Master-Thesis

Outdoor Localization and Navigation for Mobile Robots

Johannes Maurer

Graz, January 2012

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

Institute for Software Technology (IST)
Graz University of Technology
A-8010 Graz, Austria

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.

Graz, _____ _____

Place, Date Signature

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.

Graz, _____ _____

Ort, Datum Unterschrift

Abstract

To give a robot the capability to handle tasks, such as shopping, cleaning or moving goods, means that the robot can navigate in the world autonomously. Autonomous navigation is still a challenging problem in mobile robotics. A wide range of robotics is dealing with the problem of moving a robot without hitting surrounding obstacles. To cope with this challenge, abilities like sensing, planning, acting and problem solving are needed. Also special hardware and architecture has to be considered.

The Institute for Software Technology already has experience in working on robot navigation for indoor applications. Moving the robot in a controlled environment like the laboratory is a well-known issue. In the contrary, our laboratory has little practical knowledge in the area of autonomous outdoor navigation this work will consider this problem. Therefore, the goal of this thesis is to design an autonomous outdoor navigation system using present technologies. One important aspect of this thesis is the evaluation of the ground truth of the developed system.

To achieve the goal of the thesis it is necessary to provide the robot with appropriate functionality. A positioning system – based on a Kalman Filter – is designed to estimate the robot position. The knowledge about its environment is provided to the robot by a graph-based map. A navigation system is implemented to control the movement of the mobile robot.

It can be seen that for developing a truly autonomous navigation system a number of challenges have to be met, for instance unexpected large sensor errors. This thesis shows possible solutions for some of these difficulties.

Contents

List of Figures

Listings

1 Introduction

The National Imagery and Mapping Agency (NIMA) define navigation as "the process of planning, recording, and controlling the movement of a craft or vehicle from one place to another" [10, p.799].

Four questions have to be answered to fulfil the functionality of navigation [11]:

Where am I going? The answer to this question is typically determined by a human operator or specified by a mission planner on a higher software level.

What is the best way there? This is the problem of path planning. Based on the information of the world the robot should find the best way to a given goal.

Where have I been? This question covers the aspect of map making (mapping). Mapping is helpful to improve the knowledge of the robot about the world when exploring a new environment. Also it is beneficial when operating in a known domain.

Where am I? The determination of the current position is an important step in the navigation process. Addressing this issue the robot has to deal with absolute and topological positions.

1.1 Motivation

"Why can't a robot go to the supermarket and buy my lunch?"

This question was the beginning for this thesis. Who ask this question should know about the number of questions that must be previously answered. To enable a robot to move autonomously in the world is one issue to be solved in order to complete such tasks.

Autonomous navigation is still a challenging problem in mobile robotics. There are a number of systems, sensors and different techniques developed for mobile robot navigation. However, there is no simple solution that covers the whole issue to this day. A wide range of robotics is dealing with the problem, how to move a robot without hitting surrounding obstacles. To cope with this challenge, abilities like sensing, planning, acting and problem solving are needed. Also special hardware and architecture has to be considered.

The Institute for Software Technology already has experience in working on robot navigation for indoor applications. Moving the robot in a controlled environment like the laboratory is a well-known problem. Since the Institute for Software Technology (IST) has little practical knowledge in the area of autonomous outdoor navigation, this work will consider this problem.

1.2 Goal

The goal of this thesis is to design an autonomous outdoor navigation system using present technologies. This thesis presupposes existing map information and will not give attention the aspect of map making.

To achieve the goal of the thesis it is necessary to provide the robot with appropriate functionality. Another important aspect of this thesis is the evaluation of the ground truth of the developed system. The key issue of this thesis will be the development of a positioning system.

In contrast to indoor navigation the problems in outdoor environments are various. Different sensors and another prior knowledge has to be used. Typical indoors environment features are quite scarce in outdoor. Furthermore, the surrounding can change because of large moving objects.

1.3 Overview

The next Chapter (Chapter 2) introduces commonly used sensors and technologies for navigation systems. The particular characteristics, as well as advantages and disadvantages of these various systems are described.

Chapter 3 deals with the theoretical background in the field of sensor fusion. Sensor fusion is a common technique used to handle uncertainties in the physical world and to compensate disadvantages of different sensors in current robot applications.

A look at related research topics is given in Chapter 4. It presents related works on navigation systems with problems similar to those tackled in this thesis.

Chapter 5 deals with one of the key difficulties of this thesis, the positioning. The approach developed in this work to estimate the position of the robot is matching measurements coming from the IMU, GPS and robot odometry using a Kalman filter. This Chapter intends to describe the concepts and mathematical models of the designed filter to the reader.

The design of the system for the outdoor navigation is characterized in Chapter 6.

The implementation software components are described in detail. A description of the base framework, the Robot Operating System (ROS), is given and the different software and hardware components of the system are defined. Special attention is given to the communication between the different working processes and the structure of the messages.

The results of the experiments are summarized in Chapter 7. This chapter presents at first the measuring process and the determination of the reference trajectory. The quality of the positioning system and the applicability of the navigation system are tested experimentally, the temporal progression of the positioning error is shown and an analysis of the results of the experiments is given.

The last Chapter (Chapter 8) gives an overview about the problems that occurred during this thesis. Furthermore, the contribution of this work to the knowledge of the institute (IST) is shown. Also points of contact for future research projects are shown.

1.4 Definitions and Terms

Within this thesis a number of important terms and definitions are used. In order to allow the reader a better understanding of the subject these terms are listed below [1]:

Position A position is a set of coordinates and an orientation related to a well-defined coordinate reference frame.

Positioning The process of determination a position is called positioning.

Location A position in terms of topological relation is called location.

Localization Localization terms the process of obtaining a location.

Map A map is a representation of parts or all of a space and describes relationships between elements of that space.

Mapping Mapping is the process of creating a map.

Trajectory A polygon connecting subsequent positions is called trajectory.

Route/Path A route/path is a list of manoeuvres to be performed to reach a destination.

Waypoint A waypoint is a point on a route.

Routing/Path Planning The process of planning a route from one position to another is called routing (or path planning).

Guidance Guidance terms the process of guiding an object along a predefined route.

2 Sensors and Techniques

This Chapter introduces common sensors and techniques for robot navigation systems. In particular sensor specific errors, dependencies and accuracies are described. The Chapter is based on the book [1] "Navigation: Principles of Positioning and Guidance" by Hofmann-Wellenhof, Legat and Wieser and the paper [12] "Mobile Robot Positioning - Sensors and Techniques" by Borenstein, Everett, Feng and Wehe.

2.1 Odometry

Odometry is an important method in robot navigation. Odometry is simple, inexpensive, easy to implement in real-time, has a good short-term accuracy and allows high sampling rates. The disadvantage is its unbounded accumulation of error. The orientation and position errors increase proportionally with the travelled distance.

The main concept of odometry is to translate the wheel revolution into a linear displacement relative to the floor. The offset from a known starting position can be computed by monitoring the wheel revolutions and simple geometric equations.

Odometry provides good relative motion information. For this fact it is often used together with absolute position measurement like satellite-based (GPS) or landmark localization.

2.1.1 Odometry Error

A well-known downside of odometric navigation is the unbounded accumulation of error. Unexpected interactions between wheel and floor and kinematic imperfections of the robot wheel implies that wheel rotations may not translate proportionally into linear motion and cause errors. Mainly the orientation errors have great influence on the inaccurateness of odometric measurement. A small error in the orientation cause a grow of the lateral position error without bound [13].

The resulting error can be categorised into three groups: errors through equation, systematic errors and non-systematic errors.

Errors Through Equation

The odometry equations approximate arbitrary motion as a series of short straight-line segments. This approximation can cause odometry errors depending on the sampling frequency with respect to the speed. For typical sampling rates $T_S < 10ms$ and robot speeds $V < 1m/s$ this error is insignificant [12].

Non-Systematic Errors

Non-systematic errors are caused by unexpected interactions between wheel and floor. This unpredictability is a great problem for actual applications because it is impossible to predict an upper bound for the odometry error.

Error sources for non-systematic errors are [13]:

- travel over uneven floors
- travel over unexpected objects on the floor
- wheel slippage due to
  - slippery floor
  - over-acceleration
  - fast turning
  - external forces
  - internal forces
  - non-point wheel contact with the floor

Systematic Errors

Systematic errors are vehicle specific and constant over prolonged periods. This group of errors are particulaly serious because they accumulate constantly. The accuracy of odometry measurement can be improved by counteracting to the individual contributions of systematic error sources.

Error sources for systematic errors are [13]:

- unequal wheel diameters
- average of both wheel diameters differs from nominal diameter

- misalignment of wheels
- uncertainty about effective wheelbase
- limited encoder resolution
- limited encoder-sampling rate

The systematic error usually does not chance during runs unless big change in the distribution of load or physical deformation happen.

The three main sources for systematic errors are: scaling error, uncertainty effective wheelbase and unequal wheel diameters.

Scaling Error The Scaling Error appears if the average of both wheel diameters differs from nominal diameter. It affects straight-line motion and turning motion. This significant error can be corrected with just an ordinary tape measure.

Unequal Wheel Diameters Error Unequal wheel diameters are produced by asymmetric load distribution, inexact manufacture or unequally pumped wheels. This error affects only during straight-line motion.

Uncertainty Effective Wheelbase Error The wheelbase is the distance between the contact points of the wheels of a differential-drive. Uncertainty is caused by the fact that rubber contacts the floor not in one point, but rather in an area. This error affects only during turning.

A method to measure and correct unequal wheel diameters and uncertainty effective wheelbase is the University of Michigan Benchmark, developed by Borenstein and Feng.

2.1.2 UMBmark (University of Michigan Benchmark)

The University of Michigan Benchmark (UMBmark) described in [13] is a method to test and improve odometry performance. In this approach the mobile robot has to follow a predefined square path in clockwise and counter-clockwise direction, as described later. The quantitative measurement of odometry error can be quantified by comparing the absolute position to the calculated position. From a set of equations a numerical value that expresses the odometry accuracy and two calibration constants can be defined. Including the determined constants to the odometry computation will reduce the systematic errors of the mobile platform.

Description of the UMBmark Procedure

The Bidirectional Square Path Method is defined in [13] as the following procedure.

1. The starting position of the odometry of the on-board odometer has to be initialised to the absolute position and orientation of the vehicle.
2. The robot has to run through a pre-programmed 4x4 meter square path in clockwise direction.
3. Measure the absolute position and orientation of the vehicle after the run.
4. Compare the absolute position to the robot's calculated odometry position.
5. Repeat steps 1-4 for four more times.
6. Repeat steps 1-5 in counter-clockwise direction.
7. Determine the measure of odometric accuracy for systematic error.

2.2 Inertial Navigation

Inertial Navigation is widely used to determine position, velocity and alignment of vehicles. Inertial navigation systems do not need external references for this reason they are useful for autonomous navigation technique. Another advantage of the system is the high reliability and the instantaneously and continuously measurements at high data rates.

The measurement sources for the system are gyroscopes and accelerometers mounted together on a platform called Inertial Measurement Unit (IMU). The accelerometers measure the acceleration along the axes of a well-defined reference frame. The gyroscopes are used to stabilize the coordinate axes mechanically or analytically. The positioning is done integrating rate of rotation measured by the gyroscopes once and the acceleration values from the accelerometers twice in respect to the initial values.

Some difficulties affect the computation of the position in an inertial system. One of them is the superimposition by gravitational forces on or close to large masses like the earth. In addition, rotations of the reference frame with respect to inertial space affect acceleration measurement. Furthermore, inferences of the gravitational field and possible appeared forces must be accounted when integrating the measured forces.

This self-contained navigation method is unsuitable for positioning over long periods, because the data drifts over time. One source for degradation of the position measurement is constant sensor bias. Another drawback is the fact, that high-quality systems are quite expensive.

Inertial navigation systems are usually used to balance the drawback of the other systems. In robot navigation inertial navigation is integrated to minimize the orientation error of the robot, because it is not affected by jamming or spoofing.

2.2.1 Inertial Measurement Unit

Inertial Measurement Unit is the main measurement source for inertial navigation systems. An IMU is characterized by multiple accelerometer and gyroscopes mounted together on one platform, three orthogonal sensors of both types. The accelerometers measure the specific force along the platform axes and the gyroscopes detect the angular rates.

Two different types of platforms exits:

Gimballed or Stabilized Platform: In this construction the platform is isolated from the rotational dynamic of the vehicle. Using the determined alignment of the platform the platform is kept stable with respect to the chosen navigation frame.

Strapped Down Platform: In this design the sensors are mounted directly on the platform and follow the motion of the vehicle. The measurements are performed in the body frame of the moving object and are transformed to the navigation frame by an analytic system. Strapped down systems are the most frequently used implementations in practice.

Accelerometer

The basic concept of accelerometers is to measure the force acting upon a proof mass. The two common implementations of acceleration sensors are open loop and close loop. In an open loop setup the displacement of the proof mass is measured. A closed loop accelerometer generates generating an electric or magnetic force that keeps the mass in a state of equilibration. The generated force counteracts the specific force on the mass.

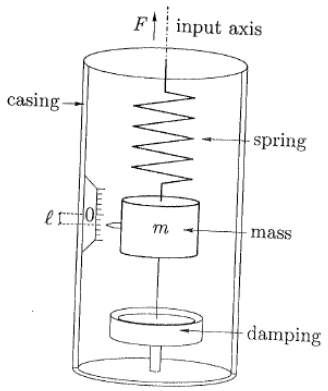

Figure 2.1: Schematic diagram of an open loop spring accelerometer [1].

The accuracy of an acceleration measurement is influences by a number of factors. The

significant considerations are thermal sensitivity and cross- axis sensitivity.

Positioning by using the acceleration date is not possible for robot navigation because of the pore signal-to-noise ratio at low accelerations.

Gyroscopes

Gyroscopes sense the angular rate of the platform with respect to inertial frame.

Mechanical gyroscopes use rotating components to determine the angular rate. The main error source for mechanical gyroscopes are mass imbalance effects, temperature sensitivity and sensitivity to external magnetic fields.

