Localization and Mapping for Outdoor Mobile Robots with RTK GPS and Sensor Fusion

An Investigation of Sensor Technologies for the Automower Platform

ODEN LOBELL

FILIP STENBECK
Localization and Mapping for Outdoor Mobile Robots with RTK GPS and Sensor Fusion

An Investigation of Sensor Technologies for the Automower Platform

Oden Lobell
Filip Stenbeck

School of Industrial Engineering and Management
Department of Machine Design
KTH Royal Institute of Technology
Stockholm, Sweden 2017
Localization and Mapping for Outdoor Mobile Robots with RTK GPS and Sensor Fusion
An Investigation of Sensor Technologies for the Automower Platform
Oden Lobell
Filip Stenbeck

Company Supervisor: Erik Bergdahl, Cybercom
Supervisor: De-Jiu Chen, Machine Design, KTH
Examiner: Hans Johansson, Machine Design, KTH

Master’s Thesis 2017:05
School of Industrial Engineering and Management
Department of Machine Design
KTH Royal Institute of Technology
SE-100 44 Stockholm
Telephone +46 8-790 60 00
Abstract

The following thesis addresses the problem of localizing an outdoor mobile robot and mapping the environment using the state of the art of consumer grade RTK GPS. The thesis investigates limitations and possibilities for sensor fusion to increase reliability and usability. The main subject of research is a robotic lawn mower from Husqvarna, the Automower 430x, connected to existing hardware on the product with an auxiliary real time kinematic global positioning system, the Emlid Reach. The test conducted showed that the auxiliary RTK GPS module is currently unsatisfactory as sole absolute position sensor for the Automower platform, mainly due to inconsistent performance. This thesis is meant as a preliminary study for future use of GNSS sensors for outdoor mobile robots and as a suggestive study of the current performance of the increasingly popular Emlid Reach GPS module.
Sammanfattning

Acknowledgements

We would like to thank our supervisor at KTH, Associate Professor De-Jiu Chen for valuable feedback in concept design and system architecture. Our company supervisor Erik Bergdahl for the cherished contribution for the report structure and logistics. The contact persons at Husqvarna, Peter Schön and Björn Mannefred for the access-ability to the Automower platform. We would also like to thank Lars Svensson for the invaluable input in regards of filter design.

Oden Lobell, Filip Stenbeck, Stockholm, May 2017
7 Recommendations and Future Work 59
  7.1 Control ................................................................. 59
  7.2 Better GPS fault detection ........................................ 59
  7.3 Mapping ............................................................... 60
  7.4 Other absolute position systems ................................. 60

Bibliography 61

A Appendix 1 I
  A.1 Code ..................................................................... I
List of Figures

1.1 Communication architecture for unit tests. .................................. 5
1.2 Communication architecture for Validation test .............................. 6

2.1 Dynamics of one drive differential mobile robot. ............................ 8
2.2 Line following robot .............................................................. 11
2.3 Emlid Reach Module .................................................................. 16
2.4 Six degrees of Freedom .............................................................. 18
2.5 The Kalman Filter Process ......................................................... 21
2.6 Spiral Coverage Path .............................................................. 27
2.7 U-turn Coverage Path .............................................................. 27

3.1 Simulation Environment ............................................................ 30
3.2 Map constructed in the simulation environment. ............................ 30
3.3 Decoupled filter architecture. ..................................................... 33
3.4 Clear Cell Function Output ....................................................... 37
3.5 Left Detection Function Output .................................................. 37
3.6 Right Detection Function Output ............................................... 37
3.7 Layout of the perimeter wire during test. ...................................... 39
3.8 Aerial photo of test garden. ...................................................... 39
3.9 Validation test ........................................................................... 39
3.10 Robot from side perspective. ...................................................... 40
3.11 Robot from above. ................................................................. 40
3.12 IMU’s positioning .................................................................... 41

4.1 Raw GPS data from four different test runs. Units are global x and y
coordinates in meters. ................................................................. 43
4.2 Test run 1. Section with correct 'fix' data points. ............................ 44
4.3 Complete Test Run 1 ................................................................. 45
4.4 Results of Test run 2 ................................................................. 46
4.5 Results of Test run 3 ................................................................. 47

5.1 IMU and odometry correlation. ................................................... 52
5.2 Example of a working angle filter .............................................. 52
5.3 Example of GPS angle correction .............................................. 53
5.4 Test run 2 with problematic GPS data marked in red ..................... 54
## Nomenclature

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGPS</td>
<td>Differential Global Positioning System</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>GDOP</td>
<td>Geometric Dilution of Precision</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite Systems</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>NTRIP</td>
<td>Networked Transport of RTCM via Internet Protocol</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RTK</td>
<td>Real Time Kinematic</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localization And Mapping</td>
</tr>
<tr>
<td>UHF</td>
<td>Ultra High Frequency</td>
</tr>
<tr>
<td>UKF</td>
<td>Unscented Kalman Filter</td>
</tr>
<tr>
<td>WAAS</td>
<td>Wide Area Augmentation System</td>
</tr>
</tbody>
</table>
1

Introduction

Autonomous robots are a relatively new type of consumer products with a growing range of applications. To achieve an autonomous operation some awareness of the surrounding environment is needed. The required extent of the awareness is dependent on the application and working environment of the robot. Autonomous lawnmowers are robots designed to automate the cutting of grass of lawns and gardens and were some of the first autonomous robots that became commercially available to the general public. But despite pioneering the field many still possess a rather basic understanding of their environment. An example of this is the state of the art and popular lawnmower "Automower" from Husqvarna. It is using a "planned randomness" approach when cutting grass, as it is unaware of its exact position in the garden. It picks a random direction and then cuts in a straight line until it either hits a physical object or a predefined edge, at which point it chooses another random direction and repeats the process. Given enough time the entire lawn will have been cut [1].

This is a rather ineffective method of cutting a lawn, a more systematic approach would be implementing a coverage path planning algorithm. A path planner could potentially decrease the cutting time as well as save energy and give the user more alternatives for the cutting operation. Implementing such an algorithm requires data about the pose of the robot, that is the combined position and orientation of the robot. Having pose data in relation to an arbitrary point is useful in some applications, but in the case of a lawnmower it is only useful in combination with additional knowledge about the environment, such as the location of obstacles and the perimeter boundaries. This data about the environment can be based on a predefined map, sensor inputs or a combination of the two [2]. In the case of a lawnmower, there is no predefined maps to use as gardens come in all forms and shapes, making it necessary for the robot to map the garden itself.

This master thesis investigates the possibilities of the Husqvarna Automower to gather reliable pose data with the sensors currently installed as well as to use a state of the art RTK (Real Time Kinematic) GPS sensor. This to explore how the pose estimation could be obtained to sufficiently map the environment for full path coverage and other future features for outdoor mobile robots.
1. Introduction

1.1 Background

The issue of navigating a robot in an unknown environment is referred to as the SLAM problem (Simultaneous Localization and Mapping). SLAM is a popular method for mobile exploration robots [3, 4, 5, 6, 7, 8]. Aulinas et al. [9] summarize the commonly used SLAM methods and evaluate them, a common feature for all of them is that they are probabilistic and that they all use long range sensors. SLAM algorithms have been developed to solve two problems at the same time. To detect and map the surroundings given the sensor data and the current pose and to localize the pose of the robot given the correspondence of the map to the current sensor data. The Automower operates in an environment where the boundaries are invisible until the robot physically reaches them, meaning they can only be detected on a short range. Although extensive research have been made on SLAM methods, most existing work relies on robots with long range sensors and substantial computation power. Relatively little research have been made on scenarios where only short range sensors are available. Zhang et al. [10] used position data from an odometer and wall collisions to successfully map an office building by utilizing rectilinear assumptions about the environment. This robot type is referred to as a bumper bot. The bumper bot problem differ from the traditional SLAM problem in that only one data point is available at any given time, instead of a sensor sweep with hundreds of data points to match with previous data. The bumper bot can not receive any feedback from the map to correct pose errors. The map generated from the bumper bot is directly correlated to the pose accuracy from the internal sensors of the robot. Although proven possible to map an environment with unreliable position data, implementing a coverage planner would likely require very accurate position data.

Sukkarieh et al. [11] used a Kalman filter to fuse sensor data from an IMU and a GPS on an outdoor mobile vehicle. By implementing GPS fault detection routines based on IMU data, they manage to get an accurate position of the vehicle. To achieve this, an advanced and expensive GPS unit was used, an alternative external positioning method is presented in [12] where Kurth uses six radio beacons as landmarks in a garden and by using triangulation calculates position data. Kurth managed to acquire a position accuracy of less than 0.2m by having additional dead reckoning measurements from an IMU and an odometer. In [13] Galceran conducts a survey of different ways to implement a coverage path planner algorithm to an unknown area, however, most methods relies on exact position data in conjunction with long range sensors. In [14] a contact sensor-based robot maps an environment by implementing a full coverage path planner, although the simulation proved successful, it relies on exact position data. The literature review shows that extensive research has been made into robotic mapping, positioning and coverage planning. The tasks are often hard to separate from each other, but mapping and coverage planning with a robot using only short range sensors and non-perfect positioning data presents a new research topic in a relatively uninvestigated part of the robotic field.
1. Introduction

1.1.1 Husqvarna Automower

Husqvarna is a world leading brand in the autonomous lawnmower product segment. They launched their first autonomous lawnmower in the Automower® series in 1995, this was also the first robotic lawnmower to become commercially available in the world. Since then a number of versions have been released, the most advanced model currently being the Automower 450X.

All the models in the Automower series use the same basic navigation principles. Apart from the lawnmower itself there is a base station which serves as a charging station, a boundary wire to define the outer perimeter of the operating area as well as guide wires to help the robot go through narrow passages and find its way back to the charging station. The basic operation consists of choosing a random direction and cutting in a straight line or a slight curve until the robot either detects the boundary wire or a physical object. After which it turns to another random direction and repeats the cycle. A current runs in the boundary wire, generating a magnetic field which is detected by three different sensors on the lawnmower, located with one in center front and two on each side on the rear of the robot. The direction of the magnetic field tells the lawnmower if it is inside or outside of the perimeter. By detecting pulses in the boundary wire, an Automower can also distinguish which boundary wire belongs to it in case several robots operate in the same area. Obstacle detection is done by physically ramming any objects obstructing the path and this will trigger push sensors attached to the chassis. In the higher end models there is also two front facing ultrasonic distance sensors that will slow the speed down prior to impact upon detecting objects in the robots path. The guide wire functions in a similar way as the boundary wire but instead of defining a boundary it defines a path that the robot can follow. When the robot encounters the guide wire it will follow it and it is meant to be placed through for example a narrow passage to guide the robot to another part of the garden or to lead it back to the base station. In addition to this there is also a sensor which can sense the height of the grass, in the case that the robot detects an area with long grass. Long grass indicates that it has not visited the area for some time and will then perform a spiral move to attempt to cover the area. The robot has tilt sensors to help protect the robot from attempting to climb too steep slopes or shutting it off to protect the user in case it is lifted. The high end models also contain a GPS sensor that is used for theft protection. The GPS module is likely not accurate enough for real time localization with navigational purposes.

1.2 Problem Description

In order to utilize a coverage planning algorithm, a map of the operating environment is required. In addition, information about the robots pose in relation to this map and the planned path is needed for a successful application. The localization and mapping problem is an extensively studied subject in the robotic community, but there is no off the shelf solution as each set of sensors and environments present their own challenges. Unique to this application is that the boundaries are invisible to the robot until it physically interacts with them, making the data collection points sparse as opposed to sonar or laser technologies commonly used when mapping unknown indoor environments. Since
the Automower is a product targeted to private consumer, constraints regarding product cost and the complexity of the system are also present.

The RTK GPS modules available were until recently only expensive units used in professional survey equipment and military applications. Little research has been done in combining the new cheaper units with other sensors and they are currently not used in any off the shelf consumer products. The background study suggest that a RTK GPS module could be a good alternative to achieve the high position accuracy that is needed for path planning features of the Automower platform. The research question for the thesis is formulated as the following:

**How can consumer RTK GPS technologies be used in conjunction with the existing hardware on the Automower platform for coverage path planning functionality?**

### 1.3 Scope

The research in this thesis will focus on the Husqvarna Automower and the existing hardware implemented on the platform. The following is considered to be the requirements in order to construct a full-coverage path planner for the Automower:

- Sub meter pose accuracy at all times.
- The created map should converge to the true map of the garden.

Modifications to the hardware set up may be done but had to be considered feasible to be installed on the real product, both in terms of budget and technical constraints. Furthermore it is assumed that there is GPS coverage within the entire area. The following constraints have been defined:

- The combined cost of auxiliary sensors must be justified in terms of increased functionality of the robot
- The position and mapping algorithms are not computed in real time but with logged data.
- The operating environment should be similar to that which is specified for the "Husqvarna Automower" today.
- The boundary wire will be used to detect the perimeters of the garden.
- No dynamic objects in the garden.

