Localization Using Laser Scanning and Minimalistic Environmental Models

Patric Jensfelt

TRITA–S3–REG–9902
ISSN 0347–1071

AUTOMATIC CONTROL
DEPARTMENT OF SIGNALS, SENSORS AND SYSTEMS
ROYAL INSTITUTE OF TECHNOLOGY
STOCKHOLM, SWEDEN

Submitted to the School of Electrical Engineering, Royal Institute of Technology, in partial fulfillment of the requirements for the degree of Technical Licentiate.
Abstract

This thesis deals with the problem of localization for a mobile robot in an indoor, structured and non-engineered environment. By localization we mean the process of determining the position and orientation of the robot from sensor measurements of the environment. We will experimentally show that minimalistic environmental models can be used to solve the localization problem, contrary to many other approaches where a higher degree of detail is used.

The thesis begins with a brief overview of the most common sensors used for localization, including odometry, sonar, laser scanner, infrared, radar, inertial, compass and vision. We argue for the use of a laser scanner, because of its high sampling rate and excellent angular resolution. We also discuss existing localization methods and divide them into feature-based and grid-based methods. We conclude that feature-based methods are best suited for combining with minimalistic models of the environment.

The first step of the localization method developed in this thesis is the derivation of models for the sensors used. The sensors used are odometry, which approximately measures the motion of the robot by counting the wheel rotations, and laser scanning. The odometry gives excellent accuracy for short distances traveled but is inherently susceptible to drift as a result of wheel slippage and modeling errors. On the other hand, the laser scanner provides lower accuracy but measures the absolute position of the robot relative to the environment.

Using the sensor models and a minimalistic model of the environment we develop a method for tracking the position of the robot. We also present an algorithm for initializing the position of the robot. Both algorithms are implemented on a Nomad200 robot using a PLS laser scanner from SICK. Experimental results are presented that demonstrate robustness of the methods and show that the position can be determined with an accuracy of 20 mm, depending on the quality of the model.
Acknowledgments

This work has been sponsored by the Swedish Foundation for Strategic Research under the Centre for Autonomous Systems contract.

First of all I would like to thank my supervisor Professor Bo Wahlberg for giving me the opportunity to continue my studies within the Centre for Autonomous Systems (CAS) here at the Royal Institute of Technology, and for continuously giving support when needed.

Thanks also to my co-supervisor Professor Henrik Christensen for always taking the time to discuss problems and being a constant source of new ideas. His vast knowledge in the field of robotics provides excellent guidance when diving into unfamiliar territory.

Olle Wijk also deserves thanks for being a great college and friend and for the nice work we have done together and will do in the future. In this context I would also like to mention Anders Orebäck and Mattias Lindström. The four of us have spent a many long nights to get the system running before various demos and deadlines. The latter two also deserve thanks for providing the Intelligent Service Robot with the good architecture we all work with.

For managing the Intelligent Service Robot (ISR) project we owe thanks to Magnus Andersson. Making us students pull in the same direction has not been easy. Thanks also to all the other members of the group.

I am forever in debt to the newest member of CAS, David Austin, for proof-reading this thesis on very short notice, providing invaluable feedback.

Thanks also to Steen Kristensen at DaimlerChrysler for being an excellent supervisor during my stay in Berlin and to Volker Hansen for proving the opportunity of the visit. The other members of the ASR team and Ola Nockhammar will not be forgotten either for taking such good care of me.

Last, but definitely not least, I would like to thank Dani for her endless support, love and understanding. Ovo je za tebe, volim te.
Nomenclature

Throughout this thesis certain expressions will be used which may not be familiar to some of the readers, or might have another meaning in some other field. A short description of the most frequently used expressions and words will follow:

- **Agent**
  The term should be interpreted as a mobile robot.

- **Odometry**
  By using wheel encoders on a wheel based platform, the change in position and orientation of the platform can be calculated. This constitutes the odometric system.

- **Platform**
  By platform we refer to mobile robot platform.

- **Pose**
  Pose is a collective term for information about both position and orientation.

- **Pose Initialization**
  The process of finding the pose of the robot at startup for example.

- **Pose Tracking**
  Given that the initial pose of the mobile platform is known, it is possible to, using various sensors, keep track of the pose as the platform moves.

- **Robot**
  By robot we mean mobile robot.

- **Sonar**
  Short for ultrasonic sensor.
Notation

In this thesis the following notation will be used:

**True Pose Information**

\( \theta \) : Orientation of the robot in a global coordinate system.
\( \Delta \theta \) : Change in orientation of the platform.
\( \Delta x \) : Change in x-coordinate in a global frame of reference.
\( \Delta y \) : Change in y-coordinate in a global frame of reference.
\( D \) : The distance traveled.
\( x \) : x coordinate in global reference frame.
\( y \) : y coordinate in global reference frame.
\( x \) : Three-tuple that specifies the pose of the robot, i.e. \( x = (x, y, \theta)^T \).

**Entities Measured by odometry**

\( \Delta x_{od} \) : Change in x-coordinate according to odometry.
\( \Delta y_{od} \) : Change in y-coordinate according to odometry.
\( D_{od} \) : The distance traveled by the robot according to odometry.
\( \Delta \phi \) : Change in orientation of the wheels.

**Odometric Model Parameters**

\( C_D \) : Scaling factor for distance traveled (wheel radius).
\( \sigma_{C_D} \) : Standard deviation for \( C_D \).
\( C_{\theta,D} \) : Constant that relates distance traveled to platform orientation.
\[ \sigma_{C_{\xi,D}} : \text{Standard deviation for } C_{\xi,D}. \]
\[ C_{\xi,\varphi} : \text{Constant that relates steering to platform orientation.} \]
\[ \sigma_{C_{\xi,\varphi}} : \text{Standard deviation for } C_{\xi,\varphi}. \]

**Pose Estimation**

\[ \hat{D} : \text{The estimated distance traveled, } \hat{D} = C_{S}D_{od}. \]
\[ \hat{\Delta}\theta_{\varphi} : \text{Estimated change in orientation caused by turning the wheels.} \]
\[ \hat{\Delta}\theta_{D} : \text{Estimated change in orientation caused by translation.} \]
\[ \hat{\Delta}\theta : \text{The estimated total change in orientation.} \]
\[ \hat{\Delta}x : \text{Estimated change in } x\text{-coordinate in a global frame.} \]
\[ \hat{\Delta}y : \text{Estimated change in } y\text{-coordinate in a global frame.} \]
\[ \hat{x} : \text{Estimated } x\text{ coordinate in global frame.} \]
\[ \hat{y} : \text{Estimated } y\text{ coordinate in global frame.} \]
\[ \hat{\theta} : \text{Estimated orientation relative to a global frame.} \]
\[ \hat{x} : \text{Estimated pose of the robot } \hat{x} = (\hat{x}, \hat{y}, \hat{\theta})^T. \]

**Uncertainties**

\[ \sigma_{x} : \text{The standard deviation in the estimate of the } x\text{ coordinate.} \]
\[ \sigma_{y} : \text{The standard deviation in the estimate of the } y\text{ coordinate.} \]
\[ \sigma_{\theta} : \text{The standard deviation in the estimate of the orientation, } \theta. \]
\[ \sigma_{xy}^2 : \text{The covariance in the estimate of the robot’s } x\text{ and } y\text{ coordinate.} \]
\[ \sigma_{x\theta}^2 : \text{The covariance in the estimate of the } x\text{ and } \theta. \]
\[ \sigma_{y\theta}^2 : \text{The covariance in the estimate of the } y\text{ and } \theta. \]
\[ \sigma_{D}^2 : \text{The standard deviation of the estimate of the distance traveled.} \]
\[ \sigma_{\Delta\theta}^2 : \text{The standard deviation of the estimate of the change in } \theta. \]
\[ \sigma_{\Delta x}^2 : \text{The standard deviation of the estimate of the change in } x. \]
\[ \sigma_{\Delta y}^2 : \text{The standard deviation of the estimate of the change in } y. \]

\[ P : \text{Covariance matrix for pose estimate, } P = \begin{pmatrix} \sigma_{x}^2 & \sigma_{xy}^2 & \sigma_{x\theta}^2 \\ \sigma_{xy}^2 & \sigma_{y}^2 & \sigma_{y\theta}^2 \\ \sigma_{x\theta}^2 & \sigma_{y\theta}^2 & \sigma_{\theta}^2 \end{pmatrix}. \]

\[ Q : \text{Covariance matrix for change in pose based on odometry.} \]
\[ R : \text{Covariance matrix for measurements used to determine pose.} \]
Contents

1 Introduction .......................... 1
   1.1 Background ........................ 1
   1.2 The Localization Problem .......... 4
   1.3 Idea and Requirements .............. 6
   1.4 Publications ........................ 6
   1.5 Contributions and Thesis Outline ... 7

2 Sensors for Localization ............... 9
   2.1 Odometry ........................... 10
      2.1.1 Odometric Models ................ 11
      2.1.2 Problems ........................ 13
   2.2 Sonar ............................... 14
      2.2.1 Basic Ultrasonic Physics ....... 14
      2.2.2 Sonar Ranging Principles .......... 15
      2.2.3 Drawbacks with the Sonar Sensor .. 17
      2.2.4 Sonar Sensor Models .............. 18
      2.2.5 Sonar Array ........................ 20
   2.3 Laser ............................... 20
      2.3.1 History and Technology .......... 21
      2.3.2 Laser Range Finders ............... 22
      2.3.3 Scanning Sensors .................. 23
      2.3.4 3D Laser Imaging Sensors ......... 23
      2.3.5 Properties of the Laser Ranger Finder . 24
      2.3.6 Advantages of 2D Laser Scanners ... 25
      2.3.7 Drawbacks of 2D Laser Scanners ... 26
   2.4 Infrared ........................... 26
   2.5 Radar ............................... 27
   2.6 Inertial ............................. 28
2.6.1 Accelerometers ........................................ 28
2.6.2 Gyroscopes .............................................. 29
2.6.3 Inclinometers ........................................... 29
2.7 Compass ....................................................... 30
2.8 Vision ........................................................ 30
2.9 Structured Light ............................................. 31
2.10 Summary ..................................................... 31

3 Methods for Localization .................................. 35
  3.1 Localization Methods Requiring Engineered Environments .................. 37
  3.2 Localization Methods for Non-Engineered Environments ................. 38
    3.2.1 Parametric Methods .................................. 38
    3.2.2 Grid Methods .......................................... 48
  3.3 Summary ..................................................... 53

4 Characterization of the Odometry and the Laser Scanner .................. 57
  4.1 Characteristics of the Odometry on the Nomad200 Platform ............. 57
    4.1.1 University of Michigan Benchmark Test (UMBmark) ............... 60
    4.1.2 Rotation Error Caused by Steering .......................... 61
    4.1.3 Rotation Error Caused by Translation ........................ 63
    4.1.4 Scaling .................................................. 65
    4.1.5 Odometric Model ........................................ 66
  4.2 Characteristics of the PLS Laser Scanner from SICK .................... 67
    4.2.1 Quantization ............................................ 68
    4.2.2 Footprint Size ......................................... 70
    4.2.3 Beam Size .............................................. 72
    4.2.4 Sensor Model ........................................... 72
  4.3 Summary ..................................................... 73

5 Pose Tracking .................................................. 75
  5.1 Theory ...................................................... 76
  5.2 Algorithm .................................................. 77
    5.2.1 The State of the Robot .................................. 79
    5.2.2 Validation Gates ....................................... 79
    5.2.3 Line Extraction ......................................... 80
    5.2.4 Measurements .......................................... 84
    5.2.5 Measurement Based State Update .......................... 85
  5.3 Implementation ............................................ 87
    5.3.1 Wall Extraction ......................................... 87
    5.3.2 Managing the Size of the Uncertainty ....................... 89
5.4 Experiments .................................................. 90
  5.4.1 Experiment 1 ........................................... 90
  5.4.2 Experiment 2 ........................................... 92
  5.4.3 Experiment 3 ........................................... 94
  5.4.4 Experiment 4 ........................................... 97
5.5 Summary ..................................................... 98

6 Pose Initialization ............................................ 101
  6.1 Theory ...................................................... 102
  6.2 Algorithm .................................................. 102
    6.2.1 Finding the Main Orientation of the Walls .......... 103
    6.2.2 Generation of Possible Poses ................. 105
    6.2.3 Finding the Pose of the Robot .............. 107
  6.3 Implementation .......................................... 108
    6.3.1 Histograms ........................................ 108
    6.3.2 Wall Candidates .................................... 108
    6.3.3 Finding the Pose of the Robot ............. 109
  6.4 Experiments ............................................... 109
  6.5 Summary .................................................... 110

7 Conclusions and Directions for Future Work ............ 111
  7.1 Conclusions ............................................... 111
  7.2 Future Work ............................................... 113

A The Range Weighted Hough Transform ..................... 115
  A.1 Hough Transform ........................................ 115
  A.2 The Range Weighted Hough Transform ................ 117
Chapter 1

Introduction

1.1 Background

During the 20th century man has invented many new technical facilities, the aircraft, the computer, etc. One facility which has always been part of the vision about what the future has in store for us, is a mobile robot which can help us perform dull or dangerous tasks. We are not there yet, but we are struggling to make the vision come true, to provide people with a helping hand where it is needed. There are innumerable applications which one can think of for a mobile robot. Some of the most apparent applications include, for example household work, construction work and cleaning up in hazardous environments.

Robots are already a key component in the car industry for spot welding. The main difference between this kind of robot and a robot which would be able to come into our houses and do work, is the level of autonomy and generality. A spot welding robot is preprogrammed and the only thing it is good for is spot welding. To be able to move freely in a home, a robot must have a higher level of autonomy and capability to solve unforeseen situations. It might be possible to get some kind of map of the building where the robot is operating, but it is unrealistic to think that everything is known down to the smallest detail. The environments in our houses are dynamic, things change place and people are moving around. This means that we need to use information about the environment which in turn calls for sensors.

There are many challenging problems which have to be solved before we can have a fully operational agent cleaning our houses. Many of the
problems are related to capabilities that we humans take for granted. It is easy to understand that we must teach the robot to locate and iron our shirts or to pick ingredients from the refrigerator to make a lasagne. However, there are far more fundamental problems that has to be solved before we come to that stage.

- **Navigation**
  Navigation deals with the problem of how to get from point A to point B. In order to move around in the house we need to be able to avoid running into things. This is a problem often referred to as obstacle detection/avoidance. Much research has been devoted to this problem, but there exists no solution which will solve the problem in all cases, as all sensors have limited regions of operation. We humans do not have a perfect system either, so striving to build a system which never runs into things might be taking it too far. In a broader scope this might be referred to as the problem of navigation, that is how to get from one place to another while not colliding with anything.

- **Localization**
  The robot also needs information about its position in the world. One can think about different ways to express this information. It could be in relation to some global coordinate system, but it could also be relative to some object. A combination is likely to be needed as every physical contact requires the robot to position itself relative to the object, whereas the robot will need its global position when reasoning about how to go from one place to another.

  A problem that is sometimes difficult for a human being as well, is the problem of finding the position when there is no information about the history of movements. That is, there is no information about how the present position was achieved. This is the problem of initializing the position of the robot. Initializing the position is more difficult than keeping track of the position when the initial position is known. Traditionally, most robot systems have only shown position tracking capabilities and have relied on manual initialization. This is not adequate if full autonomy is one of the goals.

  In order to do anything meaningful a model of the world is needed. This model, or map, can be of many different types, e.g. grid maps and feature based maps (see Chapter 3). The way the map is acquired also varies from system to system, but for a fully autonomous
system the robot must acquire the map on its own. Hence an unknown environment must be explored and a map of it built. Finding the pose of the robot and building a map are two problems that are the cause of a problem to a kind of “chicken and egg problem”, that is for localization a environmental model is required and vice versa. We refer to the next section for a more detailed discussion. Localization is closely related to the navigation problem, but we separately identify them to make it clear which problem we are trying to solve.

- **Manipulation**
  When the robot wants to interact with the environment it has to be able to perform tasks like picking up an object, opening a door, etc. This requires tight loop feedback from tactile sensors, vision and force-torque sensors, not to mention that it requires a manipulator which will play the part of the human arm. The problem can be made even more difficult by mounting the manipulator on a mobile platform and do manipulation while the platform is moving. This is difficult for a human too, something that everyone who has carried a full cup of coffee can confirm.

- **Recognition**
  In order to know what to manipulate, objects have to be recognized. In the past this problem has often been solved by limiting the set of objects to contain only objects which can be unambiguously identified by simple features, such as color or shape. In a home environment this is not feasible, so the robot has to rely on the same kind of information that we do. Shape and color of the object are likely to still be important descriptors. Object recognition has applications in many other areas than mobile robotics, e.g. surveillance and production industry.

- **Planning**
  For a semi-autonomous system the user can be expected to make some decisions regarding the operation of the robot. To achieve full autonomy it is however necessary that this capability be taken care of by the robot itself. This means that the robot must be able to reason online about high level thing like when and how to perform a certain task.
• **Learning/Adaptation**
  As mentioned above the robot must be able to acquire a model of
  the world on its own, this is one form of learning, but there are
  many more potential applications for learning. Without learning
  the user/programmer would have to prepare the robot for every
  possible situation, which would make the system highly inflexible.
  As in the case of animals, learning is an ingredients for the more
  advanced life forms. Therefore learning in full scale cannot be ex-
  pected to be part of an initial robot system.

  The six key problems mentioned above can be studied individually, but
  it is not until all of them are provided with good solutions that one can
  hope to build the foundation on top of which a household robot could
  rest.

### 1.2 The Localization Problem

In this thesis we will deal with the problem of localization. Being one
of the fundamental problems of mobile robotics, it is not surprising that
there is a great depth of literature concerned with localization. The next
two chapters will survey the available sensors and some of the techniques
for world modeling and localization, to serve as motivation for decisions
taken later when a laser based localization method is presented.

Before proceeding any further, it is useful to refine our definition
of localization. Localization is the process of finding the position and
orientation of the robot. The position and orientation, in combination,
are called the pose. Knowing the correct pose of the robot is a condition
for the successful completion of many of the tasks which a robot might
be required to perform.

The problem can be divided into three subproblems:

**Initialization** When the power to the robot is turned on, the robot
has to determine its position without any knowledge about the history
of motion. The robot has to collect evidence through sensor feedback,
and combine it with information about the properties of the environment
(the map) to find the position. The level of complexity in this task varies
with the size of the environment, but also with the level of homogeneity
of the environment.
1.2 The Localization Problem

The initialization of the position has been called e.g. absolute localization or global localization in the literature, but we believe that pose initialization is a better term. Many operating robot systems today do not have the capability to perform pose initialization. Instead they rely on the user to provide the initialization or they use a scheme where the robot is always initialized in the same configuration.

**Maintenance** When the robot is moving whilst completing its tasks it must maintain an estimate of its pose in some fixed reference frame. The process of keeping track of the pose of the robot will be referred to as pose tracking. Whereas the demand for accuracy is not paramount for pose initialization, many applications require high accuracy from the pose tracking. On the other hand, when a previous estimate of the the pose is available, it is possible to limit the search space of possible poses to those which can be reached during the time for the last pose update. This implies that the matching problem is simpler for pose tracking than pose initialization and hence the computational requirements are lower.

**World Modeling** Some of the methods for pose tracking assumes that a model of the environment is given. Further more, knowing the position of the robot is of little importance if it cannot be related to anything but the position the robot started at. Therefore a model is needed for the localization to have a meaning in the long run. This world model is either given to the mobile robot, or it has to build it itself. In order to have a fully autonomous system, the mobile robot must have the capability to autonomously build a model of the environment which can later be used to recognize places it has already been at. This is a bit of a chicken and egg problem as a map is need in order to perform tracking and the pose is needed to build a map. The solution is to do both at the same time, i.e. simultaneously build a map and keep track of the pose using the new map. Recklen [1] points at one conflict which occurs in this process. "As the robot moves features are built up and placed into the map. If it takes too long to add a confirmed feature to the map, the danger exists that the robot’s positional uncertainty has become too large. Therefore it is important that a confirmed feature be found as soon as possible. The uncertainty threshold for deciding when a tentative feature becomes a confirmed feature is therefore very important. Choosing it too high means that uncertain features are used for the localization and so the whole process diverges. Choosing it too low means that by the time a feature is available, the robot’s uncertainty has grown beyond return.”
1.3 Idea and Requirements

Our working hypothesis is that it is possible to do position tracking using a simple model of the environment. As a first step we do not require the method to include the map building stage of localization, but thought has to be given to how the method can be augmented to incorporate this as well. The only requirement we pose on the localization system is that it must not rely on altering the environment in any way, to make the localization problem easier. This requirement rules out methods which rely on e.g. bar codes or active beacons.

1.4 Publications

Work which will only be mentioned briefly in this thesis has been presented in the following document:


The part of the thesis about the laser based localization has been published in:


- Patric Jensfelt and Henrik Christensen, “Laser Based Pose Tracking”, *IEEE Conf. on Robotics and Automation*, Detroit, USA, May, 1999

In parallel to the development of the methods reported in this thesis, research has also been conducted in cooperation with DaimlerChrysler Research and Technology leading to the submission of the paper

1.5 Contributions and Thesis Outline

The main contributions of this thesis are:

- An extensive survey over existing methods for localization and a description of the most common sensors used, with an emphasis on sonar and laser sensors.
- Characterization and understanding of the data from the SICK PLS laser scanner.
- Algorithms for laser based localization, presented with successful experimental results.
- Analysis regarding the conditions under which the laser based localization algorithm will fail.

The outline of the thesis is as follows:

Chapter 2 Sensors provide information about the environment, giving the robot something like the human senses. In order to be able to use the full potential of the sensors we need to understand how they work. Models of the sensors can be utilized to interpret the data they provide. A complete model must also include the range of applicability, i.e. we must know the limitations of the model but also the of the sensor itself. Chapter 2 contains an overview over different sensors which are used for localization, with an emphasis is on odometry, ultrasonic and laser sensors.

Chapter 3 Robotics research is incremental, no method or algorithm should be designed without considering what has already been done. Improvements are made by taking what is good from the past and combining it with new parts. Chapter 3 contains an overview over what has been done in the field of localization.

The chapter concludes with a summary of available options and some ideas which will lead to a proposition for tackling problem of localization.

Chapter 4 As was previously stated, a sensor must be thoroughly understood before its full potential can be used. Chapter 4 presents the experimental results obtained when characterizing the performance of the odometry on our experimental platform. The chapter also includes
results of experiments designed to characterize the SICK PLS laser scanning sensor.

**Chapter 6** In Chapter 6 we develop a method for initializing the pose of the robot. One limitation to the initialization process is that we assume the room that we are in to be known. Experimental results indicate that the method provide means to initialize the pose in almost all situations.

**Chapter 5** Localization can be split into three subtasks, maintenance, initialization and modeling. The order in which the tasks are solved can vary, but they are all needed in order to achieve full autonomy. In Chapter 5 we present a method for tracking the pose of the robot, i.e. maintaining the pose estimate of the robot. We here assume that the initialization and modeling problems have already been solved. Experimental results show that the algorithm is highly robust.

**Chapter 7** Chapter 7 contains conclusions regarding the proposed methods and the results and some directions for future research.
Chapter 2

Sensors for Localization

In order to fully understand the problem of localization it is very important to know the characteristics of the different sensors that are available. The most common sensors will be outlined in this chapter. Odometry and the sonar and laser sensors will be dealt with in detail, as they are the sensors which this thesis is dealing with. A detailed description of all the sensors would be too lengthy here, but there are some very good books on the subject (see for example [2, 3]).