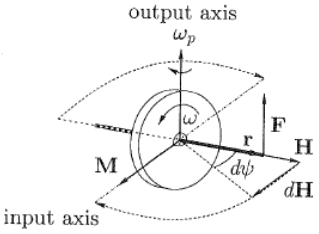

Figure 2.2: Precession of a spinning body by an external force [1].

The main concept of optical gyroscopes is based on the relativistic Sagnac effect, the increase of the path length for the laser beam in counter-rotating direction. The accuracy of optical gyroscopes is influenced by glass flow, polarization and electric effects. Furthermore, insensitivity to low angular rates.

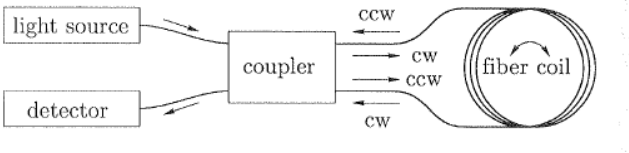

Figure 2.3: Schematic diagram of an interferometric fibre optic gyro [1].

High accurate gyros are still expensive. The main operational areas are commercial airlines.

Gyroscopes are often used to minimize the orientation error, because any small orientation error causes a growing lateral position error.

Magnetic Compass

The magnetic compass is an absolute sensor. The absolute heading is determined by measuring the earth magnetic field.

The most significant down side of magnetic sensors is the distortion of the measurement near power-lines and steel structures. Because of these facts it is difficult to use compass measurements for indoor applications.

Magnetic compasses are important for determine the heading of a robot to reduce the influence of accumulated error.

2.3 Landmark Navigation

The principle concept of landmark-based navigation is to detect features recognizable by the sensing system and derive the position of the robot from landmark position. The characteristics of the features must be known and stored in the knowledge base of the robot. Moreover, landmarks have to be in the environment around the robot.

Landmarks have to be easy to identify. Geometric shapes and bar codes are well suited for this purpose.

The common used sensor system is computer vision. Therefore, more processing is necessary. But current algorithms are able to compute that task in real-time. If the robot position is known, the processing can be simplified by looking for landmarks in a limited area around the robot. In addition, knowledge over the starting point reduces erroneous interpretation during the initialization.

An active area of research using landmarks is Visual SLAM (V-SLAM). In localization V-SLAM is used to realize drift free motion estimation [14].

A distinction is made between natural and artificial landmarks [12].

2.3.1 Natural Landmarks

Natural landmarks are objects that are already in the environment and have a function other than robot navigation. Natural landmark navigation works best in man made high structured environments like corridors, manufacturing floors or hospitals.

For example, the visually guided mobile robot ARK (Autonomous Robot for a Known environment) uses a natural landmark navigation system [15].

The advantage of this type of landmarks is their flexibility and the fact that no morti-

fication of the environment is necessary. A common problem is to detect and match the features from sensor input.

Computer vision is frequently used to catch natural landmarks. Particularly doors, wall junctions and ceiling lights are perfect features.

If a range sensors like a laser scanner is used corners, edges or long straight walls can be detected as landmarks.

2.3.2 Artificial Landmarks

Artificial landmarks are specially designed markers that are mounted in the environment. The detection is much easier in contrast to natural features because they have an optimal contrast, an exact size and a well-defined shape. Another advantage is that they are inexpensive and can encode additional information.

Figure 2.4 shows an example for an artificial landmark. This Quick Response (QR) code, a type of matrix barcode, can be mounted.

Figure 2.4: Example for an artificial landmark. QR tag with encoded text "kitchen".

Also for detecting artificial landmarks, computer vision is the common choice. Often used features are diamond-shaped landmark, reflecting material patterns or light sources.

However, also bare-coded reflectors can be used together with laser scanner in an artificial landmark positioning system.

2.3.3 Positioning Accuracy

The accuracy of a landmark navigation system depends on the quality of the feature point extraction. In fact, a precise localization of present landmarks leads to more precise results for the robot position.

Especially the exactness of vision systems depends on the relative position and angle between the robot and the landmark and ambient lighting conditions. For this reason

features with shorter distance to the robot should be favoured.

2.4 Satellite-based Navigation

Global Navigation Satellite System (GNSS) is a simple technology for determination of time, position and velocity. The basic concept is to use signals from satellites of which the position is known to determine an unknown position.

The use of satellite-based navigation systems is accurately and inexpensive. It can be used continuously and globally everywhere outdoors where a sufficient number of signals can be received.

The U.S. Global Positioning System (GPS) and the Russian Global Positioning Satellite System (GLONASS) are at present best-known examples for satellite-based positioning systems. The European project Galileo is still under construction.

2.4.1 Global Positioning System

U.S. Department of Defence (DOD) operates the Global Positioning System. It consists of twenty-four satellites in six orbits. The satellites have a period to make one complete orbit of around 12 hours and transmit encoded RF signals.

Code division multiple access (CDMA) is used to decide the signals from the different satellites. All satellites use the same two carrier frequencies. Each satellite has assigned a PRN (Pseudo Random Noise) code for modulation of the signals. The signal from the satellites contains information of the current locations of the satellites.

The satellite-based navigation system uses an advanced trilateration methods. Based on code or carrier phase measurement the travel time of the signals is identified. By multiplying the travel time of the signals with its velocity, pseudo-ranges are computed.

The determination of the position is done by using these pseudo-ranges. See Figure 2.5 to get an idea of the concept. Because of not perfectly synchronised clocks of satellites and receiver the equation for the position determination is consisting of four unknowns, three point coordinates and the clock error. Therefore, a minimum number of four satellites is necessary to compute position of the receiver.

2.4.2 GLONASS

The GLONASS satellite based system is operated by the Russian military space force (MSF). The principles are very similar to the GPS system.

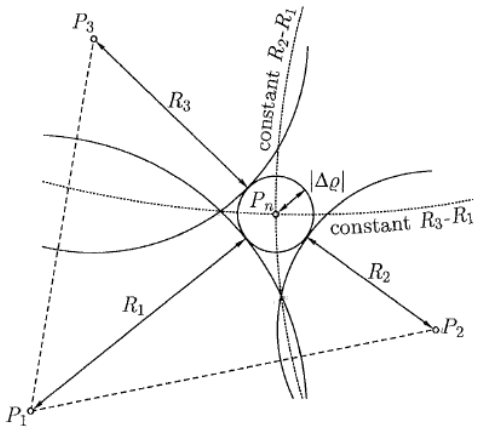

Figure 2.5: Pseudo-range position determination [1].

The most significant difference between GLONASS and the U.S. system is the transmission of the satellites signals. The Russian system uses frequency division multiple access (FDMA) to divide the signals from the different satellite. In other words, two individual carrier frequencies are assigned to any of the satellite.

2.4.3 Differential GNSS

Differential GNSS is a basic approach to improve the accuracy of satellite based positioning systems. The principle concept is to use a second receiver that experiences the same error effects. The second receiver has a fixed and known position. A typical accuracy around 2.5 meters can be achieved.

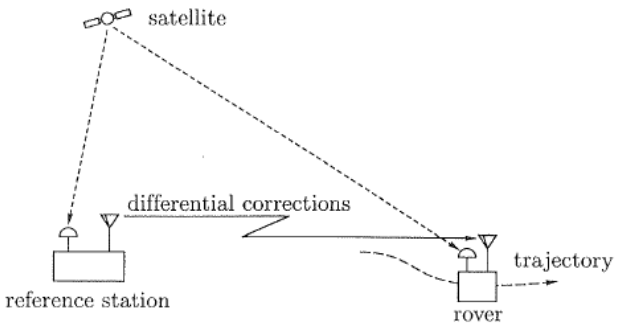

Figure 2.6: Principle of differential GNSS [1].

The fixed reference station determines the error influence in the satellite signals and

sends them to the mobile receiver. The mobile and the reference GPS must be close enough to each other. DGPS is often used in commercial systems. Local services provide the correction data. Figure 2.6 illustrates the concept. Two concepts to improve the positioning quality are in use:

The first method determines the correction data with regard to the position. The known position of the fixed receiver is compared with the determined position from the system. A disadvantage of this approach is the problem of coordinated satellite selection. The receivers have to use the same satellites.

The second concept is to determine the correction factors for the measurement. The reference station compares the pseudo-range measurements from the satellite signals with the expected values. The drawback of this approach is the higher computational effort to compute the measurement correction for each satellite signal.

3 Basics of Sensor Fusion

This Chapter introduces to the principles of probabilistic in robot applications and common algorithms are presented. It is based on the book "Probabilistic Robotics" [8] by Thrun, Burgard and Fox.

3.1 Introduction

Uncertainties are a challenge in building a robot application that precepts the physical world and acts in it. Uncertainties in measurements and controls of the robot need to be taken in tribute. Probabilistic algorithms can be used to pay respect to the uncertainty in robot perception and action. A robot system using these algorithms is robust relative the uncertainty.

The calculus of probability theory and specific probabilistic laws are used to represent uncertainty in the robot's knowledge. The state transition and the measurement are expressed by probability distributions. A dynamic System is used to model the robot and the environment. Controls, Sensor measurements, and the state of the robot and its environment are expressed as random variables. The belief of the robot is represented by the posterior distribution over the state.

The general algorithm for computing beliefs is the Bayes filter. The following Sections will introduce this recursive algorithm for state estimation and present two common implementations of the concept.

3.2 Bayes Filter

The Bayes filter is the most general algorithm for calculation belief distribution *bel* from measurements and control. The main concept is to make a Markov assumption that implies that the belief is sufficient to express the past.

The basic recursive algorithm is shown in Listing 3.1. The state is represented by x_t. The belief over the state $bel(x_t)$ at time t is estimated from the previous belief $bel(x_{t-1})$ at time $t-1$. Each recursion step gets the recent belief $bel(x_{t-1})$, the current control variable u_t and the last measurement values z_t as input. An initial belief $bel(x_0)$ is

required to start the recursion.

```
algorithm bayes-filter(bel(x_{t-1}), u_t, z_t):
    for all x_t do
        \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}
        bel(x_t) = \eta p(z_t|x_t) overline bel(x_t)
    endfor
    return bel(x_t)
```

Listing 3.1: Pseudo-algorithm for the basic Bayes filter. [8]

The two essential steps for al state variables are the prediction step and the correction step. In the first step, update or prediction step, the belief $\overline{bel}(x_t)$ over the state x_t is calculated. The next step multiplies the probability that the measurement z_t may have been observed by the computed belief $\overline{bel}(x_t)$. Finally the belief is normalized to represent a probability by the normalization constant η.

The Bayes filter is not a practical algorithm. A few algorithms implement the concept of the Bayes filter in different ways. The next Sections introduce two of them, the Kalman filter from the Gaussian filters family and the particle filter as example for non-parametric filters. Both are common used in robot applications.

3.3 Kalman Filter

Gaussian filters are an important family of recursive state estimators. One of the estimation methods most used is the Kalman filter [16] described in this Section. The principle algorithm filters and predicts the state in linear defined Gaussian systems. The beliefs in Gaussian filters are represented by multivariate normal distributions. At time t the belief is represented by the mean μ_t and the covariance Σ_t.

The state transition probability $p(x_t|u_t, x_{t-1})$ is represented by the linear equation 3.1.

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{3.1}$$

As shown in equation 3.2 the measurement probability $p(z_t|x_t)$ is described as a linear function.

$$z_t = C_t x_t + \delta_t \tag{3.2}$$

The pseudo-algorithm for the Kalman filter is shown in Listing 3.2, a computationally quite efficient algorithm. As the Kalman filter is a realization of a Bayes filter the algorithm includes the steps prediction and correction.

```
algorithm kalman-filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):
    μ̄_t = A_t μ_{t-1} + B_t u_t
    Σ̄_t = A_t Σ_{t-1} A_t^T + R_t
    K_t = Σ̄_t C_t^T (C_t Σ̄_t C_t^T + Q_t)^{-1}
    μ_t = μ̄_t + K_t (z_t - C_t μ̄_t)
    Σ_t = (I - K_t C_t) Σ̄_t
    return μ_t, Σ_t
```

Listing 3.2: Pseudo-algorithm for the Kalman filter. [8]

In the fist step the predicted belief $\overline{bel}(x_t)$ is computed. The belief is represented by the mean μ_t and the covariance Σ_t. Next the Kalman gain K_t is computed. It indicates the degree how the new state estimation is incorporated by the measurement. Finally the new mean μ_t and the new covariance Σ_t of the posterior belief is computed.

The wide range of applications for the Kalman filters comprises elegant dead reckoning navigation systems [17] and speech enhancement [18].

3.3.1 Extended Kalman Filter

The Extended Kalman Filter (EKF) is an extension of the Kalman algorithm. The standard Kalman filter assumes that the state transition and measurement are linear functions. State transition and measurement are rarely linear in practice in common applications.

For this reason the state transition and measurement are characterized by non-linear functions in the extended Kalman filter. The matrices A_t and B_t from the equation 3.1 are replaced by the function g as shown in equation 3.3. Equation 3.4 shows that for measurements the matrix C_t from the equation 3.2 is replaced by function h.

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3.3}$$

$$z_t = h(x_t) + \delta_t \tag{3.4}$$

The key idea of the extended Kalman filter is the linearization of the non-linear functions g and h. The advantage of that approach is that once g is linearized the propagation step is equal of those of the Kalman filter. A Taylor series expansion is used to approximates the non-linear function g by a linear function and a Gaussian approximation to the true belief is calculated.

The advantage of the EKF is its efficiency even the linearization causes an error.

Another Kalman filter extension to relax the linearity assumption is the **Unscented Kalman Filter (UKF)**. This algorithm performs a stochastic linearization called unscented transform. The key idea is to use weighted statistical linear regression.

3.4 Particle Filter

The particle filter is a non-parametric implementation of the Bayes filter and very popular in robotics. In contrast to the Kalman filter the posterior belief is not represented as a function, it is approximated by a finite number of values.