### 1.4 Method

This project was done in collaboration with Husqvarna, that provided software and hardware for research and development of the Automower product. The Automower platform was not available at the start of the project and had an expected lead time of one month, that later was extended to two months. During the lead time it was possible to simulate the system with an existing simulation environment provided from Husqvarna. The
1. Introduction

Figure 1.1: Communication architecture for unit tests.

The simulation environment is composed of a Gazebo-model of both the robot as well as a garden environment. Gazebo is used in conjunction with ROS (Robotic Operating System) [15]. Gazebo is a physics and visualization engine and ROS is a software framework and robotic communications network which can pipe data between different Gazebo models, external programs or to robots. In the simulation environment modular test can be made for sensor configurations, mapping algorithms and interfacing Matlab scripts. When the robot was available a validation test was conducted to evaluate the performance of the localization and mapping with the platform.

Function tests are done through the simulation where the dynamics sensor types can be simulated according to the movement of the Gazebo model, and filters can be adjusted to filter unwanted common behavior from sensors like slippage in encoders, drift in IMU and GPS signal bounces. Matlab scripts can be tested to record, log and post process sensor data that are published unto the ROS network.

The validation test is used to evaluate the performance of filtered sensor data, this can be done when both the GPS module and the Automower platform are accessible. The robot is set to "planned randomness" and sends odometer, collision and magnetic field data according to figure 1.2. A target computer logs data from the sensor gathering nodes and sends a time stamp to a secondary computer that logs GPS data. The logged data is used for later post processing to tune filter and mapping algorithms for the platform. The validation test seeks to evaluate the localization performance of this sensor configuration.

1.5 Sustainability and Ethics

This project is part of the field of robotics aiming to increase autonomy, which has the potential of revolutionize the world economy and social development. In this larger context, no single project or paper changes much, but put together with thousands of other projects such as this the effect on society is instead inevitable. In a longer time frame this project, in conjunction with other work, might contribute to make gardeners jobless as their task is made cheaper and more efficiently by an autonomous robot. It is debatable
1. Introduction

if the human race is better off using their untrained muscles to whack the forever growing grass of the garden or if this is an line of work inherently good for the robot community. The thesis take no stand on the question and we will live with the consequences of what the future bestows. There is no sustainability issues identified within the scope of this thesis.

1.6 Report Outline

This report is outlined as follows, Chapter 2 is the theoretical framework were associated research is discussed. It consists of a short survey of sensor technology, as well as positioning and path planning methods that have been deemed relevant in the scope of this research. In Chapter 3 the design and implementation is presented, this includes both the simulation as well as the implementation on the actual lawnmower. Chapter 3 describes the implementation of the Theoretical framework in the sense that it applies the theory to this specific problem. Chapter 4 contains the results from the implementation phase. Since much of this work is iterative, final results are presented alongside progressive results for a better overview. These results are then discussed and evaluated in Chapter 5 and the conclusions are presented in Chapter 6. The next section, Chapter 7, is a short section devoted to recommendations and future work that can be done in this specific topic. A nomenclature is found prior to the introduction chapter, code can be found in appendix A.1.
2

Theoretical Framework

This chapter summarizes the relevant background research that was undergone to understand the current state of the art for the field of mobile robot localization, control, mapping and path planning. First some explanation of the dynamics and the corresponding control algorithms of the differential drive robot are made. Included in the chapter is also information of the sensors that were available on the Automower as well as suitable auxiliary sensors. The RTK GPS sensor was the sensor that was deemed the most crucial after some investigation and is a large focus in this theoretical chapter. The different GPS technologies are discussed as well as some future systems that as of the time of the publication of this thesis is not yet in service.

Furthermore, different implementations of Kalman filters are investigated. Combining sensors can be done in many different ways and it was later determined that the simplest and most modular approach was implemented. To map the garden a few different mapping algorithms were explored. This includes occupancy grid map that was the final choice of mapping algorithm.

Finally some different approaches of full path coverage are mentioned. The path planner was never implemented but is seen as the final goal for this preliminary research. This is later explained in future work.
2. Theoretical Framework

2.1 Robot Kinematics

In order to simulate a robotics system it is important to model the physical system in such a way to capture the desired characteristics. This is also important in order to design a controller that will execute desired behavior. The Automower is a mobile robot in a two wheeled differential drive configuration. This is a robot that has two active wheels that can rotate independently on a common axis.

2.1.1 Dynamics in a Global frame

The dynamics of the robot can be modeled in two or three dimensions. This is a derivation of the kinematics for a two wheeled differential drive mobile robot on a plane surface with no wheel slippage. In figure 2.1 a schematic of a differential drive robot is illustrated, with a wheel spacing of length $L$. The robot rotates around a fixed point ICC with angle velocity $\omega$.

![Figure 2.1: Dynamics of one drive differential mobile robot.](image)

The robot is propelled forward by controlling the speed of each wheel, $V_l$ and $V_r$. When the speed of the two wheels differ the robot will rotate around a fixed point in the plane. This point is called ICC, Instantaneous Center of Curvature [16, 17]. The relationship between the speed of the wheels and the radius of the movement are used to calculate planned trajectories with regards to the control input of each wheel. The relationships between $V_l$, $V_r$, $\omega$ and distance $d$ can be derived according to the following set of equations.

\[
V_r = \omega (d + \frac{l}{2}) \tag{2.1}
\]

\[
V_l = \omega (d - \frac{l}{2}) \tag{2.2}
\]

From which $d$ and $\omega$ are solved to be

\[
d = \frac{l V_r + V_l}{2 V_r - V_l} \tag{2.3}
\]

\[
\omega = \frac{V_r - V_l}{l} \tag{2.4}
\]
When \( V_r = V_l \) the denominator for the radius approaches a singularity. This is when the robot is heading straight forward and should be handled with an exception in terms of implementation. Other interesting speeds are when \( V_l = -V_r \) setting the ICC in the center of the axis, rotating the robot with no translational motion. Setting the speed of either of the wheel will make the robot rotate around that wheel.

When translating the speed of each wheel to a rotation center with a rotation velocity \( \omega \) allows for transforming the odometry data to position changes within a global frame. Changes in the global positions of the robot \( \{x, y, \theta\} \) can be calculated at each time step \( \Delta t \). Calculating the position ICC \( [x, y] \) from the position of the robot can be done according to the following equation.

\[
ICC = [x - dsin(\theta), y + dcos(\theta)]
\]  (2.5)

After each timestep the new position \( \{\hat{x}, \hat{y}, \hat{\theta}\} \) of the robot is calculated as

\[
\begin{bmatrix}
\hat{x} \\
\hat{y} \\
\hat{\theta}
\end{bmatrix} =
\begin{bmatrix}
cos(\omega \Delta t) & -sin(\omega \Delta t) & 0 \\
sin(\omega \Delta t) & cos(\omega \Delta t) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x - ICC_x \\
y - ICC_y \\
\theta
\end{bmatrix} +
\begin{bmatrix}
ICC_x \\
ICC_y \\
\omega \Delta t
\end{bmatrix}
\]  (2.6)

where \( ICC_x \) and \( ICC_y \) are the \( x \) and \( y \) components of the position of ICC.

### 2.1.2 Control

In Section 2.1.1 it was shown how the controlled states of the rotational speeds of each axis affect the position of the robot in a global frame. The true position of the robot is never known, though it can be estimated in with sensory inputs and control outputs for systems with known dynamic behaviours. Regardless of how this information is computed this information of robots’ current position can be used to control future behaviour. For the two wheel differential drive mobile robot the control input to reach a target location could be calculated at each time step. The following notations are used in this chapter:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_l )</td>
<td>Velocity left wheel</td>
</tr>
<tr>
<td>( V_r )</td>
<td>Velocity right wheel</td>
</tr>
<tr>
<td>( v_\phi )</td>
<td>Rotational velocity</td>
</tr>
<tr>
<td>( v_\lambda )</td>
<td>Translational velocity</td>
</tr>
<tr>
<td>( x, y, \theta )</td>
<td>Current pose</td>
</tr>
<tr>
<td>( x_0, y_0, \theta_0 )</td>
<td>Starting pose</td>
</tr>
<tr>
<td>( x_g, y_g, \theta_g )</td>
<td>Goal pose</td>
</tr>
<tr>
<td>( x_p, y_p )</td>
<td>Control position</td>
</tr>
</tbody>
</table>

The two most critical features in order to plan a trajectory is to have rotation control and line following control. Rotation control concerns rotating the robot to a target position \( \theta_g \) with no translation, \( [x, y] = [x_0, y_0] \) and line following control is to move the robot along a line connecting the starting position \( [x_0, y_0] \) to a goal position \( [x_g, y_g] \). With these features a simple state machine architecture would suffice to make the robot move in the most complicated patterns by altering between controlling towards a goal orientation in the rotation controller case or towards a goal position in the line following controller.
Rotation control

Rotation control is used to precisely orientate the one wheel differential drive robot without changing its position. With a planned path the target angle is ideally calculated as the angle between the prior target position and the next target position. To translate the velocity of each wheel to rotation and translational dynamics of the robot the variables $v_\phi$ and $v_\lambda$ are introduced.

\[
\begin{align*}
v_\phi &= V_r - V_l \\
v_\lambda &= \frac{V_r + V_l}{2}
\end{align*}
\] (2.7) (2.8)

If the targeted translational velocity $v_\lambda$ and the targeted rotational velocity $v_\phi$ are known the velocity of each wheel can thus be calculated.

\[
\begin{align*}
V_r &= v_\lambda - \frac{v_\phi}{2} \\
V_l &= v_\lambda + \frac{v_\phi}{2}
\end{align*}
\] (2.9) (2.10)

The rotating speed could be calculated by a PID regulator [18, 19].

\[
v_\phi = K_\phi (\theta_g - \theta(t)) + \frac{1}{T_\phi} \int_0^t (\theta_g - \theta(t)) dt + D_\phi \frac{d(\theta_g - \theta(t))}{dt}
\] (2.11)

Where $\theta_g$ is the target orientation and $\theta(t)$ is the current orientation at time $t$. Translational speed is calculated to compensate for positioning error when the robot is turning. This can also be implemented with a PID regulator. A regulator with simple proportional gain would be.

\[
v_\lambda = K_\lambda [\cos(\theta(t)), \sin(\theta(t))] \left[ \begin{array}{c} x_0 - x(t) \\ y_0 - y(t) \end{array} \right]
\] (2.12)

No difference in current position and starting position leads to no translational speed. It might seem strange that this would be needed since when the rotating state is active the robot should be in the correct position. Though in practice it is almost always certain that the robot is not exactly in the correct position due to errors in dynamic modeling and sensory inputs. The translational speed tries to minimize the error in position in the direction $\theta(t)$ it is heading at time $t$. The speeds $v_\lambda$ and $v_\phi$ are written in continuous time domain but are in practise calculated in the discrete time domain. At each time step these speeds are calculated, transformed with equation 2.9 and set as control speeds for the right and left wheel respectively.

Line following control

When the robot should move position a line following algorithm would find a linear path between the starting position $x_0, y_0$ and target position $x_g, y_g$. The translational speed is calculated from how far away the robot is from the target position, assuming there is a linear path between the two points. The rotational velocity is calculated to minimize the
distance from an arbitrary point straight in front of the robot to the line connecting the
starting location to the goal location. The translational velocity is accordingly calculated
to the following equation.

\[ v_\lambda = K_\lambda \left[ \cos(\theta_g), \sin(\theta_g) \right] \left[ x_g - x(t), y_g - y(t) \right] \] (2.13)

where the angle \( \theta_g \) is defined to

\[ \theta_g = \angle([x_0, y_0], [x_g, y_g]). \] (2.14)

This minimizing the distance from the robot to the goal location but assumes that \( \theta_g \approx \theta[k] \).

The rotation speed of the robot is calculated with a regulator minimizing the distance
that the point \([x_p, y_p]\) projects on a vector \(\Phi\) that is perpendicular to the line between
the start and goal location. Varying the length \(p\) to the point in front of the robot yields
to different control behaviour, this could be chosen through trial and error. setting the
control parameter \(p\) to zero makes the control input to be the position of the robot.

\[ v_\phi = K_\phi \left[ \sin(\theta_g), -\cos(\theta_g) \right] \left[ x[k] - x_0 + p\cos(\theta[k]), y[k] - y_0 + p\sin(\theta[k]) \right] \]

This is a proportional regulator with the control input being a scalar product between
the vector \(\Phi\) and the point \([x_p, y_p]\). Like the rotational control algorithm the speeds \(v_\lambda\) and
\(v_\phi\) could be converted to velocities for the left and right wheel respectively.