As mentioned in Chapter 1 sensors are needed to get information about the environment. As we humans have different senses, there are sensors which measure different entities, such as color, distance, force, etc. The sensors can be grouped into internal and external sensors. The internal sensors measure some internal state, such as the orientation of a wheel axis or the direction of a camera. The external sensors measure e.g. distance to objects, position relative to a beacon. Another possible way to group the sensors would be to group them into sensors which are active and those which are passive. The active sensors probe the world by looking at the effect of some action. Here we find sensors like radar that probes the world by sending out a radar signal and looks at what comes back. The camera, used in ambient light, is on the other hand an example of a passive sensor.

Good physical understanding for the sensors, will greatly increase the ability to utilize their potentials. It is not until the physics are understood, that the full power can be exploited and maybe improvements made. From a localization point of view, it is also important to understand how the sensors work as they are the input to the algorithms, i.e. we
need good models for the sensors. In [4] Durrant-Whyte et al say: “We will maintain that the only way to understand and utilize the disparity between different sensor views is to explicitly model the sensor and the information it provides …”, where a sensor model is defined as “an abstraction of the physical sensing process whose purpose is to describe the ability of a sensor to extract descriptions of the environment in terms of the information available to the sensor itself”. Crowley [5] says that “the sensor model can be viewed as a form of logical sensor which provides the sensor information in a standard form”. Independent of the definition used for a sensor model, it is clear that good physical understanding is needed to construct such a model. One should be cautious when using a sensor model though, and remember that it is only a model. A good example of this is the odometric model. Borenstein & Feng point out that the odometric information is very good most of the time [6]. That is, trusting in an odometric model is warranted almost all the time. However, non-systematic errors, i.e. error that cannot be captured by the model, degrade the value of the information.

The method for doing localization will depend on the sensor which is used. When using a camera it is of little or no use to have a plan of the environment, whereas pictures of the environment would prove almost worthless when using a sonar sensor.

2.1 Odometry

As most platforms so far in robotics research have been wheeled, the use of encoders to measure the rotation of the wheel axes has become more or less standard. The common term for this kind of information is odometry. Odometry can provide information about the change in the pose of the platform. The odometric information is extracted using sensors to count the number of rotations for axes that are involved in the motion of the platform, such as wheel axes and steer axes. Typically, high resolution encoders are used to measure the number of rotations for an axes. One of the most common encoder types is the optical encoder. The optical encoder uses a light source in combination with a photo detector to detect rotations of a disk with slots in, mounted on the axis of interest. The more slots on the disk per revolution, the higher resolution of the device. The angle information will be discretized, as the number of “ticks” are counted. As the resolution is normally very high, the discretization is a problem only when measuring slow rotations.
2.1 Odometry

2.1.1 Odometric Models

There are different kinematic designs of mobile robots. This design will influence the performance of the odometry to a large extent. To construct a platform with high performing odometry, it is thus important to carefully think the kinematic design through. We will here discuss two of the most common kinematic designs, with corresponding odometric model and a short analysis of the expected performance.

**Differential Drive**

In the differential drive configuration there are two individual drive wheels, placed on each side of the platform (see Figure 2.1). The platform is turned by applying different velocities for the drive wheels. Caster wheels are used to keep the platform balanced. The number of rotations of the right and left wheel during each sampling interval can be calculated by placing encoders on the wheel/motor axes. The corresponding distance traveled, \( D_R \) and \( D_L \), can be calculated knowing the radius for the wheels, \( r \). The distance traveled for the center point \( (C) \) of the robot is given by

\[
D_C(k) = \frac{D_R(k) + D_L(k)}{2}
\]  

The change in orientation, \( \Delta \theta \), of the platform with respect to a global coordinate system is calculated according to

\[
\Delta \theta(k) = \frac{D_R(k) - D_L(k)}{b}
\]  

**Figure 2.1:** Differential drive platform.
where \( b \) is the length of the baseline. The orientation of the platform is found by summing the small changes, \( \Delta \theta \). Assuming that the movements of the platform is given by first translating the platform and then rotating it, the pose of the platform in a global coordinate system is given by the above entities, using straightforward geometry.

\[
x(k + 1) = x(k) + D_C(k) \cos(\theta(k)) \\
y(k + 1) = y(k) + D_C(k) \sin(\theta(k)) \\
\theta(k + 1) = \theta(k) + \Delta \theta(k).
\]

(2.3)

**Synchro Drive**

The synchro drive vehicle is based on the concept of having all wheels moving synchronously. Each wheel can move around two axes. One axis for translation and one for steering. The motion of the wheels are synchronized by, e.g. a chain or a belt. Figure 2.2 illustrates the idea that all wheels are connected. Figure 2.3 shows one way to make the wheel translate as well as steer. Translation is achieved by rotating the inner axis, \( \alpha \). The outer axis steers the wheel, giving them an angle \( \varphi \). The platform itself does not turn when the wheels turn, meaning that the orientation, \( \theta \), of the platform with respect to a global frame of reference
2.1 Odometry

should ideally be constant. The odometric model is very simple for a
synchro drive platform as the direction of travel is given directly by the
direction of the wheels in the global reference frame, \( \varphi + \theta \).

\[
\begin{align*}
x(k + 1) &= x(k) + D(k) \cos(\theta(k) + \varphi(k)) \\
y(k + 1) &= y(k) + D(k) \sin(\theta(k) + \varphi(k)) \\
\theta(k + 1) &= \theta(k)
\end{align*}
\]  

Here \( D \) is the distance traveled in the \( k \)'th sample and \( \varphi(k) \) is the cor-
responding steer angle. Using the notation introduced in Figure 2.3

\[ D(k) = r \Delta \alpha(k). \]  

It is possible to steer the platform without translation. This ability makes
the synchro drive platforms holonomic (or at least close to).

2.1.2 Problems

It is clear from the kinematic odometry models presented above that they
are only approximations of the true kinematics. The model assumes a
translate-and-turn behavior of the platform which is not a correct de-
scription of the true system. Given that the sampling rate is high, the
approximation is good enough. What is worse, is that even a small error
in the orientation, \( \theta \), of the platform will eventually lead to large errors
in the position. These errors in orientation will arise after applying the
last Equation of (2.3) for some time, due to model imperfections (wheel
radiuses, baseline lengths, etc). By careful modeling these errors can be
made very small. Non-systematic errors on the other hand cannot be cap-
tured by the model and will lead to devastating errors. Common sources
of non-systematic errors are wheel slippage and bumps on the ground.
It is easy to understand that a two wheeled differential drive platform is
affected more by wheel slippage than a three (or more) wheeled synchro
drive platform as as long as the slippage is limited to one wheel.

Indoor platforms normally have much better odometric quality than
outdoor once because of the non-planar surfaces which face outdoor plat-
forms. It is important to keep in mind the conditions under which the
final product is going to operate. In many labs, the floors are very smooth
and the odometric system will be very accurate. Relying on odometry to a
large extend may prove fatal if the floor surface is changed by e.g. placing
a carpet on the floor, thereby reducing the performance of the odometry.
Borenstein and Feng [7, 8] have presented a benchmark test called the University of Michigan Benchmark test (UMBmark) to evaluate the performance of the odometric system. The key is to identify the systematic errors and thereby being able to compensate for them. They report that “the vehicle’s odometric accuracy (with respect to systematic errors only) increased by at least one order of magnitude” when the compensation is made. Methods for detecting (extended UMBmark) and compensating for non-systematic errors have also been developed. In Chapter 4 the performance of the odometry of our experimental platform is evaluated using the UMBmark.

In [9], Borenstein presents results for the commercially available mobile robot called “OmniMate” which show that the IPEC-method (Internal Position Error Correction) described in e.g. [3] successfully compensates for non-systematic errors, giving an overall increase in accuracy of one order of magnitude.

2.2 Sonar

The ultrasonic sensor, the sonar, is the most widely used sensor in robotics [2]. It was originally used in underwater applications and camera auto focus systems [10]. Some of the most obvious reasons for its popularity are (as pointed out in [11]) that the sensor is widely available, inexpensive and easy to use. One more reason for the sensor's popularity must also be the amount of research which has been put down on the sensor [12, 10, 13, 14, 15, 16, 17, 18], leading to algorithms which can easily be picked up in the literature.

As the sensor is cheap many sensor units can be used without breaking the budget. Most of the commercially available platforms are equipped with a ring of sonar sensors. The number of sensors in the ring can vary, but is normally between 12 and 30. Figure 2.4 show a typical sonar sensors setup.

2.2.1 Basic Ultrasonic Physics

We will here briefly describe the sonar physics. For a more thorough description see for example [14, 2, 3, 19]. The sonar sensor uses acoustic energy for measurements.
Figure 2.4: A typical ring of sonar sensors, from a Nomad200 robot.

Ultrasonic transducers

The ultrasonic energy is created using a transducer. Three types of ultrasonic transducers can be identified [2]: i) magnetostrictive, ii) piezoelectric and iii) electrostatic. We will here only treat piezoelectric and electrostatic transducers.

Piezoelectric The piezoelectric sensor is based on a piezoelectric crystal which can be made to oscillate by applying an appropriate voltage over it. The oscillation typically have a very small amplitude, but the force can be very high. Air is a very low-density media, yielding weak coupling with the oscillations of the crystal. Water on the other hand is much more dense and the coupling is therefore better. The piezoelectric sonar sensor can be operated at very high frequencies, in the order of MHz. It takes a few cycles to start and end the oscillations due to the inertia of the crystal.

Electrostatic The electrostatic transducer consists of a foil membrane that is made to vibrate. The vibrations have larger amplitudes than the piezoelectric transducer, thus giving a better coupling to low-density media like air. The force is much lower though. The range of frequencies that can be used is much broader than for the piezoelectric transducer, but the upper limit is in the order of hundreds of kHz, while the piezoelectric transducer can be operated at MHz frequencies.

2.2.2 Sonar Ranging Principles

The idea behind the ultrasonic based ranging sensor is simple. The ultrasonic transducer transmits a short ultrasonic pulse, and a receiver reg-
isters what comes back. The returned signal can be processed in many ways, but the most common way is to measure the time-of-flight (TOF), i.e. measure the time from transmission to reception of the signal. The signal which is received is integrated until it reaches a certain threshold and then the time is measured. Based on knowledge of the transmission speed of sound in air ($\approx 330$ m/s) and the time of flight, it is trivial to compute the distance to the target which reflected the pulse. At least it would have been trivial, had one known exactly the temperature, humidity, air pressure and so on which effects the speed of sound. The parameters will not be known exactly and will therefore contribute with some uncertainty. In an indoor environment though, these parameters are more or less constant. The distance to the reflecting object is thus given by:

$$D = \frac{1}{2}cT$$

(2.6)

where $c$ is the speed of sound and $T$ is the round trip time. There are many factors which effect the propagation of the acoustic energy [2].

- The atmospheric attenuation caused in part by absorption. More important though, is the fact that the energy intensity decreases proportional to the inverse of the squared distance traveled as the energy is emitted in a cone.

- The reflectivity of the target will determine how much energy that comes back.

- Wind and temperature will effect the way the energy propagates through the air. The speed of sound that is needed to deduce the distance to the target is in fact proportional to the square root of the temperature.

- The sound pulse is diffracted in the transducer and propagates within a cone (see Figure 2.5). The most frequently used sonar sensor, the Polaroid electrostatic transducer\(^3\), has a beam width (3-dB level) of approximately 25° (compare Figure 2.5) and the range of operation is from 0.9 to 35 ft [15]. The figure page 235 in [2] shows a typical energy profile of the Polaroid sensor. It is worth mentioning that it is possible to get reflections back far outside the main lobe of the beam. In our experiments we have observed reflections from

---

\(^3\)This sensor was originally developed for use in auto focus systems for cameras.
rough materials from angles up to 30°. Flynn [15] reports that he can detect an 1-in pole at an angle up to 40°. Borenstein & Koren [20] point out that the specified beam width in many cases is a very conservative assumption.

\[\text{Figure 2.5: The ultrasonic energy is transmitted in a cone. The distance to an object is measured by time-of-flight. Using measurement from one robot position the location of the reflecting object cannot be determined. The location of the object is constrained to be somewhere on the spherical top of the cone.}\]

The signal which is received contains much more information than what is used in the TOF based technique, e.g. phase. By looking more closely at the signal which comes back it is possible to identify more than one source of reflection, i.e. more than one target. It takes a lot more processing power though, and most users settle for the standard TOF solution.

### 2.2.3 Drawbacks with the Sonar Sensor

There are many drawback with the sonar sensor as pointed out in e.g. [21, 15, 20, 17, 11, 22, 23, 24]. Some of them are:

- The wide beam, which has the effect that it is impossible from one reading to tell where the reflecting object are within the beam. To resolve this problem several readings have to be integrated. The wide beam width is not only a problem though, it gives the sensor a big field of view, i.e., it covers a large area which is desirable in some cases. In [25, 26, 27] the wide beam is a requirement for the success of a triangulation based method for fusing sonar measurements over time.

- Due to the speed of sound, the time to get a reflection back is not negligible. Since most systems use the standard Polaroid sensors,
all working with the same frequency, care has to be taken to avoid sensors picking up echoes from other sensors, so called crosstalk (see Figure 2.6). This means that one has to fire the sensors in a clever way as to avoid crosstalk as much as possible. Normally it also means that it takes a substantial amount of time to fire all of the sensors, typically 300 to 600 ms [20]. In [20] a method for almost completely eliminating crosstalk and increase the rate of fire by a factor of five to ten times is presented. The problem with the method is that it requires that one can control the sensors to a degree which most platforms do not support.

- Sensitivity to noise from the environment and other robots using ultrasound with the same frequency. Lately much interest has been directed towards cooperation of robots, see e.g. [28, 29]. When multiple agents are moving close to each other the problem with crosstalk between the platforms has to be handled.

- Specular reflections cause a big problem when interpreting the data. A surface is specular when it has a roughness of less than 5% of the acoustic wavelength [30], which for the standard Polaroid sensor means 0.3 mm. A specular surface will not give rise to a direct reflection when the sound beam hits the surface under an angle larger than the beam width. It might hit some other obstacles though and after multiple reflections come back to the sensor. In [22] it is said that almost all measurements obtained with the standard Polaroid sensor ranging system are the result of specular reflections, but also that most of the accurate readings from planes and corners are specular reflections. In [23] a scheme for handling specular reflections is developed.

There are ways to go around some of the problems with the sonar sensor. Much can be achieved by simply building your own signal processing unit, instead of using the standard Polaroid unit. Arrays of sonar sensors can be used to solve the problem of the wide beam width (see Section 2.2.5) and in [15] four different frequencies are used to counteract the dependency on reflecting material.

### 2.2.4 Sonar Sensor Models

Many models for the sonar sensor as been proposed in the literature. Crowley began by trying to fit lines to the sensor data [3] since many of
Figure 2.6: The sensor marked with 1 transmits ultrasound which through reflections can be detected by sensors 2 and 3. If sensor 2 and 3 fire before the sound pulse from sensor 1 has died, the distance might erroneously be taken as very small.

the structures which the robot faces in its environment are straight lines. He modeled the sonar in such a way that the target which gave rise to the reflection is assumed to be on the center of the acoustical axis or the far left or right of the beam depending on the situation. Later he refined the model [31] and introduced the idea of the echo coming from an arc shaped region at a distance equal to the range reading from the sensor. Moravec and Elfes [12] introduced a probabilistic profile for the sensor.

Borenstein and Koren later used a simplified version of this model where only the acoustical axis was considered. In [17] they model the sonar sensor as only giving the location of the target which gave rise to the reflection. This model is refined in [32] where the area in front of the target is also accounted for by saying that it is likely to be unoccupied. The reason for the simplification is that it reduces the computational load, giving a faster update rate.

Kuc and Siegel [14] studies the signal from the sonar sensor in order to construct a model of it. Data from Elfes (mentioned above) has “successfully been interpreted using the approach”. They consider a specular world which consists of walls, corners and edges. Based on the physical model of the sonar they can predict the response from these geometrical object. Another benefit of the model is that different frequencies and waveforms can be tested in simulation.

Leonard and Durrant-Whyte [22, 33] base their work on the physical model for the sonar developed by Kuc & Siegel, and introduce the concept
of regions of constant depth (RCD). When looking at a densely sampled scan of sonar data it can be seen that there are regions at approximately the same depth (see [22, p.381]), these are the RCDs. One conclusion which is drawn after having studied large amounts of sonar data is that “almost all measurements obtained with the off-the-shelf unmodified Polaroid ranging system in typical scenes are in fact the result of specular reflections”. This justifies the model which was based on the assumption of specular objects.

2.2.5 Sonar Array

By combining many sensors and using the phase information a more detailed picture can be obtained of the environment. The idea is to use more than one receiver in order to be able to pick out more exactly from where the echo comes. Just like when using stereo cameras the correspondence problems becomes larger the further away the receivers are, but on the other hand the accuracy becomes better. Theories from antenna array signal processing can be applied to a setup with sonars as well.

Lindstedt [34] urge us to look at the bats for inspiration. He claims that they face the same problems as we do, but that they have developed a system far superior to any of ours. One of his key ideas is that every problem should be dealt with individually and a dedicated solution should be sought. The argument is based on the fact that a bat will not use the same processing to hunt prey as to avoid obstacles. One more thing is that bats use a sweeping frequency and every bat has his own voice print, i.e. it can distinguish its sound from other bats. In most mobile robot system the same frequency is used for all transmitters and in many cases for all platforms. Lindstedt is using a custom made setup with one transmitter and four receiver placed in a quadrant around the sensor. 

Chong & Kleeman [35, 36] uses a sonar sensor array consisting of three receivers and three transmitters. The sensor is able to locate and classify geometric features into planes, corners, edges and unknown. Three transmitters and three receivers are used in a special configuration.

2.3 Laser

As the laser sensor will be the main sensor used in this thesis a short historical overview and basic description of the technology will be given of this fantastic invention, called laser. The overview is based on material
found in [37].

2.3 Laser

2.3.1 History and Technology

Laser technology is everywhere today, in CD-players, surgery, communications, etc. Laser stands for Light Amplification by Stimulated Emission of Radiation. It is based on the fact that an excited atom returns to a lower energy state by either spontaneously emitting a photon or being stimulated to do so by electromagnetic radiation. The photon emitted by stimulation is emitted in the same direction as and in phase with the stimulating radiation.

An emitted photon can help stimulate other atoms to emit photons, thereby causing an avalanche effect. As all atoms strive to be in as low an energy state as possible, few atoms are in an excited state under normal conditions. In this case the radiation would instead be absorbed and photons would spontaneously be emitted. Spontaneous emitted photons are sent out in random directions and with random phase. The electromagnetic radiation has to be of the right frequency. To be exact it has to be of the frequency that corresponds to the difference in the energy between the exited state and the lower state the atom goes to by emitting a photon.

The idea in a laser is that by pumping the laser media, i.e. exciting the atoms so that a much higher percentage than normal is excited (called inverse population), the stimulated emission of photon is going to be the dominating effect. By putting a pair of reflective surfaces at the end of the laser cavity, the wave will be reflected back and forth, increasing the energy with every pass in the direction of the cavity. In order to sustain the inverse population, energy has to be feed into cavity to pump the media. By making one of the surfaces only partially reflective, part of the wave can be taken out of the cavity. The transmitted energy forms the laser beam.

The laser was preceded by the maser (Microwave Amplification by Stimulated Emission of Radiation) in the 1950s. In July 1961, Maiman announced existence of the first operating laser. It was a pulsed solid-state ruby laser with the center frequency at about 694.3 nm. Shortly after that, in 1961, another team of researchers reported that a helium-neon, continuous wave gas laser was operating. The helium-neon laser is the most popular laser today, because it combines ease to use, with a

\[\text{Ruby is one of the most used laser media still today}\]
low cost and possibility to give continuous power in the visible frequency range (632.8 nm). A historical curiosity is that Maiman’s first paper about his discovery was rejected [37, p. 585]. Little did the people who rejected it know what an impact the new technology would have on science.

After the first discovery new medias and improvements of old techniques were found at a rapid pace. The first semiconductor laser was operational in 1962, but it could only operate in pulsed mode and only at cryogenic temperatures\(^3\). There have been semiconductor laser (also known as diode laser) operating in room temperate since the 1970s and today the technique has become irreplaceable in many applications. Semiconductor lasers are small, energy efficient, have sharp spectrum and can be modulated at very high rates.

It is clear that the laser will continue to play a major role in the field of science and in everyday life.

2.3.2 Laser Range Finders

Having found applications in so many areas, it is not surprising that the laser has been used to measure distances. Already in the 1970s NASA made use of laser techniques for this purpose. At that time the technique had not reached the right level of maturity to be applicable in a large scale. It is obvious that measuring distance with light requires high precision electro-optics.

The dominating techniques for laser based range measurements are TOF-techniques and phase-shift-techniques.

**Time-Of-Flight (TOF)**

In a TOF system a short laser pulse is sent out and the time until it returns is measured. The ranging principal is thus the same as for the standard sonar sensor

\[
D = \frac{1}{2}cT
\]

where \(c\) is the speed of light and \(T\) is the round trip time. A sensor of this type is often referred to as laser radar or lidar. In order to realize such a system, a high precision means for measuring time is needed. Thinking in

\(^3\)Cryogenic temperatures refer to extremely low temperatures, ranging from -150°C to the absolute zero at -273°C.
2.3 Laser

terms of a robot application a range resolution in the order of centimeters is desirable. The speed of light is $3 \cdot 10^8$ m s$^{-1}$. This means that the precision in time has to be in the order of 100 ps, corresponding to a frequency of 10 GHz. It is not difficult to understand that this puts high demands on the equipment. One advantage with the short pulses is that higher levels of powers can be used, giving better range coverage, but still keeping a high safety level and low power consumption. Commercially available systems today have reached below centimeter accuracy.

Phase-Shift

In phase-shift-systems a continuous wave is transmitted. The idea is to compare the phase of the returned signal with a reference signal generated by the same source. Using the Doppler shift, the velocity of the target can be measured in addition to the distance to it.

One problem with a phase-shift-based laser measuring device is that it can not distinguish between phase-shifts greater than one wavelength from a phase-shift smaller than one wavelength [38]. This means that ranges above the wavelength of the modulator cannot be distinguished from ranges below it.

2.3.3 Scanning Sensors

Most of the commercial laser sensors measure the distance in a single direction. By mounting it on a rotating body, a scanning effect can be achieved. Instead of rotating the whole sensor, there are now commercially available laser scanners based on a rotating mirror which can cover a large field of view. This kind of sensor has been used by many researchers, see e.g [39, 24, 40, 41, 42].

Compared to the sonar sensor the laser scanner is still very expensive and one has to weigh the price against the performance. In many applications a laser scanner might be too expensive (e.g. powered wheel chairs), whereas other applications are less sensitive to the price (e.g. mining trucks).

It is possible to add one more degree of freedom and let the sensor scan up-down as well, yielding a 3D scanning device [43].

2.3.4 3D Laser Imaging Sensors

There are also some 3D laser imaging sensors available (used e.g. in [38, 44, 45, 46]). Often they come in combination with a camera to form a
powerful sensor pair, which gives both information about intensity and depth. The depth information can be used to help solve the difficult correspondence problems which a user of only cameras face.

2.3.5 Properties of the Laser Ranger Finder

Some of the important properties of the laser range finder will here be mentioned. Capturing these properties in a sensor model requires the inclusion of properties of the environment which is generally very difficult.

Material Dependence

In [47, 48], objects are divided into 6 categories depending on their geometric and surface characteristics.