The basic idea of the particle filter is to represent posteriors $bel(x_t)$ by finitely many random state samples, called particles. A set of random state samples of the posterior distribution, as shown in equation 3.5, can represent a much broader space than Gaussian distribution. In additional can the number of particles be adapted dynamically to the complexity of the posterior.

$$\chi_t := x_t^{[1]}, x_t^{[2]}, ...x_t^{[M]} \tag{3.5}$$

The particle filter approximates the belief $bel(x_t)$ by the set of particles χ_t Such as the basic recursive principle of the Bayes filter, the particle filter constructs the belief recursively. Listing 3.3 demonstrate the pseudo-algorithm for basic variant of a particle filter.

```
algorithm particle-filter(χ_{t-1}, u_t, z_t):
    χ̄_t = χ_t = ∅
    for m = 1 to M do
        sample x_t^[m] p(x_t|u_t, x_{t-1}^[m])
        w_t^[m] = p(z_t|x_t^[m])
        χ̄_t = χ̄_t + ⟨x_t^[m], w_t^[m]⟩
    endfor
    for i = 1 to M do
        draw i with probability αw_t^[i]
        add x_t^[i] to χ_t
    endfor
    return χ_t
```

Listing 3.3: Pseudo-algorithm for the particle filter. [8]

As a realization of a Bayes filter the particle filter algorithm includes a prediction step and a correction step. The algorithm constructs a temporary particle set $\overline{\chi}_t$ representing the belief over the state $\overline{bel}(x_t)$. Next the particles are incorporated to the measurement and weights for each particle are determined. Finally the particles are transformed into χ_t, the posterior distribution $bel(x_t)$ of the Bayes filter.

A example application for particle filter is the simultaneous localization and mapping (SLAM) problem in mobile robot [19].

4 Related Research

4.1 DGPS and Odometry Data Fusioning System

In [20] 'Outdoor navigation of a mobile robot between buildings based on DGPS and odometry data fusion' the authors Ohno, Tsubouchi, Shigematsu, Maeyama and Yuta present an approach for a map based outdoor navigation. This Section provides a review of their aim.

The authors use an Extended Kalman Filter for modification and fusion of odometric and DGPS measurement data. The robot has to follow a path on a provided map. During the run position and orientation of the robot are continuously measured. The authors categorize the walkway environment in walkway in open space, walkway between buildings and walkway among roadside trees. The paper put special attention on the influences of buildings on the accuracy of the positioning of the robot.

4.1.1 The Test Platform

For the experiments the YM2000 a differential motion type mobile robot is used. It provides odometry (x_O, y_O, θ_O) information by integrating the driving wheel rotations. As DGPS the Trimble DSM12/212 receiver that provides position data $(x_{GPS}, y_{GPS}, \theta_{GPS})$.

The fusion of the odometry data with the DGPS data is performed by an Extended Kalman Filter. In locations near buildings the error of measured position by DGPS tends to be large. The authors aim to identify measurements with such large errors and not use them for the incorporation.

4.1.2 DGPS

As described in Section 2.4 DGPS (Differential GPS) is a satellite based positioning system. It uses radio waves from artificial satellites to determine a position. Position data is computed from time of fly of GPS radio waves. In addition DGPS can determine velocity and heading measuring the Doppler shift of GPS radio waves. A known problem is that the measurement accuracy could be wrong when buildings or threes interfere with the radio waves from the satellites.

The authors prefer to use DGPS for there approach. Even though RTK-GPS has a high measurement accuracy of several centimetres order position error it is not always effective among buildings and trees, whereas DGPS measure a position as far as there are enough satellites signals.

The authors work on the characteristics of DGPS in the different walkway environments. They show that especially on walkways between buildings multi-path phenomenon adversely affect the position measurement. The experiments show that the error of the heading direction data near buildings is small even when the position error is large.

4.1.3 Correction Method

The authors developed two correction methods. The first method is to use the position information from the GPS to correct the robot position. The second approach uses the heading direction provided by the DGPS sensor.

For the correction it is important to identify large errors in GPS data and to remove such inaccurate data. One approach would be to use the DOP (Dilution of Precision) data out of the GPS receiver to identify and eliminate GPS measurements with large error. The paper shows that using the DOP information to eliminate inaccurate data is not suitable in walkway environments near buildings. The authors propose a method using the likelihood of the GPS measurement based on the odometry position.

4.1.4 Evaluation

The authors describe the evaluation as following. The robot is driven by human accurately along a known path. The examination of the estimated error of the position using only the odometry measurements and the error ellipse after the correction method has shown that the robot position can be modified by DGPS measurements. The two correction methods are evaluated separately. Furthermore, it was observed that using only the GPS heading direction would improve the positioning.

To check the correction method of the robot position the authors also made a trial run of autonomous mobile robot.

4.2 GPS and Inertial Data Fusioning System

This section presents the paper 'An Outdoor Navigation System Using GPS and Inertial Platform' [21]. The authors Panzieri, Pascucci and Ulivi present a Kalman Filter based localization algorithm that fuses information coming from a GPS with inertial data and

map-based localization. In addition, they show difficulties and possible solutions of this sensor fusion problem.

In contrast to indoor navigation the problems in outdoor environments are various. Different sensors and another prior knowledge has to be used. Typical indoors environment features are quite scare in outdoor. Furthermore, the surrounding can change because of large moving objects. In this context, the global positioning system (GPS) is described as an interesting possibility for positioning in outdoor environment. The error is independent of the travelled distance. Until May 2000 the available precision was downgraded for military reasons. Today a simple cheap GPS unit can be used to achieve the full satellite precision. Characterize the accuracy of the GPS measurement in different situations is the real problem.

4.2.1 Performance of a Simple GPS System

A GPS receiver is able to provide values to characterize the precision of the measurement. The DOP (dilution of precision) value describes the accuracy in relation to the number and geometry of the available satellites. The EPE (estimated position error) is a statistical value that appreciates the correctness of the measurement based on information from the filter integrated in the receiver.

To evaluate the accuracy of the simple GPS receiver the authors perform some experiments. Position measurements were performed in two different points. The first position was far from obstacles affecting the radio waves from the satellites. The other one was between two buildings. The number of available satellites was different in the two points. Observing the standard deviations it can be seen that the number of satellites is just a rough indication of the precision.

The authors find that the DOP and EPE value are not suitable for the purpose in this paper. Furthermore, the authors come to the conclusion that the absolute error is slow variating. They use the GPS system as relative positioning sensor and reset the absolute position each time a known environment feature is approached.

4.2.2 Experimental Results

The experiments run on an ATRV-Jr. robot with differential kinematics moving on a parking lot. As GPS receiver the Garmin GPS35-HVS is used, the DMU-6X is used as 6-DOF inertial sensor and a LMS220 laser scanner is mounted on the robot. An EKF is used to fuse the sensor data. As a consequence that the position error is corrected near known reference points, the authors design the path planning algorithm to pass near known environment features. In addition, if DOP information is available the reconfiguration of the real time trajectory is affected.

In the experiment the robot is moving from the starting point to a goal passing three defined points. The examination of the progress of the error shows that using GPS measurement to improve the positioning of the robot when matching with the map is not possible even using inexpensive receivers. The authors note that the positioning can be improved if the strategy for path planning considers the position error and passes near easy recognizable environment.

5 Sensor Fusion Concept

Positioning is one of the key issues of a mobile autonomous outdoor robot and of this thesis. The positioning is done by matching measurements coming from the IMU, GPS and Robot odometry. This Chapter intend to describe the concepts and mathematical models of the designed filter to the reader.

5.1 Filter Design

The principle concept of the filter is shown in Figure 5.1. The incoming measurement data from the sensors is pre-processed and handed to the filter object. The sensor measurements are unsynchronised. For this reason the estimation of the position, the filter update step is data driven. This means that for each incoming measurement message a new estimation and correction step is performed.

5.2 Odometry Preprocessor

The Odometry preprocessor is designed to prepare the odometric measurement data.

When a new odometry sensor value arrives this function is executed. A sensor measurement consists of the estimated position (x, y, θ), the actual linear and angular velocity of the robot $(\Delta x, \Delta\theta)$ and the corresponding covariance. More details can be found in Section 6.3.

The Odometry equations:

$$\dot{x}_{odo} = v * cos(\phi_{imu}) * cos(\theta) \tag{5.1a}$$

$$\dot{y}_{odo} = v * cos(\phi_{imu}) * sin(\theta) \tag{5.1b}$$

$$\omega_{z,odo} = \omega_{z,odo} \tag{5.1c}$$

The covariance matrix of the Gaussian distribution of the odometric data is R_k^{odo}. The error propagation for the functions 5.1a, 5.1b and 5.1c can be computed using the equation 5.2a. The related covariance matrix and the Jacobian matrix is shown in equation 5.2c and 5.2b.

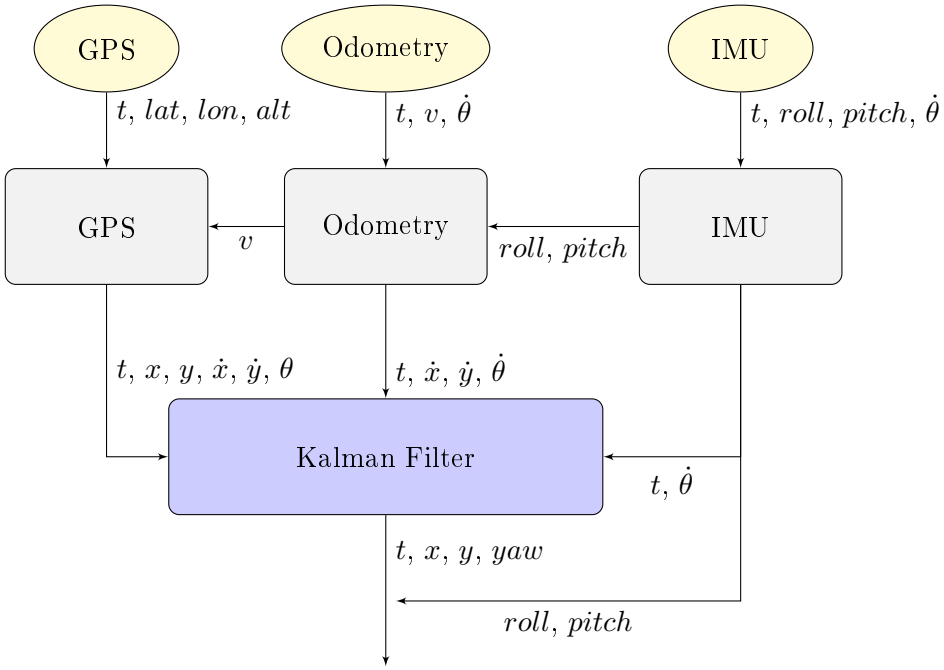

Figure 5.1: Design of the Outdoor Positioning Node.

$$R_k^{odo} = N_k * R_{k,odo} * N_k^T \tag{5.2a}$$

$$N_{k,odo} = \begin{bmatrix} cos(\phi_{imu}) * cos(\theta) & cos(\phi_{imu}) * sin(\theta) & 0 \\ -v * cos(\phi_{imu}) * sin(\theta) & v * cos(\phi_{imu}) * cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \tag{5.2b}$$

$$R_{k,odo} = \begin{bmatrix} \sigma^2_{v_{odo}} & 0 & 0 \\ 0 & \sigma^2_{\theta} & 0 \\ 0 & 0 & \sigma^2_{\omega_{z,odo}} \end{bmatrix} \tag{5.2c}$$

5.3 IMU Preprocessor

The IMU Preprocessor prepares the measurement data from the IMU for the sensor fusioning.

The function is called when new measurement data arrives. A sensor measurement consists of the acceleration in m/s, the rotational velocity in rad/sec and the orientation as in Section 6.4 described. For the positioning system the measurement of the rotational velocity around the Z-axis $\omega_{z,imu}$ and the orientation around the X-axis *roll* and Y-axis

pitch is used.

The compass information, orientation around the Z-axis *yaw*, is not used for this project. In urban environment the distortions of the earth magnetic field near power-lines and steel structures lead to inaccurate measurements.

The IMU is the only sensor that provides global orientation information. The measurement of the gyroscopes and the accelerometer are already respected for the estimation within the IMU. As a consequence, values of the orientation around the X-axis *roll* and Y-axis *pitch* bypass the filter.

The rotational velocity $\omega_{z,imu}$ is used to improve the date from the odometry and to minimize the orientation error of the robot. The sampling rate of the IMU measurements is 100Hz. To reduce the measurement noise the preprocessing function samples down the rate to 10 Hz by averaging. The corresponding equation is 5.3a.

$$\omega_{z,imu,mean} = \frac{\sum \omega_{z,imu}}{10} \tag{5.3a}$$

For every tenth message the Kalman Filter object is triggered to compute the next estimation. The measurement uncertainty matrix R_k^{imu} of the rotational velocity measurement $\omega_{z,imu}$ is computed by the equation 5.4a.

$$R_k^{imu} = \left[\sigma^2_{\omega_{z,imu}}\right] \tag{5.4a}$$

A ROS parameter defines the standard deviation $\sigma_{\omega_{z,imu}}$ of the rotational velocity measurement.

5.4 GPS Preprocessor

The GPS Preprocessor prepares the position information from the GPS for the sensor fusioning in the filter.

When a new gps message arrives this function is executed. A message contains the location latitude, longitude and altitude and the position covariance relative to a tangential plane through the reported position. For detailed information see Section 6.5.

To work with the data in the filter the position data latitude, longitude and altitude specified using the WGS-84 reference ellipsoid have to be converted in local Cartesian coordinates x_{gps} and y_{gps}. Figure 5.2 illustrates the relationships between the reference frames. This conversion is performed in the preprocessor.

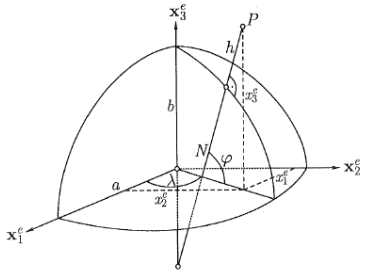

Figure 5.2: Cartesian and ellipsoidal coordinates [1].

The filter is designed to use the position change of consecutive position measurements to minimize the orientation error and stabilize the velocities relative to the tangential plane of the robot. The concept works particularly for a moving robot. Therefore, the preprocessing function respects two cases based on the velocity reported by the odometry.

