KUNGL TEKNISKA HÖGSKOLAN

Approaches to Mobile Robot Localization in Indoor Environments

Patric Jensfelt

Doctoral Thesis
Department of Signals, Sensors and Systems
Royal Institute of Technology

Submitted to the School of Electrical Engineering, Royal Institute of Technology, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Automatic Control
Department of Signals, Sensors and Systems
Royal Institute of Technology (KTH)
SE-100 44 Stockholm, Sweden
http://www.s3.kth.se

TRITA–S3–REG–0102
ISSN 1404–2150
ISBN 91–7283–135–9

Abstract

This thesis deals with all aspects of mobile robot localization for indoor applications. The problems span from tracking the position given an initial estimate, over finding it without any prior position knowledge, to automatically building a representation of the environment while performing localization. The theme is the use of minimalistic models which capture the large scale structures of the environment, such as the dominant walls, to provide scalable and low-complexity solutions.

In many cases it is enough to only maintain an estimate of the robot position. For such situation an extensively tested low-complexity, robust and accurate pose tracking method is presented which utilizes the minimalistic model in combination with a laser sensor.

When the initial position is unknown the robot must perform global localization. Two different methods are investigated. The first one is a novel localization scheme, based on the ideas of Multiple Hypothesis Tracking. The second is an, experimentally verified, significant improvement of the standard Monte Carlo Localization technique.

To automatically generate an environmental representation an hierarchical approach to simultaneous localization and mapping (SLAM) is presented. The map scaling issue is here addressed by dividing the environment into submaps, each representing a small area.

Keywords: mobile robot, laser scanner, sonar, odometric model, sensor fusion, pose tracking, global localization, SLAM, Kalman filter, particle filter, Multiple Hypothesis Localization, Monte Carlo Localization.

Credits For the Journey

Bo, for finding me and putting me on-board this ship and cheering me on. Your optimism has helped me through dark waters.
Henrik, the Danish Captain with endless energy, for always making sure that we were heading in the right direction and for providing the best material for the ship. Thanks also for always having time for a discussion.
Olle, my closest ship mate, you made the journey slightly ahead of me, but with a course very close to mine, together from day one until the end. We helped each other during many storms, some lasting as long as 50 hours and struggled together with the demo(n)s. Thanks again for reading this logbook so carefully.
David, from down under, for spending a year and a half up in the cold north and trying to heat it with chilli. You too battled the 50 hour storm and many other along my side. Great to have had you on-board mate and thanks for reading the logbook! Wadayarekon, more chilli, Dr. D?
Lars, no more sunglasses for the two of us, the end is near. No more Vostok for me and no more Tequila for you my friend!
Anders, for all your time spent on keeping the fleet up and running. For all good times that were and all to come, I am grateful, but "de' vet du!" I hope.
Mattias, for the time we spent fighting demo(n)s.
Steen, for the great cooperation we have had with MHL.
Magnus, for your efforts with ISR, a tough job that someone had to do.
All other crew members of the ship called CAS. Thanks!
Dani, wind and water, sun and rain, winter and summer, many oceans to go.

This journey has been sponsored by the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems. The funding is gratefully acknowledged.

To the Sarcastros

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today robotic systems are widely used in the industry, in particular for tasks such as welding, painting and packaging. All of these robot systems are in the form of manipulators that carry out repetitive motion. For large scale transportation such robotics systems are not particularly practical and therefore automatic transportation systems have been developed either as numerically controlled systems like automatic warehouses or through use of automatic guided vehicles (AGVs). Almost all AGVs use a guide-path system where they track a buried wire on the floor or utilize other forms of artificial landmarks.

There is currently a diffusion in the set of applications away from the factory setting towards office and domestic applications. One issue that in particular motivates this change is the aging of society. An analysis of the demographics for countries such as Sweden reveals that the number of retired people will increase with more than 50% over the next 30 years. In health-care it is widely recognized that people older than 85 years of age in general are in need of assistance for a number of everyday tasks such as getting dressed, getting out of bed, visiting the rest-room, preparing meals, etc. For this group of citizens the forecast is even more dramatic. Over the next 10 years this group will increase with 25% and over the next 30 years it will double in size. Quality of life is in some respects synonymous with "autonomy", i.e. getting by without the assistance of others. To this end there is a need for development of intelligent aids that allow people to have a significant degree of autonomy. One possible solution to this problem is to provide robotic platforms that can assist the user with tasks such as fetch-and-carry, mobility aids etc. The development of such facilities will not only be of interest to elderly but can also be used by other citizens for tasks such as automatic cleaning, entertainment, etc. The class of robot systems to carry out such tasks are termed "Service Robots".

1.1 Service Robots

A service robot must as a minimum include facilities for autonomous navigation in indoor environments. To make the systems truly useful they must in addition include facilities for interaction with the environment. This interaction can take on many different forms, from simple pushing to explicit manipulation of objects.

A fundamental component of any mobile robot system is methods for localization and navigation. Almost all deliberative tasks that a robot must carry out has as an underlying assumption that the system can answer the three questions: "Where am I?", "Where am I going?", and "How do I get there?". The first question is termed localization, while the second questions implicitly is termed localization and/or place recognition, while the third question is termed path planning.

In industrial settings it is often permissible to use artificial markers for localization. In addition it might be acceptable to provide a map of the environment a priori, e.g. a CAD model of the environment. For operation in an office environment or in a domestic setting it might not be acceptable to place artificial markers throughout the environment, in addition it is in general impossible to assume that a map of the environment can be provided. Such a map can be generated by professionals but it is unrealistic to assume that the regular citizen can. Relying on such information would thus significantly limit the potential utility of a system. Consequently it is of interest to investigate methods that allow automatic acquisition of maps of the environment based on natural features and subsequent use of such maps for robust localization and navigation. These are the issues studied in this thesis.

1.2 Working Hypothesis and the Localization Problem

The work presented in the thesis has been conducted within the context of the project "The Intelligent Service Robot" at the Center for Autonomous Systems (CAS) at the Royal Institute of Technology. The project acts as a focal point for the research and provides a platform for systems integration. It is only by putting all the parts together that a fully autonomous system can ever be realized. The setting for the service robot is an indoor semi-structured environment, which typical office and domestic areas are examples of.

The working hypothesis throughout the thesis is that it is possible to perform localization reliably using a simple model of the environment and that simplicity contributes to robustness.

The localization problem can roughly be divided into three parts, *pose tracking, global localization and map acquisition.* This separation is to a large extent based on the way people have approached the problem.

1.2.1 Pose Tracking

In many applications an initial estimate of the robot pose[1] is known. During the execution of a task, the robot must update this estimate using measurements from sensors. Using only sensors that measure relative movements, the error in the pose estimate increases over time as errors are accumulated. Therefore external sensors are needed to provide information about the absolute pose of the robot. This is achieved by matching these measurements with an environmental model (the map). Finding the correspondence between the measurements and the model of the environment is one of the hardest problems in any estimation process. In pose tracking a good initial estimate is given and the correspondence or data association problem becomes easier as it is not necessary to consider the entire space when looking for the correspondences, but rather only a relatively small region around the estimated pose.

1.2.2 Global Localization

Tracking can be used if an initial estimate of the pose is given. However, in some situations, and for a fully autonomous service robot in particular this is not possible. The process of finding the pose starting with no or very limited pose information is called global localization or pose initialization. It is considerably more difficult than pose tracking because of the data association problem. The level of complexity of this task varies with the size of the environment, but also with the level of symmetry. It is only by integrating large amounts of data over time that these symmetries can be resolved.

1.2.3 Map Acquisition

Both pose tracking and global localization require a map of the environment. In some cases a map can be acquired from a plan of a building, but in other cases the map representation is such that it must be constructed from sensor data by the robot. The position of the robot must be known to build the map using sensor data from the robot.

1.2.4 Closing the Loop: SLAM

An autonomous robot must perform both pose estimation and mapping. However, since pose estimation requires a map and mapping requires the pose, there is a "chicken and egg" problem. Which comes first? The answer to the question is that they have to be carried out at the same time. The process of building a map at the same time as estimating the pose of the robot is called *simultaneous localization and mapping* (SLAM). SLAM is different from ordinary map acquisition since the uncertainty in the robot pose is accounted for when

[1]pose: position and orientation

building the map. The correlation between the estimate of the robot pose and the map that is being constructed is thus explicitly modeled.