1. Flat and smooth surfaces (e.g. buildings, walls, wood)
2. Small isolated objects (e.g. tree trunks, poles, wires)
3. Depth texture (e.g. bushes, grass, textiles, wool, foam)
4. Transparent and semi transparent surfaces (e.g. windows, plastics, glass)
5. Reflecting surfaces (e.g. mirrors, wet smooth surfaces, polished steel)
6. Absorbing objects (mate dark surfaces, smoke, foam rubber)

It is noted that objects of type 1, 2 and 4 dominate in lab scenes whereas 1, 2 and 3 are the most common types outdoors. By designing a dedicated filter to handle data from a particular object type the accuracy of the data can be increased. Doing this requires that the reflecting objects are identified and classified.

Size of Footprint

The laser beam diverges slightly. Hence the beam will not have the same width at all distances (see Figure 2.7). The energy will be spread out over the width of the beam. When a object is hit by the laser it leaves a so called footprint. The smaller the footprint is the more energy per area giving a longer range of operation. A small footprint will give higher angular resolution, but will on the other hand require more samples to cover a certain angular range. This is a trade off between coverage and resolution. Chapter 4 presents experimental results characterizing the footprint of the PLS laser scanner form SICK.
2.3 Laser

Figure 2.7: The laser beam diverge, meaning that the beam will have different area at different distances from the source.

Beam Split Between Objects at Different Depth

In order to get accurate range measurements it is important that the whole footprint gets reflected from the same object at the same depth. If the beam is split between surfaces at different depth, the resulting measurement can be anywhere between the foremost surface and the surface furthest away. By integrating data over time collected from different positions, data points that are likely to be effected by this can be identified and handled correctly. This effect is investigated further in Chapter 4.

2.3.6 Advantages of 2D Laser Scanners

There are many advantages with 2D laser scanners e.g.:

- It is fast, i.e. the measurement can in most cases be considered as instantaneous. This means that one does not have to think about compensating for the motion of the platform between sending and receiving. What stops laser scanners from working even faster is the mechanics. In a sonar system most of the time is wasted, waiting for the pulse to return. When using laser, the problem is instead to be able to rotate the scanning device fast enough and at the same very time accurately.

- The range accuracy is fairly good. [41] reports to have a standard deviation of 2 cm and the new generation of laser scanners from SICK Electro-Optics is claimed to have an accuracy at the order of 10 mm and the angular resolution is down to 0.25°.

- The angular resolution is far much better with the laser scanner than with sonar sensors. The resolution for the PLS laser scanner from SICK has an angular resolution of 0.5° which is orders of magnitude better than for the sonar.

- The data from the laser scanner can be interpreted directly as the range to an obstacle in a certain direction. This can of course be
said for the sonar as well given that we disregard specular reflections 
and multiple reflections, but not for e.g. a camera image which takes 
a lot of effort to interpret.

\[ \text{Figure 2.8: By tilting the laser, information can be gathered from a} 
\text{larger subspace than a plan assuming the platform in moving.} \]

2.3.7 Drawbacks of 2D Laser Scanners
Among the drawbacks we find:

- The sensor provides range information limited to a plane, which 
  means that only a 2D intersection of the 3D world can be sensed. 
  To be able to get information about other parts of the environment 
  the sensor has to be moved. Another possible solution is to mount 
  the sensor so that it intersects the world under an angle $\alpha$ (see 
  Figure 2.8) and instead move the platform. This angle $\alpha$ becomes 
  a design parameter for the system which will determine the range 
  of the sensor.

- The sensor is still very expensive. The price will drop and the 
  sensor will become more widely available if an application is found 
  that will motivate mass production of it.

- Some material appear as transparent for the laser, such as glass. 
  This means that the laser sensor must be combined with some other 
  source of information in order to make a robust system.

2.4 Infrared
There is another group of sensors which can be used to measure the 
distance using light. This type of sensor does not use laser but instead
sends out normal infrared energy. By measuring the intensity which comes back to the sensor the distance to the target can be calculated. This technique has some obvious drawbacks.

- The energy is not as highly concentrated to a particular direction as in a laser system, giving a much lower range coverage, typically less than 1m.

- As the measure depends on the amount of intensity which comes back to the sensor, the material which reflects the energy will be very important. This material dependency might not be a severe limitation is the environment is stereotype, but in a normal indoor environment the type of material varies much. Our experience with this sensor is that it will in practice only be able to provide binary information, i.e. is there something in the direction of the sensor or not.

The advantages with the sensor are, e.g. that it is very cheap and is easy to use.

2.5 Radar

Radar has been used extensively in both military and commercial system for a long time. Air traffic control depends on it, and more and more applications are found every day. By radar we will understand the sensors operating with frequencies around 3 to 300 GHz. IEEE’s definition of radar [2] “an electromagnetic means for target location and tracking” is much wider and includes e.g. laser radar and IR sensors. The physical principals behind the radar is the same as those for the laser sensor, i.e. TOF, phase shift and frequency modulation. Important environmental factors to be considered in a radar system include [2]

- Atmospheric attenuation is caused by two phenomena, scattering and absorption. The extinction coefficient $\sigma$ sums up the effects from scattering and absorption to give a measure of the degree of transmission.

- Surface reflectivity depend on the wavelength, the material and the surface structure. So called specular reflections might turn an object into being invisible from certain angles.
• Differences in the air pressure, might lead to large variations in the index of refraction. A wave that hits a surface between two indices of refraction can be deflected.

• Temperature and humidity are also factors that the index of refraction depends on.

• The beam geometry is of interest as a more focused beam will increase the operating range.

Traditional radar applications use frequencies between 3 and 100 GHz [2], so called microwave radar. The reason being that microwave radar combines long-range sensing with sufficient resolution and minimal attenuation in the atmosphere [2]. Millimeter radar operates with frequencies between 30 and 300 GHz and gives less range coverage, but instead gives better range resolution. The applications for radar, besides air traffic control, include for example automatic brake system for cars [49].

2.6 Inertial

A type of sensor which has been used for a long time in aircrafts for navigation is the inertial sensor. Together with the radio beacon system and GPS, it forms the base for the navigation system in an aircraft. The inertial sensors can be used for dead-reckoning navigation.

There are different types of sensors which fall in under the term inertial sensors.

• Accelerometers

• Gyroscopes

• Tilt sensors

2.6.1 Accelerometers

The accelerometers measure, as the name indicates, acceleration. Both linear and angular accelerations can be measured. Here the output from the sensor will be a direct measure of the acceleration. Newton’s first law $F = ma$ provides means to instead measure force, which can be performed using e.g. strain gauges.
These sensors must be integrated twice to get position information and typically have very poor signal to noise ratio for low accelerations [3].

2.6.2 Gyroscopes

Gyroscopes can measure angular or linear motion. The gyroscopes can be further subdivided into rate gyros and rate integrating gyros. The rate gyros give velocity whereas the rate integrating gyros give position. The two common categories are mechanical and optical gyroscopes.

**Mechanical gyroscopes** monitor the change in angular or linear momentum. The physical law of conservation of momentum is the basis for the technique. The so called flywheel gyroscope is probably what is most often associated to when speaking of gyroscopes (see [2, p. 364] for a picture).

**Optical gyroscopes** use light to measure movement. The idea is to let two laser beams travel in the opposite direction in a closed loop. The two beams are then mixed and the interference between the beams are utilized to measure the rotation. By studying the change in the the interference pattern when the loop is rotated, the angular rate and direction can be found.

2.6.3 Inclinometers

The inclinometers typically consist of a arc-shaped tube with liquid and a gas bubble. Gravity forces the bubble to float to the highest point in the liquid. The position of the bubble is measured by for example monitoring the resistance between three electrodes that change when the bubble change position. The good thing about the sensor is that is measures the angle directly, without integration, which eliminates errors caused by integration. Due to the physics of the sensor, it is mostly useful under stationary conditions. Moving the sensor will introduce other accelerations in addition to gravity.

Inertial sensors can be used to measure both rotation and translation, and it is common that the sensor can measure the movement in relation to more than one axis, forming a multi-axial inertial sensor. For a robotic application it is a perfect add-on to an odometric system. The inertial
sensor is also subject to drift, but when combined with the odometry the overall result can be improved.

2.7 Compass

Traditionally, humans have put a lot of trust in the compass when navigating. It is a very old method to measure direction, 2643 B.C. [3]. Still today, almost all crafts, both operating in water and in air, are equipped with a compass. Traditionally the term compass has been equivalent with magnetic compass. Now there are other types, e.g. the gyrocompass [2]. The mechanical magnetic compass is the most common and oldest kind of magnetic compass, but nowadays a multitude of optional devices exist. The magnetic compass uses the magnetic field of the earth to find the direction.

For robot applications the compass typically suffers from severe deviations. These deviations are caused by large quantities of metal and the presence of motors and high power distribution systems which are normally found on a robot. It is still a good source of information when only crude orientation information is needed, e.g. to get an approximation of the initial orientation of the robot.

2.8 Vision

Vision is the sensor which has the greatest potential, but also the sensor which is most difficult to master. Even though vision has been studied for decades, few robust algorithms exists. The algorithms might perform very well under some conditions, but by for example changing the lighting conditions or the texture of the background they will not work any more. Most algorithms are written for a particular purpose, i.e. they are not general, and they are based on ad hoc rules and usually do not have any theoretical base.

The human vision system consists of two "cameras" which can be moved individually. We have one region with very high resolution, which we use to identify objects for example. Outside this region we have peripheral vision which gives us almost 180° field of view. In the peripheral region we have only limited resolution, but it still responds to stimuli like motion and color. Systems which try to mimic this setup has been constructed. One example is Brooks latest project, the robot Cog [50].
2.9 Structured Light

To overcome some of the problems with vision, structured light is sometimes used. It is still based on cameras, but to simplify the extraction of certain information, a source of structured light is projected onto the scene. In its simplest form the structured light is a thin stripe of light. By looking at how this line deforms when it falls on the objects, valuable information can be retrieved.

In [51] a light grid based on pseudo-random coding is used. The grid code makes it possible to retrieve the absolute position of certain points in the grid without seeing the whole grid, i.e. the grid contains local information about the position in the grid.

Some of the research on structured light is focused on trying to find optimal light patterns for good 3-D reconstruction.

2.10 Summary

In this Section we summarize the information about the different sensors used for localization. Table 2.1 gives a short overview of the most commonly used sensors. Range sensors can be identified as one subgroup. Among the range sensors we find laser, sonar, IR and radar. Stereo vision and structured light can also be argued to provide range information, but not directly. We here chose to make the distinction between sensors that provide direct and indirect range information. The sonar sensor, is the most widely used sensor for localization purposes mainly because they are cheap. Laser and vision are also being used extensively. We believe that vision is the most powerful sensor if its full potential can be utilized. A problem with vision algorithms are that they are typically computationally expensive. The algorithms developed for vision are also typically very sensitive to the operating conditions such as lighting, textures, etc.

When designing a localization system we need sensors that provide absolute position information, that is we need external sensors. Given that we wish to use a simple environmental model (see Section 1.3), it is clear that range sensors are the best choice for localization. Vision would require extensive computations to filter out the simple features that are captured by the model. Range sensors on the other hand provide a direct image of the environment, as described by the model, which is “easy” to interpret.
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Information</th>
<th>Accuracy</th>
<th>Adv./Disadv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Odometry</td>
<td>Pose change</td>
<td>±1mm/m</td>
<td>cheap, simple subject to drift</td>
</tr>
<tr>
<td>Sonar</td>
<td>Range</td>
<td>±2 cm, ±5-30°</td>
<td>cheap rel. well understood low angular res. rel. low scan rate</td>
</tr>
<tr>
<td>Laser</td>
<td>Range</td>
<td>±1-5 cm, ≈0.5°</td>
<td>high angular res. high scan rate expensive transparent materials</td>
</tr>
<tr>
<td>IR</td>
<td>Range</td>
<td>1-10 cm</td>
<td>cheap material dependent</td>
</tr>
<tr>
<td>Radar</td>
<td>Range</td>
<td>-</td>
<td>long range used extensively outdoor</td>
</tr>
<tr>
<td>Inertial</td>
<td>Acc, vel, pos</td>
<td>-</td>
<td>easy to use drift</td>
</tr>
<tr>
<td>Compass</td>
<td>Direction</td>
<td>-</td>
<td>easy to use sensitive to disturb.</td>
</tr>
<tr>
<td>Vision</td>
<td>Various</td>
<td>-</td>
<td>pot. very powerful difficult to use not well understood</td>
</tr>
</tbody>
</table>

**Table 2.1:** The table gives an overview of the properties of the different sensors presented in this chapter. Radar is a very wide term and we therefore find it hard to give a measure of the accuracy. The same is true for inertial sensors. The compass is very sensitive to disturbances. The accuracy of the compass itself is high (better than 0.1°). Vision has the greatest potential and can be used to solve most problems, but the understanding of it is very limited.
2.10 Summary

When selecting a specific range sensor we reason as follows. Radar is ill suited for indoor use and IR is too sensitive to the reflecting material and with poor range. This leaves sonar and laser, which provide the same kind of information. The laser has orders of magnitude better angular resolution which makes the data association problem simpler, but on the other hand the laser sensor only scans one plane of the environment. The sonar sensor, having a larger beam width, covers more but we believe that the higher angular resolution of the laser more than compensates for that. Therefore we believe that the laser sensor is the best choice for localization. We stress that the reasoning above is based on the assumptions that we want to use a simple model of the environment and that we are only considering the problem of localization. It is also worth mentioning once again the importance of having good physical understanding for the sensor being used.

No system will be robust using only one sensor. All sensors are prone to error and no single sensor normally covers the whole range of operation. It is thus of paramount importance to be able to combine different sensor modalities. Having good physical understanding and realistic models of the sensors enables the fusion of information to be theoretically founded. In this thesis we will combine information from odometry and laser scanning.
Chapter 3

Methods for Localization

Borenstein et al [3] divide the different methods for localization into four groups, depending on the technique being used.

- **Dead-reckoning**
  The idea of this method is to measure the change in position and by integration keep track of the position. A typical example is the odometry. Change in position can also be measured by looking at the change in the sensor data. The main disadvantages of a dead-reckoning system is that the estimate of the position will degrade with time. Another problem is that the method assumes that the initial position is known, i.e. that the initialization problem is solved.

  As mentioned in Section 2.1 almost all systems have access to odometric information. Inertial sensors is also a possible base for a dead-reckoning system (see e.g. [52]) and by combining them a potentially even better system can be achieved.

- **Active beacon**
  When the work space of the mobile robot is the same always, it is possible to engineer the environment in order to simplify the localization problem. Active beacons and triangulation can be used to find the location of the robot. High accuracy can be achieved but high installation cost are often involved.

  One example of an active beacon system is the Global Positioning System (GPS) which is used extensively, but it is only for outdoor
use. The unrestricted Standard Position Service (SPS) provides a horizontal position accuracy of about 100 m, whereas military version, the Precision Positioning Service (PPS), provides an accuracy of about 21 m [53]. With differential GPS (DGPS) the horizontal positioning accuracy can be below 1 m [53].

- **Landmark**
  Two types of landmarks can be distinguished, artificial and natural landmarks. The artificial have been placed in the environment for the purpose of being a landmark, examples include bar codes and floor stripes. The natural landmarks are extracted from the environment without any changes being made. The artificial landmarks have the advantage of providing arbitrarily good information. Bar codes can provide arbitrarily detailed information. The disadvantage is that the environment will have to be engineered, which in turn results in a less flexible system because the artificial landmarks have to be supplied before the robot can find its position. The natural landmarks on the other hand give a much larger flexibility, but the down side is that it might be very difficult to find good natural landmark in some environments and sometimes a human is needed in the loop for selecting them. In [54] a method for automatically identifying and learning how to recognize landmarks is presented. The motivation is that most successful localization systems are based on landmarks and to achieve high flexibility, the system itself has to find out what landmarks are useful in a particular environment.

- **Map-based**
  In the map-based methods a local map of the environment is built from sensor data. This local map is then matched against a global map to determine the position of the robot. The method is also known as map matching. Different types of maps have been discussed and utilized in the literature, including for example occupancy grids and feature maps. The advantages of this method are e.g. that the map which it is based on can be used for other tasks like path-planning, and that the methods allow for learning. Among the disadvantages we find that these methods require quite large amounts of processing and that they put high demands on the modeling of the sensors.
3.1 Localization Methods Requiring Engineered Environments

We are here only interested in methods which do not need any engineering of the environment, i.e. every kind of active beacon or artificial landmark is out of the question. The reason being that a truly autonomous system must be able to use the information provided by the environment as it is. We can not assume that every possible environment that the robot will encounter has been engineered.

The cut between landmark based techniques and map based techniques is not very distinct. The information about the position of the landmark in the environment must be stored in some way and this information could be said to constitute a map. The dead-reckoning methods cannot be used for position initialization, but they are still the backbone of many of the existing localization systems e.g. as the predictive part of a Kalman filter.

With all this in mind, we prefer to divide the localization methods into two three groups. The first one contains the dead-reckoning methods, and the other two groups are separated based on if they require engineering of the environment. We will not further discuss the dead-reckoning methods, because our interest is towards methods which potentially could be used for both initialization and maintenance of the pose information. The methods which require engineering of the environment will only be very briefly summarized, because they do not satisfy our requirement about not altering the environment in order to do localization (see Chapter 1).

3.1 Localization Methods Requiring Engineered Environments

The list of methods which use engineered environments will not be complete but is intended to give an idea of what is available.

- The most widely used method in this category is probably the GPS system. It is used extensively in outdoor applications, but it cannot be used indoors. See e.g. [33] for an overview of the technique. Active beacon systems have also been developed for indoor applications.

- In industry it is common to use a localization system which is based on different kinds of lines on the floor [55]. One obvious disadvantage is that the robot can only move in areas where there are lines on the floors. The robot is able to make small detours from the
lines on the floor, to avoid obstacles but it needs the lines to be able to find its way.

- Bar codes are also quite common. The bar codes can be used for many purposes, e.g. to do localization but also for help solving the object recognition problem by simply putting a code on the objects. Bar code could potentially also be used as part of the line in a line following method. The bar code could give absolute position information, i.e. the robot would know its position all the time.

- Lazerway is a system developed by NDC that uses a laser scanner for sensor and relies on reflective tapes being put up in the environment [55]. Using the known position of the tapes, the position of the mobile robot can be calculated. NDC is one of the leading companies for developing navigation systems for automatically guided vehicles (AGV).

3.2 Localization Methods for Non-Engineered Environments

Systems which want to be flexible, robust and have high autonomy must be able to navigate without requiring the environment to be modified. This is at least true for indoor robot. For outdoor robot one might claim that since the GPS system is already there, it does not count as engineering the environment. No single sensor localization system is perfectly reliable though, i.e. robustness will require the system to use other means for localization as well.

The existing methods for localization can be roughly divided into two main groups depending on what world modeling technique is being used. The distinction will here be made between parametric methods and grid methods. Some localization schemes utilizes both model types so the cut is not clean.

3.2.1 Parametric Methods

The idea of trying to extract features from the environment is quite natural. Extreme examples of features would be labels on doors which specify the room it is leading to, but other less discriminative features can of
3.2 Localization Methods for Non-Engineered Environments

course also be found. In a structured environment, which most office environment are examples of, lines, corners and edges are common features. The features can be parameterized by e.g. their color, length, width, position, etc. Leonard, Durrant-Whyte and Cox [56] are quite firm in their conviction that feature-based methods are the way to go, when they say: “we believe that navigation requires a feature-based approach in which a precise, concise map is used to efficiently generate predictions of what the robot should “see” from a given location”. In [4] Durrant-Whyte also promote the parametric method by saying “The advantage of describing sensor information in terms of an uncertain parametric function is that [the] geometric description itself can be transformed between different coordinate systems and different object representations, providing a simple but effective means of communicating information between different sensors”. The Kalman filter is a key component in most implementations of parametric methods, providing a good setting for pose prediction, sensor fusion and data association. [57] and [58] give good descriptions of how to use the Kalman filter for mobile robot localization.

To illustrate the ideas behind the parametric methods we believe that examples taken from real systems are the best.

Drumheller

Drumheller [13] presents a sonar based localization system which uses line segments as the only feature primitive. The model of each room is given by a set of connected line segments. Sonar data is collected at one unknown position in a room and the position is found by first extracting line segments which are then paired with the walls. One of the conclusions in the paper is that “Mobile robot localization can be performed indoors with a low resolution sonar range finder, using data obtained from only one position in a room” which is to take the result a bit too far, but under ideal conditions this might be true. A contradiction to this conclusion is found in the presentation of the results, where it is stated that the method failed e.g. when only parallel walls could be matched to sonar data. This means that normally more data than from a single position must be gathered in order to find the position of the robot. Further more, the room must be specified in advance, i.e. localization is only possible within a given room.
Crowley

Crowley was one of the pioneers in localization using parametric methods. In [5, 10] he describes the navigation part of a project that aimed at developing a low-cost intelligent mobile platform (IMP). The IMP uses a rotating sonar sensor to get sensory information from the environment. The sonar sensor has a beam spread of approximately 5° and it is mounted at a height of 30 inches. A stepper motor rotates the sensor in steps of 3°, taking 10s for a complete revolution. Due to the physics of the sonar sensor, the range reading from a sonar will correspond to the shortest distance to a target inside the sonar beam. When sampling a wall the range readings will not line up if associating the range readings with the acoustic axis. To compensate for this, a special trick is applied, consisting of systematically picking range readings from the outskirts of the beam instead. As the environment is quite densely sampled with the rotating narrow beam sonar, it is possible to extract lines directly from the raw sensor data. These lines are matched against a local map of the environment. The local map is updated partly from sensor data, but also from a global pre-defined map of the environment and is represented by a list of directed line segments. Each line segment has an attribute which represents the degree of confidence. When a line is predicted using the local map and then seen, this confidence increases, whereas the confidence will decrease when a predicted line is not seen. The orientation error is computed as the average angular error between the matched line segments. The position error is found as the average error between points connecting matched line segments.

In [31] Crowley uses a ring of 24 sonar sensors instead of a rotating sensor. The sensor model is augmented to incorporate spatial uncertainty to account for the fact that the sonars have a beam spread (see Section 2.2). A Kalman filter solution is presented for updating the position of the robot.

Montalier and Chatila

Montalier and Chatila [59] discuss the problem of simultaneously doing localization and map building. The main issue in [59] is the correlation problem which occurs between features and robot position. i.e. when the position of the robot is updated using uncertain feature estimates, the robot position estimate will be correlated with the feature estimate. The same is true for the opposite case, when the position of the features are updated using uncertain information about the robot position. A
framework for handling these correlations is developed. A shortcoming of the approach is that it implicitly assumes perfect data association, i.e. the problem of associating measurements with features is not considered. 2D laser data is used to experimentally verify the method.

Ayache and Faugeras

Ayache and Faugeras [60] presents a localization method based on geometrical primitives extracted using stereo vision. Emphasis is put on minimal representations of the geometric primitives and techniques for maintaining accurate estimates for their position in space. A Kalman filter provides the framework for the data fusion.

Cox

Cox studies the problem of localization on his experimental vehicle Blanche, which was designed to be low cost, relying only on two sensors, odometry and an optical range finder. In the early work [61] the localization is based on odometry and artificial bar codes placed at strategic places. The bar codes combine a highly reflective material with a highly absorbing material. The optical sensor provides not only distance information, but also intensity information. The intensity information can be used to recognize the bar codes, but also to detect range data which is likely to be wrong [61].

In [62, 63] Cox develops the localization, making Blanche independent of bar codes. The a priori information about the environment is instead a line segment map. The idea is that by doing position updates frequently enough, the correspondence problem is much easier to solve, i.e. what data comes from what line segment in the map. Odometry is used to predict the position of the robot before the data association is made. A position update is made every 8 s. Successful tests are performed in an structured office environment. One of the weak points is that the initial position has to be given to Blanche at startup, i.e. only the pose tracking problem has been addressed.