The first case is a stationary or slowly moving robot. The approach of supporting the velocity and heading estimation using GPS position measurements is not workable because of relative big noise in the position measurement. For example, the deviation of the measured position of a still standing robot would cause a rotation around the Z-axis of the robot. The input vector for the correction step during the filter update contains x_{gps} and y_{gps}. The values are received from the conversion from ellipsoid coordinates. The corresponding equations are 5.5a and 5.5b.

$$x_{gps} = x_{gps} \tag{5.5a}$$

$$y_{gps} = y_{gps} \tag{5.5b}$$

The associated matrix R_k^{gps} which represents the measurement uncertainty is shaped in equation 5.6a. As parameters for the Gaussian distribution of the position $\sigma_{x_{gps}}$ and $\sigma_{y_{gps}}$ are used the covariances included in the GPS message. If no covariances are submitted with the measurement, standard deviations defined as ROS parameter are used.

$$R_k^{gps} = \begin{bmatrix} \sigma_{x_{gps}}^2 & 0 \\ 0 & \sigma_{y_{gps}}^2 \end{bmatrix} \tag{5.6a}$$

The second case is that the robot is moving faster than a defined velocity. In this case the GPS measurements are used to back the orientation and the planar velocity estimation in the Kalman Filter. The input vector for the correction step during the filter update contains x_{gps}, y_{gps}, θ_{gps}, $\dot{x}_{gps}$ and $\dot{y}_{gps}$. The local positions x_{gps} and y_{gps}

are computed using the Geographic Library like described above. The yaw angle and the velocities in x and y direction are calculated out of the position change of consecutive position measurements. The corresponding equations are 5.7c, 5.7d and 5.7e.

$$x_{gps} = x_{gps} \tag{5.7a}$$
$$y_{gps} = y_{gps} \tag{5.7b}$$
$$\theta_{gps} = atan2(\frac{\Delta y_{gps}}{\Delta x_{gps}}) \tag{5.7c}$$
$$\dot{x}_{gps} = \frac{\Delta x_{gps}}{\Delta t_{gps}} \tag{5.7d}$$
$$\dot{y}_{gps} = \frac{\Delta y_{gps}}{\Delta t_{gps}} \tag{5.7e}$$

For the associated matrix R_k^{gps} (eqn.:5.8a) the effect of error propagation of the relative uncertainty had to be considered. The relative measurement uncertainty quantities are $\sigma_{x_{gps}}$, $\sigma_{y_{gps}}$ and $\sigma_{t_{gps}}$.

Equation 5.8c shows the error propagation for the function to compute the planar velocities in x and y direction. The corresponding Jacobian matrix is 5.8d

The error propagation for θ_{gps} can be calculated using the equation 5.8e and the related Jacobian matrix 5.8f.

$$R_k^{gps} = \begin{bmatrix} \sigma^2_{x_{gps}} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{y_{gps}} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\theta_{gps}} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\dot{x}_{gps}} & \sigma_{\dot{x}_{gps}\dot{y}_{gps}} \\ 0 & 0 & 0 & \sigma_{\dot{x}_{gps}\dot{y}_{gps}} & \sigma^2_{\dot{y}_{gps}} \end{bmatrix} \tag{5.8a}$$
$$R_{x,y,t} = \begin{bmatrix} \sigma^2_{x_{gps}} & 0 \\ 0 & \sigma^2_{y_{gps}} \end{bmatrix} \tag{5.8b}$$
$$R_{\dot{x},\dot{y}} = \begin{bmatrix} \sigma^2_{\dot{x}_{gps}} & \sigma_{\dot{x}_{gps}\dot{y}_{gps}} \\ \sigma_{\dot{x}_{gps}\dot{y}_{gps}} & \sigma^2_{\dot{y}_{gps}} \end{bmatrix} = N_k * 2 * R^{gps}_{x,y,t} * N_k^T \tag{5.8c}$$
$$N_{k,\dot{x},\dot{y}} = \begin{bmatrix} \frac{1}{\Delta t_{gps}} & 0 \\ 0 & \frac{1}{\Delta t_{gps}} \end{bmatrix} \tag{5.8d}$$
$$R_{\theta_{gps}} = \begin{bmatrix} \sigma^2_{\theta_{gps}} \end{bmatrix} = N_k * 2 * R_{x_{gps},y_{gps}} * N_k^T \tag{5.8e}$$
$$N_{k,\theta_{gps}} = \begin{bmatrix} \frac{-\frac{\Delta y_{gps}}{\Delta x^2_{gps}}}{1+(\frac{\Delta y_{gps}}{\Delta x_{gps}})^2} & \frac{\frac{1}{\Delta x_{gps}}}{1+(\frac{\Delta y_{gps}}{\Delta x_{gps}})^2} \end{bmatrix} \tag{5.8f}$$

5.5 Linear System Model

$$x_{k+1} = A * x_k \tag{5.9a}$$

$$x_k = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} & \omega_z \end{bmatrix}^T \tag{5.9b}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9c}$$

The design of the positioning permits a simple linear equation (5.9a) to calculate the position of the robot.

Equation 5.9b demonstrates the state vector of the filter. It contains the two dimensional position and heading (x, y, θ) as well as the linear velocities relative to the tangential plane $(\dot{x}, \dot{y})$ and the rotational velocity around the Z-axis (ω_z).

The translation from one estimated state to another is defined by the translation matrix A as described in equation 5.9c. The variable Δt in the system model represents the amount of time which has passed since the last estimation.

To respect the uncertainties in the estimation process the system noise has to be modelled. Equation 5.10a defines the system noise matrix. The basic standard deviations of the system noise are the linear acceleration in the X and Y direction $\sigma_{\ddot{x}}$ and $\sigma_{\ddot{x}}$, and the rotational acceleration around the Z-Axis $\sigma_{\ddot{\theta}}$. The error propagation of the accelerations into the deviation of the estimation is defined by the matrix 5.10*b*. Like in the state estimation also for determination of the system deviation the amount of time which has passed since the last estimation Δt is important.

The system model's extra uncertainty:

$$Q_k = N_k * R_k * N_k^T \tag{5.10a}$$

$$N_k = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \tag{5.10b}$$

$$R_k = \begin{bmatrix} \sigma_{\ddot{x}}^2 & 0 & 0 \\ 0 & \sigma_{\ddot{x}}^2 & 0 \\ 0 & 0 & \sigma_{\ddot{\theta}}^2 \end{bmatrix} \tag{5.10c}$$

5.6 Linear Measurement Models

Using pre-processor functions to prepare the sensor data for the correction step and to compute the belonging Gaussian distributions lead to simple linear measurement models.

5.6.1 Odometry Measurements

The Odometry linear measurement model for the correction step is shown in equation 5.11a. The velocity information from the odometers is used to correct the prediction of the tangential plane $(\dot{x}, \dot{y})$ and the rotational velocity around the Z-axis (ω_z).

$$z_{k+1} = H * x_{k+1} \tag{5.11a}$$

$$z_k = \begin{bmatrix} \dot{x}_{odo} & \dot{y}_{odo} & \omega_{z,odo} \end{bmatrix}^T \tag{5.11b}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.11c}$$

5.6.2 IMU Measurements

Equation 5.12a demonstrates the linear measurement model for IMU sensor values. The measurement of rotational velocity around the Z-axis $\omega_{z,imu}$ is used to correct the prediction of the rotational velocity ω_z.

$$z_{k+1} = H * x_{k+1} \tag{5.12a}$$

$$z_k = \begin{bmatrix} \omega_{z,imu} \end{bmatrix}^T \tag{5.12b}$$

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.12c}$$

5.6.3 GPS Measurements

For the correction step, using GPS measurements, two cases are distinguished, based on the velocity reported by the odometry.

For a stationary or slowly moving robot, measurements from the GPS are used to correct the estimated two dimensional robot position (x,y). In this case the equation for the linear measurement model looks like 5.13a.

$$z_{k+1} = H * x_{k+1} \tag{5.13a}$$

$$z_k = \begin{bmatrix} x_{gps} & y_{gps} \end{bmatrix}^T \tag{5.13b}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.13c}$$

If the robot is moving faster than a defined velocity, the GPS measurements are also used to correct the prediction of the orientation and the planar velocity filter. The corresponding linear measurement equation is 5.14a.

$$z_{k+1} = H * x_{k+1} \tag{5.14a}$$

$$z_k = \begin{bmatrix} x_{gps} & y_{gps} & \theta_{gps} & \dot{x}_{gps} & \dot{y}_{gps} \end{bmatrix}^T \tag{5.14b}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.14c}$$

6 System Design and Implementation

6.1 System Design

The starting point for the system design is the hardware present in the laboratory. As mobile robot platform the Pioneer 3-AT from MobileRobots Inc.[1] is selected. Next a customised body is created to mount the sensors on the robot. The robot is equipped with a laser range finder, inertial measurement unit and GPS sensor. A image of the equipped Pioneer robot can be seen in Figure 6.1.

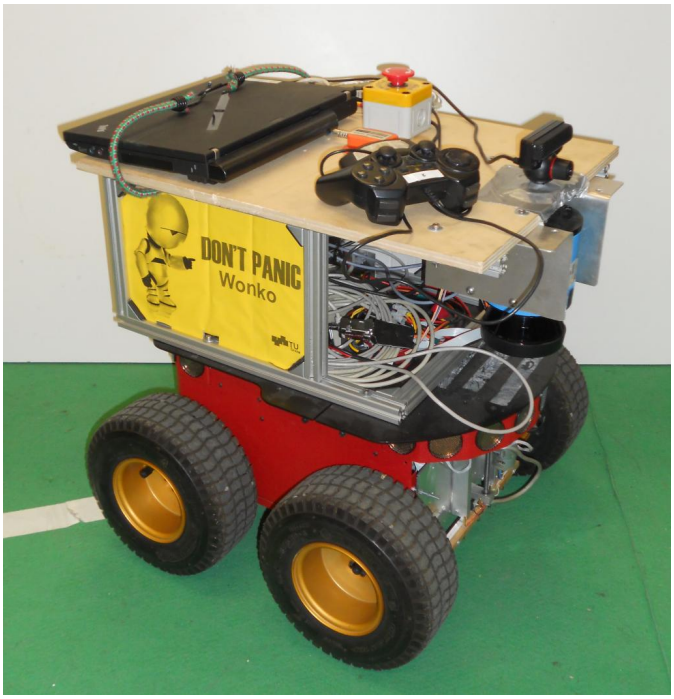

Figure 6.1: Image of the equipped Pioneer robot.

The robot software is designed to operate in the Robot Operating System (ROS) environment. The software concept is projected bottom-up. The first step is handling the hardware and provides the measurement data to the system. Next, the sensor data is

[1] http://www.mobilerobots.com/

processed to estimate the position and executing a navigation task. Finally the software to control the robot and the experiments are drawn up.

Figure 6.2 on page 33 shows the components of the software concept and the communication between them.

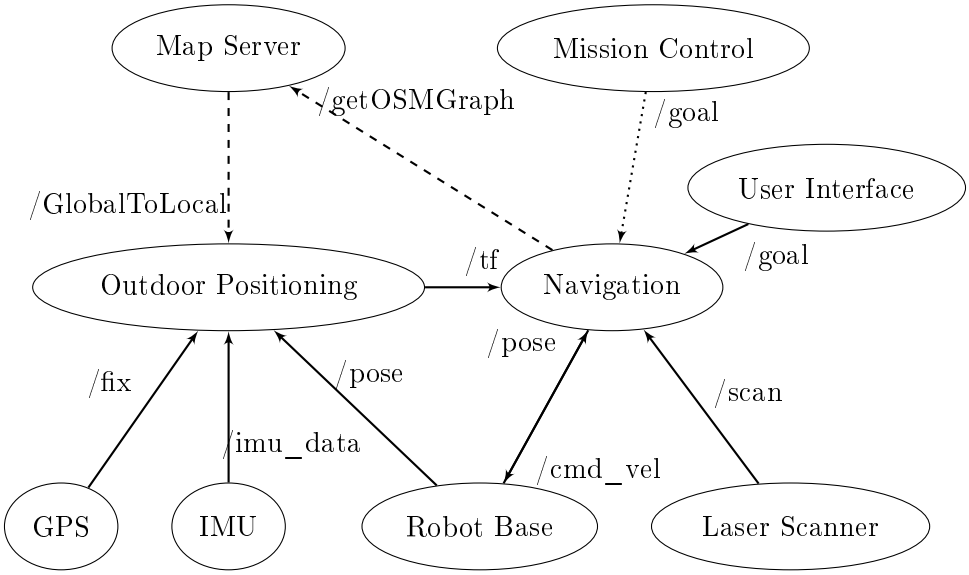

Figure 6.2: Diagram of the software system components.

6.2 ROS

The following Section describes the basic concepts of the Robot Operating System (ROS) framework.

The introduction page of the project [7, ROS/Introduction] describes ROS as an open-source, meta-operating system that runs under Unix-based platforms. Primarily it is tested on Ubuntu and Mac OS X.

ROS and is not a real-time framework. The main idea is a distributed system of processes that are individually designed and loosely coupled at runtime.

ROS provides functionalities for hardware abstraction, low-level device control and message passing between processes. It implements common used operations for high-level applications and delivers tools and libraries for obtaining, building, writing, and running code across multiple computers.

Via client libraries developers can employ the framework to write nodes, publish topics, subscribe to topics, write services, call services and use the parameter server. The

framework provides libraries for C++, Python, LISP, Java and Lua [7, Client Libraries].

The goal of the ROS project is to simplify the reuse of code in robotics research and development, to enable developers easily sharing and distributing of functionality and to support the community to collaborate.

6.2.1 Levels of Concepts

ROS has three levels of concepts to realize the project goals as aforementioned [7, ROS/-Concepts]: the File System Level, the Computation Graph Level and the Community Level.

The File System Level is a concept to allow easy sharing and distributing of functionality. Resources are grouped into Packages and Stacks. The following terms describe the main parts of this level:

Packages organize together software in ROS like runtime processes (nodes), library, datasets and configuration files.