1.3 Thesis Outline and Contributions

In this section brief reviews are given of the different chapters together with a list of publications.

Chapter 2

Localization is to a large extent a question of representing uncertain information about the pose of the robot and about the environment (the map). There is a tight coupling between how the map is represented and the way the pose estimation is performed. This chapter gives an overview of the most common approaches. The representations used for maps are divided into four groups: *topological*, *feature-based*, *grid-based* and *appearance-based*. Seven different groups for representing uncertain pose information are presented. Spanning from not representing it at all (dead-reckoning), over analytic representations using a single or a mixture of Gaussians to discretization techniques. Based on this review a strategy for further investigation is discussed. Finally the minimalistic map representation used in this thesis is presented.

Chapter 3

Pose tracking, as previously explained, is the process of maintaining an estimate of the robot pose, given a good initial pose estimate. In the chapter a Kalman filter based approach is presented utilizing the minimalistic environmental model mentioned in the previous section. By frequently updating the pose, the correspondence problem is simplified. A Gaussian PDF is used to represent the pose uncertainty, which is a good approximation as long as the data associations are solved. The minimalistic model paves the way for a low-complexity algorithm with a high degree of robustness and accuracy. Robustness here refers both to being able to track the pose for a long time, but also handling changes and clutter in the environment. The robustness is gained by the minimalistic model only capturing the stable and large scale features of the environment. The effectiveness of the pose tracker is demonstrated through a number of experiments, including a run of 90 minutes which clearly establishes the robustness of the method. The results presented in the chapter have been published in:

- Jensfelt, P. & Christensen, H. (1998), Laser based position acquisition and tracking in an indoor environment, *in* 'Proc. of the International Symposium on Robotics and Automation', Vol. 1, IEEE, Saltillo, Coahuila, Mexico, pp. 331–338.

- Jensfelt, P. & Christensen, H. (1999), Laser based pose tracking, *in* 'Proc. of the International Conference on Robotics and Automation', IEEE, Detroit, Michigan, USA, pp. 2994–3000.

- Jensfelt, P. (1999), Localization using laser scanning and minimalistic environmental models, Licentiate thesis, Automatic Control, Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

- Jensfelt, P. & Christensen, H. I. (2001), 'Pose tracking using laser scanning and minimalistic environmental models', *to appear in IEEE Transactions on Robotics and Automation* .

Chapter 4

When an initial pose estimate is not available, the pose estimate must be automatically initialized. It is needed not only when starting on the robot, but also in case the pose is lost during a mission. With global pose uncertainty, it is no longer possible to use a simple Gaussian PDF.

In the chapter a novel approach based on tracking multiple pose hypotheses is presented. The method is inspired by techniques used in radar target tracking. Although similar techniques are applied in other fields this contribution presents the first thorough investigation of how this can be applied to mobile robot localization. The method proposed is called Multiple Hypothesis Localization (MHL). The results have been presented in:

- Jensfelt, P. & Kristensen, S. (1999), Active global localisation for a mobile robot using multiple hypothesis tracking, *in* 'Proc. of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation', Stockholm, Sweden, pp. 13–22.

- Jensfelt, P. & Kristensen, S. (2001), 'Active global localisation for a mobile robot using multiple hypothesis tracking', *accepted for IEEE Transactions on Robotics and Automation* .

Chapter 5

Monte Carlo methods have recently become popular in robotics. In this chapter the performance of the Monte Carlo Localization (MCL) method is evaluated in the minimalistic model setting. Two different approaches are presented to improve the performance and reduce computational complexity. The first of these has been suggested before in the literature, but this chapter provides the first thorough evaluation of its characteristics. The second approach is a novel way of reducing the computational effort, while at the same time improving the performance significantly. These results have been published in:

- Jensfelt, P., Austin, D., Wijk, O. & Andersson, M. (2000), Feature based condensation for mobile robot localization, *in* 'Proc. of the International Conference on Robotics and Automation', San Francisco, CA, USA, pp. 2531–2537.

- Jensfelt, P., Wijk, O., Austin, D. & Andersson, M. (2000), Experiments on augmenting condensation for mobile robot localization, *in* 'Proc. of the International Conference on Robotics and Automation', San Francisco, CA, USA, pp. 2518–2524.

Chapter 6

To have a fully autonomous robot, the map that is required for pose tracking and global localization must be constructed automatically. As previously discussed this requires performing localization and mapping at the same time. Most of the approaches presented in the literature so far are based on the Kalman filter. Due to the computational complexity of these methods they are only applicable in small environments. Much of the research have been focused on different approximations to reduce the computational complexity. One such approach is to divide the map into submaps. SLAM is performed in each of the submaps and the computational complexity can thus be made independent of the size of the environment. The price for this is a suboptimal algorithm.

In the chapter SLAM algorithm is evaluated for building minimalistic models. The map scaling issue is addressed by a submap approach, where two different strategies are evaluated. A comparison is presented between the standard SLAM algorithm and the two different submap allocations strategies. Finally a scenario for how mapping can be performed in a service robot application is presented and the benefits of adding extra degrees of freedom to a laser scanner using a pan-tilt unit is investigated.

Appendix A

As most platforms so far in robotics research are 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*, which can provide information about the change in the pose of the platform. As with all sensors, a model of the odometry provides valuable information about performance and limitations.

The main contribution is the development of a model for the odometric system on a synchro drive robot. The model can be used in an iterative update procedure, such as a Kalman filter, and provides uncertainty estimates that are independent of how the path is segmented.

Appendix B

Together with the odometry, the laser sensor is the main sensor in the thesis and as such warrants a thorough investigation. The main contribution here is the characterization two laser scanners from SICK, the PLS and the LMS. The effect of quantization and the underlying range distribution are investigated as well as the beam width and effects that arise when only a fraction of the beam is reflected by an object.

1.4 Publications not covered in this thesis

Results that are only briefly touched upon in this thesis, but which concern related topics have been published in:

- Wijk, O., Jensfelt, P. & Christensen, H. (1998), Triangulation based fusion of ultrasonic sensor data, *in* 'Proc. of the International Conference on Robotics and Automation', Vol. 4, IEEE, Leuven, Belgium, pp. 3419–24.
- Austin, D. & Jensfelt, P. (2000), Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization, *in* 'Proc. of the International Conference on Robotics and Automation', San Francisco, CA, USA, pp. 1036–1041.
- Seiz, M., Jensfelt, P. & Christensen, H. I. (2000), Active exploration for feature based global localization, *in* 'Proc. of the International Conference on Intelligent Robots and Systems"', pp. 281–287.
- Jensfelt, P., Austin, D. & Christensen, H. I. (2000), Toward task oriented localization, *in* 'The 6th Int. Conf. on Intelligent Autonomous Systems (IAS-6)', pp. 612–619.

Chapter 2

Approaches to Localization

> "*While other aspects of the overall navigation problem, such as path planning, are important, in practice, the major difficulty in achieving reliable navigation is in maintaining a good estimate of the robot's whereabouts.*"
>
> (Simmons & Koenig 1995)

Localization is one of the fundamental problems of robotics, and the amount of work presented in the literature is tremendous. Before proposing any new ideas it is important to take past experiences into account, so that design decisions are well founded. Therefore, this thesis begins with a survey of different approaches to localization. The overview focuses on methods for pose tracking, global localization and ways of representing the map. The combined problem of simultaneous localization and mapping (SLAM) is further discussed in Chapter 6.

The two fundamental issues that have to be dealt with when designing a localization system is how to represent uncertain information about the environment and the robot pose. Localization is thus to a large extent about representing uncertainty. Section 2.1 presents different approaches to representing the environment, i.e. map representations. Section 2.2 is dedicated to methods for estimating the robot pose, i.e. pose tracking and global localization. Based on the material presented in the initial two sections, Section 2.3 proposes a strategy for the investigation in the remainder of the thesis, where the first step is to decide upon a representation for the map which is done in Section 2.4.

2.1 Environmental Representations

There are many ways to represent the knowledge about an environment. As stated already in Chapter 1, this thesis deals with indoor semi-structured environments. Typical examples of such environments are domestic and office settings. The assumptions and decisions made in the thesis do not necessarily apply to other types of environments, for example an outdoor scene.