### 2.2 Position Sensors

Sensors used in positioning can either provide absolute or relative data. They can be
linear, angular or multi-axis. The following are the position sensors that have been con-
sidered relevant in the scope of the thesis.
2. Theoretical Framework

Automower perimeter wire

As mentioned previously the Automower has a unique solution in order to detect the perimeter of the garden. Detailed information of the functionality of the system is a trade secret and can not be disclosed, though the basic principle will be presented. The perimeter wire is a copper wire that is dug into the garden between 1-20 cm depth [20]. The wire is connected to the charging station and loops around what the user determines is the edge of the garden. With the wire installed the charging station sends a current through the wire that induces a magnetic field. The signal in the wire is a identifier for the robot, this makes it possible to detect signals that could be received from other neighboring perimeter wires. The Automower has a magnetic sensor that detects and decodes the information in the magnetic field. If the Automower approaches a perimeter the amplitude of the signal increases, this is until the robot passes the wire and thus inverts the signal. One functionality of the Automower is to follow along a side of either the perimeter or the guide wire.

GNSS - Global Navigational Satellite System

GNSS is a common name of the different global satellite positioning systems. The GNSSs that are currently in full operation are the Russian GLONASS and the United States GPS, with the Chinese BeiDou and European Galileo not yet in full operation. This is since more satellites that are still awaiting launch are needed to complete the constellations.

All GNSSs work in a similar manner. The GPS system relies on signals received from a minimum of four out of 31 satellites to triangulate the position in both horizontal and vertical space anywhere on earth. The satellites send messages containing their position and time which is then used to calculate the receivers’ distance from the satellite. Having more than 4 satellite signals thus leads to an over defined equation system which can be solved with for example a least squares method to improve the accuracy. The system provides absolute data and each measurement is, in principle, independent from the last. Apart from the number of visible satellites, other factors contribute to the accuracy of the positioning. The configuration of the visible satellites greatly affects positioning accuracy as well as the Geometric Dilution of Precision (GDOP), which is a ratio of the position error to a satellite range error. There is also an error associated to each satellite signal due to ionospheric and tropospheric conditions affecting the time of flight of the signal. Having more channels in the GPS device in order to receive more signals simultaneously, if available, can improve the accuracy, as well as taking many measurements from a stationary point and average them over time. The largest error is usually connected to the unpredictable signal delay through the ionosphere, precise estimate of this delay is thus the key to better positioning. Advanced GPS receivers use a multiband approach to attempt correcting these errors, the satellites send signals on different bands and each frequency is affected by the ionosphere differently. As the delay is a function between frequency and total electron content along the path, comparing the delays gives a rather precise estimate of the true delay. This currently requires access to the encrypted military L2 band in addition to the civilian L1 band. It is however possible to make the calculations without decrypting the L2 signal, but it is a complex task only available on
expensive surveying equipment. Further enhancement of the accuracy requires external sources other than the GPS signal to correct the errors associated with the delay in the signal. Most augmented systems use the same method to achieve this. To have one or more stationary GPS receivers with a precisely known position, small variations in the satellite signals can be found and corrected. These corrections can include long term ephemeris and clock error estimates, as well as the ionospheric delay estimate based on calculations made on where the GPS signal pierced the ionosphere and the true time of flight. This information is then relayed to the receivers and corrected for the set of satellites that the particular unit is in view of. The method of collecting and relaying this information differ for different augmentation systems, the **Wide Area Augmentation System (WAAS)** has ground reference stations across North America that collects correction data which is sent to WAAS enabled receivers via geostationary satellites, it has a measured precision of 0.9m 95% of the time for a stationary unit under perfect conditions but the coverage is limited to North America. The European EGNOS and the Japanese MSAS system are other examples of the several satellite-based augmentation systems in operation. They each require special receivers and coverage can be limited or the correction information irrelevant outside certain areas. **DGPS**, or **Differential GPS**, works in a similar way but consists solely of ground based stations and is in operation mainly around waterways such as the Stockholm Archipelago and much of the American coast line, where a better accuracy is required. It uses long wave radio receivers to receive terrestrial radio messages containing the correction information. It can provide an accuracy of 1-2m for a stationary object, but due to legal reasons it is rated higher to 8m, swedish coast gard [21].

For more accurate local positioning, a technique called **Real Time Kinematic GPS** can be used. By adding a stationary reference receiver and not only relying on the actual content of the received signal a much higher accuracy of the relative position between the receivers can be achieved. This is done by also taking measurements of the phase of the carrier wave and compare this to the phase received at the non-stationary receiver and works as long as the units are less than 10km from each other. The principle is based on knowing the phase difference between a message received at two different receivers at the same time, and with the additional knowledge of the number of full wavelengths between the messages the difference in length to the sending satellite can be calculated. By repeating this for a number of satellite signals a precise position in local space can be determined.

The system requires more advanced receivers and fast processing as well as real time communication between the two units, usually UHF is used. The carrier frequency wavelength corresponds to a length of 19cm, a 1% accuracy in detecting the leading edge means that this component of the pseudorange error might be as low as +/-1.9mm, as compared with a typical error of 1-3m using only the information contained within the GPS signal. A significant problem is to match the leading edges between consecutive messages, a failure leads to an 'off by 19cm' problem in the accuracy. Statistical methods can be used to correct this, but it needs continuity in the tracked measurements to avoid a re-initialization of the phase-ambiguity filters, something that might be problematic in certain environments.

From a user perspective what contributes most to the performance of the GPS is the location of objects around the receiver. The GPS signal can’t penetrate more than a few
millimeters in many materials so objects obstructing a clear view of the sky can severely worsen the accuracy. Multipath errors are also a common problem associated with objects and surfaces in the vicinity of the receiver reflecting the signal making it appear to have travelled longer than it actually had. In conclusion it can be said that even the best receivers, using the most advanced correction techniques, will experience difficulties on locking a position in some environments, and the specified accuracy is usually for ideal conditions, the true results might at times differ significantly.

Table 2.1: Accuracy 95% confidence interval [22, 23, 24, 25, 21]. The data is for private users of each system, performance differs from location to location.

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>15m</td>
</tr>
<tr>
<td>DGPS</td>
<td>1-8m</td>
</tr>
<tr>
<td>Galileo</td>
<td>4m</td>
</tr>
<tr>
<td>RTK GPS</td>
<td>0.2m</td>
</tr>
<tr>
<td>GLONASS-M</td>
<td>4.5–7.4m</td>
</tr>
<tr>
<td>WAAS GPS</td>
<td>3m</td>
</tr>
</tbody>
</table>

Future of GNSS

The GNSS systems are thought to play an important role in the future, why extensive investments in getting the partly functioning systems completed have been granted. But also updates to the current fully operational GPS and GLONASS are underway, allowing for improvements in positioning also for non-military use. The new block III satellites are currently being rolled out in the GPS system, replacing the older block II satellites after a predetermined service schedule. The new block III satellites will transmit on more frequencies which will be available for civilian use, adding a L2C, L5 and L1C signals. The L2C signal will make it possible for dual frequency ionospheric correction, similar to what the military already does today. This will provide a much better theoretical precision. The L1C frequency is developed for interoperability with the Galileo and BeiDou systems, the new satellites will also transmit at higher power, allowing for better coverage under trees and other obstructions. Satellites with L2C frequencies have already been launched, the new generation block III satellites are scheduled to launch in spring 2018. Most improvements will be of limited use until the new frequencies are being broadcast from 18-24 satellites [26].

After years of decline due to turbulent political times in Russia after the collapse of the Soviet Union, restoring the functionality of the GLONASS system was made a top priority in 2001 and by 2011 it reached full operational capability. The current satellites in service are the second and third generation GLONASS-M and GLONASS-K1, with three open signals, L1OF, L2OF and L3OF, where the former two are FDMA and the latter a CDMA signal. Planned to begin launch in 2018 is the GLONASS-K2 which will add L1OC and L2OC CDMA frequencies. This will theoretically double the position accuracy according to the Russian Space Agency [24]. In the research phase is the GLONASS-KM satellite, which will also include frequencies that are shared with the Galileo, GPS block
III and BeiDou systems, making interoperability between the different systems better.

The Galileo GNSS system is currently being developed by the European Union space agency ESA. Much like its counterparts it will have a lower precision service open to anyone, and a higher precision capability available for military applications but also to paying commercial users. Under ideal conditions the Galileo system is designed to perform with an accuracy of 1m in the basic mode and with 1cm precision when using the dual frequency ionospheric correction available to paying customers. Once completed a total of 30 satellites will be in orbit, of which 6 are active spares, as of May 2017 there was 11 operational satellites in orbit [27], it is planned to be fully operational by 2020 according to the European Global Navigation Satellite System Agency in charge of the program [28]. Galileo is designed to work as a stand alone system but also for interoperability with other systems such as GLONASS and GPS when available.

China was initially involved in the Galileo project, but dropped out in 2006 to fully commit to their own GNSS called BeiDou which was already being developed. A limited test system, consisting of four satellites offering navigational services in China, have been operational since year 2000. The full scale second generation will consist of 35 satellites and will work globally upon completion, which is planned to 2020, making it a true GNSS. As of the end of 2016 there was 23 operational BeiDou satellites in orbit. The system can already offer positioning services to China and some neighbouring countries in Southeast Asia due to part of the constellation being geostationary. BeiDou will gradually expand its service area until full coverage is reached [29]. At a press conference in March 2017 a BeiDou engineer claimed that the positioning accuracy had been improved from the previous 10m to within 1-2m [30]. The BeiDou system operates in the L-band, similar to the other GNSSs, and will have two levels of positioning service, one public and one restricted for military use.

Each of these systems, once complete, will work independently from each other and are each capable of providing a user with position from anywhere in the world. Having receivers capable of receiving signals from all four systems might however greatly improve the accuracy, especially in "difficult" areas such as urban environments and other sites with limited view of the sky.

**Emlid Reach**

Emlid Reach is a RTK GNSS system that improves the GPS to sub-meter accuracy according to the manufacturer [23] and at the time of writing this report, a complete system with two units and two antennas is selling for $570 which makes it one of the most affordable RTK systems available on the market.
2. Theoretical Framework

The system is built around the Intel Edison platform with a U-blox GNSS receiver model NEO-M8T which can receive the common civilian GPS L1C/A carrier frequency as well as the GLONASS L10F, BeiDou B1, Galileo E1B/C and SBAS L1 C/A. Each module is delivered with an external Tallysman antenna. The system consists of two identical modules and they can be configured via an interface accessible from an internet browser, one unit is set to act as a base station and the other as a rover. The base station will broadcast its position as well as information about the received signals, this can be done over several different communication interfaces.

The base receives its initial position after sampling and averaging the position over time, this time period can be chosen to be between 2 to 30 minutes in the configuration tab of the reach unit. DGPS capabilities are also available for a better estimation of the position of the base unit. These corrections are commonly in the RTCM format and sent using the NTRIP protocol (Networked Transport of RTCM via Internet Protocol), which enables a stationary DGPS receiver to broadcast correction data over the internet. In Sweden there are several NTRIP broadcasting stations made available by Lantmäteriet [31]. Once the base unit has determined its position it will start broadcasting correction data to the rover, the rover will change from "Single" solution mode, meaning it is only positioning itself using the normal GPS data, to 'Float' solution once it receives the base corrections, meaning it is correcting itself in relation to the base but that it has not yet resolved the phase ambiguity correctly. When the Emlid Reach reports 'fix' the phase ambiguity is resolved and should report the position of the module with at least 19 cm accuracy. The Emlid Reach offers three modes of resolving the ambiguity, continuous, instantaneous or fix-and-hold. The continuous resolver means that the ambiguity is resolved epoch by epoch, this can be unstable and the fix might be lost frequently, but it guarantees that the fix isn’t false. The Fix-and-Hold resolver holds the initial fix constrained making it more stable, but if the first fix was incorrect it will take longer to recover, it is comparable to giving the fix point inertia, making it more difficult to move [32]. There is also an option to enable an ambiguity resolver based on the GLONASS satellite constellation, which may be used if both the rover and base support GLONASS, which is the case when using only Emlid Reach modules, but might not work when also using external NTRIP correction data. Enabling this will provide an additional source of resolving the ambiguity and works in parallel with the regular GPS AR.
Wheel rotation data

Many mechanical machines transforms rotational movement into linear movement. In order to control the position of the two wheel differential drive mobile robot the precise rotation of the left and right wheel must be controlled. This is often done with encoders or stepper motors. The former detects how much the encoder shaft has turned by either an absolute or by incremental mechanism. To control the motor speed in this way requires feedback control. The stepper motor is an open loop system, this means that the rotation of the shaft is controlled directly and requires no feedback. Both these methods control or detect the rotational position of the shaft and not the global position of the robot and it is the most widely used navigation method in mobile robot positioning [33]. They are cheap, gives high resolution and sampling time. The two wheeled differential drive mobile platform is very sensitive to angle errors. With a small angle error a very large positioning error accumulates proportional to the distance traveled when heading in a straight line, \( [x_{\text{error}}, y_{\text{error}}] = [l \cos(\theta_{\text{error}}), l \sin(\theta_{\text{error}})] \). Where \( l \) denotes the length of path since orientation error \( \theta_{\text{error}} \). The error in pose accumulates with time and quickly disorients the robot within the environment. For outdoor environments the angle error can occur due to differences in the static friction between the wheels or by topological differences where one of the wheels travel a longer path where the topological modeling of the environment does not correspond to reality. These errors are very hard to detect but could be complimented with more sensor types to calculate a better estimate of the true position of the robot, with algorithms like the Kalman filter [34].