Manifest provides information about a package like license, dependencies and compile flags.

Stacks are collections of packages that provide aggregate functionality.

Stack Manifest provides stack specific information like license and dependencies.

Messages Types define the data structures for messages sent in ROS.

Services Types define the request and response data structures for services in ROS.

The Computation Graph Level provides the functionality to realize a peer-to-peer network of processes. The main components of this level are:

Nodes are processing units that work on different tasks. Nodes are written with the use of a ROS client library. A robot control system will usually be made of many collaborating nodes.

The Master provides the name-service functionality of the ROS framework. It stores topics and service information and enables nodes to find each other, exchange messages, or invoke services.

Parameter Server stores data by key in a central location.

Messages are simple data structures that are passed between nodes to communicate.

Topics are names that identify the content of messages. The transport system routes

messages from publisher of a topic to subscriber to the appropriate topic. Multiple concurrent publishers and subscribers for a single topic may exist.

Services are functionality for request/reply interactions. The function is similar to remote procedure call. Services are defined by a pair of message structures: one for the request and one for the reply.

Bags are files for saving and playing back ROS message data. Bags are helpful for storing data for developing and testing algorithms.

On the Community Level the ROS project provides resources that enable separated communities to exchange software and knowledge. The following capacities are provided:

Distribution are collections of versioned stacks. Like Linux distributions they provide a collection of software with consistent versions.

Repositories is a federated repository model. Different institutions can develop and release their robot software components.

The ROS Wiki is a central documenting platform about ROS that provides corrections or updates, tutorials and more.

Bug Ticket System is for recording issues or request features.

Mailing Lists are communication channels about new updates to ROS, as well as a forum to ask questions.

ROS Answers is a questions and answers website for ROS-related questions.

The Willow Garage Blog provides regular updates on ROS.

6.2.2 Names

Another basic idea in the Robot Operating System is to simplify the development of large systems. Therefore, two types of names are defined [7, ROS/Concepts]: Graph Resource Names and Package Resource Names.

Graph Resource Names are used to structure all resources in a ROS Graph. Nodes, Parameters, Services and Topics are defined within name-spaces. Like in a file system names can be resolved globally or relatively. Resources can create resources within their name-space and access resources within and above their name-space. Graph Resource Names allow encapsulation of different parts of the system to avoid use of wrong resources or globally hijacking names. For example, nodes that work together can be pushed down into a name-space that defined their collection of code. The resolution is done relative to the namespace of the node.

For instance, "relative/name" will reference a relative resources, "/global/name" will reference a global resource and " private/name" a private resource.

Package Resource Names simplify referring files and data-types on Filesystem Level. Resources can be referred to by the name of the Package plus the name of the Resource. Message Types, Node Types and Service Types may be referred using Package Resource Names.

For example, the name "nav_msgs/Odometry" refers to the "Odometry" message type in the "nav_msgs" Package.

6.2.3 Higher-Level Concepts

The three concepts implemented in the ROS framework provide several different modes of communication. For this reason ROS can be used for a variety of system configurations. But for operating a robot system a multitude of mechanisms are required. For building up such systems ROS maintains several stacks that provide higher-level concepts [7, ROS/Higher-Level Concepts] as described below.

Coordinate Frames/Transforms: The transformation framework provides the functionality for representing multiple coordinate frames and calculating the transformations between them.

Actions/Tasks: This stack defines a common topic-based interface to execute long-running tasks. The library provides an interface to cancel the request and get periodic feedback during progressing.

Message Ontology: The common message stack defines base massage ontology for robot systems including messages for representing actions, diagnostic data, geometric primitives, sensor date and navigation commands.

Plugins: The pluginlib provides tools for writing and dynamically loading plugins in C++ code using the ROS build infrastructure.

Filters: ROS contains C++ libraries for processing data using a sequence of filters.

Robot Model: To represent robot models ROS defines a XML format called URDF. Additionally the functionality for parsing the XML structure is provided by the framework.

6.3 Robot Base

The mobile robot platform for the project is the Pioneer 3-AT from MobileRobots Inc.[2]. Information on the robot is taken from [2]. The vehicle has a differential drive and stands on four wheels. It can perform translational and rotational movements.

Figure 6.3: The Pioneer 3-AT from MobileRobots Inc. [2].

The Pioneer has build in high-resolution optical quadrature shaft encoders to provide odometry information. The position (x, y, θ) and the velocity of the robot $(\Delta x, \Delta\theta)$ is calculated by integrating the encoder values.

The robot is connected to the control computer over the serial port (RS-223) and a USB to serial converter.

The connection to the Pioneer robot is established by the p2os_driver node [7, p2os]. It is a driver for robots that uses either P2OS or ARCOS firmware protocol. To move the robot the velocity commands have to be send to the hardware. Furthermore, the node reads out the odometry information.

The robot motion node subscribes to the topic /cmd_vel to receive velocity commands. The message type is geometry_msgs/Twist and expresses the linear and angular velocity in free space. In the case of the pioneer robot translational velocity in the x-direction and rotational velocity around the z-axis of the robot.

The odometry information of the robot is published on the topic /pose as message type nav_msgs/Odometry. The default publishing rate is 10Hz. An odometry message contains the estimated position (x, y, θ) and the corresponding covariance with reference to the odometry frame /odom. Additionally the actual linear and angular velocity of the robot $(\Delta x, \Delta\theta)$ and the corresponding covariance in the robot frame /base_link is included.

[2] http://www.mobilerobots.com/

6.4 Inertial Measurement Unit

To estimate the alignment and the rate of turn of the robot the XSens MTi IMU from Xsens Technologies B.V.[3] is used [3]. This miniature inertial measurement unit includes magnetometers, accelerometer and gyroscopes to measure the acceleration, the rate of turn and the earth-magnetic field. An integrated processing unit computes calibrated output and orientation.

Figure 6.4: The XSens MTi IMU [3].

Over USB the IMU is connected to the computer. The communication to the IMU is executed by the xsens_mti node. This node is a driver for the Xsens MTi Inertial Measurement Unit. It reads out the measurements of orientation, angular velocity and linear acceleration from the device and streams them on the topic /imu_data.

The Message type of /imu_data is sensor_msgs/Imu. The Message holds the acceleration in m/s^2, the rotational velocity in rad/sec and the estimated orientation. The message additionally contains the corresponding covariances for each measured value. All measurements are made in reference to the /imu coordinate frame.

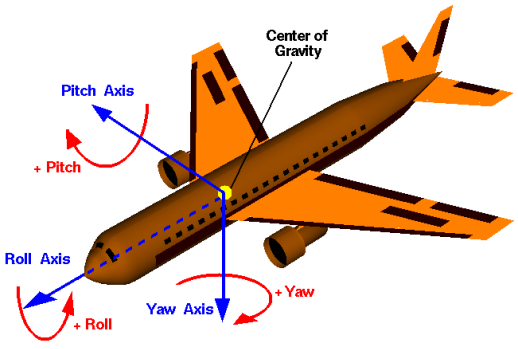

Figure 6.5: Roll, yaw and pitch axis orientation for an airplane (Source: NASA [4]).

[3] http://www.xsens.com/

6.5 GPS

For global positioning the Garmin[4] GPS 35 receiver is used [5]. This small complete GPS receiver is easy to integrate and use. The connection is done over USB. The GPS tracks up to 12 satellites to achieve superior performance.

Figure 6.6: Garmin GPS 35 receiver [5].

The GPS data is streamed to the system by the gpsd_client node [7, gpsd_client]. It connects to the GPSd service daemon and publishes location to the topic /fix.

The GPSd service daemon[5] can monitor one or more GPS devices and makes data on the location, course and velocity of the sensors available to applications. GPSd supports various GPS devices.

The Message type for GPS data is NavSatFix. It contains the location latitude, longitude and altitude specified using the WGS-84 reference ellipsoid and the position covariance relative to a tangential plane through the reported position. Besides, the message includes information on the satellite status, the measurement time and the reference frame.

6.6 Laser Measurement Unit

To avoid obstacles a laser measurement unit is mounted on the robot. The Sick[6] LMS100 [6] performs planar distance measurements in a scanning range up to 20 meters. The scanner has a field of view of maximum 270° with a resolution of angular step of 0.25°.

[4] http://www.garmin.com/
[5] http://www.catb.org/gpsd/index.html
[6] http://www.sick.com

The connection is established over an Ethernet interface. That allows flexible system configurations.

Figure 6.7: The Sick LMS100 laser measurement unit [6] .

The Sick LMS100 laser range finder is connected to the system with the LMS1xx node [7, LMS1xx]. This package connects to the unit over the network interface and publishes measurements on the topic /scan. The Message type for laser range data is sensor_msgs/LaserScan. The message holds specifications of the scanner and an array of range data in meters.

6.7 Outdoor Positioning

The outdoor_positioning node is designed to estimate the 3D pose of a robot by matching measurements coming from different sources.

The node includes a Kalman Filter to fuse measurements of odometry from the robot base, global positioning from the gps and rotational velocity and alignment from the IMU. To receive the data the Outdoor Positioning node subscribes to the odometry topic /pose, the global positioning topic /fix and the IMU message topic /imu_data. For specifications of the message types see the sensor nodes descriptions above.

As described in the filter design (Chapter 5) the sensor data is pre-processed in special functions. When a new sensor message arrives, the appropriate preprocessor callback function is executed.

The positioning node publishes the estimated 3D robot location on the topic /esti-

mated_pose as message type geometry_msgs/PoseStamped. Furthermore, the transformation from the map frame /map to the robot frame /base_link is published.

6.7.1 Coordinate Transformation

The position data latitude, longitude and altitude specified using the WGS-84 reference ellipsoid had to be converted in local Cartesian coordinates x_{gps} and y_{gps}. This conversions are performed using the Geographic Library[7] developed by Charles Karney. GeographicLib is a small set of C++ classes for solving common geodesic problems [22].

The Outdoor Positioning Node provides this functionality to other nodes. They can use two services to transform local planar map coordinates to WGS-84 ellipsoid coordinates and vice versa. The definitions of the services are shown in Listing 6.1 and 6.2.

```
float64 lat
float64 lon
float64 alt
---
float64 x
float64 y
float64 z
```

Listing 6.1: Definition of the GlobalToLocal service. It converts the WGS-84 coordinates latitude, longitude and altitude in local Cartesian coordinates x, y and z.

```
float64 x
float64 y
float64 z
---
float64 lat
float64 lon
float64 alt
```

Listing 6.2: Definition of the LocalToGlobal service. It converts local Cartesian coordinates x, y and z in WGS-84 coordinates latitude, longitude and altitude.

6.7.2 Bayesian Filtering Library

The core of the position estimation is a Kalman Filter. To implement the filter the Bayesian Filtering Library is used. The library process the computations and takes care of calculation accuracy.

In [23] the library is described as:

[7] http://www.sourceforge.net/projects/geographiclib/

> The Bayesian Filtering Library (BFL) provides an application independent framework for inference in Dynamic Bayesian Networks, i.e., recursive information processing and estimation algorithms based on Bayes' rule, such as (Extended) Kalman Filters, Particle Filters (or Sequential Monte Carlo methods), etc. These algorithms can, for example, be run on top of the Realtime Services, or be used for estimation in Kinematics and Dynamics applications.

As a result of this design the system model and the measurement models are simple linear equations. The library is initialized with the equations defined in Chapter 5.

The initial estimate of the filter is defined in ROS Parameters and set at the start of the node. Moreover, the estimated robot pose can be reset to a specific position with a message on the topic /initilpose. The message type of the position reset is geometry_msgs::PoseWithCovarianceStamped. A message contains a 3D position and a 3D orientation.

6.7.3 Dynamic Reconfigure Interface

To simplify the configuration of the filter parameters the node implements a Dynamic Reconfigure Interface. The interface allows setting of parameters during runtime of the node. Figure 6.8 shows the user interface for the reconfiguration.

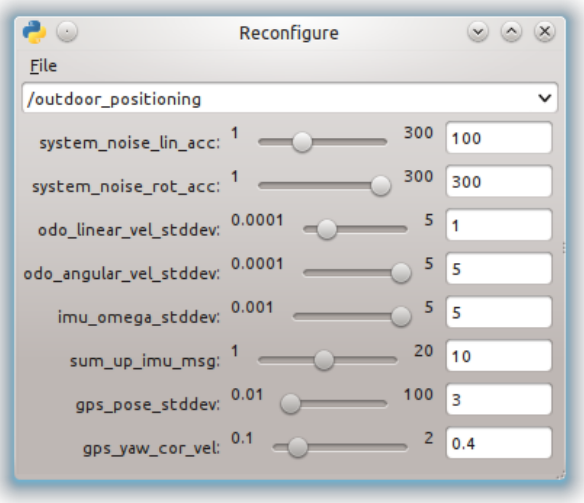

Figure 6.8: Screen shot of the reconfigure interface.

6.8 Map Server

Without information on the environment the robot cannot navigate. Within this project a graph based map is used. A node-edge graph of accessible ways is generated using geographic data provided by OpenStreetMap[8].

The map server node is called osm_server. It reads the OpenStreetMap from an OSM-file and builds up the internal data structure. The osm_server makes the graph based map available for other nodes. The graph based map can be requested by calling the service /getOSMGraph provided by the osm_server node. Listing 6.4 shows the definition of the service and the corresponding message type is shown in Listing 6.3.

The map server node publishes the generated map on the topic /map as Message Type nav_msgs/OccupancyGrid for visualization in the user interface. Furthermore, the service nav_msgs/getMap is provided to the global planner.

```
# Open Street Map Node
int32 node_id
geometry_msgs/Point coords
int32[] neighbors
```

Listing 6.3: Definition of the Open Street Map Node (OSMNode) message.

```
# request msg
---
# response msg
OSMNode[] nodes
```

Listing 6.4: Definition of the getOSMGraph service.

For the navigation a node-edge graph of accessible ways is generated. Therefore, a node class is defined which contains the geographic data. The map information is stored as a list of that node classes. The provided map information is generated out of this dataset.