In (Thrun & Bücken 1996) two major paradigms are identified for mapping indoor environments; *grid-based* and *topological*. This disregards a large part of the literature (Crowley 1985*a*, Moutarlier & Chatila 1990, Leonard & Durrant-Whyte 1991*a*, Arras & Siegwart 1997) which use feature based representations. In (Chatila & Laumond 1985) a distinguish is made between three different world models ; *geometric*, *topologic* and *semantic*. The semantic model is used at high decision making levels and is thus not of primary interest for localization. This leaves *geometric* and *topological* world models which is in agreement with the grouping made in (Engelson & McDermott 1992). A distinction is made between the grid based and feature based methods in the thesis. Furthermore, methods that use sensor data directly to represent the environment is dealt with separately under the name appearance based methods.

The border between the metric and the topological representations is not razor sharp. In (Hu et al. 1991) a hybrid approach is used, where the topologically based map is used for planning and the metric feature map is used for localization. A known example of a topological map used for planning is the TOUR model presented in (Kuipers 1977, Kuipers 1978).

When the situation allows it, there is no reason not to engineer the environment by adding for example bar codes or reflective tapes. A typical example of such a situation is a factory. The cost of adding reflective tapes or bar codes is not significant and the robot often uses the same route, going back and forth between two positions. The environment is furthermore easily controlled and not sensitive to bar codes on the walls or colorful stripes in the floor. An example of a commercially available product is the laser based navigation system LazerWay for autonomous vehicles from Netzler & Dahlgren Co AB (NDC n.d., Åström 1996).

For the consumer market, the cost issue is much more important, as installation of bar codes or reflective tapes typically requires a level of knowledge that an ordinary customer does not possess. Furthermore we have much higher aesthetic demands in our own homes than in a factory. One might argue that using the global position system GPS would not need a change of the environment. The catch here though is that it does not work indoors.

2.1.1 Outline

The following four map representations are considered in more detail in Sections 2.1.2-2.1.5:

Topological maps:
The world is represented as a connected graph. The nodes in the graph correspond to places of importance and the edges are the connections, such as corridors.

Feature maps:
Geometric features are used to represent the world. Common examples are points and lines.

Grid maps:
Instead of restricting the map representation to specific features, typically determined by the user, the world is divided into a grid. Each cell represents a small area/volume of the world and is given a certainty value expressing the chance of that part of the environment being occupied.

Appearance based methods:
Instead of having an intermediate representation of sensor data in the form of grids or features, sensor data is used directly to form a function from sensor data to pose space.

2.1.2 Topological Maps

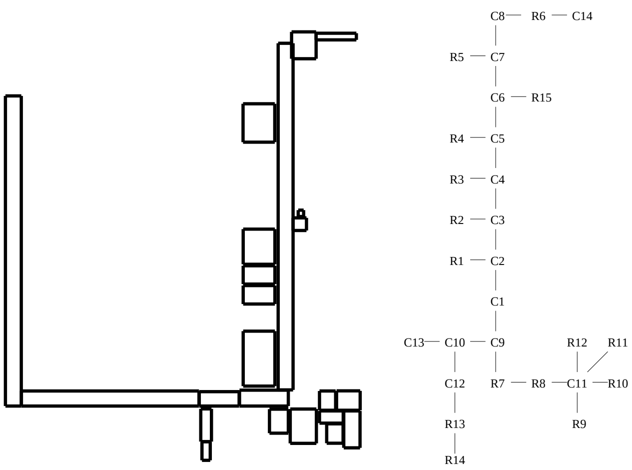
The topological map is often defined by the structure of the environment. In (Chatila & Laumond 1985) the concept of *place* is defined as an area that is a functional or topological unit. Examples of topological units are corridors and rooms, whereas a printer is a functional unit. Connectors are used to connect the places, e.g., doors, stairways and elevators. The topological world model is a connected graph where the places are nodes and the connectors are edges. Topological maps can be organized as a hierarchical structure. For example, at a low level a place might be a room, but at a higher level a building or a city.

Brooks builds a topological map based on visual information. Here the building blocks are called freeways and meadows (Brooks 1984). Freeways correspond to edges and meadows are nodes.

One important special case of an indoor environment is an office space. The office environment typically consists of long corridors, some open spaces and offices. The corridors often form a grid with each corridor running either left-right or up-down. Most mobile robot research is conducted in such office environments, which is reflected in many of the approaches.

One of the clear advantages of the topological approaches is that they are abstract and can be constructed without knowing the exact geographical relation between the different nodes. It is enough to know on which edge the robot is traveling when leaving a node. In the office environment discussed above there are only 90° angles between the corridor parts, and thus only four directions to distinguish between. This can easily be achieved using odometry (Horswill 1998).

Figure 2.1 shows a floor plan of the lower floor of the mobile robot lab at KTH along with a possible topological description of it. In the right subfigure R:s mark rooms and C:s are corridor connections or corridor areas outside a door.



(a) Metric map **(b)** Topologic map

Figure 2.1: *(a) A metric description of the mapped part of the lower floor of the mobile robot lab. (b) Topological map of the same environment. R marks rooms and C is a corridor connection or the area outside a door.*

To achieve localization in a topological map the robot must be able to tell the nodes apart. For a human this is often quite easy, but for a robot this can be a challenging task. Associated with the nodes, there are landmarks, signatures or features. "*A place corresponding to a node must be locally distinctive within its immediate neighborhood by some criterion definable in terms of sensory input*", as put by (Kuipers & Byun 1991), where 16 sonars and a compass are used to distinguish between places.

In (Kortenkamp & Weymouth 1994) so-called *gateways* are used as nodes. These gateways mark the transition between two spaces in the environment. The idea is that the sonar sensor does not give a rich enough description of a place to make it distinct. That is, it cannot decide which place it is, but it can however still be used to detect a place. Place recognition is performed using vision. Examples of gateways are doors and corridor intersections. Gateways are typically traversed in two or a small number of directions and thus provide a limited set of view for the robot at these locations.

In (Koenig & Simmons 1998) the topological map is augmented with rough metric information about the length of the edges (corridors). Assumptions about the environment also include all corridors being two meters wide, intersections having 90 degree angles and doorways being no more than ten meters apart. It is pointed out that there might be valuable information in between the topological nodes which is discarded in a purely topological approach.

A topological map representation is also used in (Cassandra et al. 1996) where the focus is on how to choose the action to take. POMDPs (Partially Observable Markov Decision Processes) form the basis for the decisions and various heuristics are presented. As noted in for example (Simmons & Koenig 1995) a full implementation of POMDP is still computationally infeasible.

In (Ulrich & Nourbakhsh 2000) the robot is led through the environment taking pictures with a panoramic color CCD camera at 1 Hz. The user then labels the images with the location where they where taken. The labeling is made easy because of the sequential images. It is claimed that the color images contain enough information so that range sensor data is not needed. Instead of using the raw images, one-dimensional histograms are extracted. The histograms have the advantage of being two orders of magnitude smaller in memory than the raw images and being invariant under rotation.

2.1.3 Metric Maps: Features

The idea of trying to extract features from the environment is quite natural, this is for example how most city maps are constructed. Extreme examples of features are labels on doors which specify the room it is leading to, but other less discriminative features can also be found. In a structured environment, which most office environments are examples of, lines, corners and edges are common features. The features can be parameterized by, for instance, their color, length, width, position, etc. A feature based map can in general be written

$$\mathcal{M} = \{f_j \mid j = 1, \ldots, M\}, \tag{2.1}$$

where f_j is a feature and M is the number of features in the map.

Leonard, Durrant-Whyte and Cox are quite firm in their belief 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*" (Leonard et al. 1992). Durrant-Whyte also promotes 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*" (Durrant-Whyte 1988).

In (Chatila 1985) a polygonal map is used to predict the readings from a laser range finder. Crowley also uses lines (Crowley 1985*b*, Crowley 1985*a*), where each line segment has an attribute which represents the degree of confidence. When a line is predicted using a local map and later detected, its confidence increases, whereas the confidence decreases when a predicted line is not seen. Drumheller also uses the line primitive for doing sonar based localization (Drumheller 1987). Arras and Tomatis complement horizontal lines extracted from laser range data with vertical lines extracted from vision (Arras & Tomatis 1999). The list of researchers using the line feature can be made long, some more examples are (Cox 1989, Moutarlier & Chatila 1990, Forsberg, Larsson, Åhman & Wernersson 1993, Weckesser & Dillmann 1997, Jensfelt & Christensen 1999).