When the robot is rolling in the environment this could be used to exactly determine change in position and would then be no need for auxiliary sensors. What becomes troublesome is when the friction coefficient between the wheel and the ground becomes too small to overcome the torque of the motor. Slippage occurs and the odometry data no longer corresponds to the actual position of the robot. There are also other errors like misalignment of wheels, error in wheel diameter and encoder or step resolution. The odometry errors can be divided into systematic and non-systematic errors. Where the systematic errors are connected to the imperfections of the kinematics of the robot. Some papers investigates ways of quantifying the systematic errors in a repeatable way [35] [36]. Borenstein and Feng [37] developed a method to detect the systematic errors by moving a robot in a quadratic movement and detecting the orientation error when rotating 90 degrees. This method is called UMBmark (University of Michigan Benchmark) and correlates asymmetry in center of gravity in respect of clockwise rotation and counter clockwise rotation to systematic errors. Rekleitis, [36] traced the orientation error for rotational and translational movement for a one drive differential robot in different indoor environments. The empirical data suggested orientation error of \( \pm 1.5^\circ \) when performing rotational movement and \( \pm 0.5^\circ \) for translational movement with lower orientation error for faster translational velocities. Standardized methods for non-systematic error measurement have been proposed before with the extended UMBmark [37]. The non-systematic errors are heavily dependant on the surface, this makes it difficult to compare systems for slippage detection purposes. Often non-systematic errors are not modeled. This relies for the Kalman filter or other software solutions to detect and compensate for.
2. Theoretical Framework

Compass
An electronic compass is essentially a magnetometer, or a number of magnetometers, showing the direction to earth's magnetic north pole. It can be used to determine an object's orientation, or more specifically, an object's direction in relation to the magnetic north pole. It is an absolute sensor that can provide data independent of previous measurements. The compass can be used to determine the angle of the robot with respect to a coordinate system before making the initial move, something which is not possible with the other sensor technologies discussed. Relatively good accuracy, typically of less than one degree error, can be achieved even with cheap off the shelf electronic compasses and state of the art equipment is capable of less than 0.3 degree error. However, large errors might occur due to the operating environment, as the compass will react to any magnetic field in its vicinity, regardless if it is an artificial magnetic field or from earth. Apart from magnetic metals, all current carrying wires produces a magnetic field, the disturbances might be significant in certain areas and close to unshielded electronic equipment. Furthermore, earth's magnetic poles are not located in the geographic poles, when travelling over distances and using a map this would have to be taken into consideration and compensated for, however, in the short distances of a garden this would likely not produce a measurable error unless placed in close proximity to one of the magnetic poles. As the vertical direction of the magnetic field changes with longitude, approaching a 90 degree angle at the actual poles, the measured component of the force vector becomes smaller the further north or south the sensor is moved, which contribute to a larger error at these longitudes.

IMU
An Inertial Measurement Unit (IMU) is a device used to track the specific force and angular rate of an object using a combination of accelerometers and gyroscopes, sometimes magnetometers to provide additional information about the magnetic field can be included. To fully orient an object with 6 DOF in 3D space, see figure 2.4, at least three gyroscopes and three accelerometers are needed, one of each for each axis respectively.

![Figure 2.4: Six degrees of Freedom](image)

The data provided can be used to orient and position an object in space by integrating the acceleration twice to get the position or integrating the angular velocity to get the
angle, a method known as dead reckoning. The position is estimated by advancing from a previously estimated position with the data provided by the IMU, which means that all new position data will be dependent on previous data as opposed to for example a GPS which provides absolute data. The performance of the IMU is dependant on the sensors it contains, and a range of performance options are available, usually ranging from $0.1\ ^\circ_s$ to $0.001\ ^\circ_h$ for the gyroscope and $100\ mg$ ($0.982\ \frac{m}{s^2}$) to $10\ \mu g$ ($9.82\cdot 10^{-5}\ \frac{m}{s^2}$) for the accelerometer. Regardless of the quality of the sensor some errors will still be present. As the position is based on a previous estimate, any error, regardless how small, accumulates over time in something known as sensor drift. If this error is a spread around a constant offset it is known as sensor bias, the bias is the difference between the real value and the output value of the sensor, denoted by $b$ in equation 2.15, which is the single axis case with $x$ being the output, and $S$ the scale factor.

$$x = Sf(x) + b$$ (2.15)

This can be broken up into two components to describe the behaviour of the bias over time; Bias Repeat-ability and Bias Stability [38]. The former describes how much the bias differ between each power up, the offset might vary due to physical properties and conditions in the IMU itself, a very repeatable bias allows for a quick compensation with calibration. The latter describes how the initial bias change over time while running. This change is often related to change in temperature, time or mechanical stress on the system and could be somewhat compensated for with external sensors and accurate models, and a more stable bias makes a more accurate correction possible. In a perfect case, the sensors would also be placed orthogonal to each other so that each axis is independent from one another. In practice however, there will be alignment errors between the axis, this is referred to as sensor non-orthogonality and will lead to a correlation between the sensors and inaccurate readings. Careful manufacturing can help reduce this error, but even in expensive units this will still be present, which makes it necessary to make an estimation of the offset between the IMU measurement frame and the sensor frame and continuously correct for this during operation. A rotation transformation matrix might be used to translate the axes into the local frame of the robot, given that the angles $\alpha$, $\beta$ and $\gamma$, which are the rotation around the x, y and z axis respectively, are known. This is described in Equation 2.16, where $c$ represents cosinus and $s$ represents sinus.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c(\beta)c(\gamma) & c(\beta)s(\gamma) & -s(\beta) \\ s(\alpha)s(\beta)c(\gamma) - c(\alpha)s(\gamma) & s(\alpha)s(\beta)s(\gamma) + c(\alpha)c(\gamma) & s(\alpha)c(\beta) \\ c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) & c(\alpha)s(\beta)s(\gamma) - s(\alpha)c(\gamma) & c(\alpha)c(\beta) \end{bmatrix} \begin{bmatrix} x_{\text{IMU}} \\ y_{\text{IMU}} \\ z_{\text{IMU}} \end{bmatrix}$$ (2.16)

Equation 2.16 only works if the axes are orthogonal to each other, in the case that they are not, another transformation matrix has to be used to first translate the readings to a right hand orthogonal coordinate system. The IMU samples data at discrete time steps that creates truncating errors, each data point is treated as an average for the sample period, which isn’t necessarily true, especially for objects that’s changing acceleration quickly. A typical IMU use a measurement frequency of 30 kHz, but state of the art sensors can have much higher update frequencies. Further more, the bandwidth of the sensors affects the possible input rate, a low bandwidth sensor won’t be able to measure high
frequency vibrations, which by aliasing might appear as a lower frequency acceleration. As the estimates are made by integration of data, even small errors can produce significant drift quickly as a constant error in acceleration results in a linear error in velocity and a quadratic error in position. In many positioning applications an absolute data source is used to correct the IMU and reset the drift at a regular interval before it grows too large, this could be done by using known landmarks, a GNSS module or any other source of absolute position data. The most exact units are both expensive, bulky and usually reserved for military applications.

2.3 Kalman Filter

The Kalman filter is an algorithm used to calculate estimates of unknown variables based on unreliable and noisy measurements observed over time. This is a way to fuse sensor data. The data from each sensor can tell something about the state of a process, but only indirectly or with some uncertainty or inaccuracy. Based on models an estimation of the next state could be made from the current state, but this prediction also has some uncertainty or inaccuracy. There is a whole range of possible states that might be true, but some of them are more likely than others. The Kalman Filter combines any number of state estimations from sensor readings and the model of the process and using probability theory attempts to find an estimate that’s better than any of the estimates would be by themselves. It works in a two step process, the prediction step and the update step. The prediction step uses prior knowledge of states to estimate the state for the next time step. In the update step this prediction is compared to the observed data and a gain is adjusted for the sensors readings depending on how well the prediction and the reading match.

The Kalman filter aims to estimate the current states $x_k$, given the prior states of $x_{k-1}$ and control input $u_{k-1}$ in the prediction step. This require some correlation between these variables.

$$x_k = f(x_{k-1}, u_{k-1}) + q_k$$

$q_k$ is some process bias or noise. Similarly the update process takes new observed data $z_k$ and corrects those to the current states and requires a correlation between the two.

$$z_k = h(x_{k-1}) + r_k$$

$r_k$ is observation noise. Following is the process to implement the Kalman filter and is also illustrated in Figure 2.5, notation is partly taken from Zarchan et al [39].

**Prediction Step**

$$x_{k+1} = Fx_k + Bu_k$$

$$P_{k+1} = FP_kF^T + Q_k$$

**Update Step**

$$K_k = P_kH^T(HP_kH^T + R)^{-1}$$
2. Theoretical Framework

\[ x_k = x_{k-1} + K_k(z_k - Hx_k) \]  \hspace{1cm} (2.22)

\[ P_k = P_{k-1} - K_kH P_{k-1} \]  \hspace{1cm} (2.23)

**Figure 2.5:** The Kalman Filter Process

The filter assumes that every sensor reading is having a Gaussian error distribution. Each variable has a mean, which is the most likely true value, and a variance, which is the uncertainty associated with the reading. The variables might be correlated, which means that the state of one variable will tell something about the state of another, which is captured in the **covariance matrix** \( P_k \). This is a symmetric matrix where each element represents the degree of correlation between different states. It is updated in the prediction step, thus also making non apparent state relationships appear after a few iterations. The state vector \( \hat{x}_k \) represents the current best estimate, or the mean from a statistical point of view. The **state transition matrix** \( F_k \) contains a function model of the system that predicts the next state given the current state estimate, this could for example be basic kinematics formulas. There might be additional external influences on the system, such as issued commands by an operator to control the system. This is represented by the **control vector** \( \vec{u}_k \) which contains the input and the **control input model** \( B \) translates these to the system model. There is also external influences on the system from the environment it operates in, over which there is no control, such as wheel slippage due to uneven surface or a gust of wind. This leads to an uncertainty in the prediction regardless of the accuracy of the previous estimate and these untracked influences are modeled with **process noise** \( Q_k \), this co-variance could be functions and vary depending on for example operating mode. Together these matrices and vectors makes up Equation 2.19 and 2.20 in the prediction step, the **best estimate** \( \hat{x}_k \) is a prediction made from the previous best estimate \( F_k \hat{x}_{k-1} \), plus a correction for known external influences \( B_k \vec{u}_k \). The new uncertainty \( P_k \) is predicted from the old uncertainty \( F_k P_{k-1} F_k^T \) with some additional uncertainty from the environment \( Q_k \) [40].

This estimation is then refined with the sensor readings, these readings might need to be converted into the same units and scales as the states that are being tracked. This
is done in the observation model $H_k$ matrix which models the sensors to the system. Every state in the original estimate might result in a range of sensor readings, and from every observed reading a guess about the system being in a particular state can be made. But because there is uncertainty associated with each reading, some states are more likely than others to have produced the observed reading, this uncertainty is due to the sensor noise and the co-variance of this sensor noise is described in the observation noise matrix $R_k$, this matrix could consist of constants or of functions depending on the sensor noise behaviour. The readings from the sensors can be regarded as the means of the distributions and are contained in the observation vector $\mathbf{z}_k$. The predictions from the previous step could be wrong, or the sensor readings could be miss-measured, in any case they will likely differ from each other to some extent. One could think of each prediction and each sensor reading of being a Gaussian distribution, which could be more or less concentrated around a mean and when plotted upon each other they will overlap in certain areas. By multiplying these means and co-variance matrices, a new mean and co-variance of this overlap is found. This represents the configuration for which all estimates are the most likely and is therefore also the best guess of the states given the available information. This calculation happens in equation 2.22 and in this equation there is also the Kalman gain matrix $K'$, and is calculated in the step before. The Kalman gain is adjusted in the update step in Equation 2.21 and controls how much trust should be put on the measurements over the estimations. The last part of the update step updates the co-variance matrix given the new Kalman gain.