6.8.1 OpenStreetMap

OpenStreetMap (OSM) is an online database, which contains geographic data. The map of the whole world is written and constructed collaboratively by volunteers around the world. All maps are without legal or technical restrictions on their use. All information on the OpenStreetMap project is taken from [9].

The Data Primitives of the OSM database are:

[8] http://wiki.openstreetmap.org

Node: Is a point defined as latitude and longitude coordinates. Nodes can be used to define the path of a way or can be standalone point features.

Way: A way is an ordered list of nodes. Ways can have one or more tags for detailed information.

Relation: To group one or more data primitives relations are used.

Listing 6.5 shows an example of an OpenStreetMap XML file. The ROS Service /GlobalToLocal provided by the positioning node is used to convert the WGS-84 ellipsoid coordinates in local Cartesian coordinates. Within this project the robot should drive only on OSM Ways of the type footway and cycleway.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
  <bounds minlat="54.0889580" minlon="12.2487570"
          maxlat="54.0913900" maxlon="12.2524800"/>
  <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO"
        uid="46882" visible="true" version="1" changeset="676636"
        timestamp="2008-09-21T21:37:45Z"/>
  <node id="261728686" lat="54.0906309" lon="12.2441924"
        user="PikoWinter" uid="36744" visible="true" version="1"
        changeset="323878" timestamp="2008-05-03T13:39:23Z"/>
  ...
  <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO"
        uid="46882" visible="true" version="1" changeset="676636"
        timestamp="2008-09-21T21:37:45Z"/>
  <way id="26659127" user="Masch" uid="55988" visible="true" version="5"
       changeset="4142606" timestamp="2010-03-16T11:47:08Z">
    <nd ref="292403538"/>
    <nd ref="298884289"/>
    ...
    <nd ref="261728686"/>
    <tag k="highway" v="unclassified"/>
    <tag k="name" v="Pastower Strasse"/>
  </way>
  <relation id="56688" user="kmvar" uid="56190" visible="true"
            version="28" changeset="6947637"
            timestamp="2011-01-12T14:23:49Z">
    <member type="node" ref="294942404" role=""/>
    ...
    <member type="node" ref="364933006" role=""/>
    <member type="way" ref="4579143" role=""/>
    ...
    <member type="node" ref="249673494" role=""/>
    <tag k="name" v="Kuestenbus Linie 123"/>
    <tag k="network" v="VVW"/>
    <tag k="operator" v="Regionalverkehr Kueste"/>
    <tag k="ref" v="123"/>
    <tag k="route" v="bus"/>
    <tag k="type" v="route"/>
  </relation>
  ...
```

```
</osm>
```

Listing 6.5: A shortened example of a complete OSM XML file [9]

6.9 Navigation

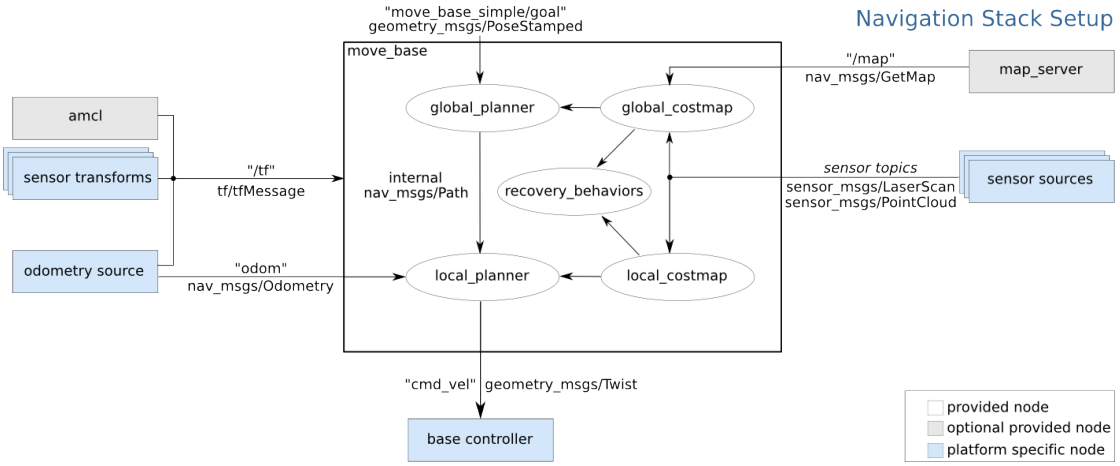

Figure 6.9: Diagram of the principle move_base node structure [7, move_base].

The navigation task is performed by the move_base node [7, move_base]. Figure 6.9 on page 45 shows a diagram of the principle move_base node structure. The package provides an action interface to give the robot a goal in the world and to monitor the task.

To navigate the robot to the global goal, move_base links together a global and local planner. Planners are implemented as plugins that adhere to the interfaces defined in the navigation core [7, nav_core]. The class nav_core::BaseLocalPlanner defines the interface for the local planners in. The interface for global planners is defined in nav_core::BaseGlobalPlanner. The global planner provides the functionality to compass a global path for the robot using the map information. The local planner performs the task to generate the movement commands to navigate the robot locally avoiding obstacles. For both planners a costmap is provided.

If the robot perceives itself as stuck the move_base node performs recovery behaviours. Recovery behaviours are also implemented as plugins defined in the navigation core [7, nav_core]. The standard interface is nav_core::RecoveryBehaviour.

Now let us have a closer look to the different components of the navigation.

6.9.1 Costmap

The costmap_2d [7, costmap_2d] builds a 2D or 3D occupancy grid for the robot navigation. It takes in sensor data from the world to store and update information about obstacles in the world. A costmap can be initialized with a static map from a map_server or as a rolling window based costmap.

In the static map configuration the costmap makes a service call to the map_server to obtain a 2-D grid map, which represents the probability of occupancy of each cell. Map data is stored as data-type nav_msgs/OccupancyGrid.

Sensor data can be taken from laser_scan_topics or point_cloud_topics. The sensor data is used by Costmap_2d to update map cells as free, occupied or unknown. The Message type for laser scans is sensor_msgs/LaserScan and for point cloud data sensor_msgs/PointCloud.

A detailed description of all options can be found at [7, costmap_2d].

6.9.2 Local Planner

The dwa_local_planner [7, dwa_local_planner] plugin for the move_base node provides local robot navigation using the Dynamic Window Approach algorithm [24]. This package has a large number of ROS Parameters to customize the behaviour of the planner. The parameters are dynamically reconfigurable.

The planner takes a global plan to follow and a local costmap and produces velocity commands for the mobile base. The velocity commands are published to the topic /cmd_vel. The message type is geometry_msgs/Twist and expresses the linear and angular velocity in free space.

The dwa_local_planner subscribes to the odometry information of the robot on the topic /pose to get the current speed of the robot. The Message type of the odometry information is nav_msgs/Odometry. The velocity information in this message is assumed to be in the robot frame.

For a detailed description of the parameters see [7, dwa_local_planner].

6.9.3 Global Planner

For global path planning the OpenStreetMap planner is implemented. The package is inherited from the nav_core::BaseGlobalPlanner interface specification. The plugin is called osm_planner and plans the global path based on the road network exported from the OpenStreetMap database.

The planner internally uses the same data-structure as the Map Server (6.8). The osm_planner requests the graph based map from the osm_server node at start-up by calling the service /getOSMGraph. The base class for the Global Planner is the nav_core::BaseGlobalPlanner interface. The OpenStreetMap Planner is implemented as ROS Plugin.

The path planning algorithm implemented in the node is an A* heuristic. A detailed description of the algorithm and a pseudo code can be found in [25, p.527]. Figure 6.10 shows a visualization of a planned path. The computed path is provided internally as nav_msgs/Path to the local planner.

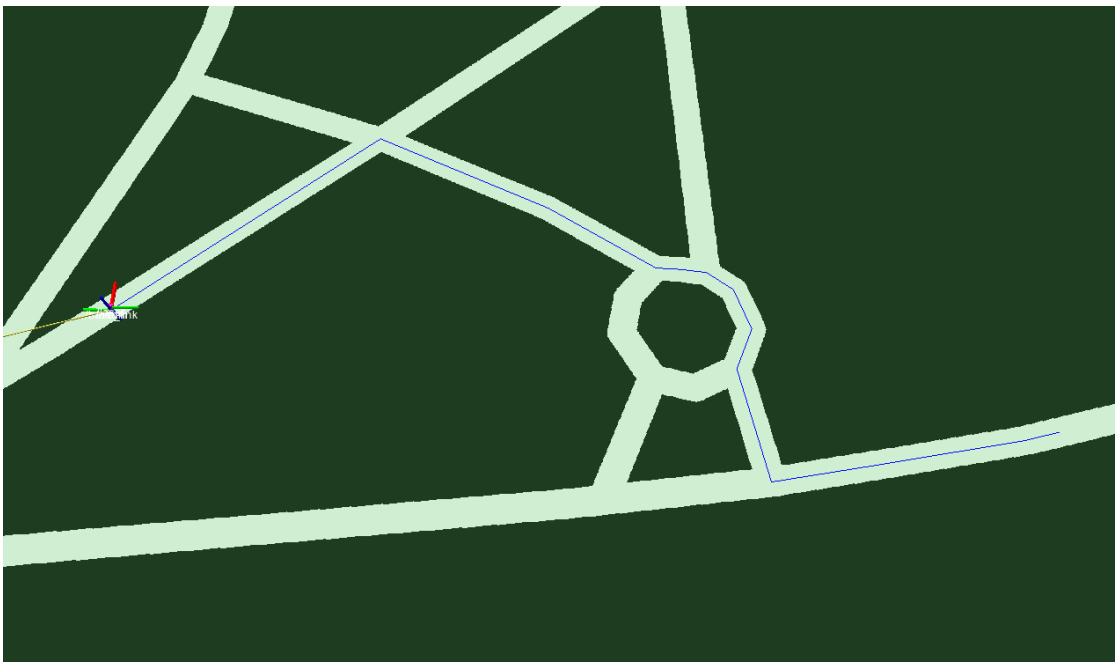

Figure 6.10: Map generated by the osm_server geographic data and a planned trajectory computed by the osm_planner.

6.9.4 Recovery Behaviour

Figure 6.11 on page 45 illustrates the standard recovery behaviour. The description is taken from [7, move_base].

As first step the robot will clear obstacles outside of a specified region from the map and try to perform an in-place rotation to clear out space. If this fails, the robot will clear its map, removing all obstacles outside of the rectangular region in which it can rotate in place and makes another in-place rotation. If all this fails, the robot will consider its goal infeasible and the task will be aborted.

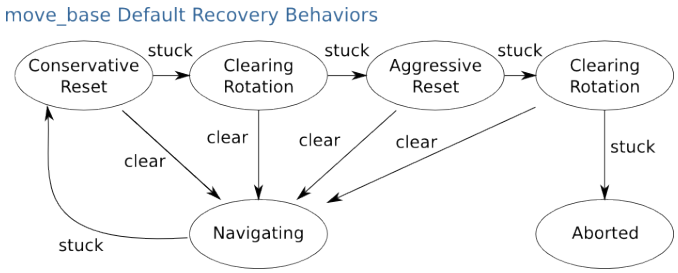

Figure 6.11: Expected robot recovery behaviour of the principle move_base node.

6.10 User Interface

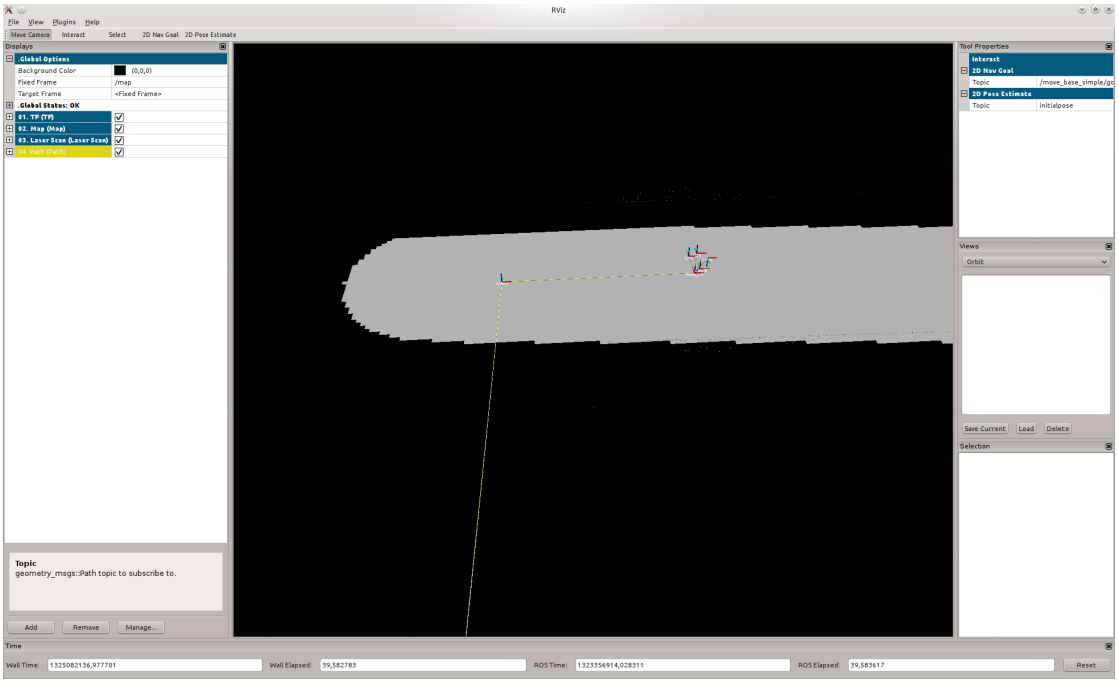

Figure 6.12: Screen shot of the User Interface RViz.

Within this project RViz is used as a high-level control user interface [7, rviz].

RViz is a visualization tool in ROS that can be used to collect and display information from different sources. Sensor data, coordinate frames, robot models and maps can be visualized in one 3D view of the world. The tool also provides functionality to interact with the system, like setting the initial robot position or a navigation goal.

Rviz has a plugin-based architecture so the tool can be adapted easily to different needs. By adding specific modules the interface can be customized to display relevant

sensor data or provide useful commands [7, rviz/UserGuide]. Figure 6.12 on page 48 shows a screen-shot of RViz in the used configuration.

6.11 Mission Control

The mission_control node is designed to run the trial experiment. The robot should navigate between defined positions on a known road network. This node connects to the action interface of the move_base node and triggers the robot to go to the goals. The status of the navigation node is tracked during the walk.

The experiment positions are defined in a YAML file. The website [26] describes the make-up language as:

> YAML is a human friendly data serialization standard for all programming languages.

To read the list of positions the library yaml-cpp[9] is used. This library is delivered as part of the ROS framework. Listing 6.6 shows an example of a simple position list.