John Leonard and Hugh Durrant-Whyte develop a localization method based on tracking natural geometric beacons (Leonard & Durrant-Whyte 1991*a*, Leonard & Durrant-Whyte 1992), where 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 (Kuc & Siegel 1987), lines, corners and edges are considered as candidates for geometric beacons. The beacons are extracted from densely sampled sonar data.

Localization using artificial landmarks is well understood and reliable, but requires modified environments. When using natural landmarks for localization, one of the problems is to find suitable candidates for such, a process which often requires the designer of the algorithm to give geometrical specifications, e.g. points, lines, corners, etc. This is a cumbersome task. In (Thrun 1998) a method based on artificial neural networks is presented for automatically selecting good landmarks. Results are presented to show that it outperforms systems where the landmarks are selected manually.

In (Faugeras et al. 1986) 3D features are points, lines and planes from visual data, while in (Rencken 1993) lines and edges are extracted from sonar data.

In (Castellanos & Tardós 1999) many different geometric features are used. Besides the common point and line features, edges, corners, semi-planes and vision edges are used. A scheme based on so-called binding matrices is used when pairing features of different types.

An extreme example of feature based localization is the work by Christensen *et al* (Christensen et al. 1994) where a complete CAD model of the environment is used for localization and pose tracking through feature matching and 3D recovery using stereo vision.

2.1.4 Metric Maps: Grids

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 (Hager & Mintz 1990) points out, "*The grid-based method is only an approximative solution, but it is much less sensitive to assumptions about the particular form of the sensing system*". In (Raschke & Borenstein 1990) it is stated that the grid-based models are better suited for dealing with sonar data than line-type models (features).

Moravec and Elfes made the grid based techniques popular with their paper (Moravec & Elfes 1985), where 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 occupancy grid method, the world is rasterized into cells. Each grid cell contains information about whether or not it is occupied. Different frameworks are used for updating the grid, e.g. Bayesian (Moravec & Elfes 1985, Cho 1990) and fuzzy logic (Fabrizi & Saffiotti 2000, Oriolo et al. 1995). Figure 2.2 gives an example of an occupancy grid built using a Bayesian approach. The trajectory followed by the robot is shown along with the outline of the room. The figure is borrowed from (Wijk 2001), and is a map of the living-room at CAS.

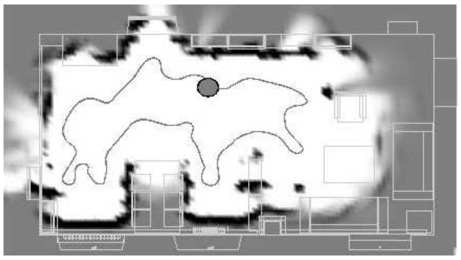


Figure 2.2: *Example of a sonar based occupancy grid, built using a Bayesian approach.*

In the thesis by Alberto Elfes, a thorough description is given of the occupancy grid from the basics to more advanced applications of it, for example, the transformation between a geometric description of the environment to a grid description and vice versa (Elfes 1989). Results are also presented with other sensors than sonar and with multiple sensors.

Already in (Moravec & Elfes 1985), but also in, for example, (Elfes 1989), techniques for finding the position of a robot using the grid representation are discussed by correlating a local map of the environment to a global one. The correlation is performed with different map resolutions to reduce the computational cost.

One the disadvantages of the occupancy grid technique presented by Moravec and Elfes is the computationally heavy updating routine. In (Borenstein & Koren 1989, Borenstein & Koren 1991) a simpler grid updating scheme is applied where only the cells along the acoustic axis of the sonar sensor are updated. A realistic probability distribution is still achieved by fast sampling. The computational burden is reduced significantly.

Schiele and Crowley (Schiele & Crowley 1994) 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) locally extracted line segments to global grid, iii) local grid to global line segments and iv) local and global line segments. 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 segments. Schiele and Crowley also conclude that the results using grids "*are comparable or more accurate than the ones we obtain with previous work using a parametric model of segments extracted directly from sensor data*".

2.1.5 Appearance Based Methods

Appearance based methods represent the environment using raw sensor data or rich descriptions thereof. The downside of the metric methods is that the sensor data itself typically contains a richer description of the environment than the features or the grids. Thus, representing the environment using raw sensor data is not a farfetched idea. By recording sensor data from different positions, the sensor data acts as signatures of these positions. The idea is similar to the way topological places are recognized. The difference is that an attempt is made to create a function from sensor data to an exact robot pose and not only to, for example, a room. If a local one-to-one mapping can be found from sensor data to robot pose, it can be tracked. Having a global mapping allows for global localization by inverting the function. Global one-to-one functions can almost never be found in practice because most environments, exhibit a large degree of symmetry, e.g. corridors.

In (Weiss & von Puttkamer 1995) the environment is divided into grid cells. Whenever the robot enters a grid cell, in which it has not been before, a scan is acquired. These scans acts as reference scans. When returning to an old position the reference scans can be used to update the robot pose estimate. The scan matching algorithm presented in (Lu & Milios 1995, Lu & Milios 1997*b*, Lu & Milios 1997*a*) also uses laser reference scans. In (Gutmann & Schlegel 1996) different scan-matching algorithms are compared. Figures 2.3 gives an example of how a map built using laser scans can look. The raw laser scans are acquired approximately every meter, in groups of four, throughout the environment, giving a total of 148 scans. The raw scans are shown in Figure 2.4, where the odometric drift is clearly visible.

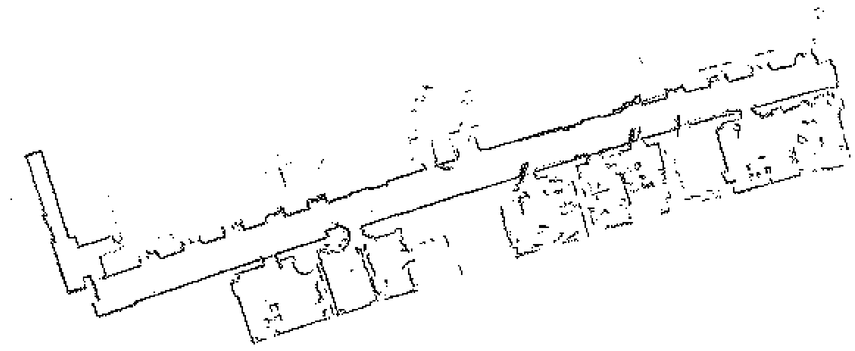


Figure 2.3: *Reference laser scans that are aligned using scan matching. Scans are collected in the area around the main corridor in the mobile robot lab.*

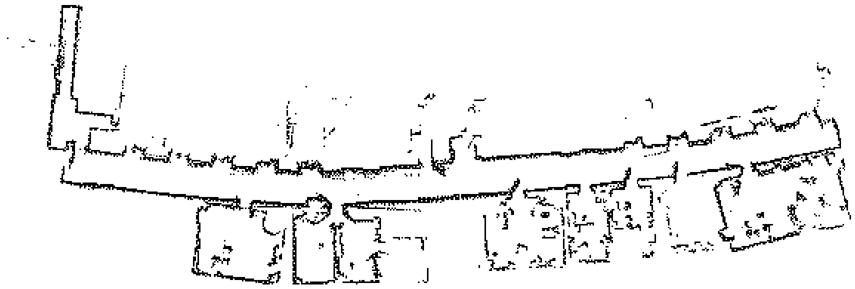


Figure 2.4: *Raw laser scans before being align by scan matching. The scans are acquired approximately every meter in groups of four.*

A method quite different from the others presented above is given in (Wallner 1997, Crowley et al. 1998), where principal components analysis (PCA) is applied to laser range based localization. In (Sim & Dudek 1999) a PCA technique is used to extract landmarks from images. The map is a set of such landmarks. Localization is performed by extracting landmarks and comparing them to the map. PCA techniques are successfully applied to other domains of robotics as well, e.g. object recognition (Murase & Nayar 1995).