### Extended Kalman filter

If the system is non-linear, an Extended Kalman Filter may be used instead, that is a slight modification to the regular filter. A non-linear system has a non linear state transition matrix $F$ or observation matrix $H$. This requires these matrices are redefined as the equivalent Jacobian matrices for equation 2.17 and 2.18.

\[
F = \frac{\partial f}{\partial x} \tag{2.24}
\]

\[
H = \frac{\partial h}{\partial x} \tag{2.25}
\]

For known systems this can be linearized around the current state mathematically. If the matrices are unknown, the derivation is usually done by approximation with the finite difference method, which is just taking difference between two consecutive readings and dividing it by the time step between them as seen in Equation 2.26.

\[
f'(t) = \lim_{\Delta t \to 0} \frac{f(t + \Delta t) - f(t)}{\Delta t} \approx \frac{y_{k+1} - y_k}{\text{timestep}} \tag{2.26}
\]

### Unscented Kalman filter

The Unscented Kalman Filter is another closely related sensor filtering method that works on non-linear systems. It is especially useful when the predict and update functions, $H_k$ and $F_k$, are highly non-linear in which case the Extended Kalman Filter can give particularly poor performance. The issue with having a non-linear model is that although
the input variables are Gaussian random variables, after they have been passed through the model they are not. The EKF solves this by propagating the estimates through a linearization of the nonlinear system, thus making the output from the model a random Gaussian variable as well, but this does not always give a good result since it depends on how well the functions can be linearized. One solution is to pass a number of values through the true nonlinear functions and although the output isn’t Gaussian distributed, one can create one by calculating a mean and the standard deviation from these values. This is however time consuming and not always possible to apply in real time, there is also some difficulties associated with choosing what points and how many to pass through the functions to create the mean. The unscented transform is a method to deal with these issues, it uses the above approach of finding the mean and co-variance and the method is focused around selecting good points to base the calculations on. These points are called sigma points and are located at the mean and spread symmetrically along the main axes of the co-variance, giving two symmetrical points per dimension. If \( x \) is an \( n \)-dimensional Gaussian variable with the mean \( \mu \) and co-variance \( \sigma \) there will be \( 2n + 1 \) number of sigma points which are determined by the following rule:

\[
\chi^{[0]} = \mu
\]

\[
\chi^{[i]} = \mu + (\sqrt{(n+\lambda)} \sigma)_i \quad \text{for } i = 1, \ldots, n
\]

\[
\chi^{[i]} = \mu - (\sqrt{(n+\lambda)} \sigma)_{i-n} \quad \text{for } i = n+1, \ldots, 2n
\]

Where \( \lambda = \alpha^2(n+\kappa) - n \) with \( \alpha \) and \( \kappa \) are scaling parameters that can be chosen to determine how far the sigma points should be spread from the mean. Each of the sigma points also have two weights associated to it, \( w_{m}^{[i]} \) and \( w_{c}^{[i]} \), one used when computing the mean after the points have been passed through the nonlinear functions, and another for recovering the covariance [41]. These are calculated as follows:

\[
w_{m}^{[0]} = \frac{\lambda}{n + \lambda} \quad (2.30)
\]

\[
w_{c}^{[0]} = \frac{\lambda}{n + \lambda} + (1 + \alpha^2 + \beta) \quad (2.31)
\]

\[
w_{m}^{[i]} = w_{c}^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n
\]

\( \beta \) can be tweaked if more information about the underlying Gaussian representation is available, if the distribution is an exact Gaussian then \( \beta = 2 \) is the optimal choice [41]. If \( g(x) \) is a nonlinear function passing the sigma points through this gives, \( \gamma^{[i]} = g(\chi^{[i]}) \) from which the mean and covariance can be calculated as

\[
\mu' = \sum_{i=0}^{2n} w_{m}^{[i]} \gamma^{[i]} \quad (2.33)
\]

\[
\sigma' = \sum_{i=0}^{2n} w_{c}^{[i]}(\gamma^{[i]} - \mu')(\gamma^{[i]} - \mu')^T \quad (2.34)
\]
The implementation of the Kalman filter differs depending on the application, the underlying system model, sensor choice and other system design choices. There is rarely an out of the box solution to apply to a given problem, instead numerous more extensions and generalizations have been developed that are then tweaked for the specific problem. A way to make the implementation easier is to use a Decoupled Kalman filter, which basically divides the problem into several filters that are run independently from each other, only feeding the relevant state estimations to the next one. In robotics it is not uncommon to apply nonlinear transformations to measurements before feeding them into a filter [42]. Having the filter separated makes it possible to do the non-linear transformations outside of the filters. Then simple Kalman filters can be applied to the linear equations instead, making otherwise non-linear filters into multiple linear filters.

2.4 Maps

Robotic maps are representations of the environment and comes in two main categories Metric maps and Topological maps. Metric maps captures the geometric properties of the environment in a coordinate system. They make sense from a human point of view but as all entities are described within one absolute coordinate system it is not possible to leave relations between certain entities completely unspecified, even if the true relation is unknown. This is a problem as they require precise coordinate data or else global inconsistencies might occur [43]. Topological maps on the other hand describes the connectivity of different places in the environment, the map is then represented as a graph, where significant places are in a list of nodes and the arcs corresponds to the paths. Each arc is usually annotated with information about how to navigate from one significant place to another. One big advantage of this compared to the metric map is that only relations that have been directly observed, and thus could be considered reliable, are listed [44]. Furthermore, robotic maps can be divided into world-centric and robot-centric maps, where the former is represented in a global coordinate space and the entities in the map does not carry information about sensor measurements associated with them. The robot-centric maps are instead described in measurement space, with the sensor measurements a robot would receive in different locations. The latter has the advantage of not having to translate what is measured into world coordinates, but there is often difficulties associated with extrapolating individual measurements from measurements at nearby, unexplored places. There is also substantial difficulties with telling similar places apart. For these reasons the world-centric approach is the dominating method used for robotic mapping [43].

Occupancy grid maps

Occupancy grid maps is a type of metric map where a slice of a 3D world is represented by a 2D grid. The resolution is determined by the grain-size of the grid, where each cell can be either occupied or unoccupied. An unoccupied cell block means that the robot can transverse it and an occupied cell means it can not. In order to be used in conjunction with probabilistic mapping methods an uncertainty about the occupancy state might be associated with each cell block. This is then continuously altered depending on if the sensor readings suggest occupancy of a cell block or not [44]. This allows for raw sensor
2. Theoretical Framework

data to be used directly to build the map rather than process the readings in order to identify landmarks, meaning no information has to be discarded. However, this type of mapping contains a huge number of map features, one for each grid cell, this complexity limits the type of localization and mapping techniques that can be used [45]. Since each grid cell is a binary value of occupied or free it leads to that the number of possible maps are \(2^n\) where \(n\) is the number cells in the map. The Automower can cut an area of 5000 \(m^2\) and with a cell size of 0.1 \(m\) leads to a map with \(2^{50000}\) possibilities.

The following notations are taken from Thurn et al, [46] where occupancy grid maps are explained in further detail. Occupancy grid maps intend to estimate the most likely map \(m\) given the prior observations \(z_{0:t}\) and prior poses \(x_{0:t}\) between time \(0\) and time \(t\).

\[
\arg\max(p(m|z_{0:t}, x_{0:t}))
\] (2.35)

for all possible maps \(m\) where \(\arg\max\) is the maximum argument. In order to find a good map some smart algorithms need to be implemented since computing the probability of all the maps by brute force is not feasible. The complexity can be reduced through filtering each cell independently.

\[
p(m|z_{0:t}, x_{0:t}) = \prod_{i=0}^{n} p(m_i|z_{0:t}, x_{0:t})
\] (2.36)

In order to calculate the probability of cell \(m_i\) the the prior value is needed \(p(m_i|z_{0:t-1}, x_{0:t-1})\), as well as the probability of the cell given the current observation and pose \(p(m_i|z_t, x_t)\). If possible a few assumptions can be made to simplify a system.

- The true pose of the robot is known.
- The environment is static.
- Each grid cell is independent.
- Cells are just as likely to be occupied as free.

Thrun et al, [47, 46] derive an expression through Markov assumptions and Bayes rule to the following.

\[
\frac{p(m_i|z_{0:t})}{1-p(m_i|z_{0:t})} = \frac{p(m_i|z_t)}{1-p(m_i|z_t)} \frac{p(m_i|z_{0:t-1})}{1-p(m_i|z_{0:t-1})}
\] (2.37)

Equation 2.37 can be used to calculate a new probability for cell \(m_i\) from new observations. To solve for \(p(m_i|z_{0:t})\) in 2.37 requires large computational power, this can be reduced with computing in logarithms instead to turn the products into sums.

Position changes of the robot makes nearby cells update their probability of occupancy. The updated probabilities for the cells can then at any time be used to compute an updated map. In Matlab each cell has a occupancy probability associated to it, and setting probability threshold parameters defines if the cell is occupied, unoccupied or if its occupancy is unknown.
2. Theoretical Framework

2.5 Coverage Planning Algorithms

A coverage path planner is an algorithm designed to offer an optimized complete coverage path of an area as opposed to the more researched topic point-to-point path planning which only deals with the most efficient way to move from one point to another [48]. Typically, this involves minimizing the length of the total travelled path and keeping the repetition rate as low as possible while still achieving 100% coverage. The area to be covered might also be dynamic with moving objects or not fully known, in which case an online algorithm has to be used. If the entire path is calculated beforehand and not adapted during operation this is referred to an offline coverage path planner. Coverage path planners are commonly used for example in cleaning robots, agricultural vehicles, search and rescue robots and mapping UVs [48].

There are many approaches for constructing a coverage path planner, the most common way relies on dividing the area into a grid. Each cell block of the grid holds data about if the cell has already been covered by the robot or not, or if there is an obstacle in that particular block. This could be binary data, or if the position data is uncertain, a probability value. The coverage planner is by this method closely linked with the choice of map representation. Some examples of commonly used simple complete coverage path planning algorithms are the random coverage method, which is currently used in the Automower, the internal spiral method and the U-turn method [48], each of these have their strengths and weaknesses.

The random coverage method is basically just a robot driving around in a random pattern, and over time the probability that the entire area will have been covered will converge towards one. This is not cost or time effective and has a low path repeat-ability, it is however simple to implement, works in unknown areas and it works well in dynamic environments. The internal spiral method is a more systematic approach, by following the edges of an area and turning every time either an obstacle or a previously covered cell block is encountered, simple geometries can be covered more efficiently, but as soon as the geometry of the area is a little bit more complex the robot will lose track and full coverage can’t be ensured, see figure 2.6. The U-turn method goes straight until an edge is encountered, where it makes a 180 degree turn while shifting its position to not return over the same path and does this until the entire area is covered, similar to how a tractor plows a field. This also works well on simple geometries but objects in the path or even a slightly more complex shape will make it difficult to ensure full coverage, see figure 2.7. However, these simple methods of planned path coverage can form a basis for a coverage algorithm that works in more general environments [49].
A helpful tool when constructing a coverage path planning algorithm is the *Exact Cellular Decomposition* which breaks down the mapped environment into a collection of non-overlapping regions, these are then associated to each other with a connectivity graph representing the adjacency relation between these regions [50]. The complete coverage can be ensured if every region in the decomposition have been visited and covered. The advantage of this is that it subdivides a complex problem into a set of smaller and less complex tasks. A simple coverage algorithm can then be applied within each region and another coverage algorithm to make sure each region is visited, making it a much simpler task to optimize the entire problem. It is also useful in case dynamic objects appear, as only the path within the affected region has to be recalculated. A problem associated with this approach is that moving between regions might require overlapping a path already taken, which leads to a less optimized solution [49]. In a dynamic, unknown environment an exact cellular decomposition called the *Boustrophedon cellular decomposition* may be used [49]. The basic principle is based on creating each region, or cell, such that a full coverage path of the cell can be readily determined with a simple pattern such as the basic back and forth pattern used in the U-turn method as shown in figure 2.7, and thus reducing the full coverage planning problem to the planning of motions from one cell to another [51], which can be treated as an optimization problem, where each cell block is a node in a graph with a certain cost associated with it. The coverage path can then be optimized as a travelling salesman problem. The cell blocks are created by moving a sweep line across the map perpendicular to its direction, each time the connectivity of this line is changed due to the presence of an object, a new cell is created, and when the connectivity of the sweep line increases, two or more cells are closed in addition to creating a new cell block [51].
2. Theoretical Framework
This chapter discusses some of the steps taken to implement theory discussed in Chapter 2. No hardware was available in the first part of the project and resulted in simulation of the system to develop communication software, as well as to understand the software framework the Automower platform uses. The software architecture is shortly described in the beginning of the chapter. Afterwards some of the reasoning behind sensor choice is presented as well as more detailed information about usage of the Emlid Reach GPS module. Implementations of the filter design and mapping algorithms are also presented in detail. The final part of the chapter covers detailed information of how the test was constructed. It includes software architecture for logging, physical setup of the perimeter wire and hardware setup up of the auxiliary sensors.
3. Design and Implementation

3.1 Simulation

The main objective with the simulation environment was to develop functions for mapping, fusing sensor data and communicate with the ROS (robotic operating system) network and Matlab. The simulation environment provided from Husqvarna is designed to replicate the output from existing sensors on the platform. Though using the system to tune the Kalman filter was proven to be quite ineffective. This was due to the sensors reported very simple errors that does not represent reality, mainly just added white noise to the simulated output value. Some efforts were made to artificially create slippage between wheel and ground, but was later discarded.