```
- position: { x: 159.464279175, y: -442.224456787, z: 0.0}
  orientation: { x: 0.0, y: 0.0, z: -0.476974266337, w: 0.878917259617}
- position: { x: 114.952140808, y: -476.165008545, z: 0.0}
  orientation: { x: 0.0, y: 0.0, z: 0.915350526474, w: -0.402657936322}
- position: { x: 81.7606048584, y: -404.174194336, z: 0.0}
  orientation: { x: 0.0, y: 0.0, z: 0.7414528362, w: 0.671004986338}
- position: { x: 119.024230957, y: -393.008972168, z: 0.0}
  orientation: { x: 0.0, y: 0.0, z: -0.43713753446, w: 0.899394671969}
- position: { x: 103.11390686, y: -468.373626709, z: 0.0}
  orientation: { x: 0.0, y: 0.0, z: -0.55469777397, w: 0.832051909169}
```

Listing 6.6: Example position list for the mission control YAML file

[9] http://code.google.com/p/yaml-cpp/

7 Experimental Evaluation

This Chapter will have a closer look on the quality of the designed system. The accuracy of the position estimation is essential for the navigation system. For the evaluation of the positioning algorithm a reference to make an effective comparison is necessary.

7.1 Ground Truth

The evaluation of the ground truth is an important element in the analysis of the developed system. For the determination of the true robot position a second high precision positioning system would be great, which unfortunately is not available.

The position estimation is evaluated using a reference trajectory and typical sensor measurements. To determine the error of the positioning the normal distance of the estimated position and the reference path are processed. Clearly this approach for error determination is not very precise but that approach is the best choice that can be put into practice.

The robot is driven by joystick control along the reference path. Because of the human control of the robot it is not possible to move on the white ground line with precision. During the run it was tried to keep the ground marking in-between the wheels of the mobile platform.

7.1.1 Reference Trajectory

To evaluate the position estimation a reference trajectory is required. As path for the robot the ground markings on the cycleway close to the laboratory are used. The measuring of the ground marking is performed using an orthophoto from the spatial data service Geoimage-Austria©[1]. These pictures have a high positional accuracy although the inaccuracy of the position is up to thirty centimetres.

Figure 7.1 shows the sector of the orthophoto with the reference ground marking. The high-resolution aerial pictures from the spatial data service have a high positional

[1] http://www.geoimage.at/

accuracy. The white ground markings are well visible. To process the reference trajectory in the software an OpenStreetMap data file is generated that contains the trajectory.

Figure 7.1: Reference trajectory to verify the position estimation. The trajectory is shown in purple. Orthophoto: www.geoimage.at (c)

7.2 Evaluation of the Positioning

A recorded experimental run is used to parameterize the Kalman Filter for the position estimation. Table 7.1 shows the standard deviations used for the evaluation runs.

Standard Deviation	Symbol	Value
Linear acceleration (System noise)	$\sigma_{\ddot{x}}$	50
Rotational acceleration (System noise)	$\sigma_{\ddot{\theta}}$	100
Linear velocity (Odometry)	$\sigma_{v_{odo}}$	0.005
Rotational velocity (Odometry)	$\sigma_{\omega_{z,odo}}$	0.5
Rotational velocity (IMU)	$\sigma_{\omega_{z,imu}}$	1.0

Table 7.1: Parameter-Set for the Kalman filter.

The position estimation is examined in different runs using the same filter settings. In the experiment the robot is driven by joystick control along the reference path.

Two typical example runs are described below. They differ from each other in the maximum robot motion speed, the travel direction and the initialisation time of the gps receiver.

7.2.1 First Run

In the first run the robot was driven from the south-east end of the trajectory to the north-west point of the path. In this travel direction the path has a slight incline, the maximum motion speed was set to $0.8m/s$ and the GPS receiver was operating for more than 30 minutes.

Figure 7.5 on page 55 demonstrates the tracks of the filter-estimated way of the robot, the recorded GPS track, as well as the position estimated from the odometry measurements. It can be seen that, the deviation between reference path and odometry is increasing rapidly.

Figure 7.2 compares the progression of the normal distance from the reference trajectory of the estimated position, the GPS position and the position estimated by odometry. Compared with the filtered position and the GPS measurement, the accumulation of error in the odometry position estimation is clearly visible. Taking into account the fact that the computed error indicator represents the normal distance to the nearest path segment the absolute error is even higher.

For a better observation of the temporal progression of the normal distance from the reference trajectory Figure 7.2 only shows the filter position error and the GPS position error. The mean error of the positioning is drawn as cyan line exceeds the average error of the gps position measurement.

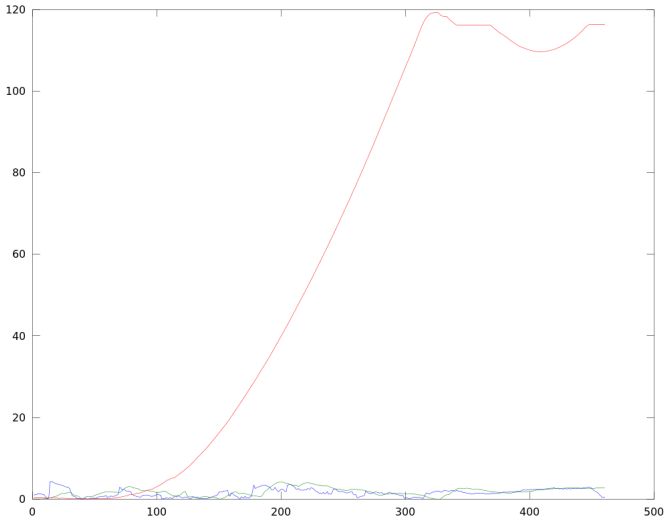

Figure 7.2: Position error: estimated position (green), GPS (blue) and odometry (red).

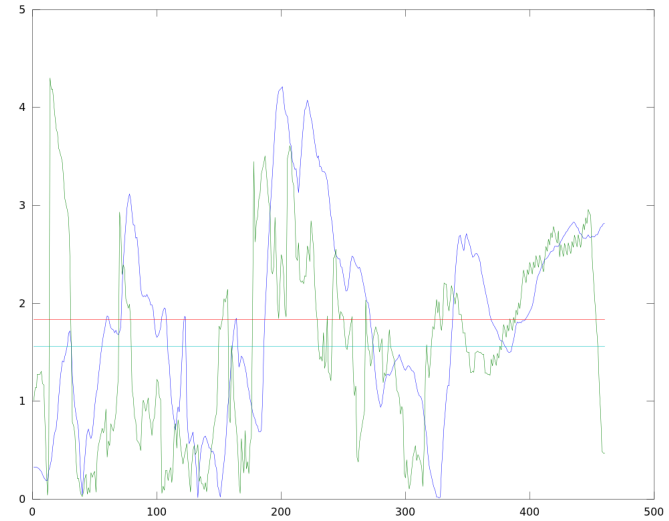

Figure 7.3: Position error: estimated position (blue) and GPS (green); Average position error: estimated position (red) and GPS (cyan).

The temporal progression of the heading error of the Kalman filter and the odometry estimation is shown in Figure 7.4. The average heading error of the filter estimation is a factor of five smaller. Those data show that the approach of using consecutive GPS position measurements to back the orientation helps to correct the estimated orientation of the robot.

It can be seen that the temporal progression of the odometry heading error is constantly increasing with a rapid change in the value. It could perhaps be argued that this jump is caused by the inadequate method to measure the heading error.

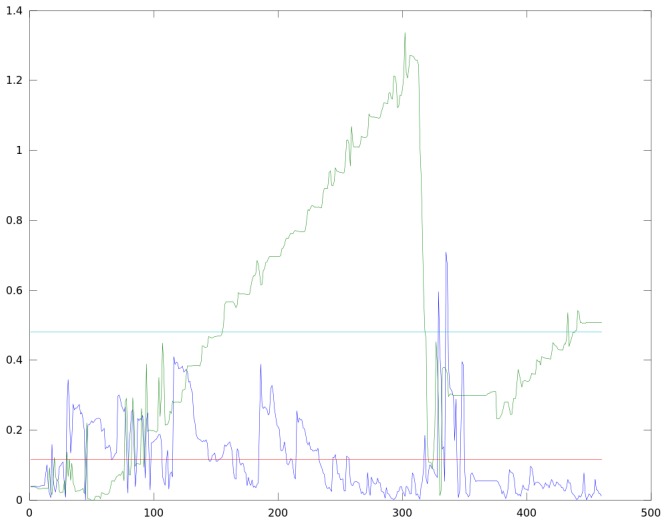

Figure 7.4: Heading error: heading estimation (blue) and odometry heading (green); Average heading error: estimated heading (red) and odometry heading (cyan).

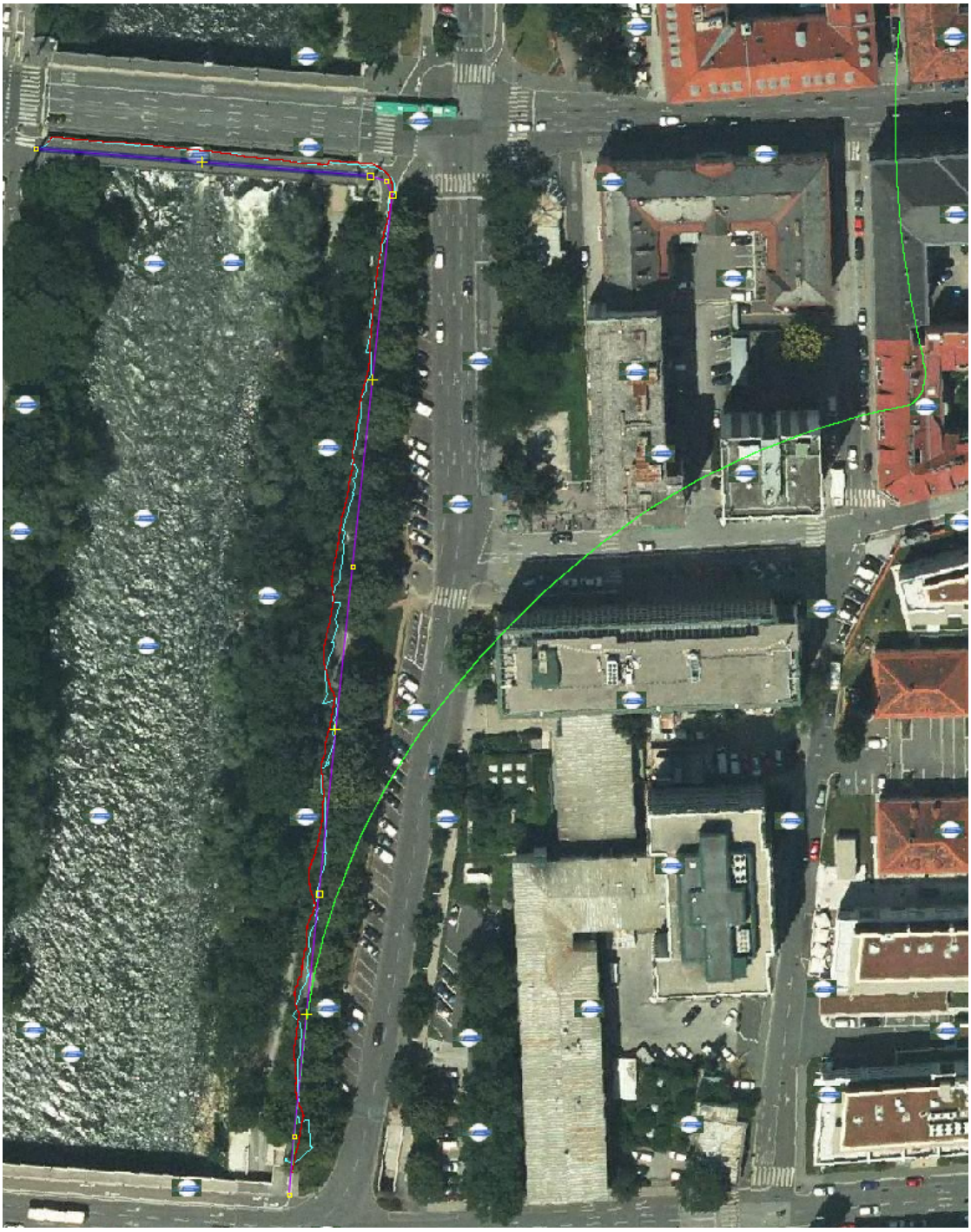

Figure 7.5: Tracks of the different position sources in run one: reference path (purple), GPS (cyan), estimated position (red) and odometry position (green). Orthophoto: www.geoimage.at (c)

7.2.2 Second Run

In the second run the robot was driven from the north-west point of the trajectory to the south-east end of the path. In this travel direction the path has a slight downward slope. The maximum motion speed was set to $0.5m/s$ and the GPS receiver was switched on shortly before the run.

The tracks of the filter-estimated way of the robot, the recorded GPS track, as well as the position estimated from the odometry measurements are shown in Figure 7.9 on page 58.

Figure 7.6 compares the progression of the normal distance from the reference trajectory of the estimated position, the GPS position and the position estimated by odometry. Compared with the filtered position and the GPS measurement, the accumulation of error in the odometry position estimation is clearly visible. Compared with the odometry position error in the first run the progression is not that grave. This might be taken as indication for different odometry error behaviour for inclines and gradients.

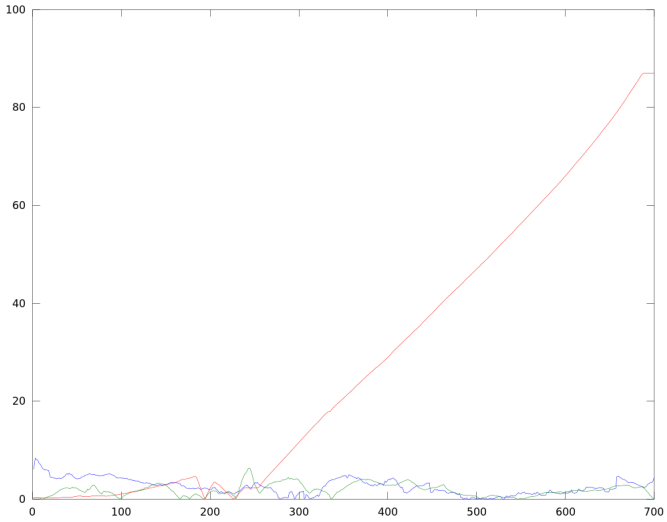

Figure 7.6: Position error: estimated position (green), GPS (blue) and odometry (red).

For a better observation of the temporal progression of the normal distance from the reference trajectory Figure 7.6 only shows the filter position error and the GPS position error. In this run the mean error of the positioning is drawn as cyan line is smaller than the average error of the gps position measurement.

For the second run the temporal progression of the heading error of the Kalman filter and the odometry estimation is shown in Figure 7.8. It can be observed that the estimated

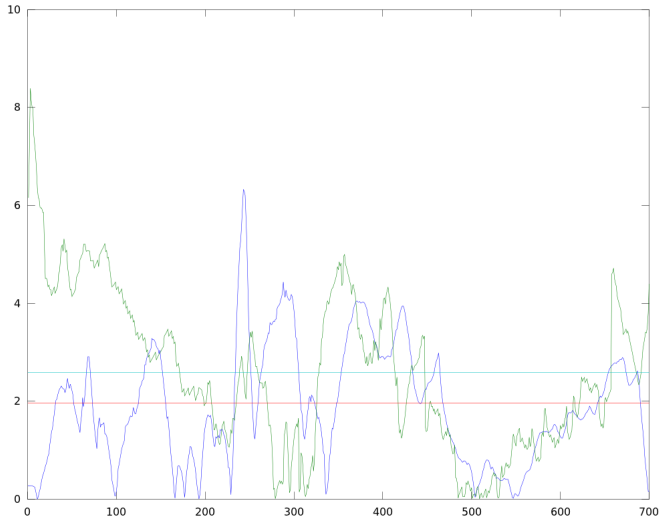

Figure 7.7: Position error: estimated position (blue) and GPS (green); Average position error: estimated position (red) and GPS (cyan).

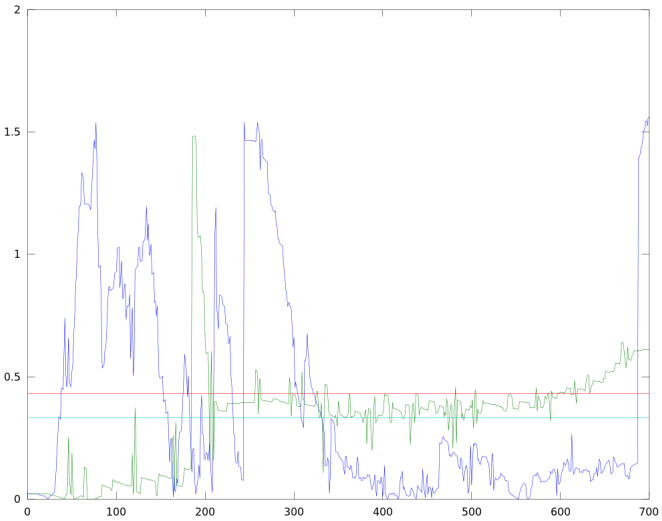

Figure 7.8: Heading error: heading estimation (blue) and odometry heading (green); Average heading error: estimated heading (red) and odometry heading (cyan).

heading is very inaccurate during the first 300 seconds of the experimental run. This could be taken as result of the incomplete initialisation of the GPS receiver and the larger error involved that have a negative effect in the sensor fusion.

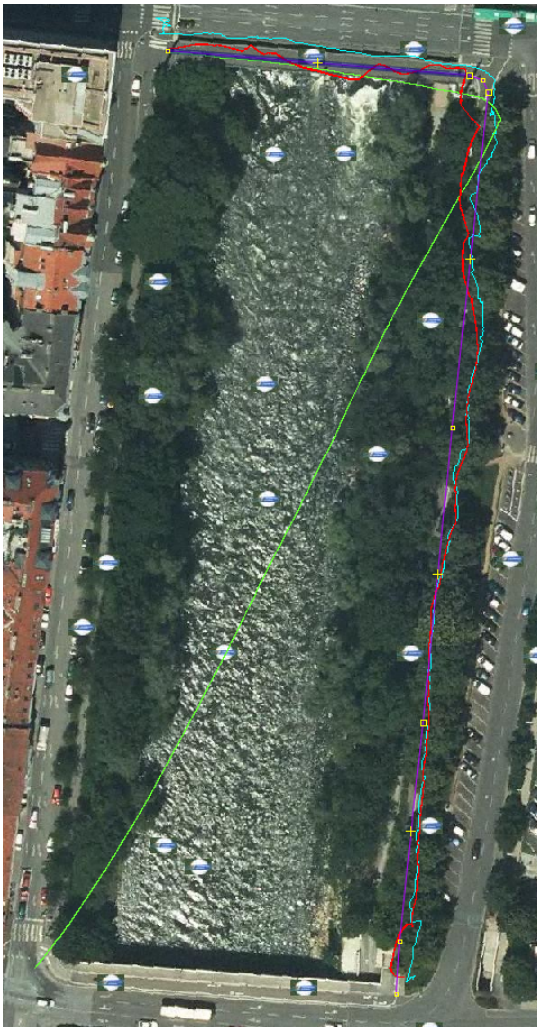

Figure 7.9: Tracks of the different position sources in run two: reference path (purple), GPS (orange), estimated position (blue) and odometry position (green). Orthophoto: www.geoimage.at (c)

7.3 Trial Run

In order to test whether the position estimation is useful for an autonomous outdoor navigation a trial run is performed. In the setup for the experiment the robot have to move autonomous in the "Augarten" a park facility with several paths near the laboratory. Figure 7.10 shows aerial photograph of the park.