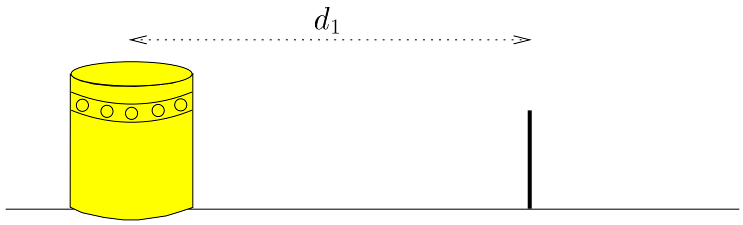


Figure 2.5: *One dimensional example of pose uncertainty situation.*

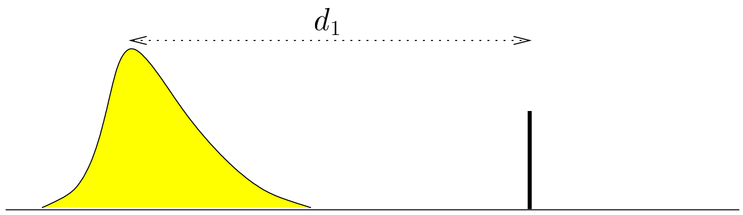


Figure 2.6: *Example of a probability density function after incorporating measurement and knowledge about the sensor.*

2.2 Pose Estimation

In this section an overview of some of pose estimation techniques is given. Some of the methods are only applicable if an initial estimate of the robot is given. Others can cope with global initial uncertainty.

In the Bayesian framework the aim is to find $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$, where $\mathbf{x}_k$ is some time-varying signal and $\{\mathbf{z}_i\}$ are measurements (Sorenson & Stubberud 1968). This problem is known as the Bayesian filtering problem, optimal filtering or stochastic filtering. A more complete description of the state of the system than $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ is not possible (Alspach & Sorenson 1972). However, in general it is inadequate to describe the state with a finite set of parameters. When $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ is well represented by a Gaussian, the mean and the covariance give a full description of the state. Most of the localization methods below represents a way of approximating the distribution $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$. Estimating the pose is thus very much about representing uncertain pose information.

Consider the simple one-dimensional example given in Figure 2.5. The robot is positioned at some distance d_1 away from a vertical structure. Assume for the time being that this is the only structure in the environment. The probability density function (PDF) after a measurement of the structure may look like in Figure 2.6.

In a real world scenario the pose space is not one-dimensional, but rather three-dimensional[1]. Sensors are not perfect and the environment contains many, many structures. Some of the structures might also be hard to tell apart. The real challenge in mobile robot localization lies in performing correct data associations between sensory data and the environmental model. If the matching can be done, the localization problem is reduced to a standard estimation problem, where the sources of the all measurements are known.

Most indoor mobile robots are equipped with wheel encoders. Knowing the size of the wheel and the wheel configuration makes it possible to estimate the motion of the robot. The information given in this way if referred to as odometric information. Figure 2.7 shows the PDF after the robot has moved to the right. Using the odometric information to predict the new pose of the robot makes the PDF spread out. The better the odometric information is, the closer the predication step is a plain translation of the PDF. From this example it is easy to understand that a good understanding of the odometry is needed to make an accurate prediction.

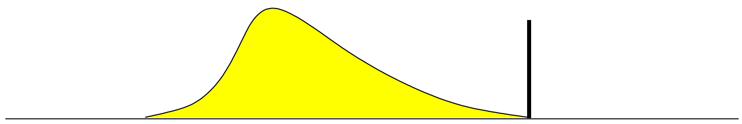


Figure 2.7: *Probability density function after robot motion before measurement update.*

The pose estimation problem can be broken down into two steps, *prediction* and *measurement update* as in Figure 2.8. The prediction step is in principle not needed if the robot at any point in time can determine its pose with enough accuracy directly and unambiguously from sensor data. In real world applications this is rarely the case.

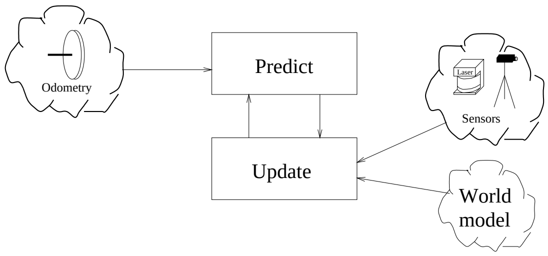


Figure 2.8: *Pose estimation can, in general, be divided into two steps,* prediction *and* measurement update.

[1]In general it is a six dimensional space for a rigid body. Motion in the plane and rotation around one axis reduces it to three dimensions though.

2.2.1 The Data Association Problem

One of the fundamental problem in mobile robot localization and all sensor based estimation is that of data association. Data association refers to the process of determining the origin of the measurements. An important part is also to reject outliers, as unmodeled objects in the environment cannot be used to localize. Either the map is augmented with the new structure or the measurement coming from it must be neglected.

Let $\mathbf{z}$ be a measurement and let $\mathcal{M}$ be the environmental representation (the map). Data association can then be defined as for each $\mathbf{z}$ finding a correspondence in $\mathcal{M}$. How this correspondence is defined depends on the representation of $\mathcal{M}$. With a feature based map (2.1) the data association problem consists of classify $\mathbf{z}$ as either

- a measurement of a map feature $f_j \in \mathcal{M}$, or
- a measurement of something not represented in the map.

2.2.2 Outline

In the remainder of this section different methods for estimating the robot pose are discussed. The focus is on the representation of pose uncertainty. The methods are grouped into

Dead-Reckoning:
Having an initial estimate and odometric information makes it possible to calculate the motion of the robot. No absolute pose information is used and thus the error in the pose estimate is unbounded.

Topological:
Just like the map can be represented as a graph, the knowledge about the pose can be kept in a graph structure. The location of the robot is not given as coordinates, but rather as being in a particular place, such as a room.

Gaussian PDF:
The most used representation for uncertain robot pose information is the Gaussian PDF, often applied in conjunction with the Kalman filter.

Gaussian Sum PDF:
Multiple Gaussians can be used to represent multi-modal distributions which arise as soon as the uncertainty becomes too large. A bank of Kalman filters is one way to implement this approach.

Position Probability Grids:
By dividing the pose space into small cells, each representing one possible pose, a discrete approximation of the PDF can be made. With a fine enough discretization of the state space, any functional form of the PDF can be represented.

Monte Carlo Methods:
The Monte Carlo methods use a set of samples to represent the PDF. Any PDF can be represented given that the number of samples is infinite. Finite sized sample sets allow for an efficient approximation of the PDF.

2.2.3 Dead-Reckoning

Some tasks only require the robot to move short distances. If the robot has a good initial pose estimate, odometric information can be used to keep track of the pose of the robot. Here the odometric information is considered deterministic. As soon as the robot moves over longer distances, external sensor are required to keep the pose uncertainty bounded.

2.2.4 Topological

Localization in a topological map is not about finding the (x, y)-position of the robot, but rather finding on which node the robot is located. In the topological approaches, the accuracy of the odometric information is often less important. However, odometry is still used to indicate approximately in which direction the robot is moving.

In (Nourbakhsh 1998) it is argued that "*the real-world is so fine grained that any attempt to plan using a highly detailed model of reality to solve this problem (localization) is doomed because of enormous computational complexity*". A simple sonar based recognition system is used to determine if the robot is in a hallway, an intersection or if there are any doors open or closed on the sides. Dervish, as the robot is called, operates in a typical hallway environment. When going from one node to the next in the graph, the sonar sensors are monitored. Knowing the width of the corridor makes it possible to tell if a new node is reached. The hallways have the same width everywhere and thus a small change in width can be associated with a closed door. Open doors or intersecting hallways both show up as openings to the sonar sensors. To tell the two apart, odometry is used to measure the width of the opening. Doors are assumed to be more narrow than hallways. Each node in the graph is given a certainty values, which represents the relative likelihood of the robot being there. When the robot moves the certainty values are propagated along the edges. When reaching a new place the expected characteristics of the place can be compared with the measured one and through this process the true position can be singled out.

By making the recognition of places better the robot does not have to move as much. This is the idea pursued in (Kortenkamp & Weymouth 1994) where the sonar sensor is only used to detect a place. Vision is then used to recognize which place the robot is at.