ROS is a set of software libraries and tools that can assist when building robot applications, which are all open source [15]. The operating system functions mainly as a communication network to send data between nodes in the network. Data is sent in between nodes through routing messages with topics, that can either be published or subscribed to. It follows the same logic as a strongly typed message bus [52]. In the simulation environment the ROS node sends data to a Gazebo node to simulate the robot. The Gazebo node handles the kinematics and dynamics of the robot with the world. In order to visualize the robot a Matlab node was constructed. The Matlab node subscribes to the topics that are of interest, like wheel encoder data, GPS module etc. Matlab supports ROS through the 'Robotics System Toolbox’. Real time visualization and sensor fusion can be done as well as more computational extensive post processing like generating a map given observed data.

The framework for communication between Matlab and ROS was constructed using the simulation environment. In figure 3.2 mapping functions written in Matlab are tested in a simulated area. The mapping algorithm is a occupancy grid map, occupied cells are detected through the collision sensor in a garden made of walls in an empty gazebo world seen in figure 3.1. The robot is programmed to simulate the 'planned randomness' functionality that is used in the validation test.

![Figure 3.1: Simulation Environment](image1)

![Figure 3.2: Map constructed in the simulation environment.](image2)
3. Design and Implementation

3.2 Hardware Implementation

The following is a summary of the process for the choice of sensors, filter design in terms of mapping and sensor fusion. The implementation for a test setup is also presented.

3.2.1 Sensor Choice

The Automower is already equipped with a number of sensors. This includes wheel encoders as well as a simple GPS unit, an IMU and sensors to detect the perimeter wire. The background study suggested that a sensor reporting accurate absolute data at certain intervals was needed. An absolute sensor could report data to compensate for drift and let IMU and wheel encoders provide data in between these data points. After a review of the built-in GPS it was concluded that this would not be able to provide accurate enough absolute positioning data for coverage path planning capabilities. Instead a number of sensors were considered for use. Using a compass for absolute angle data was ruled out due to strong magnetic fields in the vicinity of the perimeter wire, where most turns would take place. Other options considered included a radio beacon system and the use of a camera for visual odometry and location recognition. Both of these could have the potential to work, but was ruled out in favour of a RTK GPS unit. The decision to use the RTK GPS system over the other options was based on it being similar to the already implemented system, only requiring an extra receiver in the base. The background research shows that there’s a lot of planned development in the GNSS field, with improved accuracy in the near future. Although an RTK GPS work a little bit differently than these, it was concluded that by using it, this research might provide some insight in what could be possible once these new systems are deployed as well. In addition to the RTK GPS, it was decided to add two additional IMUs to the Automower, mainly since the internal IMU was not available to read through the supplied software libraries from Husqvarna, using two IMUs also allows for averaging the measurements to reduce some of the sensor noise. The IMU’s used is the MPU6050 from InvenSence, which is the same type of unit as is built into the Automower according to the documentation.

Using this sensor configuration, position data can be derived from three different sources, the IMU, the GPS and the wheel encoders. The IMU is fixed in the local frame of the robot and provides the acceleration in three axis in relation to the robot, it also provides the angular rate in relation to a fixed starting point. The accelerometer needed to be integrated twice for position changes. After investigation of the integrated acceleration this was shown to have too low signal to noise ratio to be used for pose estimation. The wheel encoders provides the number of ticks that each wheel generates as they turn, which is converted into approximate speed and distance travelled, comparing the distances travelled by each wheel and knowing the geometry of the robot the angle can also be deduced. The GPS provides the position in all three directions in a global frame as well as giving a global heading during linear movement.
3. Design and Implementation

3.2.2 Emlid Reach Implementation

The two Emlid reach modules were configured to connect via an external WiFi router and having the base station send corrections to the rover unit over TCP/IP. The rover unit was configured to broadcast its position in a XYZ format, also using the TCP/IP protocol, with a Matlab TCP/IP server in the receiving end. The unit acting as the base station was fixed in position and configured to broadcast its position after a 5 minute averaging period, the rover unit was attached to the Automower with the antenna mounted on top with a clear unobstructed view of the sky.

For good results, the Emlid Reach requires a clear unobstructed view of the sky at least 30 degrees above the horizon, it is also sensible to radio interference from electronic devices. The Tallysman antenna was for this reason placed on a metal ground plane and as far away from other electronic components as possible. On the Automower there are several sources of RF noise, apart from the Automower itself and its built in electronics, the laptop is attached to the top of the mower, the Arduino with an IMU unit and the Emlid Reach unit are all susceptible sources of RF noise. To shield the antenna from both RF noise as well as limit the multipath errors from GPS signals bouncing from the ground, the antenna was mounted onto a 15 cm by 15 cm piece of metal to act as a ground plane, according to the Emlid Reach documentation this should also improve the signal reception [53]. To ensure better sky visibility and to further distance the antenna from the electronics attached to the Automower it was decided to elevate the ground plane and the attached antenna to approximately 1 m above ground of the rover. For the base station the antenna ground plane was mounted on top of a signpost, approximately 3 m above ground.

3.2.3 Filtering Design

Including the local frame and the global frame into the same filter would create non-linearity in the filter. Instead two linear filters was used. One filter was constructed to estimate the global angle of the robot and one filter to estimate the position of the robot. This removes the non-linearity in the position estimation filter, due to that the non-linear functions are no longer a function of an internal state but a variable. This is often preferable since it increases the modularity and debugability of the system.
3. Design and Implementation

Angle filter

The angle filter is a linear Kalman filter that estimates the angle $\theta_k$ at each instance $k$ with the corresponding vectors:

\[
\begin{align*}
    &x = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} \\
    &u = (0) \\
    &z = \begin{pmatrix} \dot{\theta}_{\text{gyro}} \\ \dot{\theta}_{\text{odom}} \\ \theta_{\text{GPS}} \end{pmatrix} \\
    &F = \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix} \\
    &H = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \\
    &B = (0)
\end{align*}
\]  \hspace{1cm} (3.1)

This follows the notation that is given in Section 2.3. The measurement vector $z$ consists of raw sensor data from the gyro, calculated angle rate from rotation of each wheel and an angle from a function which interpolates data from the GPS.

The GPS function waits until a certain distance has been travelled after a turn, and then starts calculating the angle from two GPS positions some distance apart using simple trigonometry. For every new GPS data point the angle is calculated again. The angles are then averaged by converting them to complex representation and finding the argument of the sum as Equation 3.2. The result is the true mean of the entire set of angles which is then converted back to polar representation and fed to the filter.

\[
\bar{\theta}_{\text{GPS}} = \arg\left( \sum_{j=1}^{n} e^{i*\theta_j} \right) \hspace{1cm} (3.2)
\]

This presumably makes the estimated GPS angle more accurate with time until the robot makes another turn and it is reset. This is reflected in the measurement variance matrix which is set to dynamically reduce the variance of the GPS angle input as more measurements are collected according to Equation 3.3. If no GPS data is available the $H$ matrix is adjusted to disregard the GPS angle entirely.

\[
K_{\text{GPS}} = 10(1 + \frac{10}{n_{\text{avg}}}) \hspace{1cm} (3.3)
\]
3. Design and Implementation

Where $n_{\text{avg}}$ is the number of averages. The variance of the measurement noise for the odometer and the IMU were not dynamically adjusted, instead they were fixed at 30 and 20 respectively.

Some special considerations were made for when the robot is backing, as the angle calculated from the encoder and IMU data would still contribute to showing the true angle where as the GPS would provide an angle 180 degrees off. As the backing motion only is undertaken for a short duration and for small distances, the calculated GPS angle would risk being rather inaccurate why the GPS component is instead completely ignored while backing by setting the corresponding value in the $H$ matrix to zero. As angles are a circular quantity, some care has to be taken when they wrap around. The IMU and odometer angle data are both fed into the filter as angle rates, which are not circular quantities and thus not creating any issues, the GPS data is however represented as the calculated angle which is a circular quantity. If the GPS angle is close to $\pm \pi$ some data points may wrap around, and due to the output of the filter being dependent on previous data this may result in some angle estimations being considerably off. To prevent this a function to detect GPS wrap around was constructed and the filter was made to filter out unreasonable outputs upon detecting a GPS wrap around.

Positioning filter

The position filter is a secondary filter that combines the GPS position with the odometer data. In theory the GPS data is a absolute sensor that drifts around a mean value with some added flicker and white noise. After observing the raw data from the GPS it was concluded that this was not the case. Some of the data collected seemed to be within the specification according to the Emlid reach website, but other data seemed to drift away and not ever again converge to previous data. With this, two different position filter was constructed. One that took GPS data as absolute positions and one that used relative position changes from the GPS as inputs. The former filter is designed as the following:

$$ x = \begin{pmatrix} x \\ y \end{pmatrix}, \quad u = (R) \quad z = \begin{pmatrix} x_{\text{GPS}} \\ y_{\text{GPS}} \end{pmatrix} $$

$$ F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} $$

As the RTK GPS doesn’t always provide absolute data, but can also be "off by 19cm" as described in Section 2.2 a second position filter was created for this situation. When the RTK GPS doesn’t have a fix, the relative distance between two data points might still be correct with a high precision, however they are misplaced in the global frame. A filter assuming that the GPS is absolute would drift off, but if instead choosing to look at the measured speed calculated from the GPS data and feed this into a position filter together with the odometer data a correct estimation of the position could still be made. Assuming that the given initial position was correct. The relative filter was constructed as follows:
3. Design and Implementation

\[ x = \begin{pmatrix} x \\ y \\ \Delta x \\ \Delta y \end{pmatrix} \quad u = (R) \quad z = \begin{pmatrix} \Delta x_{GPS} \\ \Delta y_{GPS} \end{pmatrix} \quad (3.5) \]

\[
F = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad H = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \\ 0 \end{pmatrix}
\]

Where the control variable $R$ is the distance traveled since the last sample, this is directly calculated from the ticks of the encoders. For both the angle and position filters the tuning is done by changing the co-variance matrices for the measurement and process noise that is explained in 2.3.

Based on the assumption that the wheel encoders can never have a negative error, any slippage would only contribute to making the apparent travelled distance appear longer than it should be, a weighting factor between the sensors could be implemented. This was made by dividing the relative distance according to the GPS with the relative distance recorded by the encoders as in the following equation:

\[
R_{quotient} = \frac{\sqrt{\Delta x^2_{GPS} + \Delta y^2_{GPS}}}{\Delta R_{encoder}} \quad (3.6)
\]

If the quotient is larger than one, which indicates that the GPS data might be unreliable, the GPS contribution to the update phase of the Kalman filter is weighted down as follows:

\[
weight = \frac{1}{(K \cdot R_{quotient})^2} \quad (3.7)
\]

Where $K$ is a constant to tune the weight parameter.

GPS fault detection

After evaluating the first test run it was found necessary to implement GPS fault detection conditions. These conditions applied mainly to what GPS data to be fed to the angle filter, which is especially sensitive to bad data. These checks consist of sorting out GPS points with too big or too small distance between them. The distance between points that meet this criteria are divided with the corresponding time stamp difference to check that the velocity is within bounds of the dynamics of the robot, this is thought to reduce the impact of high speed drift in the GPS. Furthermore, to identify high frequency flicker noise, the velocity between two consecutive GPS points is constantly checked as well, if this velocity is high it indicates that the data is noisy and is disregarded from the computations. A function for comparing a calculated angle with the average and ignoring it if it differs considerably was also found necessary to be implemented. Related functions for determining the robots operating mode and disregarding GPS data close to turns and while the robot was stationary were also tweaked after an assessment of the first test run.
3. Design and Implementation

**Determine Control Commands**

The Automower interface would not permit read access to the internal control commands, these are necessary for determining the current mode of the robot which is important for correctly resetting filters and making pose estimations. For example it is unwise to rely on GPS data to calculate the angle while the robot is stationary or reversing. It was found necessary to determine if the robot was stationary, reversing, making a turn or going forward. To determine if the robot is turning the orientation data from the odometer is read, a change of angle of more than 5 degrees over a one second period will report that a turn has been commanded. To make this more robust, any angular rate over or under $\pm 1 \text{ degrees/second}$ from the IMU will also register as a turn command. By looking at the raw wheel encoder data, a reverse command is defined as both wheels rotating backwards. A forward command is registered when both wheels are turning forward at a somewhat similar speed. The robot is stationary if both wheel encoders report zero ticks for a few measurements. As the data sampling frequency is rather high, all calculations are made with several data points to ensure there’s enough time to register a change. For the mapping function it was necessary to determine the direction of rotation when turning after encountering a boundary wire. This was done by noting the estimated angle when hitting the boundary and comparing this with the estimated angle after the turn.

3.2.4 Mapping

After an evaluation of the common map types presented in Section 2.4 the occupancy grid was chosen as the most suitable method of constructing the map. The robotics package of Matlab comes with a built in function for creating a simple occupancy grid map which was decided to be used. In this map the gradient of a cell is associated with its probability of being occupied, where lighter shades indicate a higher probability of the cell being clear and a darker shade indicate a higher probability of occupancy. A grid size and total map area has to be predefined in this function, why the global position data had to be shifted to a local map frame to fit within the map area which always extends from the origin. The grid size describes the size of a cell by defining how many cells there should be per meter. In the implementation the map size was decided depending on the input data and the grid size was set to 40, meaning each cell has a side of 2.5cm which was found to be sufficient. The Matlab occupancy grid is based on the binary Bayes filter for calculating the occupancy probability of a cell [54] and also includes parameters for setting properties such as a "clear cell"-threshold and a "occupied cell"-threshold, saturation limits for the probability as well as function calls to check the occupancy of a cell. The saturation limits where set to 0.01 and 0.99 in the implementation. Thresholds for when a cell is occupied or clear was left at the predefined values as they would not influence the results at this stage.

