The MIT Press, 2006.

[53] D Vivet, P Checchin, and R Chapuis. Radar-only localization and mapping for ground vehicle at high speed and for riverside boat. *IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA*, 2012.

[54] C Weyers and G Peterson. Improving occupancy grid fastslam by integrating navigation sensors. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[55] WS Wijesoma, KRS Kodagoda, and AP Balasuriya. Road-boundary detection and tracking using ladar sensing. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 2004.

[56] W Xu, J Wei, M Dolan, H Zhoa, and H Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. *IEEE International Conference on Robotics and Automation*, 2012.

[57] H Zeltwanger. Standardized higher-layer protocols for different purposes. *CAN in Automation (CiA), iCC 2012*, 2012.

[58] X Zhang, X Chen, J Li, and X Li. Vision-based monte carlo - kalman localization in a known dynamic environment. *Control, Automation, Robotics and Vision, 2006*, 2006.

[59] Z Ziaei, R Oftadeh, and J Mattila. Global path planning with obstacle avoidance for omnidirectional mobile robot using overhead camera. *International Conference on Mechatronics and Automation*, 2014.