Hinkel and Knieriemen

Hinkel and Knieriemen introduce an interesting method for finding the position and orientation in a rectangular room in [64]. The method, called the Angle Histogram, is a two steps process based on range sensor data histograms. After acquiring a densely sampled 360° scan of the
environment, a histogram is built over the relative angle between nearly scan points, looking at the Cartesian representation of the scan, with the aim of trying to find the main orientation of the scan. To reduce the influence of noise, the relative angle is calculated between points which have a scan angle separated by 10°. In an uncluttered rectangular room, the histogram will show four distinct peaks, corresponding to the four wall orientations. If the histogram is calculated modulo 180° the two main directions of the rectangular room will show up as two peaks. Using modulo 90° will give one peak and might help finding the main direction in a cluttered room.

A scheme for solving the localization problem is presented, based on a hierarchical structure of maps. At the top, a set of so called Topels is kept, one Topel for each room. These are stored between missions and form the robot’s knowledge about the environment. Each Topel has its own coordinate system found using the angle histogram. The base feature for each Topel is a so called box feature, which represents the best rectangular approximation of the room. The Topels also contain geometric information about lines. Localization within a room is done using odometry. The position can be corrected by correlating the current histograms with the box feature in current Topel. During a mission, a new Topel is only created when entering an unexplored room, but it is updated every time new information is available. Global localization can be achieved by correlating a local geometric map to the stored Topels.

Hoppen et al [39] build on the framework presented above and extend it with planning capabilities, both local and global planning.

Edlinger and von Puttkamer [65] look at the problem of determining when one room ends and a new starts, in order to correctly divide the environment into room elements in the topological map structure. They introduce a virtual bubble, limited by virtual and real borders. A virtual border is a border to unknown area, caused by limited sensor range or occlusions. The real borders are caused by object such as e.g. walls. To keep the bubble from covering more than one room, the bubble is not allowed to deform through holes which are narrower than a certain threshold. These holes are classified as doors and constitute the border to a new room. Each room is completely explored before a new room is entered. Which room to enter is determined by the distance to the door and information about if it leads to an already explored room.
3.2 Localization Methods for Non-Engineered Environments

Leonard and Durrant-Whyte

Together with his student John Leonard, Hugh Durrant-Whyte develops a localization method based on tracking natural geometric beacons [22, 66, 67, 56, 33]. In [22] a geometric beacon is defined as: “a special type of target that can be reliably observed in successive sensor measurements and that can be accurately described in terms of a concise geometric parameterization”. Based on the work by Kuc and Siegel [14], lines, corners and edges are considered as candidates for geometric beacons. The idea is that as active beacons are not feasible for a fully autonomous and flexible system, the system itself has to find naturally occurring beacons and track them. Knowing the position of the beacons makes it possible to track the pose as well. The key components in the method are: i) prediction of new position based on control input, ii) sensor input processing to yield beacon positions, iii) matching and iv) pose estimation. An extended Kalman filter is the backbone of the method, providing a theoretical basis for doing prediction and pose estimation. The matching is done by using validation gates. A measurement is only used if it falls inside a validation gate. The size of the validation gate depends on the uncertainty in the robot pose and the uncertainty in the measurements. These uncertainties are given by the Kalman filter.

In his thesis [33], Leonard also addresses the problem of simultaneous map building and localization. The correlation problem which Moutalier and Chatila addressed is solved by only updating the robot position with features which are confirmed and whose position are known accurately. The features are in the same manner only updated at locations where the position of the robot is known with high accuracy. By doing this, it is claimed that there is no need to explicitly represent cross-correlation between robot position and feature position. A drawback with the method, as pointed out in [1], is that the method require servo mounted sensors on the platform to initiate the algorithm, i.e. get the first good feature estimates which estimations of movements and positions of future features are based on.

Rencken

In [1], the problem of localization is defined as: “Given a set of measurements and a set of features, what is the robot’s position?”. The problem of map building is defined as: “Given the robot’s position and a set of measurements, what are the sensors “seeing”?”. In [68] Rencken says: “The main challenge lies in finding a generic method which is able to deal
with the bootstrapping problem of concurrent localization and map building in general”. The work takes off where Leonard’s thesis [33] ends, and tries to remedy some of its shortcomings. The main contribution is the extension of Leonard and Durrant-Whyte’s work regarding the initiation problem mentioned in the previous paragraph. Results are presented in [68] based on the implementation of the proposed method on the mobile robot ROAMER. The speed of ROAMER during the experiments is 12 cm/s and the duration of the tests is 20 minutes. The results show that ROAMER is able to keep the position error relatively small during the whole mission without any prior map information given.

Csorba and Durrant-Whyte

Another solution to the cross-correlation problem mentioned earlier in this section is presented by Csorba et al [69, 70]. The problem arises when the position of features are measured in an absolute frame of reference. The solution is thus to use a relative frame of reference, i.e. the position of the features are given only in relation to other features and not to some absolute frame of reference. The benefit of the method is that the computational burden is decreased, as the cross-correlation relations takes a lot of effort to update when using an absolute reference frame. One disadvantage which is pointed out [70], is that there is no good way to go from the relative map to a map in an absolute frame of reference, leading to a situation where all users of the map (e.g. for planning) must work in the relative frame of reference.

Wernersson

Wernersson [41, 71, 72, 73, 74] studies the problem of localization with several of his students. The basis for the research is the Range Weighted Hough Transform (RWHT), a modified version of the standard Hough Transform [75, 76] used extensively in the field of computer vision, but also for localization purposes by e.g. Crowley et al [77]. The Hough Transform is a kind of filtering technique which allows for parametric search of geometric features by a voting scheme where each data point votes for possible parameters. The parameter space is divided into bins and the best sets of parameters are found by searching for local minima in a tessellated parameter space. Wernersson et al uses the Hough Transform to robustly extract lines from 2D range data. The extension with using range weights in the Hough Transform is justified by the fact that
objects which are close to the sensor will accumulate more data from a scanning sensor which samples equidistantly in angle. Lines which are further away will thus be given less votes than a line which are close, but not as long. Therefore, assuming the number accumulated points to be proportional to the range, the vote of each data point is weighted with its range to the sensor.

In [41, 71] a localization scheme is developed where the pose of the robot is estimated in a rectangular room using a Kalman filter. The distance to the walls and the relative angle of the robot in relation to the walls are measured using RWHT. Passing through a door in one of the walls and following a corridor is given as experimental examples of the method. Problems are reported to be related to, e.g. data association.

The algorithm for localization is augmented in [72, 74] with the capability to build maps of the environment consisting of line features. The lines are extracted with RWHT, and a Kalman filter handles the updating of robot pose and feature poses estimates. The method is inspired by the work of Leonard et al [33], but a probability based algorithm for discarding false associations of extracted lines and line features in the model is introduced.

Weiβ et al

The angle histogram [64] was originally used for finding the main orientations of the walls in an unknown room, and for finding the position and orientation of a mobile robot in a known room. In [78] the angle histogram is used to find the relative movement between two consecutive scans by using a correlation technique. The angle histogram is well suited for this purpose, as it has good invariance properties regarding position and orientation changes of the robot.

In [42] the method is enhanced by storing reference scans from positions the robot has been to. This way the robot can keep track of the position and use the stored reference scans for localization when revisiting an area.

Wijk and Jensfelt

Wijk et al [25, 26, 27] presents a sonar based localization method using natural landmarks extracted through triangulation. Instead of looking at the wide sonar beam as a disadvantage in the sense of low angular resolution, the method utilizes this property to fuse sonar information.
The basic idea is to use triangulation to improve the quality of the sonar data. Sonar data typically suffers from, e.g. specular reflections and cross talk (see Section 2.2). To be able to get more accurate information, integration over time is necessary. Keeping in mind the physics of the sonar sensor, the information about the position of the target which reflected the sound is limited to knowing that it is somewhere on an arc\textsuperscript{1}. Assuming that the target which caused the reflection is “sticking out”, the point of reflection of the target will be almost the same even if the robot has moved. If it somehow can be established that it was the same target which gave rise to two reflections, triangulation can be used to find the true position of the target. The true position is given by the intersection of the circular arcs as illustrated in Figure 3.1. The accuracy in the triangulation depends on the angle to the target relative to the direction of robot motion, as well as the distance traveled between the two sonar readings. By doing the triangulation, outliers are filtered out and a better estimate of the true position of the target can be established. Triangulation can of course be made using more than two sonar readings. The number of sonar readings successfully triangulated can be taken as a measure of the quality of the triangulation and the probability that it really stems from a true target. The crux is of course to know which readings to use for the triangulation. A scheme called Triangulation Based Fusion (TBF) has been developed to solve this problem and

\textsuperscript{1}A 2D assumption has here been made about the world.
we refer to [25, 27] for more details.

Localization using artificial landmarks is well understood and reliable, but requires that the environment be engineered. When using natural landmarks for localization one of the problems is to find suitable landmarks, a process which often requires the designer of the algorithm to give geometrical specifications on the landmarks, e.g. points, lines, corners, etc. The TBF algorithm provides a method to automatically generate natural point landmarks from sonar data without any geometrical model of the landmarks. It puts requirement on the environment though. The environment must contain targets that are good, angularly stable, sonar reflectors, a requirement which is met in a normal office environment. The triangulation landmarks can be stored in a map and be used for localization [26, 27] describes such a technique which involves first building a map of the environment, i.e. collecting the strongest triangulation points as landmarks and storing them in a map. Later the robot can do pose initialization by building a local map of the environment based on natural landmarks and match it against the stored map.

Leonard et al [33] used so called geometric beacons to do localization, these beacons were extracted from densely sampled sonar data. The triangulation landmarks could potentially be used in a similar framework to provide means for doing pose tracking as well as pose initialization and map building.

Wallner

A method quite different from the others presented above, is given by Frank Wallner in his PhD thesis [79] (also presented in short in [80]), where principal components analysis is applied to laser range based localization. Principal component analysis techniques have been applied to, e.g. object recognition [80] before. In short, principal components analysis is about extracting the most important information from a large set of data by doing eigenvalue decomposition. By taking many scans from different positions in a room the characteristics of the scans (room) can be represented by a few so called eigenscans (eigenvectors). The most important information in the scans will be represented by the eigenscans corresponding to the largest eigenvalues. By only using the most significant eigenscans, a eigenspace with reduced dimensions is given. A single scan can be approximated by a linear combination of these eigenscans, given by projecting the scan on the eigenvalues. The more eigenscans used, the better the approximation is of the real scan.
To be able to use principle components for localization the mapping from pose to eigenspace must be determined, i.e. given the projection on the eigenscans, a probable pose for the robot should be given. It is important that the whole area be covered when building the eigenspace, so that every part is given equal weight when calculating the eigenscans. This would be a terribly cumbersome process to perform by hand. Therefore a method is presented for synthetic generation of scans taken at equidistant poses. First a set of scans are taken and the transformation between the scans are calculated using the correlation technique presented by Weiß et al [42], to give a composite scan in a common frame of reference. From this composite scan, the synthetic scans can be generated at arbitrary poses. The projection of each synthetic scan on the eigenscans are stored along with the corresponding robot pose, giving a mapping from pose to eigenspace. When using the map for localization, the robot takes a scan and projects it onto the eigenscans and then uses the mapping to determine the pose.

Normally, in a structured environment full of symmetries, a point in eigenspace corresponds to many points in the real world, giving many hypotheses of the pose. To solve this, a set of possible robot tracks are initiated. By assigning a probability to each track, based on how many hypotheses that are found to support it as the robot is moving, the true pose of the robot can be found after a number of iterations.

3.2.2 Grid Methods

The parametric methods have the disadvantage that an explicit model is needed for all the information which is used. Another thought is to divide the work space into a grid where each cell in the grid represents a part of the world. One advantage of this approach compared to the parametric is as Hager and Mintz [81] points out: "The grid-based method is only an approximate solution, but it is much less sensitive to assumptions about the particular form of the sensing system". Raschke and Borenstein [11] states that the grid-based model are better suited for dealing with sonar data than line-type models (features). Just as in the case of parametric methods we believe that example will illustrate the idea best.

Elfeś and Moravec

Moravec and Elfeś made the grid based techniques popular with their paper [12] in 1985. Here what would later be called the occupancy grid
is presented. As the authors say in the paper: "one range measurement contains only a small amount of information", which means that the only way to be able to build a map of the environment or find the location of the robot is by combining many readings. In the parametric setting, explicit models (feature models) are needed to fit the sensor data to, i.e. all sources of information must be modeled beforehand.

In the occupancy grid method, the world is rasterized into a grid. Each grid cell contains information about whether or not it is occupied. In their implementation, a sonar sensor is used as source of information, but other sensor can be used just as well. The sensor model presented in Section 2.2 is used to update the occupancy grid. The robot uses a stop-and-move strategy to explore the world. At each position sensor data is processed to update the world model and the model is used for planning the next move. The grid based methodology has great potential and has since then been used by numerous researchers.

Already in [12], but also in [21, 82, 83, 84], techniques for finding the position of the robot using the grid representation of the world is discussed by correlating a local map of the environment to a global one. The correlation is performed at different map resolutions to reduce the computational cost.

In Elfes’s thesis [83], a thorough description is given of the occupancy grid from the basics to more advanced applications of it, like for example the transformation between a geometric description of the environment to a grid description and vice versa. Results are also presented with other sensors than sonar and with multiple sensors.

Borenstein

Johann Borenstein has, by writing the book *Navigating Mobile Robots: Systems and Techniques* [3], become one of the most referenced researchers in the community. The research of Borenstein has mostly been focused on getting as much as possible out of the sonar sensor, with a lot of emphasis on obstacle avoidance and has to a large extend been performed in cooperation with Yoram Koren.

Among the disadvantages with the occupancy grid technique presented by Moravec and Elfes are a computationally heavy updating routine and the fact that the robot cannot move continuously. In [17, 85, 32, 38, 86] Borenstein and Koren apply a much simpler updating scheme of the grid, by only updating the cell which corresponds to the center of the

---

2The book is written together with H.R. Everett and Liqiang Feng
acoustical axis at the distance of the range reading. Instead of using a floating point values to represent the probability, integer values are used to measure certainty. The higher the certainty value a cell has, the more likely that the cell is occupied. A realistic probability distribution is still achieved by fast sampling. The computational burden is decreased significantly and the robot can also move during the map building. In [87] all cells on the acoustical axis are updated. The certainty value of the cell corresponding to the target is increased, whereas the certainty value of the cells in front are decreased.

As mentioned before, the main issue for Borenstein and Koren’s research is obstacle avoidance and therefore the certainty grid map is only local, consisting of a 33 by 33 cell grid (cell size 10x10 cm). Obstacle avoidance is achieved by combining the potential field method introduced by Khatib [88] and the certainty grid, by associating a virtual force to each occupied cell, in what is called the Vector Field Histogram technique [86].

**Schultz et al**

Schultz and Adams presents a localization system based on the occupancy grid in [89]. The method is the same as originally suggested by Elfes, i.e. a so called short-term perception map is build from sensor data and correlated to a global map to find the pose of the robot. Localization is performed not only when the odometric error grows too large or when a position error is detected, but more often to make the correlation problem easier to solve. Different methods to match the short-term map to the long-term map are evaluated.

In [90] Graves et al present a method to handle changing environments, i.e. making the long-term map adapt to the changing environment (the short-term map). Experimental results are presented which indicate an increase in performance when the long-term map is adapted. The increase is marginal though, from 10.02 inches error to 9.29 inches after a 50 minutes run.

Yamauchi et al [91] complement the continuous localization method presented above with a method for exploring the environment to build grid maps. The method is called Frontier-Based Exploration and is very similar to the virtual bubble approach presented by Edlinger and von Puttkamer [65] (see above). In short, it all boils down to finding edges between unknown and known area, i.e. parts of the grid which are occupied or empty and those which are unknown. The frontiers are explored
with a nearest-first approach.

Simmons

Few mobile robots can show so extensive experimental results as Xavier at CMU [92]. Xavier was designed to do research on delivery tasks for autonomous mobile robots. Xavier can be commanded via a World Wide Web interface to go to a set of given positions. This is all fully autonomous and Xavier had traveled a distance of 63 km autonomously, while completing 1467 missions, at the end of 1996 [92].

Xavier’s localization capabilities are based on Partially Observable Markov Decision Process models (POMDPs). Xavier’s movements are limited to one building. This building has been divided into a grid of possible positions. As the localization is only of interest on a gross scale, the building is tessellated with a cell size of 1x1m. Each position is also given four possible directions, i.e. each position corresponds to four possible poses. Each possible pose will form one possible state for the robot, and each state is given a probability that the robot is in that state. Probabilities are given to the transitions between states and probabilities are also given regarding detecting certain features in certain states.

Odometric information is received in the form of so called motion reports, these motion reports are given relative to the desired heading, when the robot has turned 90° or translated one meter, i.e. when the robot is predicted to change state. These motion reports will redistribute the probability of the robot’s position, effectively moving it in the direction of the motion report and blurring it due to uncertainty in the odometric information. Sonar sensor are used to build a local occupancy grid which is used for feature classification. Wall, door, closed door, open door, etc are examples of features which are used. Three so called sensor reports are delivered regarding features in front, to the left and to the right of the robot. These sensor reports will decrease the uncertainty about the robot’s pose by concentrating the probability distribution.

One obvious advantage of the approach is that every pose is considered as an potential robot pose, i.e. no hard decision has to be made regarding the pose of the robot and data association. Among the disadvantages are i) that it takes quite a lot of effort to design the POMDPs, ii) updating the probability is a computationally heavy operation.
Thrun

Sebastian Thrun et al present a solution inspired by the Xavier project in [93]. The state space of the robot is here much more densely sampled, the orientation of the robot is discretized in steps of 1° and the cell size is in the order of 10 cm in position. Instead of a occupancy grid this is referred to as a Position Probability Grid, i.e. each grid cell holds the probability that the robot is in the corresponding state (pose).

The localization problem is here divided into two parts, namely position tracking and global localization, but it is also stated that in reality it is the same problem, namely “determining the position of the robot under global uncertainty”. The map of the environment is given by an occupancy grid. Different methods for updating the position probability grid is discussed, e.g. landmarks, wheel encoders, sonar modeling and matching a local occupancy grid to the global one. Sonar modeling refers to a technique where the probability of the getting a certain range reading in a certain sensor direction and robot position is calculated. A neural network is used to interpret groups of sensor readings, and outputs local occupancy grids used for map matching. To solve the global localization problem, evidence is collected and the position probability grid is updated until a single pose state can be said to be the true one. It is a very costly process, as a single sensor reading will effect the whole probability distribution. In order to make it perform in real time, approximations are made. When doing position tracking the problem is simplified as only one position has to be considered in the beginning. The problem is here only to reliably track the change in pose relative to a known initial pose.

An extensive number of papers has been published on variation on the above by the authors, see e.g. [94, 54, 95, 96, 97, 98]. Some of the issues which have been discussed are the real-time performance problem and automation of the landmark selection process.

Schiele and Crowley

Schiele and Crowley [77] evaluate different methods for matching a local occupancy grid to a global one. The methods under consideration are to match i) local grid to global grid, ii) local extracted line segments to global grid, iii) local grid to global line segments and iv) local and global line segments. The line segments are extracted using the Hough Transform and the position of the robot is updated using a Kalman filter framework. The result of the evaluation is that it is best to match representations at that same level of abstraction, i.e. grid to grid or line segments to line
3.3 Summary

In his thesis, Elles [83] states: "The recovery of robust, coherent, and useful spatial models from sensor data is currently probably the single largest bottleneck in the development of autonomous robotic systems, and constitutes itself into one of the fundamental challenges in Robotics and Computer Vision". The statement was made about ten years ago, but it is still true.

Given this and the knowledge encompassed in the last two chapters, we now summarize the design criteria:

- Odometry is a resource that should be utilized to make the problem of localization simpler. The odometry is known to be very reliable over short distances under normal circumstances, and not using it when it is available is a waste of information. Care must be taken though, not to make the system rely too much on the odometry, giving a system that is very sensitive to non-deterministic odometric errors.

- As already stated in the beginning of Section 3 we are only interested in methods that need no engineering of the environment. This is so because methods that require the user to modify the environment will limit the applicability of the method. Much interest recently has been directed towards the Sojourner Rover project [99] and its successors, this is an extreme example of a case where environmental engineering is not practical. Other less extreme examples include systems operation in hazardous environments.

- Sonar sensors have been used extensively in localization researched. We too have developed a method based on sonar (see [25, 26, 27]). The sonar sensor has proven to be reliable and cheap, but it has clear limitations in its use. Due to the wide beam width it requires heavy post-processing to extract the important information. The experimental platform under consideration is equipped with a scanning laser sensor, a sensor that thus far has not been the focus of
as much attention as the sonar and its potential is not yet fully explored. We therefore let the laser sensor be the range sensor on which the method for localization, presented in this thesis, will rely.

- Much effort has been spent on modeling the world very accurately, to achieve better performance. We chose to turn this upside down and instead ask ourselves how simple can the map be and still be useful for solving the localization problem? What are the limitations of such a map? Therefore, one design criteria is to use as simple a model as possible.

- In order to achieve full autonomy in the system, it is vital that the environmental modeling task can be automated. By automated we here mean that the robot itself must be able to acquire a usable map of the environment. In many cases the modeling is performed by the user, but this is not realistic for a fully autonomous mobile robot. If a plan of the building under consideration is available, it could be used to build an environmental model. One requirement is then that the world model be simple enough so that the information can be acquired from the drawing. A case where this might not be practical is the model used in [25, 26, 27], where the model consists of natural point landmarks extracted using a triangulation technique.

- To use the laser sensor in an optimal way, an active sensing strategy is required. We propose that the method to be developed must perform satisfactory even without active sensing, as there might be other bidders for controlling the laser sensor, e.g. to pass through doors or servo on objects. This means that the pose tracking routine can not rely on being able to control the sensor at all times.

- The long term goal is to combine information from many sensors, to gain reliability from redundant information. But a first step towards that, is good understanding of the individual sensors.

With these design criteria in mind and with reference to the knowledge encompassed in the last two chapters, we propose the following plan of attack:

1. Develop a good understanding for the available odometric information. Questions that must be answered include for example what factors that effect the performance.
2. Characterize the performance of the laser sensor with respect to noise, beam width, etc, to gain good understanding for the sensor. This is necessary to be able to construct a reasonable sensor model.

3. Try the simplest model first. In a structured office environment it is not far fetched to try to model each room as a rectangle. This means that each room can be represented by two numbers, the length and the width of the room.

4. Trying to solve the whole localization problem is beyond the scope of this thesis, therefore the problem is broken down into components as follows:

   (a) Solve the pose tracking problem, as that capability is needed to be able to do environmental modeling. Pose tracking will also help the system to reach a high level of autonomy, allowing other parts of the whole system that rely on localization to be tested.

   (b) The pose tracking needs to be initialized at startup and when the robot has lost its pose. The startup initialization problem could be made easier by letting the robot start in the same pose every time. If the pose is lost during a mission though, pose initialization is the only possible solution, to insure robustness.

   (c) The environmental modeling capability must be automated to achieve full autonomy, but as a start it will be done by the user. Given that the initial attempt to model each room as a rectangle is successful, the modeling can be considered easy, only consisting in taking two measurements in each room, length and width.
Chapter 4

Characterization of the Odometry and the Laser Scanner

In order to be able to use information from different kinds of sensors, it is crucial that the characteristics of these sensors are known. This chapter will summarize the results that we have obtained in different tests, performed on the odometric system on our experimental platform, a Nomad200 from Nomadies, and on the SICK PLS laser scanner that is mounted on this platform, see Figure 4.1.