The implementation consist of feeding the position and orientation data together with the boundary wire detection sensor status to different functions. If no wire is detected a function to lower the occupancy probability is called on, this function use the position and orientation to determine what cells of the grid is under the robot and then set the probability for occupancy depending on how close to the center of the robot the cell is. Figure 3.4 shows the output of a call to the clear cell function. To ensure that the probability of a cell isn’t adjusted multiple times during the same passing due to the high
frequency position data output a function that checks if the cell has been written to lately is also implemented. If instead a wire is detected the estimated angle is registered before the robot begins to turn, once the sensor is inside the perimeter again the difference in angle is noted and depending on what direction the robot turns a function to register occupancy to the left or the right of the detection sensor is called. Figure 3.5 and 3.6 shows the output of a left and right hit respectively with the white cell showing the centre of rotation, which is the same as the middle of the rear wheel axle. Functions for occupying cells just in front or behind the robot was also made for when it would collide with something when either going forward or reversing.

**Post Processing of the Map**

Due to positioning inaccuracies, incomplete data and other errors it is unlikely that the occupancy grid map can be used for coverage path planning without some post processing. Examples of errors that may appear in the occupancy grid are gaps in the edges due to incomplete data in combination with a fine grid size, any gap with a width smaller than the width of the robot can be assumed to be occupied as well as the robot wouldn’t be able to transverse it in any case. Other examples are jagged edges due to small positioning inaccuracies or single cells inside the perimeter appearing occupied due to an inaccurate hit that only was partly cleared again. Post processing the map can speed up the convergence towards the true map and ensure that the area is "closed", which is a condition for a coverage path planner to work. It will also have to be converted to a binary occupancy map, consisting only of occupied or free cells, as a coverage path planner can’t take cell blocks of unknown occupancy into consideration when planning its path. The method of choice for doing this is to convert the map into a gray-scale image and dilating and eroding this several times with different parameters before converting the image to a binary map where a threshold shade of grey defines if a cell is occupied or free. Erosion and dilation are the two fundamental operations in morphological image processing. Gray-scale dilation operation takes two inputs, the image to be dilated and a so called structuring element. The structuring element is a small set of coordinate points that creates a shape with a certain size. This structuring element is superimposed on the original image and the pixel corresponding to the centre point will then be adjusted to the
3. Design and Implementation

lightest shade of gray within the superimposed shape. The structuring element is then moved to a new centre point on the original image, corresponding to the next pixel and the process is repeated, this is iterated until every pixel has been diluted and a complete new image created. Gray-scale eroding is based on the same principle but instead of adjusting the centre point pixel to the lightest shade of grey, it is instead adjusted to the darkest shade of grey found within the superimposed shape [55]. Matlab has built in functions for erosion and dilation and these were used to post process the map. A ten pixel long line is used as a structuring element to first dilate and then erode the image, this is repeated 180 times, changing the angle of the structuring element line one degree between the runs, this method intends to connect neighbouring elements and make jagged lines straight. Another dilation and subsequent erosion with a 20 pixel wide disk shaped structuring element is then applied in an attempt to remove small islands of falsely occupied cells within the perimeter. The image is then converted to a binary map, where the occupancy probability threshold is set to 0.65 and anything less is identified as a free cell.

3.2.5 Test case

The validation test is used to evaluate the performance of filtered sensor data. To test the mapping and filter functions the "planned randomness" feature from the Automower was used, this is the built in normal cutting operation of the robot that has been thoroughly tested. With the planned randomness function no control commands are needed and thus no real time computations are required. The robot bounces around the garden and all sensor data is logged into files for later processing. The sensor data consists of IMU, odometer, GPS, collision and magnetic field data. With all sensor data acquired the state of the robot and the map of the environment can be estimated.

Test Area

An area which shares some common features with a typical garden was selected and the boundary wire was laid out in the shape seen in Figure 3.7. Figure 3.8 shows an aerial photo of the test area, the garden wire is highlighted by the red lines. The area was in most part flat, but some parts had slopes, however the total elevation difference was less than 10cm for the entire area. The ground was in part covered with pine cones which were intentionally left during the test to ensure some slippage in the wheel encoders similar to what might happen in a real world environment. Two pine trees, visible just outside the marked test area in Figure 3.8, in each end of the test area obstructed the view of the sky to some degree, but was considered to be in line with what might be encountered in a common garden. The only identified anomaly to the test environment compared to a typical garden was the presence of the nearby Victoria Tower, obstructing part of the sky and presumably also reflecting GPS signals. Initial tests with the Emlid Reach suggested however that the system was working well despite this. The RTK base station antenna was attached on top of a signpost nearby, approximately 3m above ground and with a clear view of the sky above 20 degrees of the horizon, apart from the aforementioned Victoria Tower in the south direction. The base station was located at the latitude, longitude 59.40784, 17.956386 (WGS84).
Test setup 1

The Automower was configured to run its normal cutting sequence, relying on its own "random pattern" navigation system but with the cutting disc switched off in order to save battery. Piggybacked on the Automower was a laptop connected to the mower via USB to read and log the published ROS topics. Connected to the computer was an Arduino running the two IMUs which published data on its own ROS topic which was recorded by the laptop. Also attached to the lawnmower was the Emlid Reach and its antenna, but the recording of data was done on an external computer over TCP/IP. The external computer logged the GPS data as well as the time stamp from the ROS network, this in order to match the data from each sensor for post processing. An overview of the system architecture for logging data can be see in Figure 3.9.

**Figure 3.7:** Layout of the perimeter wire during test.  
**Figure 3.8:** Aerial photo of test garden.  
**Figure 3.9:** Validation test
3. Design and Implementation

Placement of the IMUs and the GPS antenna in relation to the center of rotation is important for analyzing data. The distances to the center of the rear wheel axis are presented in Figure 3.10 and a side perspective of the setup showing the height of the GPS antenna can be seen in Figure 3.11.

![Figure 3.10: Robot from side perspective.](image1)

![Figure 3.11: Robot from above.](image2)

If using the simplified 2D pose estimation this means that the sensor readings would have to be converted as follows:

\[
\begin{pmatrix}
    x \\
    y
\end{pmatrix} =
\begin{pmatrix}
    x_{GPS} - 0.10 \cdot \cos(\theta_R) - x_{ref} \\
    y_{GPS} - 0.07 \cdot \sin(\theta_R) - y_{ref}
\end{pmatrix}
\]

(3.8)

Where \( \theta_R \) is the angle of the robots local coordinate system in relation to the global coordinate system and \( x_{ref} \) and \( y_{ref} \) are reference values to make the GPS coordinates more manageable by defining a local origin by for example the base station. There is an added uncertainty in this transformation, since the variable \( \theta_R \) is calculated from other sensor readings. The IMUs were considered to have their axes orthogonal to each other, since any offset would be very small and not affect the test much. Only the acceleration in the forward direction of the lawnmower, corresponding to the x-axis, and the rotation about the robots z-axis are of interest. To compensate for the misalignment between the IMU coordinate axes and the robot coordinate axes equation 2.16 was used.

Test setup 2

A second test was performed at the same test area and with a very similar test setup. The main difference being that the GPS antenna was instead centered above the middle of the rear wheel axle, placing it in the center of rotation and thus removing the source of error that comes from estimating the robots angle in order to get the correct global x and y position coordinates. The two IMU’s were also moved, from the top to the bottom of the lawnmower, and centered on the middle of the rear wheel axle see figure 3.12. This
removes the need of using conversion matrices to correctly read the data, removing a possible source of error. The Emlid Reach was configured with a 7 minute single status averaging of the base station and the ambiguity resolver set to 'fix-and-hold' and only utilizing the GPS satellites and not the extended GLONASS ambiguity resolver.

Consecutive test setups

In the subsequent test runs the only thing modified was the configuration of the Emlid receivers. The major modifications were connecting the base unit to a DGPS over NTRIP for better base position estimation and testing different modes of the ambiguity resolver. Some of these runs will not be presented due to configuration errors, but in the run from here on referred to as Test Run 3 the ambiguity resolver was set to continuous mode with GLONASS AR off and DGPS NTRIP switched on and in Test Run 4 the GLONASS AR was switched on and the ambiguity resolver set to fix-and-hold.
3. Design and Implementation
In total four different test runs were conducted with the setup described in the former chapter. The weather conditions during the tests were clear to partly cloudy, suitable for optimal GPS reception. There were a slight change in positioning of the inertial measurement sensor after the first test run. In figure 4.1 it can be seen that the raw GPS data that was collected from the test runs. The test runs are illustrated with different colours. The robot moved around within the perimeter at all times in the "planned randomness" mode.

Figure 4.1: Raw GPS data from four different test runs. Units are global x and y coordinats in meters.

Following is sensor fusion and mapping the perimeter from one of the test runs, coloured
pink in figure 4.1. This test run had approximately the first 45 minutes of coherent GPS data and later exhibit drift and flicker noise like the other runs. In figure 4.2 only the first 45 minutes of data is fused and used to map the boundary of the garden and in figure 4.3 the complete test run is filtered to map the garden. The Automower base station is placed in the rightmost corner of the area, why no "hits" are registered there. When the GPS data is coherent it can be used as an absolute sensor, whilst must be used as a dead reckoning sensor when flicker noise exists. The same filter is used in figure 4.4 on another test run with a majority of "float" data points. Test run 4 held a GPS fix for a long time but unfortunately a software bug in the communication protocol made the time stamp connecting GPS data with other sensor data unavailable, why no sensor fusion can be shown. The graph shows however the GPS drifting, although in this test run NTRIP corrections were switched on.

4.2 (a) Schematic of boundary wire layout
4.2 (b) raw GPS data
4.2 (c) Occupancy grid map
4.2 (d) Binary occupancy grid map

**Figure 4.2:** Test run 1. Section with correct "fix" data points.
4.3 (a) Raw GPS data, green is status 'fix' and black is GPS status 'float'.

4.3 (b) Filtered signal, grey is absolute filter, black is relative.

4.3 (c) Occupancy grid map

4.3 (c) Map after post-processing

**Figure 4.3:** Complete Test Run 1
4. Results

4.4 (a) GPS data and estimated position

4.4 (b) Occupancy grid map

4.4 (c) Map after post-processing

Figure 4.4: Results of Test run 2
4. Results

4.5 (a) GPS data and estimated position

4.5 (b) Occupancy grid map

4.5 (c) Map after post-processing

Figure 4.5: Results of Test run 3
4. Results
Discussion

Four test runs were made at different times, these can be seen in figure 4.1. None of the runs gave robust sensor data from the Emlid Reach module, even though the circumstances were far better than would be considered average, with little cloud-coverage. The GPS data seem to exhibit large drift and flicker noise at certain points during the test runs.

In figure 4.2 a part of one of the test run is filtered with coherent GPS data and 4.3 is the corresponding complete test run. The filters were tuned for GPS fault detection with some improvements to the raw GPS data. The same filters were used for the test run shown in figure 4.4. This test run had a majority of incorrect GPS data points, setting high demands on the angle filter. The relative filter was running for 88 minutes with no absolute GPS data. The estimated position did not drift away. This indicates that this filter configuration would not require a high frequency absolute GPS input for a good position estimate.

In the following chapter some of the issues encountered will be discussed. This involves the performance of the RTK GPS module, evaluation and possible improvements of the angle filter, position filter and mapping algorithms used.
5. Discussion

RTK GPS - Emlid Reach

A basic assumption during the whole project was that the GPS would exhibit a random error with very little to no bias and could be seen as an absolute sensor with a mean value corresponding to the true position of the sensor. The current state of this module seems not to agree with this assumption. By observing the data from figure 4.2 and 4.3 large differences of the map are visible, this due to the incorrect GPS data points. The data suggest that reaching a very high precision with the GPS module is indeed possible, as long as the GPS reports a consequent 'fix'. However, although the robot never moved outside the perimeter, positions marked as fix are registered far outside of it, suggesting that the ambiguity resolver in the RTK GPS is not trustworthy. A basic assumption for this project was that the RTK GPS could be considered an absolute sensor with some noise around a true position when reporting a fix, otherwise reporting a 'float', something the results seem to suggest is not the case. This is demonstrated in figure 4.3a, where 'fixed' data points are illustrated in green and 'float' in black.