Simmons and Koenig (Simmons & Koenig 1995) use a partially observable Markov model to estimate the pose of the robot. The topological model is similar to (Nourbakhsh et al. 1995), but is augmented to incorporate the approximate distance between the topological nodes. Where Nourbakhsh *et al* only encode the pose of the robot to be on a node or an edge, Simmons and Koenig discretize the edges into one by one meter squares. Each of these squares are further divided into four different states, one per possible direction of motion, i.e. north, west, south and east. Using only four directions is motivated by the assumption of all corridors having right angles. In (Koenig & Simmons 1998) it is said that *"for office navigation, the added expressiveness of being able to model arbitrary pose distributions outweighs the decrease in precision from discretization, especially since coarse-grained representations of uncertainty is often sufficient"*. The corridor edges connecting the topological nodes are modeled as Markov chains (see Figure 2.9).

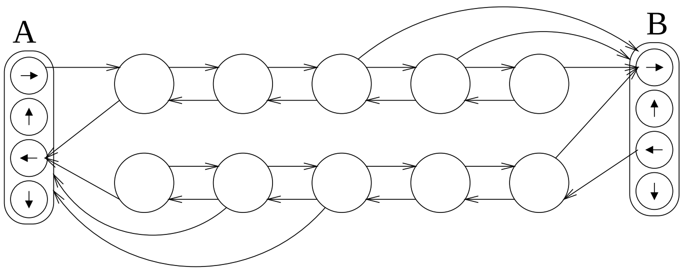


Figure 2.9: *The corridor edge connecting two topological nodes A and B is modeled as two Markov chains in (Simmons & Koenig 1995). Intermediate states corresponding to the same position are shown as one circle for illustration purposes.*

Depending on if the robot moves from A to B or from B to A it enters a different chain. This setup incorporates the fact that it is often easier to calculate how far the robot has traveled from a node than it is to calculate how far there is to the next one. To account for the uncertainty in length, a finite transition probability is given to jump from intermediate states to the end node. In the example in Figure 2.9 the corridor length is between three and five meters. Each state holds a probability of the robot being there. The control action, a, sent to the robot is used to update the probability of the different states, s, according to (Simmons & Koenig 1995)

$$p(s'_{k+1}) = K \sum_{s_k} p(s'_{k+1}|s_k, a)p(s_k), \tag{2.2}$$

where $p(s'_{k+1}|s_k, a)$ is the transition probability from s_k to s'_{k+1} given the action a and K is a normalization factor. Observations, z, from external sensors are used in a similar way to update the probabilities

$$p(s_{k+1}) = Kp(z|s'_{k+1})p(s'_{k+1}), \tag{2.3}$$

where $p(z|s'_{k+1})$ is the probability of making observation z from state s'_{k+1}.

In (Ulrich & Nourbakhsh 2000) the map consists of reference image histograms acquired with a panoramic color CCD camera. When using the map for localization new images are captured and the same type of image histograms are created. Each image gives six histograms, one for each color band in the HSV and normalized RGB spaces. These histograms are then matched against the reference histograms. Each color band histogram carries one vote. The output is an image classification and a confidence measure. Knowing the initial position of the robot limits the range of reference histograms that has to be considered in each step.

2.2.5 Gaussian PDF

The Gaussian distribution is the most commonly used way to represent the uncertainty of the robot. There are good reasons for this as pointed out in for example (Maybeck 1979). First, because there are often many different independent sources of uncertainty. When they are added together they approach the Gaussian function. Secondly, the knowledge about the uncertainty is seldom larger than knowing the first and second moment. The strongest motivation though ought to be that with a Gaussian assumption the filtering distribution $p(\mathbf{x}_k|\mathbf{z}_0,\ldots,\mathbf{z}_k)$ can be evaluated exactly (Doucet 1998). In one dimension the Gaussian function is completely specified by the mean $\bar{x}$ and the variance σ_x^2 according to

$$f(x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\bar{x}}{\sigma_x}\right)^2}. \tag{2.4}$$

Using a Gaussian approximation of the PDF in Figure 2.6 might yield Figure 2.10.

Figure 2.10: *Using a Gaussian approximation.*

When going to higher dimensions, the value to estimate becomes a vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ and the scalar variance becomes a covariance matrix $P = E(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T$. The Gaussian PDF is now given by

$$f(\mathbf{x}) = \frac{1}{\sqrt{|P|(2\pi)^N}} e^{-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})P^{-1}(\mathbf{x}-\bar{\mathbf{x}})^T}. \tag{2.5}$$

The Kalman Filter

The Kalman filter is a key component in many implementations using the metric map representations (especially the feature based ones), providing a good setting for pose prediction, sensor fusion and data association. It originates from the 1960's (Kalman 1960) and is described in numerous books, e.g. (Gelb 1974, Maybeck 1979, Anderson & Moore 1979). How to use the Kalman filter for mobile robot localization is presented in (Crowley 1995) and (Durrant-Whyte 1994). An introduction to the Kalman filter can be found in (Kailath et al. 2000, Welch & Bishop 2001).

As put by (Maybeck 1979), "*the Kalman filter is simply an optimal recursive data processing algorithm*". The key here is that it is a recursive algorithm, which means that measurements can be incorporated into the estimate as they arrive and that no batch processing is required. Having to process all measurement data at every time step soon becomes too time consuming. The Kalman filter provides the optimal estimate of the quantity $\mathbf{x}$ given some measurements $\mathbf{z}$ if three assumptions hold. The system must be *linear* and the noise associated with the process model and the measurements must be *white* and *Gaussian*. White noise is not correlated in time and has the same power for all frequencies, which in turn implies infinite noise energy. Hence, white noise does not exist in reality. Having the same power for all frequencies is only of practical importance if the system effected by the noise has an infinitely large bandwidth. Such systems do not exist either and thus as long as the noise has the same power for all frequencies within the bandwidth of the system it can be considered white.

The Kalman filter algorithm can be divided into two parts, a prediction or time update step and a correction or measurement update step (see Figure 2.8). Let $\mathbf{x}_k$ denote the true pose at time step k. Let furthermore $\mathbf{u}_k$ be the input signal to the system, which could be control signals to the motors or odometric data. Here $\mathbf{x}_k$ is referred to as the state of the system and captures everything there is to know about the robot pose. This is the key to the recursive algorithm, everything that previous measurements have provided about the pose is encoded in $\mathbf{x}_k$. Assuming that the system is linear it can be written as

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + G\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{2.6}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{2.7}$$

where F is referred to as the system matrix. The matrix G specifies how the input effects the state (the pose), H is the measurement matrix, $\mathbf{w}_k \sim N(0, Q_k)$ is the process noise and $\mathbf{v}_k \sim N(0, R_k)$ is the measurement noise. The processes $\mathbf{w}_k$ and $\mathbf{v}_k$ are assumed to be independent. The matrix H encodes the map, i.e. the relation between robot pose and measurements.

In a perfect world[2] where F and G are known without error and with no process noise, (2.6) can be used together with the input signal $\mathbf{u}_k$ to keep track

[2]Some might argue against this definition of a perfect world

of the robot pose if the initial pose is known. This is basically the assumption made when the odometry is used without any external measurements, in dead-reckoning.

In the presence of noise and with an imperfect model there are errors. Let $\hat{\mathbf{x}}_{k|k-1}$ denote the *a priori* estimate of the robot pose at time step k using all measurements until that time $k-1$. Let also $\hat{\mathbf{x}}_{k|k}$ denote the *a posteriori* estimate of the pose when all measurements until time k are used. To keep track of the quality of the measurements, $P_{k|k-1}$ denotes the *a priori* estimation error covariance and $P_{k|k}$ the *a posteriori* ditto. The prediction or time update step of the Kalman filter is given by

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= F\hat{\mathbf{x}}_{k-1|k-1} + G\mathbf{u}_{k-1} \\ P_{k|k-1} &= FP_{k-1|k-1}F^T + Q_{k-1}. \end{aligned} \tag{2.8}$$

The measurements $\mathbf{z}_k$ taken from the system are used to correct the estimate. The innovation $\nu_k = \mathbf{z}_k - H\hat{\mathbf{x}}_{k|k-1}$ provides information that the Kalman filter is unable to predict and is small when the prediction is good.

$$S_k = HP_{k|k-1}H^T + R_k \tag{2.9}$$

$$K_k = P_{k|k-1}H^T S_k^{-1} \tag{2.10}$$

$$\nu_k = \mathbf{z}_k - H\hat{\mathbf{x}}_{k|k-1} \tag{2.11}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k\nu_k \tag{2.12}$$

$$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{2.13}$$

where K_k is the Kalman gain and S_k is the covariance of the innovation process.