4.1 Characteristics of the Odometry on the Nomad200 Platform

As described in Section 2.1, odometry is a standard technique for measuring the change in pose of the robot. It is based on encoders measuring the rotation of the wheel axes. By knowing how the rotation of these axes map to translation and rotation of the platform, it is possible to calculate the change in pose. Because the mapping is not exact and because there are non-systematic effects coming in, like slippage, the odometric information will not be perfect. In most systems the odometry will show clear signs of systematic errors. These systematic errors can potentially be compensated for. Compensating for the non-systematic errors is much
Figure 4.1: The SICK laser scanner is placed on top of a Nomad200 platform equipped with an array of sensors.

harder. Borenstein presents a solution to the problem of compensating for non-systematic errors in [3] where two coupled platforms are used. By using information from both platforms and information about the angle of the link between the two platforms, it is possible to compensate for some of the non-systematic errors. Such a solution is not possible here as we only have one base. Borenstein also points out that large errors typically occur during very short time intervals [6]. By being able to detect these events, it is possible to ignore the data from the odometry during that time and instead get the information from another sensors, like an inertial sensor. The Nomad200 is a synchro drive platform and as such expected to have a well performing odometry [3, p. 109].

Several tests are presented in the following sections to characterize the performance of the odometry on the Nomad200 platform. The results will lead us to a model for the information given by the odometry, including a noise model. This model will be used when estimating the change in the robot’s pose, based on odometric information. The model will be on
the form
\[ x(k + 1) = x(k) + g(k) + w(k) \]  \hspace{1cm} (4.1)

where \(g(k)\) is the odometric pose change information\(^1\) and \(w(k)\) is the corresponding noise. The noise is assumed to be white, zero mean and Gaussian with covariance matrix
\[ E[w(k)w(k)^T] = Q(k). \]  \hspace{1cm} (4.2)

These assumptions are in reality not true, but similar assumptions are made in the literature with good results. Figure 4.2 illustrates the relation between the global and the platform fixed coordinate system. The robot fixed coordinate system will thus translate with the robot. We will in the following understand all coordinates to refer to the world coordinate system if not otherwise stated. To separate the two when needed we will use index \(W\) for the world coordinate system and \(R\) for the robot fixed coordinate system. It should be pointed out that the world coordinate system might not be the same at all time. As will be evident in chapters to come each room will have its own world coordinate system.

The Nomad200 platform is, as said before, a synchro drive platform and the base of the platform should not rotate when the platform is moving. That is, under ideal conditions the angle \(\theta\) should be fixed, meaning that the third element of \(g\), \(\Delta \theta = 0\). Due to drift that will be discussed later, the orientation of the base will drift when the platform is moving. What we need to find out is if there is a systematic drift which can be modeled as \(\Delta \theta \neq 0\), or if it should be captured in the noise, \(w\).

\(^1\)\(g(k) = (\Delta x, \Delta y, \Delta \theta)^T\)
4.1.1 University of Michigan Benchmark Test (UMBmark)

Borenstein and Feng has devised a method for finding the systematic errors in the odometry. The test is called the University of Michigan Benchmark test (UMBmark). The test was designed as a result of the confusion that existed regarding comparisons of odometric performance, as no standard measures were available. The test is very simple, but requires a certain amount of free space. It consists of driving the robot in a 4x4m square. Ideally the robot should end up at the starting point, but due to errors this is not true in general. To be able to account for systematic errors that might cancel each other when driving in a certain direction around the square, the test is performed in both directions. 5 tests are performed in each direction. Looking at the center of gravity for the x,y-errors in each direction, clockwise (cw) and counter clockwise (ccw), four coordinates \((x_{e.g.\,cw}, y_{e.g.\,cw}, x_{e.g.\,ccw} \text{ and } y_{e.g.\,ccw})\) can be identified, two for each direction. The offset from the origin for the two centers of gravity are denoted by \(r_{e.g.\,cw}\) and \(r_{e.g.\,ccw}\). A single valued performance index is taken to be the largest absolute offset of the two centers of gravity from the origin, i.e. \(\max(r_{e.g.\,cw}, r_{e.g.\,ccw})\).

The UMBmark will only determine the size of the systematic error, but will not provide evidence regarding the non-systematic errors. A test called the extended UMBmark is presented in e.g. [3]. The extended UMBmark consists of one standard UMBmark test and a similar test were artificial bumps are introduced along the path. A synchro drive robot is less sensitive to these kinds of disturbances then a differential drive robot as it has three or more wheels, which gives better ground contact and thereby improved odometric information.

<table>
<thead>
<tr>
<th>direction</th>
<th>(x_{e.g.}) [mm]</th>
<th>(y_{e.g.}) [mm]</th>
<th>(e\theta) [deg]</th>
<th>(r_{e.g.}) [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>cw</td>
<td>238</td>
<td>-279</td>
<td>3.8</td>
<td>367</td>
</tr>
<tr>
<td>ccw</td>
<td>145</td>
<td>65</td>
<td>5.6</td>
<td>139</td>
</tr>
</tbody>
</table>

Table 4.1: The results of the UMBmark test on the Nomad200 platform.

The full description of the results can be seen in Table 4.1, from which it is clear that the error is much larger when driving in the clockwise direction than in the counter clockwise direction. \(e\theta\) stands for the rotational error. The performance of the odometric system can thus be said to be 367 mm. The performance of the platform is comparable to that of a
4.1 Characteristics of the Odometry on the Nomad200 Platform

TRC LabMate which is reported to have 310 mm in [3]. A synchro drive platform from Cybermotion is reported to have a performance of 63 mm. The TRC LabMate is a differential drive platform and we would expect it to have much worse odometric performance than our synchro drive platform. The fact that we get a performance of 367 mm indicates that we have to expect bad performance.

The results can also be illustrated in graphical form as in Figure 4.3. It should be noted that due to limited space the length of the sides of the square is only 3.5 m.

Figure 4.3: Left: Graphical illustration of the results of the UMBmark test. The x’s marks the \((x_{cw}, y_{cw})\) and the o’s the \((x_{cw}, y_{cw})\). Right: A possible path for the robot given that all the rotational error is caused by steering slightly more than 90°.

4.1.2 Rotation Error Caused by Steering

In the process of trying to determine the sources of the systematic errors we devised a test to investigate the effect that turning the wheels has on the rotation of the platform. It is a synchro drive platform, where the platform itself should not rotate when rotating the wheels. The test can be described as:

1. Place the robot in front of a wall at a known distance, \(d\).
2. Measure the orientation of the platform.
3. Turn the wheels 360° slowly, not to let slippage and other effects influence the result unnecessarily.
4. Measure the orientation of the platform and calculate the rotation caused by the turning.

Step 2 can be removed if instead the rotation can be measured directly by some means. In our test setup we used a laser pointer to point at the wall. Knowing the distance to the wall and the displacement of the laser point, \( l \), the rotation can be approximated by \( \Delta \theta_s = \arctan \left( \frac{\Delta s}{l} \right) \approx \frac{\Delta s}{l} \), assuming the laser is pointing approximately perpendicular to the wall. The results are presented in Tables 4.2. As seen in Tables 4.2 the rotation

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{deg} & \text{mm} & \text{deg} & \text{mm} & \text{deg} & \text{mm} \\
\hline
\Delta \phi & l & \Delta \theta_{\phi, \text{cw}} & l & \Delta \theta_{\phi, \text{cw}} & l \\
\hline
+360 & -55 & -0.5252 & -360 & +45 & +0.4297 \\
+360 & -3 & -0.0286 & -360 & +38 & +0.3629 \\
+360 & -32 & -0.3056 & -360 & +45 & +0.4297 \\
+360 & -25 & -0.2387 & -360 & +23 & +0.2196 \\
+360 & -25 & -0.2387 & -360 & +49 & +0.4679 \\
+360 & -30 & -0.2865 & -360 & +16 & +0.1528 \\
+360 & -5 & -0.0477 & -360 & +36 & +0.3438 \\
+360 & -43 & -0.4106 & -360 & +30 & +0.2865 \\
+360 & +3 & +0.0286 & -360 & +45 & +0.4297 \\
+360 & -40 & -0.3820 & -360 & +30 & +0.2865 \\
\hline
\text{Average} & -25.5 & -0.25 & \text{Average} & 35.7 & +0.34 \\
\hline
\end{array}
\]

Table 4.2: Results in rotation of the platform when turning the wheels a full \( 360^\circ \) in both directions.

of the platform will be in the opposite direction of the rotation of the wheels. When turning the wheels in the positive direction, i.e. counter clockwise, some of the measurements seem to be outliers. If the second, the seventh and the ninth values are removed, i.e. classified as outliers, the average \( \Delta \theta_{\phi, \text{cw}} \) becomes \(-0.35\). This is very much like the average value when turning clockwise (0.34). A reasonable approximation is thus that the rotation of the platform caused by a rotation of the wheels, can be modeled as

\[
\Delta \theta_s = -C_{\theta, \phi} \Delta \phi, \quad C_{\theta, \phi} > 0,
\]

where the constant \( C_{\theta, \phi} \) is approximately \( \frac{0.35}{360} \approx 0.0009 \). The standard deviation in this modeled to be of the same size as the drift. Even though most of the measurements fall in a much tighter region, it is important
not to make the odometric model too optimistic as the features may be lost. The noise in the odometric model is also intended to capture at least part of the non-systematic errors. Given that the size of the uncertainty is of the same size as the drift, $\sigma_{\delta, n} \approx 0.001$.

### 4.1.3 Rotation Error Caused by Translation

Our Nomad200 has a drift in the orientation of the platform during normal translation, i.e. the orientation of the platform is effected not only by turning the wheels as shown in Section 4.1.2, but also by simple translation. We model this as

$$\Delta \theta_D = C_{\delta, D} D$$

(4.4)

where $D$ is the distance traveled. A algorithm based on the use of a laser scanner is devised as follows:

1. Place the robot at the starting position, with the wheels in a direction where it is possible to drive for a long distance, a corridor or such. Aligning the wheels will minimize the amount of steering during the test, as steering was found to make the platform rotate. Mark the position of the robot in order to be able to measure the distance traveled later. Note the direction of the laser in relation to the platform so that it can be replicated at the end point. Take a laser scan and determine the angle of the wall relative to the platform, $\alpha_1$ by looking at the data. Fitting a line between two points on the wall is accurate enough.

2. Let the robot move automatically or by hand in as straight a line as possible to minimize the effect of steering, but also to make the approximation of distance traveled more accurate.

3. Take another scan and once again calculate the angle of the wall relative to the platform, $\alpha_2$. Be sure that the sensor is facing in the same direction relative to the platform as when the first scan was taken.

4. Measure the distance traveled, $D$.

5. The rotation per traveled meter can then be calculated according to

$$C_{\delta, D} = \frac{(\alpha_2 - \alpha_1)}{D}.$$ 

(4.5)
where the minus sign in the numerator make the change be that of the platform and not the wall.

![Diagram of CVAP building](image)

**Figure 4.4:** The first floor of the CVAP building.

The test is performed from one side to the other in the corridor, see Figure 4.4. In the first test the start position is the right corner of the corridor in Figure 4.4 and the end position is at the mailbox side. The robot is commanded to drive forward. The orientation of the platform is shifted approximately $6.7^\circ$. The distance traveled is $D = 46.2$ m, which gives $C_{\theta, D} \approx 0.14^\circ$/m. In the next test we drive the robot backwards. Here we get $C_{\theta, D} \approx 0.56^\circ$/m.

Further tests are needed to be able to statistically characterize the behavior. Table 4.3 show the result of these tests. The values differ much from one test to the other. Getting an accurate model for this behavior, only using the distance traveled as an input, will be difficult. We settle for capturing the trend, i.e. that the platform will rotate when

<table>
<thead>
<tr>
<th>$D$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\Delta \theta_D$</th>
<th>$C_{\theta, D}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.2</td>
<td>87.63</td>
<td>80.99</td>
<td>6.65</td>
<td>0.14</td>
</tr>
<tr>
<td>46.2</td>
<td>90.24</td>
<td>73.49</td>
<td>16.74</td>
<td>0.36</td>
</tr>
<tr>
<td>46.2</td>
<td>90.00</td>
<td>72.82</td>
<td>17.18</td>
<td>0.37</td>
</tr>
<tr>
<td>46.2</td>
<td>88.97</td>
<td>68.85</td>
<td>20.12</td>
<td>0.44</td>
</tr>
<tr>
<td>-46.2</td>
<td>90.67</td>
<td>116.53</td>
<td>-25.85</td>
<td>0.56</td>
</tr>
<tr>
<td>-46.2</td>
<td>90.00</td>
<td>121.81</td>
<td>-31.81</td>
<td>0.69</td>
</tr>
<tr>
<td>-46.2</td>
<td>92.47</td>
<td>96.84</td>
<td>-10.68</td>
<td>0.09</td>
</tr>
</tbody>
</table>

**Table 4.3:** Table showing the results when translating in the corridor.
it translates. We model the drift with $C_{\theta,D} = 0.35$ °/m. Using the same reasoning as above (Section 4.1.2) we set the uncertainty to be of the same magnitude, i.e. $\sigma_{C_{\theta,D}} = 0.35$ °/m.

4.1.4 Scaling

Another source of systematic errors is the wheels having different radius than assumed in the model. Borenstein et al [3] presents the model for the translation of the platform to be:

$$D_{od} = \frac{2\pi N}{C_r} R,$$  \hspace{1cm} (4.6)

where $D_{od}$ is the estimated translation based on the number of encoder counts ($N$) registered, the number of encoder counts per wheel revolution ($C_r$) and the wheel radius ($R$). Errors in the estimation of the wheel radius may result from deformations due to changing payloads [3]. From Equation (4.6) it is clear that the error in estimating the translation of the platform will be proportional to the error in the radius. We assume the true distance traveled by the robot to be given by

$$D = C_D D_{od},$$  \hspace{1cm} (4.7)

where $C_D$ is a scale factor. If the error in measuring the distance traveled only depends on an error in measured wheel radius the scale factor should be $C_D = \frac{R_{true}}{R}$. The simplest way to find this scaling factor is to let the robot drive a known distance. The scaling factor can then be found by looking at ratio between the known distance and the distance measured by odometry.

For an accurate measurement of the constant, $C_D$, it is best to use a long distance. However, we now know that we can expect the robot to drift in orientation, i.e. the local coordinate system of the robot will drift. Simply commanding the robot to follow a line in the local coordinate system will result in an arc in the global reference frame. Therefore, if we want the robot to drive along a straight line we have to compensate for the drift, i.e. we have to steer the robot to keep it going straight. Hence let the robot drive parallel to a wall in the corridor at a constant distance. We use the laser sensor to extract a line from the corridor wall from which the distance is calculated. The odometry is sampled at approximately 3 Hz and $D_{od}$ is calculated based on consecutive odometry readings. Using the information about the angle of the line (wall) we then project $D_{od}$
in the direction of the corridor and sum these increments. By letting the robot drive a known distance we can calculate the corresponding scaling factor to be $C_D = 1.006$. In experiments where the robot is commanded to drive much shorter distances, for which the drift can be neglected, we get the results shown in Table 4.4. In these short distance experiments

<table>
<thead>
<tr>
<th>[m]</th>
<th>D</th>
<th>[m]</th>
<th>$D_{od}$</th>
<th>[-]</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2554</td>
<td></td>
<td>2532.38</td>
<td></td>
<td>1.00854</td>
<td></td>
</tr>
<tr>
<td>2556</td>
<td></td>
<td>2532.38</td>
<td></td>
<td>1.00933</td>
<td></td>
</tr>
<tr>
<td>9190</td>
<td></td>
<td>9110.98</td>
<td></td>
<td>1.00867</td>
<td></td>
</tr>
<tr>
<td>9137</td>
<td></td>
<td>9110.98</td>
<td></td>
<td>1.00505</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Results of short distance experiments to determine $C_D$.

the measurement accuracy is lower. The trend is clear though, $C_D > 1$. In the same spirit as before, we model the uncertainty to be slightly larger than expected and use, $\sigma_{C_D} = 0.05$, to capture some of the non-systematic errors, such as slippage in this case.

4.1.5 Odometric Model

Given the results of the tests presented above, a model for the odometry can be designed. The odometry is only used to measure change in pose. With the notation introduced above, updating the position of the robot in a global frame of reference follow:

$$D_{od} = \sqrt{(\Delta x_{od})^2 + (\Delta y_{od})^2}$$

$$\alpha = \arctan(\Delta y_{od}/\Delta x_{od})$$

$$\dot{D} = C_D D_{od}$$

$$\dot{x} = \dot{D} \cos(\dot{\theta} + \alpha)$$

$$\dot{y} = \dot{D} \sin(\dot{\theta} + \alpha)$$

$$\dot{\theta} = \dot{\theta}_D + \dot{\phi} = C_{\phi,D} \dot{D} + C_{\phi,\phi} \Delta \varphi$$

That is, the deterministic part of Equation (4.1), $\mathbf{g}$, is modeled to be:

$$\mathbf{g} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{D} \cos(\dot{\theta} + \alpha) \\ \dot{D} \sin(\dot{\theta} + \alpha) \\ C_{\phi,D} \dot{D} + C_{\phi,\phi} \Delta \varphi \end{pmatrix}$$
To complete the odometric model, we need to known how errors will propagate. The uncertainty in distance traveled and orientation

\[
\sigma_T^2 = (\sigma_C \delta \hat{D})^2
\]
\[
\sigma_{\theta}^2 = (\sigma_{C_{\theta}} \delta \hat{D})^2 + (\sigma_{C_{\phi}} \Delta \phi)^2 + (C_{\theta \phi} \sigma_D)^2
\]
\[
= \hat{D}^2 \left( \sigma_{C_{\theta}}^2 + C_{\theta \phi}^2 \sigma_D^2 \right) + (\sigma_{C_{\phi}} \Delta \phi)^2
\]

will transform into uncertainties in \( x \) and \( y \) according to (using a first order approximation)

\[
\sigma_{x,\Delta}^2 = \hat{D}^2 (\sigma_{C_{x}}^2 \cos^2(\theta + \alpha) + \sigma_{\theta}^2 \sin^2(\theta + \alpha))
\]
\[
\sigma_{y,\Delta}^2 = \hat{D}^2 (\sigma_{C_{y}}^2 \sin^2(\theta + \alpha) + \sigma_{\theta}^2 \cos^2(\theta + \alpha))
\]
\[
\sigma_{\theta,\Delta\theta}^2 = \hat{D}^2 (\sigma_{C_{\theta}}^2 + C_{\theta \phi}^2 \sigma_D^2) + (\sigma_{C_{\phi}} \Delta \phi)^2
\]
\[
\sigma_{y,\Delta\theta}^2 = \hat{D}^2 \sigma_{C_{y}}^2 C_{\theta \phi}^2 \cos(\theta + \alpha)
\]
\[
\sigma_{x,\Delta\theta}^2 = \hat{D}^2 \sigma_{C_{x}}^2 C_{\theta \phi}^2 \sin(\theta + \alpha)
\]

The covariance matrix, \( Q \) is thus

\[
Q = \begin{pmatrix}
\sigma_{x,\Delta}^2 & \sigma_{y,\Delta}^2 & \sigma_{\theta,\Delta\theta}^2 \\
\sigma_{x,\Delta\theta}^2 & \sigma_{y,\Delta\theta}^2 & \sigma_{\theta,\Delta\theta}^2 \\
\sigma_{x,\Delta\theta}^2 & \sigma_{y,\Delta\theta}^2 & \sigma_{\theta,\Delta\theta}^2
\end{pmatrix}
\]

\[
= \hat{D}^2 \begin{pmatrix}
(\sigma_{C_{x}}^2 \cos^2(\theta + \alpha) + \sigma_{\theta}^2 \sin^2(\theta + \alpha)) \sigma_{C_{x}}^2 \cos^2(\theta + \alpha) + \sigma_{\theta}^2 \sin^2(\theta + \alpha) \\
(\sigma_{C_{y}}^2 \sin^2(\theta + \alpha) + \sigma_{\theta}^2 \cos^2(\theta + \alpha)) \sigma_{C_{y}}^2 \sin^2(\theta + \alpha) + \sigma_{\theta}^2 \cos^2(\theta + \alpha) \\
(\sigma_{C_{\theta}}^2 \cos^2(\theta + \alpha) + \sigma_{\theta}^2 \sin^2(\theta + \alpha)) \sigma_{C_{\theta}}^2 \cos^2(\theta + \alpha) + \sigma_{\theta}^2 \sin^2(\theta + \alpha)
\end{pmatrix}
\]

\[
+ \Delta \phi^2 \begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \sigma_{C_{\phi}}^2 \Delta \phi^2
\end{pmatrix}
\]

where \( c_{\theta} \) and \( s_{\theta} \) are short for \( \cos(\theta + \alpha) \) and \( \sin(\theta + \alpha) \) respectively.

### 4.2 Characteristics of the PLS Laser Scanner from SICK

The PLS 200 from SICK Electro-Optics is an example of a TOF laser scanner (see Section 2.3). It can provide 180° scans of the environment.
at a rate of 2.5 Hz using a rotating mirror. Each scan thus takes 40 ms, out of which 20 ms provide data and the next 20 ms is just rotating the mirror. Due to hardware limitations (max serial speed 38.4 kBaud), we can only use a maximum sampling rate of \( \approx 3 \text{Hz}. \) The angular resolution is 0.5°, resulting in 361 range measurements per scan. The sensor is placed approximately 930 mm above ground on the platform to scan a horizontal plane, as shown in Figures 4.1 and 4.5.

![Figure 4.5: The laser scans the environment in a plane.](image)

### 4.2.1 Quantization

The sensor provides a polar range map of the form \((\alpha_k, r_k)\). \(\alpha_k\) is the angle, 0° - 180° discretized into 0.5° bins \((2 \Delta \alpha = 0.5^\circ)\). The angle of \(\alpha_k\) is thus given as

\[
\alpha_k = k \times 0.5^\circ, \quad k = 0, \ldots, 360.
\] (4.10)

For each angle \(\alpha_k\), there is a range measurement, \(r_k\). To be able study the behavior of the sensor in more detail we perform an experiment where the sensor is placed in front of a wall and data is collected for different distances. It is clear that, as the data is discretized with a step of 50 mm, the range readings all end up on circular arcs, separated by 50 mm. This means that given one range reading, \(r_{base}\), all other values can be written as

\[
r_k = r_{base} + i \cdot 2\Delta r, \quad i \in \mathbb{Z}, \quad k = 0, \ldots, 360.
\] (4.11)

Figure 4.6 show a sequence of scans taken with the robot moving back from the wall approximately 10 mm between the scans. The first scan (top left) is taken with the robot approximately 5450 mm from the wall.
Figure 4.6: The robot starts \(\approx 5450\) mm in front of a wall (top left corner). Between each scan the robot moves \(\approx 10\) mm away from the wall. The data is quantized and is restricted distances \(r_{\text{base}} + i\Delta r\).

We see that the reading in the center region end up giving values between 5450 and 5500 mm. As the robot move away from the wall the readings move discontinuously out on the half circles to increasingly larger radii, making jumps of 50 mm. When the distance to the wall is in between two range levels, the range measure will vary between the two levels due to noise.

The factor \(r_{\text{base}}\), specifying the absolute location of the quantization levels is not a constant. It too is subject to small variations, as is seen in Figure 4.7. When \(r_{\text{base}}\) changes to \(r'_{\text{base}}\) all other measurements follow to give range measurements \(r'_{\text{base}} + i\Delta r\). The mid figure in Figure 4.7 we see that \(r_{\text{base}}\) has decreased with 10 mm.