The experiments show that the robot can move along a way for a short period of time. In certain cases the system loses its way. In other words the robot movement becomes unpredictable and human intervention is needed. The author thinks that different disorders in the sensor input cause an erroneous orientation. The local navigation control algorithm tries to compensate the orientation error but this behaviour lead to an ever-growing error. Also can be seen, that even though the robot can move along a way for a short period of time the positioning system can not guarantee that the robot is on the way.

Figure 7.10: Aerial photograph of the "Augarten", a park facility with several paths near the laboratory. Orthophoto: www.geoimage.at (c)

8 Conclusion

In this thesis a basic robot setup for mobile outdoor navigation is presented. The system consists of the following components: a mobile robot platform, a GPS receiver, an IMU sensor, a laser range finder and a computer to control the system. The developed software runs on Ubuntu, a Linux operating system and uses the Robot Operation System (ROS) as framework. Geographic data that is received from an online database is used to provide knowledge on the environment to the robot. A graph-based map is used to store the knowledge of the environment and guide the robot through the road network.

The position determination is evaluated using a reference trajectory and typical sensor measurements. Clearly this approach for error determination is not very precise but gives adequate indications of the temporal progression of the positioning error. The evaluation method is limited by the fact that the determination of the used reference path and the measurement of the difference between the real and the estimated robot position have some limitations. The determination of the used reference trajectory has been done using high-resolution aerial pictures from a spatial data service. These pictures have a high positional accuracy although the inaccuracy of the position is up to thirty centimetres. In additional, only normal distance of the estimated position and the reference path could be determined.

The designed positioning system uses a Kalman Filter to fuse the sensor information from the GPS, the IMU and odometric sensor. The filter equations are modelled to improve the accuracy of the positioning by compensating the disadvantages of the different sensors. For instance the position change of consecutive GPS position measurements are used to back the orientation and the planar velocity estimation. Typical sensor measurement data, which was observed, shows that the heading direction error is improved through this correction method.

During the work it was discovered that the use of magnetic compass measurements to back the absolute heading of the robot could not be implemented. Tests in urban outdoors environment has shown that the distortions of the earth magnetic field near steel structures like buildings lead to inaccurate measurements. In addition to the unexpected large errors in compass measurement the influence of multi-path phenomenon to the accuracy of the GPS position measurement was not expected.

Because of a number of technical problems no running DGPS (Differential GPS) positioning could be operated. In open environment the accuracy of the positioning could have been further improved by taking advantage of differential correction data. In urban

environment the benefit is reduced because of incorrect positioning results near buildings and trees as a result of multi-path phenomenon.

This thesis has realized a system to move the robot in outdoor environment. Through practical observations could be seen that the chosen approach is not sufficient to operate the robot full autonomously. To develop a reliable autonomous robot navigation system many factors have to be taken into account and different sources of error have to be determined. Within the project the filter parameters are set to higher standard deviation as assumed. The results show, that the position estimation works in the majority of cases with sufficient accuracy to locate the robot, but it cannot be guaranteed that the robot is on the way or navigate autonomously.

The navigation task consists of a global-path planning based on a graph based map and a local path planning based on a Dynamic Window Approach algorithm. In a local area laser range information is used to avoid collisions. This implemented functionality allows the robot to react on local occurring obstacles but not any types of obstacles have been taken into account. For instance stairs and kerbstones.

8.1 Future Research

Many problems still have to be sorted out for developing a full autonomous mobile robot that moves in urban environment without presenting a danger in the surrounding area.

Future research could improve the developed positioning system by integrating a system to identify large errors in compass and GPS measurements and eliminate them. A possible approach is described in "Using qualitative and model-based reasoning for sensor validation of autonomous robots" [27].

In addition, it is possible to combine technologies like SLAM (Simultaneous Localization And Mapping) or landmark navigation with the system to balance out the drawback of the designed system. Furthermore, a more realistic and precise modelling of sensor characteristics could improve accuracy.

In future research the accuracy of the map data (OpenStreetMap) could be improved and additional relevant knowledge about the environment around the robot, like traffic lights or crosswalks, could be inserted. Furthermore, an improved perception of the environment might increase the robustness of the outdoor navigation system, an advanced vision system to detect the way and obstacle could be used to balances the drawback of the inaccuracies in the positioning.

Bibliography

[1] B. Hofmann-Wellenhof, K. Legat, and M. Wieser. *Navigation: Principles of Positioning and Guidance*. Springer, Wien, 2003.

[2] MobileRobots Inc. *Pioneer 3 Operations Manual*, version 5 edition, July 2007.

[3] Xsens Technologies B.V. *MTi and MTx User Manual and Technical Documentation*, revision g edition, March 2006.

[4] National Aeronautics and Space Administration NASA. Aircraft rotations. Online, Jul 2008.
http://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html.

[5] GARMIN Corporation. *GPS SMART ANTENNA TECHNICAL SPECIFICATION*, rev. e edition, March 2000.

[6] SICK AG. *Laser Measurement Systems of the LMS100 Product Family.*

[7] ROS.org. Ros wiki. Online.
http://www.ros.org/wiki/.

[8] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* The MIT, 2001.

[9] OpenStreetMap. Openstreetmap wiki. Online, November 2011.
http://wiki.openstreetmap.org/wiki/.

[10] NIMA National Imagery and Mapping Agency. *The American Practical Navigator: Bowditch.* Paradise Cay Publications, 2002.

[11] Robin R. Murphy. *Introduction to AI Robotics.* The MIT Press, Massachusetts, 1st edition, 2000.

[12] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning - sensors and techniques. *Journal of Robotic Systems*, Vol. 14 No. 4:231–249, 1997.

[13] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, Vol. 12 No. 6:869–880, 1996.

[14] H. Lategahn, A. Geiger, and B. Kitt. Visual slam for autonomous ground vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1732 –1737, may 2011.

[15] M. Jenkin, E. Milios, P. Jasiobedzki, N. Bains, and K. Tran. Global navigation for ark. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 3, pages 2165 –2171 vol.3, jul 1993.

[16] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[17] KyuCheol Park, Dohyoung Chung, Hakyoung Chung, and Jang Gyu Lee. Dead reckoning navigation of a mobile robot using an indirect kalman filter. In *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, pages 132 –138, dec 1996.

[18] Ning Ma, M. Bouchard, and R.A. Goubran. Speech enhancement using a masking threshold constrained kalman filter and its heuristic implementations. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):19 – 32, jan. 2006.

[19] A. Rusdinar, Jungmin Kim, and Sungshin Kim. Error pose correction of mobile robot for slam problem using laser range finder based on particle filter. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, pages 52 –55, oct. 2010.

[20] K. Ohno, T. Tsubouchi, B. Shigematsu, S. Maeyama, and S. Yuta. Outdoor navigation of a mobile robot between buildings based on dgps and odometry data fusion. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1978 – 1984 vol.2, sept. 2003.

[21] S. Panzieri, F. Pascucci, and G. Ulivi. An outdoor navigation system using gps and inertial platform. *Mechatronics, IEEE/ASME Transactions on*, 7(2):134 –142, jun 2002.

[22] Charles Karney. Geographic library. Online. http://geographiclib.sourceforge.net/html/index.html.

[23] Klaas Gadeyne. BFL: Bayesian Filtering Library, 2001. http://www.orocos.org/bfl.

[24] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23 –33, mar 1997.

[25] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[26] Clark C. Evans. The official yaml web site. Online. http://www.yaml.org/.

[27] Kleiner A., Steinbauer G., and Wotawa F. Using qualitative and model-based reasoning for sensor validation of autonomous mobile robots. In *Twentieth International Workshop on Principles of Diagnosis (DX 2009), (Stockholm, Sweden)*, 2009.