Many, not to say most, real systems are non-linear. Approximation techniques to handle non-linear systems surfaced shortly after the Kalman filter was suggested (Cox 1964). The extension to the non-linear case is called the extended Kalman filter (Jazwinski 1970), and is nothing but a linearization of the system around the current state estimate. The success of the EKF depends on how well the system is approximated by the linearization. The non-linear systems considered in the thesis can be written as

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \tag{2.14}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k). \tag{2.15}$$

To get the *a priori* estimation error covariance, the Jacobian of the function $\mathbf{f}$ has to calculated with respect to $\mathbf{x}$ and $\mathbf{w}$. Defining

$$F_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, 0} \tag{2.16}$$

$$W_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}_k}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, 0}, \tag{2.17}$$

the time update equation of the EKF are given by (2.18)

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, 0) \\ P_{k|k-1} &= F_k P_{k-1|k-1} F_k^T + W_k Q_{k-1} W_k^T . \end{aligned} \tag{2.18}$$

For the measurement update the Jacobians of the function $\mathbf{h}$ with respect to $\mathbf{x}$ and $\mathbf{w}$ are needed,

$$H_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}\right|_{\hat{\mathbf{x}}_{k|k-1},0} \tag{2.19}$$

$$V_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{v}_k}\right|_{\hat{\mathbf{x}}_{k|k-1},0} . \tag{2.20}$$

The measurement update equations are given by

$$\begin{aligned} S_k &= H_k P_{k|k-1} H_k^T + V_k R_k V_k^T \\ K_k &= P_{k|k-1} H_k^T S_k^{-1} \\ \nu_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, 0) \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + K_k \nu_k \\ P_{k|k} &= (I - K_k H_k) P_{k|k-1} . \end{aligned} \tag{2.21}$$

Data Association

The standard estimation techniques, like the Kalman filter, assumes that the correspondence between the measurement and the map is known. It is only by knowing this that the measurement equation can be defined correctly. One common way of handling the data association problem is to use the Mahalanobis distance, which loosely speaking is the Euclidean distance normalized with the covariance. The Mahalanobis distance, ρ, is defined as

$$\rho_k = \nu_k S_k^{-1} \nu_k^T . \tag{2.22}$$

The Mahalanobis distance defines a χ^2-distribution (Bar-Shalom & Fortmann 1988). To accept a match between a measurement and its believed origin in the environment the Mahalanobis distance has to fulfill

$$\rho_k \leq \gamma, \tag{2.23}$$

where γ determines the size of the so-called validation gate. The value of γ depends on the dimension of the measurement and with what probability a true match should be classified as such. The size of the validation gate increases with $P_{k|k-1}$ and R_k according to (2.9) in combination with (2.22).

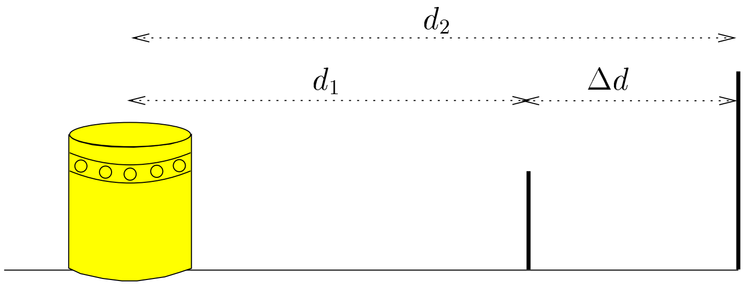


Figure 2.11: *The robot must associate the measurement with the correct vertical structure and thus has a matching problem.*

Consider Figure 2.11, where a second vertical structure is introduced behind the first one. Note also that the closest vertical structure is somewhat lower than the one further away. If the range sensor is mounted high on the platform there is a risk that the sensor overshoots and instead of d_1 measures the distance d_2. If the uncertainty in the pose estimate or the uncertainty in the measurement is large, both vertical structures might pass through the gate. In such a situation the nearest neighbor technique is often applied and the match corresponding to the smallest Mahalanobis distance is taken to be the true one. This example illustrates that the uncertainty which can be allowed in a Kalman filter based system is a function of the complexity of the environment and the quality of the measurements. The smaller Δd is, the harder it is to tell two structures apart and thus the smaller the uncertainty must be kept to avoid ambiguous matchings.

The simple Gaussian representation works well when the uncertainty is small and the PDF is uni-modal, i.e. as long as measurements can be matched to the map unambiguously. When the matching is ambiguous, the Gaussian representation is not anymore suitable. Using the same sensor model and also taking into account the probability of measuring d_1 and d_2 respectively one might arrive at the PDF shown in Figure 2.12 after processing a measurement.

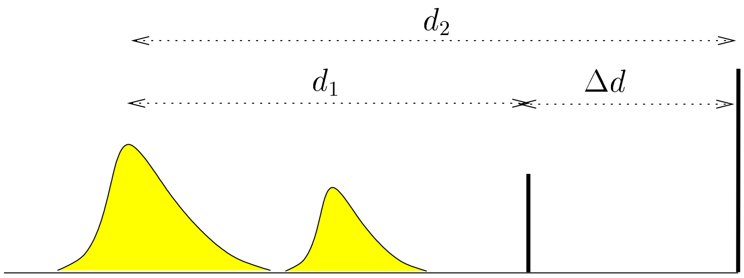


Figure 2.12: *Assuming the nearest object is more likely to reflect, it gives the highest probability for the robot's position.*

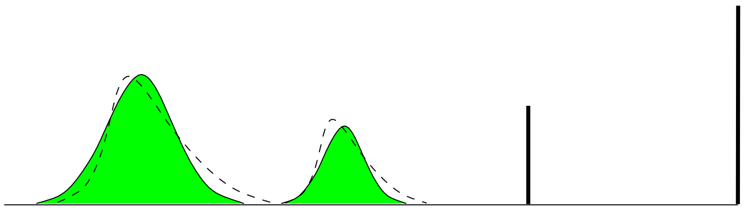


Figure 2.13: *Using multiple Gaussians to represent the PDF.*

2.2.6 Gaussian Sum PDF

It is quite natural to look for an extension of the single Gaussian representation. Figure 2.13 suggests that using two Gaussians is quite close to the true PDF. The PDF is thus a sum or mixture of Gaussians, where each Gaussian can be viewed as one hypothesis about the robot pose (Jensfelt & Kristensen 1999, Roumeliotis & Bekey 2000, Reuter 2000). The advantage of this representation is that it is defined by few parameters, one vector and one matrix for each Gaussian. The EKF can still be applied which allows for efficient computations.

Every time a measurement is matched to the map there is chance that the match is wrong. A proper implementation must take this into account. Each ambiguous match leads to a branching into two hypotheses, one which corresponds to the possibility that the match is correct and one where it is assumed that the match is false. This causes an explosion in the number of hypotheses. To keep down the computation cost the hypotheses tree must be pruned.

2.2.7 Position Probability Grids

The single Gaussian distribution presented in Section 2.2.5 provides high accuracy but is limited to uni-modal distributions. The purely topological methods from Section 2.2.4 can handle multi-modal distributions but the pose information is coarse. The Gaussian sum approximation can both provide high accuracy and represent multi-modal PDFs, but demands advanced data association schemes and pruning techniques. Inspired by the occupancy grids (Moravec & Elfes 1985) and the topological approaches (Nourbakhsh et al. 1995, Simmons & Koenig 1995) the position probability grid came about (Burgard et al. 1996, Fox, Burgard & Thrun 1999). Each cell in the grid corresponds to one possible pose of the robot or, using the same terminology as for the Kalman filter, each cell corresponds to one possible state of the robot. Whereas (Simmons & Koenig 1995) use a discretization of one by one meter and 90° the position probability grid is finer grained to improve the accuracy in pose.

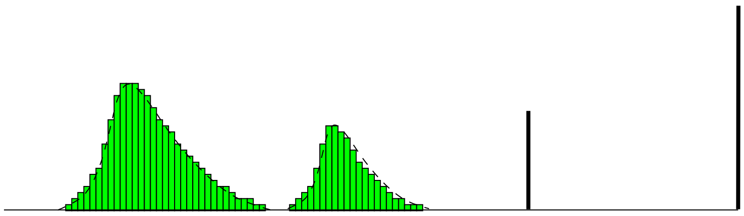


Figure 2.14: *Using a grid approximation of the PDF.*

Going back to the true PDF from Figure 2.12 this could be represented as shown in Figure 2.14. Using the position probability grids for localization involves two steps (Burgard et al. 1996), which are more or less the same as the ones used to update the partially observable Markov model in (Simmons & Koenig 1995). Let x denote a robot state (a cell) in the grid G. Assuming that the robot can only be inside the known world, the probability of cells outside can be put to zero.