In some situation we also observe that individual measurements change, i.e. Equation (4.11) no longer holds. Figure 4.8 shows this individual change for a number of examples. From the examples it seems that this individual chatter is limited to the center most readings, but this is not true in general.
Figure 4.7: The factor $r_{base}$ is not constant. It changes level on occasions, resulting in a new set of possible range readings.

Figure 4.8: Sometimes individual measurement might deviate, i.e. give values that do not match those allowed by Equation (4.11).

For the moment we chose to neglect the fact that individual measurements may chatter. Using this assumption we model the measurements as being uniformly distributed over each quantization bin, both in angle and range.

4.2.2 Footprint Size

The laser energy propagates in a cone, just like the ultrasonic energy. The difference is in the beam width. The beam width of the laser is less than a degree whereas the standard Polaroid sensor has a full beam width of approximately 25° (see Section 2.2).

The footprint of the laser sensor is the shape that the laser beam has when it hits an object. To be able to determine the beam width and hence the size of the footprint, we performed a series of experiments. The laser sensor is placed approximately 4.7 meters away from two wooden boards. The range measured by the center laser beam, i.e. $r_{180}$, is studied. By sliding the boards perpendicular to that laser beam and monitoring the measurement the size of the beam can be estimated. Figure 4.9 illustrates the experimental setup. When the boards are not detected, the measured distance will be the distance to the wall behind.
Figure 4.9: The footprint of the laser beam is determined by sliding two boards towards each other with the laser beam in between.

**Phantom Measurements**

In the first experiment we use only one of the boards. The idea is to look at what happens when the boards in near the border of the beam. Introducing the coordinate $x$ as a measure of the board’s position perpendicular to the beam, Table 4.5 show the resulting measurements. As

<table>
<thead>
<tr>
<th>$x$ (mm)</th>
<th>$r_{180}$ (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5450</td>
</tr>
<tr>
<td>1</td>
<td>5450</td>
</tr>
<tr>
<td>2</td>
<td>5450</td>
</tr>
<tr>
<td>3</td>
<td>5450</td>
</tr>
<tr>
<td>4</td>
<td>5050-5450</td>
</tr>
<tr>
<td>5</td>
<td>4900</td>
</tr>
<tr>
<td>6</td>
<td>4950</td>
</tr>
<tr>
<td>7</td>
<td>4850</td>
</tr>
</tbody>
</table>

**Table 4.5:** When the laser beam is split between two objects at different distances, the resulting range measurement will be somewhere in between the distances to the two objects.

...can be seen from the table the range measurements do not jump discon-
timously from the wall distance to the board distance. When the board is at position \( x = 4 \) the range measurements jump over a range of 400 mm, giving measurements in between 5050 mm (in between) and 5450 mm (wall). In the range \( x \in [4, 11] \) the measurement are in between wall and board. This range corresponds to an angle of approximately \( \frac{7}{360} = 0.0015 \text{ rad} = 0.085^\circ \), i.e. using a wooden board the laser beam has a region of \( \approx 0.9^\circ \) on each side that will give spurious measurements.

### 4.2.3 Beam Size

To be able to determine the size of the beam we use the two boards, sliding them towards each other. By finding the largest possible separation that results in a range measurement corresponding to the distance to the boards, the size can be estimated. The result is that the board can be separated approximately 20 mm. This separation at a distance of 4.7 m is equivalent to a beam width of approximately \( 0.25^\circ \). The size of the laser beam, that is strong enough to give the correct correct measurement, is thus \( 0.25^\circ \). The size of the beam that is capable of detecting that there is something in front of the wall, but not correctly measure the distance, is approximately \( 0.43^\circ \). As the laser beams are separated by \( 0.5^\circ \), there is a blind region between the beams.

### 4.2.4 Sensor Model

If we, as above, neglect the fact that individual measurements sometimes chatter, the uncertainty of the laser sensor data can be modeled as being uniformly distributed over the ranges \([-\Delta r, \Delta r]\) mm and angles \([-\Delta \alpha, \Delta \alpha]\), respectively. Converting the polar data into a Cartesian frame of reference we obtain:

\[
\mathbf{r} = \begin{bmatrix} a \\ b \end{bmatrix} = r \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix}.
\]  

(4.12)

The density functions can be written as

\[
f_r(r) = \begin{cases} \frac{1}{2\Delta r} & r \in [\bar{r} - \Delta r, \bar{r} + \Delta r] \\ 0 & \text{otherwise} \end{cases} 
\]  

(4.13)

\[
f_\alpha(\alpha) = \begin{cases} \frac{1}{2\Delta \alpha} & \alpha \in [\bar{\alpha} - \Delta \alpha, \bar{\alpha} + \Delta \alpha] \\ 0 & \text{otherwise} \end{cases} 
\]  

(4.14)
where \( \bar{r} \) and \( \bar{\alpha} \) are the mid values of the discretization bins in range and angle respectively. The variance in the \( a \) and \( b \) directions, respectively, can be derived from straight forward calculations assuming that the uncertainty in \( r \) and \( \alpha \) is independent. The result being:

\[
\sigma_{aa} = \frac{1}{2}(\bar{r}^2 + \frac{1}{2} \Delta r^2) \left(1 + \cos(2\bar{\alpha}) \cos(\Delta\alpha) \frac{\sin(\Delta\alpha)}{\Delta\alpha}\right) \quad (4.15)
\]

\[
\sigma_{bb} = \frac{1}{2}(\bar{r}^2 + \frac{1}{2} \Delta r^2) \left(1 - \cos(2\bar{\alpha}) \cos(\Delta\alpha) \frac{\sin(\Delta\alpha)}{\Delta\alpha}\right) \quad (4.16)
\]

\[
\sigma_{ab} = \frac{1}{2}(\bar{r}^2 + \frac{1}{2} \Delta r^2) \cos(2\bar{\alpha}) \cos(\bar{\alpha}) \frac{\sin(\Delta\alpha)}{\Delta\alpha} \quad (4.17)
\]

In our localization experiments we settle for using a, mathematically, much simpler model of the uncertainty. We assumed the uncertainty to have an independent Gaussian distributions in the \( a \) and \( b \) directions with standard deviations of 50 mm each. This a very crude model, but one that will prove to be adequate in experiments.

### 4.3 Summary

In this section we have characterized the performance of the odometry and the laser scanner, leading to models for these sensor modalities.

It was found that the performance of the odometry was approximately 370 mm, using the UMBmark with a reduced square path (3.5 m × 3.5 m). By comparing this to other reported systems, we can see that the odometry our Nomad200 platform is relatively poor. At the same time we must acknowledge that the odometry is very good over short distances, that is when moving only in the order of 1 m. This indicates that the odometry is a good source of information for doing short time predictions about the change in pose.

The data from the laser scanner was found to be discretized in steps of 50 mm. Due to noise the range readings are subject to a certain amount of perturbations, that is the measurements jump between two possible discretization levels. We have also observed phantom measurements when the laser beam is split between two surfaces at different distances, meaning that the measured distance will end up somewhere in between the
two surfaces. This effect could potentially be very important to keep in mind when estimating certain features, such as jump edges. While the laser scanner has a lower resolution than odometry, it has a bounded uncertainty and therefore the combination is ideal for localization.
Chapter 5

Pose Tracking

By pose tracking we understand, as was previously stated, the process of keeping track of the pose of the robot while it is moving in the environment, given that the initial pose is known. We will in this chapter assume the initial pose to be known. Later, in the next chapter, we will look at how to automate the initialization process.

First we recapitulate the design criteria that will effect the choice of method for pose tracking.

- We should use odometry to simplify the localization process, but the algorithm must not be too dependent on the odometric information. Chapter 4 provides us with a model for the odometry.

- We are not allowed to use artificial landmarks and any other form of environmental engineering.

- We want a simple world model. In a typical office or home environment most rooms are rectangular. A rectangle only requires two number to be represented, the length and the width, thereby being a very simple model. Hence the initial attempt will be to use a rectangular model of each room.

- We should consider how to incorporate active sensing, but in the first version it need not be implemented. Fair performance should be achieved without active sensing, to guarantee robustness.

The rectangular model we propose to use is a simple special case of a line based model. In Chapter 3 we found that models based on line segments
are quite common. Both parametric methods and grid methods have been used in conjunction with line-based models. We make no assumption as to how cluttered to rooms will be, hence the method must be able to handle a fair amount of clutter. The rectangular model is not a very rich description of a room. Using such a simple model we hope to be able to realize a low-complexity pose tracking routine. If that is not possible, a more complex model of the environment might actually prove to reduce the complexity. We believe that a parametric method is the best way to fully utilize the simplicity of the model. A grid-based method would involve either matching heterogeneous data (not a good way according to [77]), or to transform either the global line model to a grid model or to extract lines from the local grid. We believe that the parametric approach is better suited for capturing the simplicity of the model.

5.1 Theory

Given that we know the initial pose of the robot, a good approximation of the robot at every time instant can be calculated using odometric information. Knowing the approximate pose of the robot would also make the data association problem considerably easier. Laser scanner data that is likely to belong to the walls (the sides of the rectangle model) can be filtered out effectively.

There are many ways to mathematically handle a problem of this nature. It is clear that it is an estimation problem and in some sense an optimization problem. How do we incorporate the information that we get from the sensors, both odometry and laser scanner, to calculate the best possible estimate of the pose? In order to answer this we need to specify, best in what sense? As is often the case, we chose to define best in the least square sense. The problem is then to, in as good a way as possible, fit the data from the odometry and the laser scanner to get the least square error between estimate and data. We have already in Chapter 4 learned how the pose of the robot can be predicted using odometry. The characteristics of the laser scanner are also known from Chapter 4 and hence we stand a good chance to use some of the standard techniques available for estimating the pose. We will assume that the probability density function for the robot’s pose can be represented by a unimodal function, i.e. one having a single maxima. This puts high demands on the data association. In certain situations a large uncertainty in the pose of the robot might introduce ambiguities when associating
data, leading to a need for a multi-modal description of the probability distribution.

The robot only needs to search for walls in the room corresponding to its current location, i.e. no attention has to be paid to walls in other rooms or any other features but walls for that matter. This means that there will only be four features at most to track at each time instant. Under normal conditions there will in fact be only two or three features that are in the field of view of the sensor (see Figure 5.1). Compared to many other approaches, four features to be tracked must be considered a small number.

**Figure 5.1:** Normally only two or three walls are in the field of view of the laser scanner.

By tracking the walls, an estimate of the pose of the robot can be maintained. The prediction made by odometry will be corrected using information from sensor data. In order to robustly find a parametric descriptions of the walls, we need to solve the data association problem. This is possible by using the approximate information about the pose of the robot, relying on odometry for short time predictions.

## 5.2 Algorithm

In order to perform the least square estimate of the robot pose in real-time we need a recursive algorithm. The alternative would be to keep all the information in a batch and do a complete calculation at every time instant. The price we pay for real-time performance is that once a piece of information has been used and is incorporated into the estimate of the pose, it is lost forever. That is, we cannot go back and reconsider a decision concerning data association for example. Our decision will be
final and therefore it is of vital importance that the number of error in data association be kept to a absolute minimum. As we cannot guarantee perfect data association the robot also needs to be able to recover from errors in the pose estimate. This will be dealt with in the next chapter.

We will use the Kalman filter framework that has been used by numerous researchers (see e.g. [22, 56, 71, 68, 31, 80]) and proven to provide a good setting for sensor fusion. The Kalman filter will give an optimal estimate of the pose given the information at hand, assuming that the model of the system is correct and that all sources of noise are Gaussian. The Gaussian assumption is normally not fulfilled, which yields a sub-optimal estimate. Despite the non-Gaussian nature of the noise, the Kalman filter is still reported to perform very well.

Let us, just like Hinkel and Knieriemen [64], provide each room with a separate coordinate system. As the model for each room is a rectangle, let us further more assume the coordinate system to be defined by two orthogonal walls, i.e. two neighboring sides of the rectangle (see Figure 5.2). Finding the pose of the robot will in this setting be equivalent to finding the distance from the walls and the angle relative to these walls (compare Figure 5.2).

![Diagram](image)

**Figure 5.2:** As the model for each room is a rectangle, let us further more assume the coordinate system to be defined by two orthogonal walls, i.e. two neighboring sides of the rectangle. The pose of the robot can be calculated based on the distances to and orientations of the walls. Each wall is given an identification number.
5.2.1 The State of the Robot

In the Kalman filter setting we introduce the state of the system to be the pose of the robot, \( \mathbf{x} = (x, y, \theta)^T \). We also introduce \( \hat{\mathbf{x}} \) as the estimate of the state, with a corresponding covariance matrix \( \mathbf{P} \).

\[
\mathbf{P} = \begin{pmatrix}
\sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\
\sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\
\sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta}^2
\end{pmatrix}
\]

Equation (4.1) in Chapter 4 provides the means to update the estimate of the pose using odometric information. The best estimate of the pose of the robot given the pose change information from the odometry, is to assume that the noise, \( \mathbf{w} \), in Equation (4.1) is zero (the noise assumed to be zero-mean). The notation follow that of Bar-Shalom (see e.g. [100]).

\[
\hat{\mathbf{x}}(k+1|k) = \hat{\mathbf{x}}(k|k) + \mathbf{u}(k) \tag{5.1}
\]

where \( \hat{\mathbf{x}}(k+1|k) \) should be interpreted as the pose estimate at time \( (k+1) \) using sensor data up to and until time \( k \). The deterministic input, \( \mathbf{u} \), is the pose change according to the odometric model, i.e. \( \mathbf{u} = (\Delta x, \Delta y, \Delta \theta) \). The predicted state covariance is given by

\[
\mathbf{P}(k+1|k) = \mathbf{P}(k|k) + \mathbf{Q}(k) \tag{5.2}
\]

where \( \mathbf{Q}(k) \), just as before, is the covariance matrix for the noise of the odometric model.

5.2.2 Validation Gates

In target tracking literature the problem of data association has always been in the focus. Using validation gates is a common way to handle the problem. A validation gate define a region around some predicted value in which a measurement will be accepted as associated to the corresponding feature\(^1\). Validation gates can be applied at different levels of abstraction. In target tracking applications it is customary to first extract possible targets from sensor data and then try to match these with established target hypotheses. The validation gates could also be used directly on raw data to filter out data points that are likely to be associated with a certain feature/target. After this, the parametric description of the

\(^1\)Here feature should be understood in its widest sense, including targets like aircrafts but also geometric features as lines and edges.
target feature is extracted. We chose the latter, i.e. we first try to filter out data points that are likely to be associated with the walls and then we extract parametric descriptions of these walls in the form of lines. The reason for this is that due to clutter the walls might be very hard to find without filtering. The location of the gates will be functions of the estimated robot pose and the local room model. The size of the gates will depend on the quality of the sensor data, the method used to extract line parameters and the uncertainty in the robot pose. The gates will open up, i.e. let more data through, when the uncertainty in the pose grows and vice versa. If the sensor data is very noisy, the gates have to be more open than if the sensor data is very "clean".

Let the validation region be describe by the six-tuple

\[ G = (\hat{\rho}, \hat{\phi}, \delta, \gamma, \alpha_1, \alpha_2). \]

Here \( \hat{\rho} \) is the predicted distance to the wall and \( \hat{\phi} \) is the predicted angle of normal to the line. These two entities define the pose of the gate. \( \delta \), the smallest width of the gate, and \( \gamma \), the opening angle, define the size of the gate. \( \alpha_1 \) and \( \alpha_2 \) constitutes the visibility constraint. The visibility constraint amounts to not trying to detect a wall that is not in the field of view of the sensor (compare Figure 5.1). Walls that do not fulfill the visibility constraint will not be searched for. Figure 5.3 shows a situation where the robot is in a room, having a slightly wrong estimate of the robot’s pose. The uncertainty in the pose estimate is shown by the ellipse and the true pose is marked with the dark dot in the top corner of the ellipse. The validation gates are marked by the dotted lines and the estimate of the wall locations are shown with dashed lines. Figure 5.4 show a more detailed illustration of the parameters that define the location and size of the validation gates.

5.2.3 Line Extraction

The walls constitute the feature that we want to track. The walls can be described by lines. There are many possible ways to represent a line. As we are only considering one room at a time and the model for each room is a rectangle, the description of a line can be very simple. We have chosen to represent each line with its perpendicular distance from the robot, \( \rho \), the orientation of the normal, \( \phi \) (see Figure 5.5), the length, \( l \). We also store the indices of the data points of that constitutes the line. The equation describing the relation between points on the line and the
Figure 5.3: The estimate of the robot’s pose is used to determine the location and size of the validation gates. The size of the validation gates are increased for illustration purposes. The orientation estimate is assumed perfect for the same reason.

The parameters $\rho$ and $\phi$ is:

$$\rho = x \cos \phi + y \sin \phi. \quad (5.3)$$

**Range Weighted Hough Transform**

In order to track the walls reliably, we need to be able to extract them despite a large amount of clutter in the room. Just like Crowley *et al* [77],

Figure 5.4: Validation gate where the error in the prediction of the wall and the size of the validation gate were scaled for illustration purposes.
and Wernersson et al [41, 71, 72, 73, 74] we use the Hough Transform. To be specific, we use the modified version introduced by Wernersson et al called the Range Weighted Hough Transform (RWHT). We refer to Appendix A for a description of the Range Weighted Hough Transform. In our algorithm the RWHT is part of the filtering process that aims at providing a least squares based line fitting algorithm with as clean range data as possible. The first stage is to run the data through a gate to remove as many of the outliers as possible. We use a local version of the RWHT with a limited Hough Space with parameters, \( \rho \in [\hat{\rho} - N \Delta \rho, \hat{\rho} + N \Delta \rho] \) and \( \phi \in [\hat{\phi} - N \Delta \phi, \hat{\phi} + N \Delta \phi] \), where \( \hat{\rho} \) and \( \hat{\phi} \) are the expected values. That way we do not have to perform all the calculations for lines that are not of interest to us.

The purpose of the pose tracking can, in view of the validation gates, be seen as handling situations where \( \hat{\rho} \) and \( \hat{\phi} \) are not perfect estimates of the line parameters for a particular wall. That is, when the wall does not fall in the middle of the gate. Standard least squares algorithms are sensitive to outliers which calls for a narrow gate. As we are searching for the best possible estimate of the pose of the walls, we must make sure to remove as many outliers as possible. We will use the RWHT for pre-processing. The idea is that the RWHT allows the validation gates to be more open as it has proven to be very robust with respect to outliers. Uncertainty in the orientation of the robot will increase the size of the gate much (parameter \( \gamma \) will increase). By using the local RWHT we can get a better estimate of the pose of the wall and thereby define a tighter gate to filter the data before finally performing the least squares fit. The data that is left through to the least squares algorithm is with a high degree of certainty associated with the walls.
### 5.2 Algorithm

**Least Square Line Fitting Algorithm**

The least squares line fitting algorithm we are using is taken from Deriche et al [101]. In [101] a line is described by

\[
x \sin \phi' - y \cos \phi' + \rho = 0
\]  

(5.4)

where \( \rho \) is the perpendicular distance to the line and \( \phi' \) is the direction of the line. Note that the angle specifying the direction of the line in Equation (5.3), \( \phi \), is angle of the normal to the line, whereas \( \phi' \) in Equation (5.4) is the angle of the line itself (compare Figures 5.5 and 5.6).

Let the measurement points which have passed the two validation gates be called \((x_{2,i}, y_{2,i})\), \(i = 1, \ldots, n_2\), where \( n_2 \) is the number of data that was let through the second stage validation gate. Let furthermore

\[
\Sigma_i = \begin{pmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_{yy}^2 \end{pmatrix}
\]  

(5.5)

be the covariance matrix for measurement point \( i \). The parameters in Equation (5.4) are given by

\[
\phi' = \frac{1}{2} \arctan \left( \frac{b}{a - c} \right)
\]  

(5.6)

\[
\rho = \bar{y} \cos \phi' - \bar{x} \sin \phi'
\]  

(5.7)

where

\[
\bar{x} = \frac{1}{n_2} \sum_{i=1}^{n_2} x_{2,i}
\]  

(5.8)

\[
\bar{y} = \frac{1}{n_2} \sum_{i=1}^{n_2} y_{2,i}
\]  

(5.9)

\[
a = \sum_{i=1}^{n_2} (x_{2,i} - \bar{x})^2
\]  

(5.10)

\[
b = 2 \sum_{i=1}^{n_2} (x_{2,i} - \bar{x})(y_{2,i} - \bar{y})
\]  

(5.11)

\[
c = \sum_{i=1}^{n_2} (y_{2,i} - \bar{y})^2
\]  

(5.12)
Using a first order approximation, the covariance matrix $\Lambda$, for the line with parameters according to Equations (5.6) and (5.7), is given by ([101]):

$$
\Lambda = \frac{a \sigma_{xy}^2 - b \sigma_{xy} + c \sigma_{xx}^2}{(a - c)^2 + b^2} \begin{pmatrix}
1 & -d \\
-d & d^2
\end{pmatrix}
$$

$$
+ \begin{pmatrix}
0 & 0 \\
0 & \frac{\sigma_{xy}^2 \cos^2 \phi' + \sigma_{yy}^2 \sin^2 \phi' - 2 \sigma_{xy}^2 \sin \phi' \cos \phi'}{n_2}
\end{pmatrix}
$$

$$
d = \bar{y} \sin \phi' + \bar{x} \cos \phi'
$$

(5.13)

The length of the line, $l$, is given by

$$
l = \sqrt{(x_{2,n_2} - x_{2,1})^2 + (y_{2,n_2} - y_{2,1})^2}
$$

(5.15)

where $(x_{2,1}, y_{2,1})$ is the first point in the scan that is classified as belonging to the wall and $(x_{2,n_2}, y_{2,n_2})$ is the last point.

In order to make sure that very short lines that might belong to other structures than walls do not disturb the tracking we also enforce a condition for length of a line. A line is considered to be a measurement of a wall if it fulfills one of the conditions below.

$$
l > \tau_1
$$

(5.16)

$$
l > \tau_2^2 \text{ and } \frac{l}{L_{vis}} > \tau_r
$$

(5.17)

where $\tau_1 > \tau_2^2$ are length thresholds and $\tau_r$ is the minimum ratio between the extracted line and the visible length according to the visibility constraint.

### 5.2.4 Measurements

An extracted line will provide information about the $x$ or the $y$ coordinate of the robot in the local room coordinate system by means of the distance to the robot. Two of the walls will provide the $x$ and $y$ coordinates respectively directly and the other two will provide that information by knowing the size of the room. Introducing a numbering of the walls according to Figure 5.2, walls 2 and 4 will provide information about the $x$ coordinate and walls 1 and 3 will provide information about the $y$ coordinate. Each wall will also give information about the orientation of
the robot. A wall measurement can therefore be modeled as
\[ z_i = C_i \mathbf{x} + \mathbf{m}_i + \mathbf{w}_i, \ i = 1, 2, 3, 4 \] (5.18)
where \( z_i = (\rho, \phi)^T \). \( \mathbf{m}_i \) represents the model of the room. \( \mathbf{w}_i \) is the noise associated with the measurement of the \( i \)th wall, with a covariance matrix \( R_i \). \( \mathbf{w}_i \) also includes the uncertainty in the model of the room. We will assume, as is often done, that the odometric noise is uncorrelated with the measurements noise. The predicted measurement corresponding to wall \( i \) will be
\[ \hat{z}_i = \mathbf{m}_i + C_i \bar{\mathbf{x}}. \] (5.19)
The \( C_i \) matrices in Equations (5.19) and 5.18 are given by
\[ C_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \ C_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \ C_3 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \text{ and } C_4 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \] (5.20)
and the \( \mathbf{m}_i \)'s by
\[ \mathbf{m}_1 = \begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix}, \ \mathbf{m}_2 = \begin{pmatrix} 0 \\ \pi \end{pmatrix}, \ \mathbf{m}_3 = \begin{pmatrix} \frac{y_{\text{size}}}{2} \\ \pi \end{pmatrix} \text{ and } \mathbf{m}_4 = \begin{pmatrix} \frac{x_{\text{size}}}{2} \\ 0 \end{pmatrix}. \] (5.21)

Figure 5.7 illustrate how the angle of the wall relative to the robot relates to the orientation of the platform. The orientation of wall 1 is predicted as \( \frac{3\pi}{2} - \theta \) (compare seconds row of \( C_1 \) and \( m_1 \)). Each scan will provide between zero and four measurements depending on the pose of the robot and the presence of moving objects that might block the field of view.