The documentation of the Emlid Reach unit states that the ambiguity resolver 'fix-and-hold', can under some circumstances report a false fix as it applies 'an inertia' to the fix position which may take time to recover if an incorrect fix position is calculated to begin with. Changing this mode to 'continuous' exhibited a similar behaviour suggesting that the Emlid Reach is not operating according to specifications. After consulting the Emlid Reach forum it was proposed that the error comes from an ambiguity resolver with too generous conditions for reporting a fix point [56]. The necessary modifications are not readily available in the Emlid Reach interface but by downloading the raw data and post processing it using the RTKLIB software, which the Reach is based on, it should be possible to put more constrains on the resolver. Due to limited time this was never attempted, instead GPS fault detection checks were made in MATLAB.

Even without these issues, the equipment was found to be sensitive to the operating environment, requiring tuning of masking variables and antenna placement to achieve a somewhat consistent result. The units were only evaluated under similar conditions, but it is likely that the settings would have to be reviewed when changing to a new operating environment. Choosing the correct settings is crucial to the performance and requires some prior knowledge about RTK GPS systems.

Position filter

It was suggested that although the GPS position was sometimes incorrect it could still be used. The relative distance between two measurements could still be correct, even if they are misplaced in relation to the global frame. In this case a relative position filter was used instead of the absolute filter.

Since neither the odometer data, nor the RTK GPS in certain modes, are random Gaussian distributions some tweaks could be made to improve the performance. The main source of error in the odometer data is slippage, this adds an error which is unbounded upward but can not be negative. The reported travelled distance is therefore likely the
maximum possible true distance, this was used to evaluate the GPS data before feeding it to the filter. This makes it necessary to average GPS and odometer data as just comparing distances between two GPS and two odometer points might contain noise and simply removing the points would exclude half of the Gaussian distribution. Further tweaking of the number of averages, the distance between two measurement points and the weight factors associated to each input might improve the performance. An evaluation and comparison between the two position filters is hard to do since the relative position filter is heavily dependent on the angle filter where as the absolute filter operates more independently.

In Figure 4.3b it can be seen that the absolute position filter appears to be working very well, much thanks to very accurate input data. The relative position filter is however harder to evaluate. It appears to estimate the total travelled distance correctly, but due to bad angle data, what should be straight lines are sometimes a little bit crooked, making the total distance too short. The angle errors emerge due to incorrect sensor data or wheel slippage that was discussed in section 2.2. A function for dynamically correcting past estimated positions as better angle data becomes available might be a possible method for solving that issue and reducing position drift in general, this would however be something done outside of the Kalman filter.

**Angle filter**

The angle filter is used to determine the orientation of the robot and is especially important when no absolute position data is available. The filter relies on the measured angular rate from the IMU, angular data from the odometry recalculated to an angular rate and the GPS feeding it an absolute angle from measuring the angle between data points. IMU and odometry data are available at all times, making this part of the filter easy to implement. The GPS can however only calculate an angle correctly if the robot have travelled some distance forward, making it important to determine when it’s suitable to pass this data to the filter. The calculated angle from the odometry and the IMU was found to correlate well as can be seen in Figure 5.1, even seem to detect very small signals. The plot is wrapped around $[-\pi : \pi]$ and shows the integrated angle rate from IMU and odometry data.

In figure 5.2 an example of correct behavior during three turns is shown. The figure shows the angle in radian for odometry, IMU, GPS and output of the filter. Only the GPS data effect the absolute angle. The odometry and IMU are passed in as a dead reckoning sensor and effect the output of the filter in a relative correlation. The GPS angle is reported at all of the straight travels and can be seen as the true angle of the robot. The odometry and IMU data seems to approximate the angle during rotation well, except the third turn that has a slight angle error. This figure suggest that the angle filter can work as the theory predicts. With a correct IMU calibration and GPS fault detection good angle estimation can be achieved.
Figure 5.1: IMU and odometry correlation.

Figure 5.2: Example of a working angle filter
Figure 5.3: Example of GPS angle correction

Figure 5.3 shows the performance with the relative position filter when the GPS is turned on and GPS turned off. The data is taken during correct 'fix' GPS data points and the 'true path' line shows the output of the absolute position filter. 'GPS off' shows the path with the angle filter only using odometry and IMU with the GPS switched off, and in line 'GPS on' the GPS angle corrections are passed to the filter, which represents the complete angle filter. In this part of the test run significant errors are present in the IMU and odometry data, making some of the turns appear too small which quickly results in a large position error. In line 'GPS on' this drift is reset with the GPS once it becomes available, as can be seen in the small angle corrections in the beginning of the previously straight lines, greatly improving the results. The overall performance of the angle filter is relatively good, but even a single bad estimation will greatly affect the subsequent position estimations, and are irrecoverable without reliable absolute position data. Due to a heavy weight on GPS angle data, passing an incorrect GPS angle has a large negative impact on the results. The problem arise when the GPS is drifting, better GPS fault detection and prefiltering would be necessary as the RTK GPS errors are not a random distribution but often appear to be correct and the drift is only visible over longer distances, spanning hundreds of data points. An example of this behaviour can be seen in Figure 5.4. The GPS raw data and the corresponding filter output are marked in red, the GPS fault detection detects some of the drift, but when the drift becomes smaller it is easily confused with a true movement and gets passed on to the filters by the fault detection checks. It is possible to write stricter conditions for passing this type of data, but this also means that more correct data will be filtered out as well. The choice comes down to having strict conditions and rarely receive an absolute angle correction, or to have less strict conditions and receive it more often, but run a higher risk of passing less accurate angle estimations. What works best is highly dependent on the input data, and a general solution is hard to find. A more accurate gyroscope and a second source of
5. Discussion

absolute data, for example by incorporating a compass, could be beneficial for preventing drift in between GPS readings, and could also improve the GPS fault detection. Having trustworthy fixes from the GPS at certain intervals would also help recover angle errors. It is worth noting that the error seen in Figure 5.4 only occurred after over an hour of receiving poor GPS data, suggesting that an absolute position fix, if accurate, doesn’t have to be available very often.

![Figure 5.4: Test run 2 with problematic GPS data marked in red](image)

**Figure 5.4:** Test run 2 with problematic GPS data marked in red

**Unknown Control Commands**

A major difficulty with letting the robot run the default cutting program during the test runs is that the velocity and turn commands are not available as the ROS topic relating to them doesn’t publish anything. It is only possible to send control commands to the robot, not read what control commands it sends internally. In order to correctly estimate angles and distances travelled the filters are heavily dependent on knowing when a turn is commanded or when the robot is stationary or reversing. Since this information was not readily available the status of the robot had to be determined by sampling sensor data instead. This was done by simple if-statements in conjunction with looking at for example odometry data to determine if the wheels are reversing, determine if a turn is commanded by noting that the wheels are rotating in different directions or by noting that IMU registers a high angle acceleration. These if-statements are loosely written, as it was found that even a single missed turn command had far more severe consequences than several false positives. The result is that the robot registers false commands which resets the filters during normal operation which has a negative impact on the results. Furthermore, there’s a slight time delay in registering a change in command mode when a change actually does happen, which further affects the results negatively. Having access to the real command data would likely significantly improve the ability to determine the position of the robot correctly.
5. Discussion

Mapping algorithms

The functions for creating the occupancy maps displayed a good performance if the input data was correct to a high precision as can be seen in Figure 4.2(d), but the map quickly deteriorates if the position estimation is incorrect. This is due to the same probability weight being given to a "hit" or a "clear" call regardless of how accurate the position is. As the data sets are rather short, quite a heavy weight was given to the probabilities already in the first passing, this means that any map will quickly be destroyed if the position estimate start drifting and incorrectly reporting clear areas where it is occupied and vice versa, visible in Figure 4.4(c). In a real world application, where a lawnmower will be in the same area for a long period of time, thus making it able to collect much larger sets of data, the need for this heavy weighting is no longer necessary. Assuming that the position estimate will be correct more often than not, which is also suggested by the results, a correctly implemented map based on the same principles would converge towards the true map with time.

When a hit was detected it was necessary to estimate the orientation of the robot in relationship to the perimeter wire. This to occupy the correct cells. It is clear that the Automower was not designed to do this, but with more boundary wire detection sensors this would be more accurate and easier to implement.
5. Discussion
6

Conclusions

The results indicate that the Emlid Reach RTK GPS cannot be relied on as an absolute position sensor. The installation of a RTK GPS system requires knowledge of antenna placement and tweaking of variables to suit the local environment to achieve good performance, making it unsuitable for a plug-and-play consumer device such as the Automower. The results show that under some circumstances, in conjunction with other sensors and with robust fault detection, it is possible to achieve acceptable results, even when a GPS "fix" is unavailable for an extended period of time. This indicates that with further research in GPS fault detection and an improved filter design it might be possible to use the RTK GPS for mapping and coverage path planning, but a lot of effort would have to be made to make it work in a more general case. The current state of consumer grade RTK GPS is unsatisfactory for direct implementation for positioning and mapping in the Husqvarna Automower.
6. Conclusions
7

Recommendations and Future Work

7.1 Control

The vision of this thesis is to apply trajectory control. This requires a good estimate in both pose of the robot and map of the environment. Only the premise for trajectory control is evaluated in this report and no test with active control was conducted. Some efforts were made to control the robot from an external computer. This was not continued due to the large time delay from command to actuation. The two control algorithms mentioned in chapter 2, subsection 2.1.2 and 2.1.2 are recommended to be implemented into the firmware. The control functions are needed to be implemented as close to the hardware as possible to quickly send new control commands when detecting dynamic objects or the boundary wire. With these two functions the simplest trajectory controller can be constructed, the Spiral and U-turn coverage path planner that are depicted in Figure 2.6 and 2.7, these coverage paths can then be used to cover much more complex geometries by the Exact Cellular Decomposition method as discussed in Section 2.5. Having active control may significantly reduce the time it would take to map a garden by having the robot follow the boundary wire around the garden and continuously register hits along the entire perimeter.

For full-coverage path planning some computations must be made in real time. As mentioned in section 1.3 this was excluded for the thesis and data was only post processed. It is likely that computational time of the position and mapping algorithms might be a limiting factor if implemented on the current hardware on the Automower platform. Since the code is not computational time optimized there are probably large room for improvements and most likely necessary to reach the final goal of full-coverage path planner. This involves to reduce complexity of GPS fault detection and log past positions collisions for post processed map generation.

7.2 Better GPS fault detection

The Emlid Reach reported false "fix" status during all of the tests. An example of this was shown in figure 4.3 (a). The settings for status reports in the Emlid reach can be set through a set of user-configurable parameters in the RTKLIB software. The RTKLIB software is "an open source program package for standard and precise positioning with GNSS" [57]. The Emlid Reach could possibly be configured in such a way to reduce the number of false fixes. This would likely also increase the number of float data points where the ambiguity-resolved RTK solution is not obtained. There is a counterbalance between
the false positives and false negatives of the system. To tune Emlid reach to avoid all false positive fix reports while having enough true fix reports to map the environment might be difficult. The RTKLIB configuration was not altered from the default setting, therefor it is suggested that future work would include this. The data was in many ways be improved with simple tricks and filter techniques that was implemented and can certainly be improved in multiple ways. The RTKLIB has most likely more sophisticated methods to detect false GPS reports and it is recommended to explore this.

7.3 Mapping

In its current form, the map is based on position data but the selected probability of occupancy of a cell is not tied to the variance in the estimated position. A future implementation should take this into consideration and weight position data with a high accuracy heavier when constructing the map. This could be done by looking at the GPS status and check the time since last reported fix and also by looking at the estimated GPS covariance. Estimating the precision and accuracy in the position reported from the Kalman filters is possible, but complex due to them being separated into two filters, but by looking at the time elapsed since the last GPS fix and the variance between the sensors a rough estimation should be possible to calculate. Furthermore, by incorporating more magnetic sensors for detection of the boundary wire, the angle between the robot and the boundary wire could be more precisely determined. Having this angle would be an aid when selecting what cells to occupy when registering a hit. The weight associated with a 'hit' and a 'clear' should also be modified in any future implementation as it is much too aggressive at the moment due to the relatively short test runs.

As mentioned in Section 7.1 actively following the boundary might improve the mapping, especially in terms of time. Utilizing the fact that the base is a known landmark could be used in conjunction with this to determine when to "close the loop", further improving mapping results. Active control would also allow for scheduling more frequent visits to the base station when the GPS fix has been unavailable for some time, which could serve as an aid to reset drift over time. This has not been investigated in this report due to lack of control implementation, but would be the logical next step in any future work.

7.4 Other absolute position systems

The literature review that was made during the thesis showed that the new RTK GPS module Emlid Reach was a good choice as an absolute position sensor. Other systems could however give similar precision and accuracy. Systems that were of interest was radio beacon systems and visual landmark based positioning. The Emlid Reach module used had a better theoretical accuracy than was deemed necessary for path planning features but exhibited robustness problems. A GNSS module will always be affected by environmental constraints in some extent. Further investigation in other absolute positioning systems is recommend, to increase overall system robustness.
Bibliography

Bibliography


Appendix 1

A.1 Code

The code can be found on the github webpage. Clone the repository and run 'Main.m' with the whole repository added in the matlab.

https://github.com/nakceb/Master_Thesis.git