### 5.2.5 Measurement Based State Update

In the standard Kalman filter formulation the measurements are combined to form one large measurement vector. For example, assuming that walls 1, 2 and 4 are detected, the measurement vector will be
\[ \mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_4 \end{pmatrix} \]
with a corresponding covariance matrix

\[ R = \begin{pmatrix} R_1 & R_{12} & R_{14} \\ R_{21} & R_2 & R_{24} \\ R_{41} & R_{42} & R_4 \end{pmatrix}. \]

The updating is then performed using the complete measurement vector, \( z \) (see e.g. [100])

\[
\begin{align*}
S(k+1) &= P(k+1|k) + R(k+1) \\
W(k+1) &= P(k+1)S(k+1)^{-1} \\
\nu(k+1) &= z(k+1) - \hat{z}(k+1) \\
\hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + W(k+1)\nu(k+1) \\
P(k+1|k+1) &= P(k+1|k) - W(k+1)S(k+1)W(k+1)^T.
\end{align*}
\]

This is called block processing and is quite demanding computationally. \( \nu \) is the innovation and represents the new information in the measurement, i.e. the part that could not be predicted. The innovation drives the estimates in the right direction, towards the true pose of the robot as long as the data has been correctly associated. \( W \) is the so called Kalman gain that will weight measurements and predictions to form an optimal pose estimate, assuming that the noise levels and the model are correct.

We will make the simplifying assumption that the measurement noise from individual measurements are uncorrelated, i.e. that \( R_{ij} = 0 \; \forall \; i \neq j \). Because of this, we can instead use so called sequential processing, where the update is performed with one component of the measurement at a
Once again using the notation from Bar-Shalom [100], let $\hat{x}(k|k, j)$ denote the pose estimate based on wall measurements with indices up to $j$. This means that $\hat{x}(k|k, 0) = \hat{x}(k|k - 1)$ and $\hat{x}(k|k) = \hat{x}(k|k, 4)$. $P(k|k, j)$ is the obvious extension of this notation to the covariance matrix. Figure 5.8 provide a description of the data flow in the pose tracking module.

![Figure 5.8: The data flow in the pose tracking module. Odometry is used for prediction in the Kalman filter. Using the predicted pose of the robot in combination with map information, validation gates can be defined. The first validation region provide a local RWHT with data. The line estimate given by the local RWHT parameterizes the second validation gate which provides the least squares line fitting algorithm with data.](image)

## 5.3 Implementation

The above pose tracking algorithm has been implemented on a Pentium 133 MHz running standard Linux on a Nomad200 platform.

### 5.3.1 Wall Extraction

As described in Section 5.2.3 the extraction of the walls is performed in four steps, two validation gates, one local Range Weighted Hough Transform and a least squares line fitting algorithm (see dashed rectangle in Figure 5.8).

---

\(^2\) Even though the measurement are correlated, the sequential processing technique can be applied by first transforming the measurements to a diagonal form using a linear transformation.
First Stage Validation Region

The first validation region must incorporate the uncertainty in the robot’s pose, the model uncertainty and the uncertainty in sensor data. Extending the notation introduced in Section 5.2.2, let $\mathcal{G}_{i,j}$ denote the $i$'th stage validation region for the $j$'th wall. Let further more $\hat{\rho}_{i,j}$ be the expected distance to wall $j$ in the $i$'th stage validation gate and so on. The validation region parameters are chosen as:

$$\begin{pmatrix} \hat{\rho}_i \\ \phi_i \end{pmatrix} = \mathbf{z}_i, \ i = 1, 2, 3, 4$$

$$\delta_i = 50 \text{ mm} + \begin{cases} \sigma_y, & i = 1, 3 \\ \sigma_x, & i = 2, 4 \end{cases}$$

$$\gamma_i = 1^\circ + \sigma_\theta.$$  \hspace{1cm} (5.27, 5.28, 5.29)

Local RWHT

The grid size of the Hough Space has been chosen to be $\Delta \phi = 0.1^\circ$ for $\phi$ and $\Delta \rho = 20 \text{ mm}$ for $\rho$. The size of the local Hough Space is chosen as

$$N_{\phi} = \frac{\gamma}{\Delta \phi}$$

$$N_{\rho} = \frac{\delta}{\Delta \rho}.$$ 

Second Stage Validation Region

For the second gate, the smallest width, $\delta$, is set to be 50 mm to incorporate the uncertainty in the sensor data. The opening angle, $\gamma$, is set to 0 as we now assume that the direction of the line is known with fair accuracy from the RWHT.

Least Squares Algorithm

We have chosen to assume that the covariance matrix for each measurement point is given by

$$\Sigma_i = \Sigma = \begin{pmatrix} 50^2 & 0 \\ 0 & 50^2 \end{pmatrix},$$

i.e. that the $x$ and $y$ coordinates are independent and that variance is equal to the quantization of the sensor. This is obviously a major simplification, but one that has proven to give satisfactory results.
5.3 Implementation

Line Length Based Filtering

The values of the thresholds in (5.16) and (5.17) are given by

\[
\begin{align*}
\tau^1_l &= 2000 \text{ mm} \\
\tau^2_l &= 1000 \text{ mm} \\
\tau_r &= 0.25
\end{align*}
\]

That is, either the line is longer than 2 meters or it is longer than 1 meter and more than 25% of its length is extracted. These values are based on empirical results.

5.3.2 Managing the Size of the Uncertainty

Lower Bound on State Covariance Matrix

The size of the Kalman filter gain, W in Equation (5.23), depends on the size of the uncertainty, P. If the uncertainty is small the Kalman gain will be small. A small gain means that the measurements will be heavily filtered, i.e. they will not influence the pose estimate much. When the robot is standing still and getting many readings without the motion of the platform increasing the uncertainty, the gain will become very small. Then, when the platform does move, the tracking will be very slow due to the small gain. A very simple, but effective, way to deal with this is to put a lower bound on the uncertainty when the robot is moving. Thereby we can assure good tracking performance at every time instant. Putting a lower limit on the uncertainty will also prevent the validation gates from getting unnecessarily small. We have used a lower limit for the diagonal elements of P according to:

\[
\begin{align*}
P_{00,\text{min}} &= (30 \text{ mm})^2 \\
P_{11,\text{min}} &= (30 \text{ mm})^2 \\
P_{22,\text{min}} &= (0.065 \frac{\pi}{180} \text{ rad})^2.
\end{align*}
\]

Noise Increase When Changing Room

Another situation when special care has to be taken regarding the uncertainty is when the robot passes from one room to another, i.e. when the robot goes from one coordinate system to another. The model for each room is very simple to build, but it is more difficult to get the transformation between two different coordinate systems correct. This is dealt
with by inducing noise in the system when passing from one room to another, i.e. noise is induced when the uncertain transformation is applied. We have simple chosen to implement this as an increase of the diagonal elements of \( P \). The elements corresponding to the position information is increased by \((100 \text{ mm})^2\) and the element corresponding to the orientation is increased by \((2^\circ)^2\). This means that when changing room the \( P \) matrix will be updated according to:

\[
P = P + E
\]

with

\[
E = \begin{pmatrix}
100^2 \text{ mm}^2 & 0 & 0 \\
0 & 100^2 \text{ mm}^2 & 0 \\
0 & 0 & \left(\frac{2\pi}{180}\right)^2 \text{ rad}^2 \\
\end{pmatrix}.
\]

### 5.4 Experiments

The pose tracking algorithm described in this chapter has been extensively tested. Tests have been conducted to evaluated the accuracy of the algorithm as well as its robustness. The main part of the tests has been conducted in the mobile robot lab which is furnished with products from IKEA to provide a realistic home environment. The mobile robot lab is nicknamed “the living-room”. When robustness is concerned we have tested for robustness over time, but also robustness to different kinds of environments, i.e. different rooms.

#### 5.4.1 Experiment 1

In this experiment the robot is given a chain of goal points it has to visit. Figure 5.10 shows the living-room with the chain of goal points marked as s(tart/end), 2, 3, 4, 5, 6. The robot is not forced to reach every goal point exactly, only within 100 mm of it. The reason for this is that we are interested in testing the robustness and for that the robot must move around. At the end position though, the requirements are harder and the robot is told to stop within 5mm of the start/end point to be able to tell whether or not the estimated pose is still good. Seven laps in the chain is driven, which takes close to 20 minutes for the robot. By calculating the distance between the goal points in the chain and multiplying by the number of laps around the chain, the theoretical distance the robot travels is found to be 180 m. The average speed is slightly over 0.15 m/s based on the theoretical distance traveled.
5.4 Experiments

The robot is driven totally autonomous, with an avoid behavior actively making sure that the robot does not run into any obstacles. The direction of the sensor is constant during the test, facing approximately towards wall 2. The test can be described by:

1. Place the robot at the desired start/end position.
2. Zero the robot, i.e. reset the odometry.
3. Measure the pose of the robot by some means.
4. Initialize the pose estimate of the robot.
5. Let the tracking unit run until the uncertainty in the position has decreased under a certain threshold, to give the system a good estimate of the starting position to which it is supposed to go back.
6. Note the pose of the robot according to the pose tracking module.
7. Let the robot make the trip around the chain of goal points back to the start/end point.
8. Read the pose according to the odometry.
9. Measure the pose of the robot.
10. Note the pose according to the pose tracking unit.

Figure 5.9: Sketch of the living-room seen from above.
\[
\begin{array}{|c|c|c|c|c|}
\hline
 & \text{Run 1} & \text{Run 2} \\
\hline
x & 2976 & 2960 & 3054 & 3053 \\
\Delta x & 15 & 42 & 14 & 6 \\
\Delta x_{od} & - & 723 & - & 1220 \\
y & 7276 & 7288 & 7300 & 7282 \\
\Delta y & 42 & 39 & 44 & 49 \\
\Delta y_{od} & - & 48 & - & 30 \\
\theta & 91.96 & 120.0 & 73.64 & 102.1 \\
\Delta \theta & 1.85 & 2.1 & 2.33 & 2.12 \\
\text{Time} & 1074 \text{ sec} & 1125 \text{ sec} \\
\text{Dist} & 181 \text{ m} & 181 \text{ m} \\
\text{Speed} & 0.17 \text{ m/s} & 0.16 \text{ m/s} \\
\hline
\end{array}
\]

Table 5.1: Results of Experiment 1, designed to test robustness over time, repeatability and make a comparison with odometry. \( x \) denotes the true x-coordinate, \( \Delta x \) is the error in the tracking units estimate, \( \Delta x_{od} \) error in odometry based estimate, etc... The total time for the tests is given along with the theoretical distance traveled and the average speed.

As the robot moves around in the room for almost 20 minutes when completing the chain of goal points and is able to keep track of its position, robustness can be argued for. Table 5.1 shows that the repeatability is good since the robot returns to almost the same position as it starts from. The platform drifts about 30° during each test, resulting in an odometric error in the order of 1 meter. A bias can be noted in the results. In the \( y \) direction the bias is almost the same in all four measurements (about 40 mm). The source of this bias is an offset in the laser data due to poor measurements of the position and orientation of the sensor in relation to the platform and a non-perfect model of the environment.

### 5.4.2 Experiment 2

The second experiment is constructed to test the accuracy of the algorithm and is described as follows:
Figure 5.10: The chain of goal points in the living-room used to test the robustness of the pose tracking algorithm.

1. Initialize the pose estimate.

2. Direct the robot to go to a given goal point with high accuracy (here 5 mm).

To test the effects of the direction of attention, the experiment is performed with the sensor facing in different directions. The direction of the sensor is constant during each test (not counting the drift).

The results of this experiment show that the method provides means to position the robot at a given location with an accuracy of approximately 20 mm. Regarding the effect of using different directions of the sensor we can say that as long as the sensor can pick up enough points from a wall to form an estimate of the position of the wall it does not matter much if the wall is partially obscured by bookshelves, waste bins, etc.

The largest difference regarding the sensor's direction is noted in situations when the field of view of the sensor is blocked as the robot is moving. In this case the uncertainty will grow and thereby the validation gates will become larger. When the sensor once again gets data, the estimate of the pose of the wall might be disturbed. When the sensor is facing an uncluttered wall this effect will not be noted, but in case the wall is partially occluded, objects in front of the wall might give rise to a stronger line response than the wall itself. This is because we are not looking for a connected line, only points that are align along a line. This
is a weakness, but also a strength. As long as the estimate of the pose is good, i.e. the uncertainty is low, it does not take many data points from the wall to sustain an accurate estimate. When the estimate degrades though, the data association problem becomes harder. Making an error in the data association is a serious error as described before. In the scenario discussed above, with clutter being taken as the wall, the uncertainty in the estimate might indicate that the pose is very accurate whereas in reality the pose estimate is biased and wrong. Means to detect similar situations are needed to ensure robustness.

Due to the fact that the odometry is very good over short distances, especially when there is compensation for the drift in orientation, it was found that it was not necessary for the sensor to see the wall at all times and still keep accurate track of the position and orientation of the robot.

5.4.3 Experiment 3

In this, the third, experiment the robustness to different environments are evaluated. The robot is equipped with the lift mechanism seen in Figure 4.1. The avoid behavior has been augmented to account for presence of the lift, actively trying to turn it as far away from obstacles as possible. When the lift is mounted on the robot, turning the lift to avoid obstacles has higher priority than actively controlling the direction of the sensor. The lift and the sensor are fixed with respect to each other. Having the avoid behavior avoiding obstacles by turning the top of the robot means that even without an active sensing strategy the sensor will not face the same direction at all times. In this context, it is easy to understand why we required the pose tracking algorithm to give fair performance without an active sensing strategy.

One of the objectives with this experiment is to show that even though some rooms are very cluttered, leaving almost no clear view of the walls the method still keeps an accurate estimate of the robot’s pose. We will present the results achieved in three different rooms.

Living-room

The living-room is approximately 8.6 by 5 m. Two pictures, taken from different view points, are shown in Figure 5.11. Figure 5.9 show a sketch of the living-room seen from above. As can be seen from the figures above the living room is quite cluttered. The only wall that provides a somewhat clear view is the right wall in Figure 5.9. The conclusion after
many test is that the robot made a significant error in the estimation of the pose only once. This occurred after having moved for minutes without having had any input from neither the top or the lower wall (refers to Figure 5.9). This is an example of what was discussed at the end of Section 5.4.2. Eventually the validation gates are open enough to let data that comes from the bookshelves and the pillar on the lower wall form a hypothesis strong enough to be interpreted as the wall. When this occurs, the estimate in the position is offset by approximately 300 mm, corresponding to the depth of the bookshelves.

To investigate a possible solution to the problem, noise is injected in the estimate to open the gates and the robot is given a clear view of three of the walls (lower, left and top walls). This results in the robot finding the pose again, indicating that given that the direction of the sensor can be controlled, the correct pose can be retrieved, possibly by opening the validation gates to allow the correct data to come in again. When to inject noise is still on open question, or in other words. How do we detect when the robot is tracking the wrong feature?

Office

The second room is a typical office in our laboratory. The room is so small that the allowable movements of the robot are very limited. The rectangular model of this office room is approximately 5 by 3 meters. The office is divided by a cubicle divider into two parts, leaving very limited sight of two of the walls. No problems are found in this room, partly because the limited space does not give odometric drift a chance to
increase the uncertainty enough to cause any data association problems.

**Corridor**

The corridor at the ground floor of our laboratory is approximately 55 m long and the width is about 2.3 m (see Figure 5.13). The corridor is modeled as a single room in our experiments. It is obvious that the problem in the corridor is going to be to maintain a good estimate of the position along in the direction of the corridor. Since the corridor is modeled as a rectangle, the only two possible sources of information about the position along the corridor comes from the two short walls. When moving far away (>≈ 15 – 20 m) from the short walls they can no longer be used reliably, as too few points are accumulated. The detection of the short walls are made even more difficult if there is clutter in front of them. Moving the platform also makes the detection more difficult because of lack of registration between the sensor data and the corresponding odometric data. This implies that almost half the length of the corridor has to be driven using only odometric information for position updating along the corridor. The biggest problem with the odometry is usually that rotational errors result in big translational error, especially over long distances. This is true for our Nomad200 platform also, as was shown in Chapter 4. In the corridor this could potentially lead to very big errors. The fact that the orientation of the platform is estimated with high accuracy compensates for this though. The biggest error left is the error that comes from mapping the number of wheel rotations to distance traveled. Many experiments are performed and the robot is able to drive from one end of the corridor to the other while maintaining an
5.4 Experiments

Figure 5.13: The corridor seen from north-south and south-north.

accurate estimate of its pose. The estimate of the robot’s position along the corridor becomes increasingly more uncertain along the corridor, but the orientation and the position perpendicular to the corridor are very accurate. As the main part of the odometric error stems from drift in orientation the pose tracking algorithm performs very well even though no measurements are available of the short walls. When coming close enough ($<\approx 15$ m) to the other end, the short wall is tracked and the uncertainty decreases.

5.4.4 Experiment 4

In the last experiment the pose tracking method is tested when moving around between different rooms, at the ground floor of the laboratory. As was seen when testing the algorithm in the corridor, the robot gets no measurement update of the position along the corridor in the mid area of the corridor. It is therefore not clear that the robot will be able to find its way into a room at the other side of the corridor. In one of the tests we let in the robot drive autonomously from the living-room, to the manipulator lab and back again to the living-room. Figure 5.14 show how the uncertainty of the robot grows in the mid area of the corridor and how it decreases when the short wall on the other side of the corridor is detected. The size of the uncertainty ellipses are made bigger for illustration purposes. Figure 5.15 shows a close up of Figure 5.14 at the end of the corridor. When moving in a normal room (not a corridor) the uncertainty is kept small and the uncertainty along the two wall directions are about the same.
5.5 Summary

In this section we summarize the results presented in this chapter. A pose tracking algorithm has been proposed which has shown to combine robustness with accuracy. The algorithm is tested extensively in a non-engineered environment. Situations when the algorithm is likely to fail have also been identified.

As a result of the experiments we conclude that the simple model where each room is modeled as a rectangle is sufficient for keeping track of the robot’s pose in a non-engineered but structured environment. Difficulties are encountered in large rooms, especially if they are very narrow like corridors, thereby seriously limiting the number of data points that can be associated with a distant short wall.

The only precondition for the successful operation of the method is that parts of the walls can be detected and that we have a sufficiently good initial estimate of the robot’s pose. A strong advantage of the method is its ability to accurately track the robot’s pose in the presence of a large amount of clutter. As long as a few data points associated with the wall can be detected, the pose estimate can be maintained. This is at the same time the main weakness in the sense that a few points suffice to define a line. When the pose estimate is poorly known the validation gates will open up and may erroneously let through lines formed by a few points. If this happens then the estimate of the pose will converge.
5.5 Summary

Figure 5.15: A close up of the situation when passing from the corridor to the manipulator lab.

towards an incorrect value.

The method has proven to provide an accuracy in the order of 20 mm. The lower limit for the accuracy depends heavily on the world model and the parameters defining the placement of the sensor relative to the robot. To get better accuracy we could improve the environmental model, reduce the speed of the robot and decrease the lowest allowed level of uncertainty in the estimate, reducing the filter gain and the size of the validation gates. By reducing the filter gain, the fluctuations in the estimate will decrease. With the levels specified in Section 5.3 the position estimate typically chatters with an amplitude in the order of 5 mm. When the estimate is fluctuating it is difficult for the robot to determine when it has reached the goal point, thus making the accuracy evaluation more uncertain. Another complicating factor is that the uncertainty inherent in the manual measurements is of the same magnitude as the localization error, meaning that the true localization error could in fact be smaller.

The robustness has also been proven to be excellent. The pose tracking has been shown to be highly robust for different environments (different rooms) as seen in Experiment 3 and over time, as shown by the experimental results presented here and also the tens of hours that the system has successfully kept track of the robot's pose when performing various other tasks.

One of the design goals was to be able to operate without active control of the direction of sensing to ensure robustness. This is also
fulfilled by the suggested method. Experiment 1 and 2 were successfully performed without changing the orientation of the sensor at all. The robot is equipped with a lift mechanism which also reduces the ability to actively control the direction of the sensor, as priority is given to avoiding hitting obstacles with the lift. However, the same lift-avoid behavior ensures that the sensor is changing direction when moving, allowing for better robustness because the best wall to look at is more likely to be in the field of view at least part of the time.
Chapter 6

Pose Initialization

By pose initialization we understand the process of finding the pose of the robot with no or very vague prior pose information. Situations which match these conditions are at startup and during a mission when the pose is lost. We have already discussed, in Chapter 1 what we understand to be the three main components of localization, initialization, maintenance and modeling. Maintenance requires the initial pose of the robot to be known. Hence the pose initialization problem is of utmost importance as pose tracking potentially can be implemented much more computationally efficient as the initial pose is known which limits the search region.

In a typical scenario for a autonomous robot we find that the initial pose of the robot could be provided by the user. This could be done either by making sure that the robot is initiated at a pre-defined position or that the user specifies the position of the robot relative to some other object, such as between the sofa and the chair in the living room. During the execution of a mission on the other hand, it cannot be assumed that such assistance is available. Two situations where complete or partial pose initialization is needed can thus be identified.

- At startup, potentially without any prior pose information at all.
- During the execution of a mission, when the robot loses track of the pose for one reason or another.

We will here limit our discussion to a situation where prior knowledge of the robot’s pose is available in the form of knowing that the robot is in one of a very small number of possible rooms, in most cases only one room.
The limitation caused by this assumption is justified by the fact we hope to be able to determine when the pose is lost during the execution of a mission, by for example monitoring the uncertainty in the pose estimate. At startup we furthermore assume that the initial room is known. This room might very well be the room where the robot recharges.

Just as when determining a suitable method for pose tracking we must consider the world model we are working with, where each room is modeled as a rectangle. Drumheller [13] claims that pose initialization can be achieved using data acquired from a single position in a room using a low resolution sensor. The method relies on extracting lines from sensor data to match against the world model. Hinkel and Knieriemen [64] present a more promising approach, using what they call the angle histogram. Grid methods can also be used, but just as for the case of the pose tracking we believe that the parametric methods provide better means to capture the simplicity of the rectangular model. We propose to use the angle histogram in combination with a voting scheme to find the pose of the robot.

6.1 Theory

The angle histogram is well suited for finding the main orientation of the walls of the room relative to the robot. Given that we know what room we are in, and hence the size of the room, we can concentrate our search for walls in certain directions. The Hough Transform, used as a pre-filter in the pose tracking algorithm, has proven to be very robust for extracting lines from a cluttered environment. The Hough Transform could be used to extract lines here as well which are then matched against the model of the room. Another possibility is to follow the example of Hinkel and Knieriemen [64] and create histograms over the $x$ and $y$ distribution of data. Given that the room is rectangular, the Hough Transform and the $x,y$-histogram technique will provide approximately the same kind of information. The $x,y$-histogram technique can be seen as an example of a local Hough Transform where the Hough Space has been limited to lines having directions close to the four wall directions.

6.2 Algorithm

We chose to use the angle histogram in combination with the $x,y$-histograms to find the pose of the robot. The reason for choosing the $x,y$-histograms
Figure 6.1: Raw laser data from the living room. Two scans are taken with the robot rotated 180° between the scans.

over the Hough Transform is that we believe that the \(x,y\)-histograms contain the needed information in a better format. In the \(x,y\)-histograms we can easier cluster data points that are situated close to the walls, regardless if they lie on a straight line. This could be useful in a cluttered room. In the following sections we will describe the steps involved in finding the initial pose.

6.2.1 Finding the Main Orientation of the Walls

The main orientation of the walls can be found using the angle histogram that will be outlined here. A 360° laser scan of the environment is created by combining two scans of 180° where the robot turns 180° in between the scans. Figure 6.1 show such a complete scan taken in the living room. The angle of virtual lines between nearby scan points is the basis for the angle histogram. As sensor data is noisy in nature, virtual lines between neighboring scan points give noisy estimates of the angles. It is better to let the scan points be separate by angle \(\beta\). In [64] \(\beta\) was chosen as 10°. Figure 6.2 shows an illustration of such a virtual line with corresponding angle, \(\alpha_k\). Let the points of the scan be denoted with \(P_i = (x_i, y_i), i = 1, \ldots, N\) where \(N\) is the number of points in the complete scan. The coordinates are given in the robot fixed reference frame. Let the angle of the virtual line between point \(P_k\) and \(P_{k+M}\) be denoted \(\alpha_k\). \(M\) is the spacing between the points used to avoid unnecessary noise,
corresponding to a separation in scan angle of $M \Delta \phi$, where $\Delta \phi$ is the angular resolution of the scan. In a rectangular room, the orientations, $\gamma_i$, of the walls can be written

$$\gamma_i = \gamma^0 + i \frac{\pi}{2}, \quad i = 1, 2, 3, 4$$

(6.1)

where $\gamma^0$ is the direction of one of the walls. Hence, if we build a histogram over the angles, $\alpha_k$, $k = 1, \ldots, N$, of the virtual lines we would expect to see four peaks corresponding to the four wall directions. Better accuracy can be achieved by calculating the angles modulo $180^\circ$, or even $90^\circ$. Figure 6.3 shows an angle histogram corresponding to the complete scan in Figure 6.1 where the angle have been calculated modulo $180^\circ$. The peaks of the angle histogram corresponds to direction along which many points are aligned. In a rectangular room the peaks are expected to be found at angles $\gamma_i$ according to Equation (6.1). Given the orientation of the walls relative to the robot fixed reference frame the orientation of the robot can be found by identifying which wall corresponds to which in model. As in the pose tracking algorithm we here assume each room to have a local coordinate system, two walls coinciding with the coordinate axes. We also use the same numbering of the walls in the room, i.e. wall 1 coincide with the $x$-axis, wall 2 with the $y$-axis, etc (see Figure 5.2).
6.2 Algorithm

Figure 6.3: The angle histogram, created by calculating the angle between nearby scan points in the raw laser scan data (see Figure 6.1). The accumulator bins in the histogram have size $\frac{\pi}{180}$ rad. The peak at -0.25 rad ($\approx -14^\circ$) corresponds to two short walls and the peak at 1.3 ($\approx 75^\circ$) the long walls.

6.2.2 Generation of Possible Poses

To find the position of the robot we need to identify the walls in the scan. As suggested in Section 6.1 we will use $x,y$-histograms to find the walls. The idea is that given the position and orientation of the walls relative to the robot, the pose of the robot can be found in the local coordinate system. By rotating the data in the complete scan an angle $-\gamma^0$, the walls will be parallel with the coordinate axes of the robot fixed reference frame. Figure 6.4 show the result of rotating the complete scan in Figure 6.1 by $\approx 14^\circ$. When the data has been aligned with the coordinate axes, $x,y$-histograms can be created by projecting the scan data on the axes. Figures 6.5 and 6.6 show the $x$ and $y$-histograms corresponding to the rotated data in Figure 6.4. Introduce $\mathcal{W}^x$ as a set of wall candidates, corresponding to direction $x$ and $\mathcal{W}^y$ as a set of wall candidates corresponding to direction $y$, where it is important to remember that $x$ and $y$ refers to the robot fixed coordinate system. Introduce also $\mathcal{W}$ as the set of all wall candidates, i.e. $\mathcal{W} = \mathcal{W}^x + \mathcal{W}^y$. The elements of $\mathcal{W}$ will be denoted $w$, where the indices are chosen so that the first $N_w^x$ elements belong to $\mathcal{W}^x$ and the following $N_w^y$ belong to $\mathcal{W}^y$, giving a total of $N_w = N_w^x + N_w^y$ wall candidates. We here assume the wall candidates in sets $\mathcal{W}^x$ and $\mathcal{W}^y$ to be given by the strongest peaks in the $x,y$-histograms. This means that $w_1$ corresponds to the peak at
distance $-3000$ mm in the $x$-histogram in Figure 6.5, $w_2$ to the peak at
$-1800$ mm, and so on. Let the wall distance of candidate $w_i$ be denoted
$d_i$ and its voting strength $v_i$. The voting strength corresponds to the
accumulated number of data for wall distance $d_i$, i.e. $v_i \approx 70$, etc.

Let $C^x$ denote a set of combinations of wall candidates, representing
possible pairs of walls, parallel to the $x$-axis of the room coordinate sys-
tem. $C^y$ denotes the corresponding set for the $y$-axis. We know that for
a pair of correctly associated wall candidates $w_i$ and $w_j$ in $C^x$ we have
$|d_i - d_j| = y_{size}$ (compare Figure 5.2). The elements in these sets will
be denoted $c_{i,j}^{x,k}$ and $c_{i,j}^{y,k}$ respectively, where the sub indices refer to the
indices of the elements of $W$. The elements in $C^x$ must fulfill

1. $|d_i - d_j| = y_{size} < \tau_x$

2. $w_i, w_j \in W^x$ or $w_i, w_j \in W^y$.

The first condition just means that the walls should have the right dis-
tance between to be walls parallel to the room $x$-axis. The second con-
tdition is just to make sure that only parallel walls candidates are combined.

The conditions of for elements in $C^y$ must fulfill

1. $|d_i - d_j| = x_{size} < \tau_y$

2. $w_i, w_j \in W^x$ or $w_i, w_j \in W^y$. 

**Figure 6.4:** By rotating the data with an angle given by the highest
peak in the angle histogram, i.e. the angle corresponding to the main
direction of the walls, the walls will be aligned with the coordinate axis.
6.2 Algorithm

**Figure 6.5:** The $x$-histogram is formed by summing the number of data points with a certain $x$ coordinate in the rotated data (see Figure 6.4. Here the size of the accumulator bins is 200 mm.

**Figure 6.6:** The $y$-histogram is formed by summing the number of data points with a certain $y$ coordinate in the rotated data (see Figure 6.4. Here the size of the accumulator bins is 200 mm.

We let the concept of combination be wide in the sense that we allow a single wall to form a combination. The voting strength of combination $i, j$ is given by the function $g(c_x, y)$.

### 6.2.3 Finding the Pose of the Robot

Finding the pose of the robot is equivalent to finding the best matching elements $c_x^k$ and $c_y^l$. We chose to define best in the sense of voting strength. Hence, the two elements $c_x^k$ and $c_y^l$ of $C_x$ and $C_y$ respectively is given by

$$\arg \max_{k,l} \{g(c_x^k) + g(c_y^l)\}. \tag{6.2}$$

One problem associated with finding the pose of the robot, given the simple rectangular model, is that there will be two ambiguous hypotheses about the pose of the robot given by $c_x^k$ and $c_y^l$. To disambiguate these hypotheses we will assume that the initial orientation of the robot, $\theta^i$, is known within $\pm 90^\circ$. This is no serious limitation as we can assume that gross orientation information is available from a compass.

The orientation of the robot is

$$\theta = -\gamma_x \tag{6.3}$$

---

1 four for a square room
where $\gamma_x$ is the angle of the wall corresponding to the room $x$-axis, i.e. we need to identify wall 1.

6.3 Implementation

6.3.1 Histograms

Creating the histograms requires discretization. In general a gross discretization will reduce noise but will decrease the resolution. Therefore the choice of bin size will be a trade off. Let $\Delta \alpha$ denote the bin size in the angle histogram. Let further more $\Delta x$ and $\Delta y$ be the bin size in the $x$ and $y$-histograms respectively. The following bin sizes have been used

$$\Delta \alpha = \frac{\pi}{100} \text{ rad}$$
$$\Delta x = 200 \text{ mm}$$
$$\Delta y = 200 \text{ mm}.$$ 

For calculating the angle histogram we have used a scan angle separation, $\beta = 10^\circ$, just as in [64].

6.3.2 Wall Candidates

When selecting wall candidates we use the 20 largest peaks from the $x$ and $y$ histograms, i.e. $N_{x\text{c}} = N_{y\text{c}} = 20$. Parallel wall candidates, i.e. taken from the same set, either $\mathcal{W}^x$ or $\mathcal{W}^y$, will be combined to form a pair according to the conditions in Section 6.2.2. The thresholds $\tau_x$ and $\tau_y$ are chosen as $2\Delta x$ and $2\Delta y$ to reflect the fact that the distances given by the peaks in the histograms are quantized with steps of $\Delta x$ and $\Delta y$, i.e.

$$\tau_x = 400 \text{ mm}$$
$$\tau_y = 400 \text{ mm}.$$ 

The function for the voting strength of a pair of wall candidates are chosen as

$$g(i, j) = \begin{cases} 
2(v_i + v_j) & i \neq j \\
v_i & i = j 
\end{cases}$$ (6.4)

to promote wall combinations where two walls are used.
6.4 Experiments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Δx</th>
<th></th>
<th>Δy</th>
<th>θ</th>
<th>Δθ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos1</td>
<td>3420</td>
<td>-30</td>
<td>7650</td>
<td>60</td>
<td>119</td>
<td>-0.2</td>
</tr>
<tr>
<td>Pos2</td>
<td>2470</td>
<td>120</td>
<td>6440</td>
<td>-30</td>
<td>119</td>
<td>-2</td>
</tr>
<tr>
<td>Pos3</td>
<td>1380</td>
<td>10</td>
<td>7870</td>
<td>-7070</td>
<td>118</td>
<td>+0.8</td>
</tr>
<tr>
<td>Pos3b</td>
<td>1380</td>
<td>10</td>
<td>7870</td>
<td>-50</td>
<td>118</td>
<td>+0.8</td>
</tr>
<tr>
<td>Pos4</td>
<td>3940</td>
<td>-50</td>
<td>3900</td>
<td>10</td>
<td>116</td>
<td>-0.8</td>
</tr>
<tr>
<td>Pos4b</td>
<td>3940</td>
<td>-50</td>
<td>3900</td>
<td>10</td>
<td>116</td>
<td>-0.8</td>
</tr>
</tbody>
</table>

Table 6.1: Results of the absolute localization. Four different locations are tested and in two of the locations two different environmental settings are tested. Δx, Δy and Δθ here represent the errors in the three different components.

6.3.3 Finding the Pose of the Robot

The initial guess of the orientation of the platform corresponds to the highest peak in the angle histogram. If the best match according to Equation (6.2) indicates that the x-axis in the rotated robot fixed coordinate system is equivalent to the y-axis of the local room coordinate system, the orientation of the robot is given by a rotation of 90°, to yield the orientation θ. At this point the pose of the robot is given as one of two poses. Here we use the information about the initial orientation of the robot. The orientation is finally given by the angle

$$\theta = \begin{cases} 
\theta, & |\theta - \theta^0| < 90^\circ \\
\theta - 180^\circ, & \text{otherwise} 
\end{cases}$$  \hspace{1cm} (6.5)

As the walls have been identified the position of the robot can be found by using the distances to the walls, \(d_i\).

6.4 Experiments

In the experiment the robot is placed at different locations in the living room and the pose initialization algorithm is run to estimate the pose of the robot. At each point five tests were performed. Table 6.1 show the result of the experiment. Pos3 is the only position that show a error larger than the resolution of the algorithm. The reason for the failure in this position is that the robot is placed between one of the short walls
and a line of three chairs. Due to the limited resolution of the method the chairs cannot be distinguished from a wall and the they falsely taken to be the other short wall.

6.5 Summary

We have in this chapter presented a method for determining the pose of the robot given that the room is known. The initial orientation of the robot is also assumed to be known ±90°, which is not such a limitation as a compass can be used to get a rough estimate of the orientation. The method can additionally be augmented to handle a situation where there is a set of possible rooms. To do so we need to introduce a measure of how likely a given room is to be the right room. Such a measure should encode the degree of match between the data and the rectangular model. The measure should also ideally be robust against clutter.

The presented pose initialization method has proven to provide good estimates of the pose of the robot. Situations where it fails have been identified, but this problem can be resolved by simply moving the robot to another position. One scheme would be to run the pose initialization algorithm at a few different positions in the room and then use simple voting to determine the pose of the robot. In the experiments performed in the living room, only one failure to find the pose occurred. Assuming that three random positions from this experiment were to be chosen, simple majority voting could be used to determine the true pose of the robot.

We believe that the very simple, rectangular, model is too simple for doing complete pose initialization in an efficient manner. By complete we mean without using the compass. Furthermore, larger rooms like corridors will present problems as the end of the corridor will only provide very limited response. Open doors can in such a situation easily be mistaken for the end of the corridor, giving the wrong pose.

An interesting question to answer is: How much more information we need to be able to discriminate the pose? A possible extension of the model would be to incorporate the positions of doors. The positions of the doors are needed to be able to go from one room to another. This information is already in the system, it is just not being used for localization purposes. Using doors as features would also reduce the need for knowing the initial orientation of the robot, as the doors can be used to resolve ambiguous situations.
Chapter 7

Conclusions and Directions for Future Work

7.1 Conclusions

In this thesis we have presented an overview of the most common sensors used for localization. The emphasis has been on laser, sonar and odometry. We concluded that for our purposes, localization using a minimalistic model of the world, the laser sensor in combination with odometry is the best choice. An overview of existing methods for solving the localization problem has also been presented. We limited the overview to methods which do not require the environment to be engineered. The conclusion was that the parametric methods or feature-based methods are most likely to be successful when using a simple model.

We have throughout the thesis stressed the importance of having good understanding of the sensors being used. Therefore, Chapter 4 was partially dedicated to characterizing the odometric system on our Nomad200 platform. The result being a model, complete with noise characteristics.

One conclusion was that the odometric system provides information of excellent quality over short distances. Because of drift in the orientation, the use over long distances is limited. This implies that the odometric system would benefit greatly form being able to compensate for the drift in orientation by some means. Inertial sensors or a compass are possible
solutions.

The second half of Chapter 4 evaluated the performance of the PLS laser scanner from SICK. Various effects were studied, such as quantization, chatter and phantom measurements. The beam width was determined to be less than 0.5°, underlining that the laser scanner is sensor with high angular resolution. A model for how the uncertainty in polar coordinates are transformed to uncertainty in Cartesian coordinates was presented. The model is very complex. Experiments show that a Gaussian model is sufficient though.

In Chapter 5 we introduced the model of the environment. Each room is modeled as a rectangle, parameterized by the size of the room in the $x$ and $y$ directions. A four stage filtering algorithm was presented which was able to robustly extract the position of the walls in a cluttered environment. Assuming that the initial pose of the robot was known we showed, through a series of experiments, that the algorithm is capable of robustly keeping track of the robot’s pose for very long periods of time. In fact, during the missions performed after the development of this pose tracking algorithm, the robot has not even once lost track of its position. High accuracy can also be argued for as the method has proven to provide estimates with an accuracy of the order of 20 mm. Situations where the method fails has been identified as those where no measurements are available in one or both direction over a long time, when the robot is moving. The reason for the failure being that, as the uncertainty grows, the validation gates open up and the risk of making a mistake in the data association is increased. We must also acknowledge that we have benefited greatly from the high quality of the odometry over short distances. The pose tracking algorithm provides the necessary compensation for the drift giving a system that can function without continuously detecting the walls. Moving the robot to an environment with other surface conditions, more inclined towards slippage, would increase the need for wall detection.

Finally, a method for initializing the pose of the robot has been presented. It has been shown to be reliable through experiments. As the method is based on information gathered at only one location, there may be situations where the walls are obscured and so the pose cannot be estimated. The experiments indicate that such situations can be resolved by simply repeating the procedure at a few locations and then voting for the most likely pose of the robot.

The main conclusion in this thesis is that pose tracking can be implemented to provide robustness and accuracy using a very simple model
of the environment. Pose initialization is likely to require more information, but we have showed that the simple model is sufficient for pose initialization in many cases.

7.2 Future Work

Many open questions still remain. Some of them can be seen as modifications or extensions to the existing methods. Other open up whole areas of research and constitute fundamental problems. We list some of the issues below.

Extensions

- The pose tracking method presented uses odometry for predicting the next position. Due to wheel slippage, etc., the errors in the odometric information are occasionally large. Time varying odometric noise might be one possible way of capturing these events.

- The pose initialization method should be extended to detect cases when the position from which the data is collected is insufficient to resolve the true pose of the robot. As suggested in Chapter 6 one way might be to run the algorithm at a few locations resulting in multiple hypotheses. Voting based techniques could be used to find the true pose of the robot.

Fundamental Problems

- Just as detecting large errors in the odometry is important, it is equally important to be able to detect when the robot has lost track of the pose. Simply requiring that at least two walls are seen continuously is not good enough, as there will always be situations where the sensor is blocked momentarily, e.g. by a walking person. In the case of the corridor, one method to better handle the growing uncertainty in the direction of the corridor is to let the robot temporarily go into another room on the way to decrease the uncertainty. This is an example of active sensing, i.e. we actively control what part of the environment to extract information from to get the information needed to solve the task in the best possible way. The laser sensor has so far been restricted to extracting information from a plane. By mounting the sensor on a pan-tilt-unit the active sensing
strategy can be designed for finding the best sensing direction not only in the pan-direction, but also in the tilt-direction.

- Laser in itself will not provide a completely robust system. Fusion with other sensor modalities is therefore necessary. We believe that to create a truly robust system we need heterogeneous sensory information, i.e. sensory information from other types of sensors than range sensors.

- In the past, the methods for pose estimation have employed an implicit assumption of a uni-modal distribution of the uncertainty in pose. I.e. the matching of data to a model is unique. We believe that in a structured, office type environment with a high degree of symmetry a multi-hypothesis representation is needed. For the case of multiple competing matchings a hypothesis scheme has been employed. Each possible matching alternative is then modeled as a hypothesis that can be approximated by a uni-modal error distribution. Independence is assumed between different hypotheses. In future research it is suggested that a multi-modal framework is employed, where multiple matchings can be encoded in a single coherent representation. Such an approach has recently shown great promise in the area of tracking of objects in digital images. It is expected that similar results can be obtained for other sensory modalities.

- No system can be said to be fully autonomous until it has the ability to, by itself, construct a model of the environment. The construction of the model should be an ongoing process that constantly adapts the model to best describe the environment in its present state.
Appendix A

The Range Weighted Hough Transform

The Range Weighted Hough Transform is a special version of the standard Hough Transform that has been used extensively in other disciplines in the robotics field, such as computer vision. [76, 75] give a brief description of the standard Hough Transform. We will outline the standard Hough Transform and then describe modifications leading to the Range Weighted Hough Transform.

A.1 Hough Transform

The Hough Transform was developed to find curves from a certain family (lines, circles, etc) by connecting edge points extracted in an image. The edge points can be generalized to any point of interest. A family of curves can be parameterized, e.g. all lines passing through \((x_i, y_i)\) can be parameterized as

\[ y_i = k x_i + m \]  \hspace{1cm} (A.1)

where \(k\) is the slope and \(m\) is the intersection with the \(y\)-axis. The idea is to go through the points of interest and look at all possible curves passing through that point. The parameters for lines passing through \((x_i, y_i)\) can be written

\[ m = -x_i k + y_i, \]
i.e. a line in the parameter space. Each point of interest corresponds to a line in the parameter space. All points on a line will give lines in parameter space that intersect at a point corresponding to the line parameters. Figures A.1 and A.2 illustrate how points on a line are mapped to the parameter space. It is clear that the representation for the line given by A.1 is not ideal, as the line parameters vary over an infinite range. A better representation is

\[ x_i \cos \phi + y_i \sin \phi = \rho \]  

(A.2)

where \( \phi \) is the direction of the line and \( \rho \) is the perpendicular distance.

In a real application the points of interest are subject to noise and there might be more than one curve present in the data. In such a situation there will not be one unique intersection in the parameter space. To solve this problem the parameter space can be quantized, i.e. the space is divided into a grid of discrete parameter values. Let \( \Delta \rho \) and \( \Delta \phi \) be the quantization step in distance and angle.

Each point in point space will now update all the grid cells or bins along a discretized curve in parameter space. With the parameterization as in Equation (A.2), for each point \((x_i, y_i)\) the update of the parameter space can be performed by iterating over all angle, \( \phi_k \), and calculate the corresponding \( \rho_k \) according to Equation (A.2). Each parameter pair \((\rho_k, \phi_k)\) give one vote for the corresponding line. The line is found by searching for the bin with the largest number of votes. If there are many lines, they can be found by searching for local maxima.

\[ \text{A.1} \]

**Figure A.1:** Two points \((x_i, y_i)\) and \((x_j, y_j)\) on the line \( y = k'x + m' \).

\[ \text{A.2} \]

**Figure A.2:** The intersection of the lines corresponds to the line parameters.

\[ \text{A.3} \]

\[ m = -x_jk + y_j \]

\[ m = -x_i k + y_i \]

\[ k' \]

\[ k \]

---

\( \text{A.4} \)

3The parameter space is sometimes referred to as the Hough Space.
In the case where the points of interest are edge points in an image, there is more information attached to each point than the position. Each point has a direction associated to it. Hence the parameter space that has to be updated can be limited. Using the example with the line above, the direction of a point will put constraints on the angles, $\phi$. That is, only angles close to the angle predicted by the edge detection need to be considered. This will speed up the update. Constraints on the parameterization will reduce the parameter space that has to be updated and hence reduce the number of computations.

### A.2 The Range Weighted Hough Transform

Even though the original intent for the Hough Transform was to extract curves from edge points in an image, the idea to apply the Hough Transform on range sensor data is not farfetched. Crowley et al use the Hough Transform to extract lines in [77]. Wernersson et al [41, 71, 72, 73, 74] use a scanning laser sensor for localization. The laser scanner samples the environment equidistantly in angle. Objects that are close to the
sensor will be represented by more points in the data. Each point will give votes in the Hough Space. Therefore, curves that are close to the sensor will accumulate more votes than curves further away. The Range Weighted Hough Transform (RWHT) aims at giving a certain curve equal vote, no matter what distance from the sensor it is at. The points are given a voting strength proportional to the distance from the sensor. An exact compensation would require the voting strength to be $\frac{\phi}{\cos \phi}$. Due to noise, this does not lend itself to be implemented because of the term $\cos \phi$ in the denominator. Using the weighting strength $\rho$ has proven to give good results though.
Bibliography


Bibliography


[55] Netzler & Dahlgren Co AB, SE-429 80 Säro, Sweden, phone: +46 31 93 80 00, fax: +46 31 93 81 00, NDC News No.11.