1. In the first step the probability values in the grid are updated using information from the odometry. This step has a smoothing effect on the PDF (compare Figure 2.7). The new probability of each cell is given by the probabilities to arrive there from any cells in the grids. Let $p(x|x', \mathbf{u})$ denote the probability that the robot gets to cell x from cell x' given the odometric information $\mathbf{u}$. The probability of each cell is thus given by

$$P(x) = \alpha \sum_{x' \in G} P(x')p(x|x', \mathbf{u}) \tag{2.24}$$

   where α is a normalization factor. As $P(x \notin G) = 0$ by assumption, $\alpha = \sum_{x \in G} P(x)$. This does not account for the fact that parts of the space is occupied according to the environmental map. Therefore the probability of each state is scaled with a factor $\beta \cdot (1 - p(occ(pos(x))))$, where $\text{pos}(x)$ is the position corresponding to state x and $occ(pos(x))$ is the probability that $pos(x)$ is occupied.

2. In the measurement update step, each grid cell is updated with the probability of obtaining the observed measurement from that pose given the map of the environment. This has the intuitive effect that grids cells from which the predicted measurement is close to the actual measurement is strengthened. The computational cost for this step depends to a large extent on the representation used for the map. In (Burgard et al. 1996) an occupancy grid is used as map, in combination with sonar sensors. To make the computations feasible, only a discrete number of possible range readings are considered.

How well the true PDF is approximated depends on the size of the cells in the grid. In (Burgard et al. 1996) each cell is 15×15 cm^2 and an angular resolution of $2°$ is used. Using only 8 sonar sensors in a small 4×4 m^2 single room environment, each update takes 6 seconds on a 90 MHz Pentium computer. To deal with the computational issues (Fox et al. 1998) introduces a more efficient implementation. Two techniques are combined: i) pre-computation of the expected distance reported by the sensor from all states and ii) only use cells with probability higher than some threshold in the computations. The computational burden can in this way be reduced several orders of magnitude. For a larger environment integrating all the information from a sensor scan takes 120 seconds even with the efficient implementation. The grid based methods do not only require a massive computational burden, the memory requirements are also high, in excess of 100MB according to (Fox, Burgard, Dellaert & Thrun 1999).

In (Burgard, Fox & Henning 1997) it is pointed out that the position probability grid can approximate the true PDF better than the single Gaussian used in Kalman filter based methods. The example brought up is the case when the robot is close to an known obstacle, e.g. a wall. In this case the probability should be zero that the robot is located in the wall. The single Gaussian however puts a non-zero probability everywhere which is wrong.

2.2.8 Monte Carlo Methods

Monte Carlo methods have become increasingly popular over the last few years in robotics. The methods have been around since the 60's (Handschin & Mayne 1969, Handschin 1970), but was prevented from being widely used because of their dependency on computer power. During the last couple of decades the computer power has reached a level high enough to make the methods applicable. For a historical overview of the Monte Carlo methods see (Neal 1993). Having been applied in other research fields, they came into robotics primarily via (Isard & Blake 1996, Isard & Blake 1998). Here the so-called CONDENSATION[3] algorithm is presented and used for image contour tracking.

The Kalman filter approaches of Sections 2.2.5 and 2.2.6 uses a Gaussian approximation for $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ and the position probability grid of Section 2.2.7 discretizes it. In the Monte Carlo methods a sample set is used to represent $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$.

The grid approach from the previous section is also, in a way, a sample based method, where the environment is uniformly sampled. With a sample set, the sampling can instead be done where there is a need. Each sample, $s_k^{(i)}$, in the sample set has a corresponding state, $\mathbf{x}_k^{(i)}$ and weight $\pi_k^{(i)}$. Letting $N \to \infty$ the sample set

$$S = \left\{ (\mathbf{x}_k^{(i)}, \pi_k^{(i)}) \mid i = 1, \ldots, N \right\},$$

can be made to approximate $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ arbitrarily well.

[3] CONditional DENSity propagATION

A thorough discussion on Monte Carlo methods can be found in for example (Pitt & Shephard 1999, Doucet 1998) and an introduction to the methods in (Mackay 1996). The Monte Carlo methods were brought into the field of mobile robot localization in 1999 (Dellaert, Fox, Burgard & Thrun 1999, Dellaert, Burgard, Fox & Thrun 1999, Fox, Burgard, Dellaert & Thrun 1999) under the name Monte Carlo Localization (MCL).

Importance sampling is one component in many Monte Carlo methods, and requires that $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$ can be evaluated up to a multiplicative constant, i.e. $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k) = C_p p^*(\mathbf{x}_k)$. Let $q(\mathbf{x}_k) = C_q q^*(\mathbf{x}_k)$ be another function that can be evaluated up to some multiplicative constant. Samples $s_k^{(i)}$, drawn from $q(\mathbf{x}_k)$, are given weights

$$\pi_k^{(i)} = \frac{p^*(\mathbf{x}_k^{(i)})}{q^*(\mathbf{x}_k^{(i)})} \tag{2.25}$$

to compensate for the difference between $q(\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)$.

MCL as well as CONDENSATION is based on what is known as Sampling Importance Resampling (SIR). Here a second step is added to the algorithm, which can be characterized by survival of the fittest and is closely related to genetic algorithms. Looking at the sample set at time k, the samples with the highest weight (the fittest) are more likely to be part of the next sample set. MCL is nothing but SIR applied to mobile robot localization. The algorithm can be summarized in the following steps:

1. Initialize the set by drawing N samples from the initial PDF. In case of global localization, this is typically a uniform distribution, i.e. the known environment is uniformly sampled, both in position and in orientation. The weights of the initial samples are uniformly set to $\frac{1}{N}$.

2. Propagate the density approximation forward using information from the odometry measurements. This corresponds to the prediction step of Figure 2.8. For each sample draw a new sample with pose $\mathbf{x}_{k+1}^{(i)}$ from the distribution $p(\mathbf{x}_{k+1}^{(i)}|\mathbf{x}_k^{(i)}, \mathbf{u}_k)$ where $\mathbf{u}_k$ is the odometric data.

3. Set the weights of the samples to $\pi_{k+1}^{(i)} := p(\mathbf{z}_k|\mathbf{x}_{k+1}^{(i)})\pi_{k+1}^{(i)}$. This gives high weights to samples that corresponds to poses from which it is likely to measure $\mathbf{z}_k$.

4. Create a new sample set by drawing samples from the current set in a way that the probability of letting a sample move on to the next generation is proportional to its weight. After the resampling step all samples are given equal weight again, $\frac{1}{N}$.

5. Go back to step 2.

In loose terms the Monte Carlo localization method can be viewed as a clever way of allocating computational resources where they are needed. In MCL sampling is primarily done where the probability is high. This makes approximation of the PDF better at the same time as resources are not wasted in regions where it is not as important to have a good approximation.

Depending on the size of the environment and the initial uncertainty different sample set size are needed. With 10,000 samples the computational load on a Pentium III 500 MHz is about 14% (Thrun, Fox & Burgard 2000). As a comparison, what took 120 seconds to compute using position probability grids takes 3 seconds (Fox, Burgard, Dellaert & Thrun 1999) using MCL.

For the SIR algorithm to work, enough samples have to be used so that there are samples at the true robot pose. It is only by having samples at the correct pose that the filter converges to the true pose. The required number of samples increase with dimension. This is often referred to as the curse of dimensionality.

2.3 Strategy for Investigation

The goal of the thesis is to investigate different methods for real world, indoor, mobile robot localization, typically in a domestic or office type setting. The localization process must work in real-time and leave enough computational resources for other parts of the system to run in parallel, i.e. it must have a low complexity. Another requirement is that the method is robust and that it scales to large environments and are not only applicable in a small lab area.

From a map complexity point of view the topological map representation is by far the best. However, due to the lack of metric information this representation is ruled out, since this is a prerequisite for some tasks. The grid based methods are general in that few assumptions have to be made about the characteristics of the environment. The price is a high level of complexity that makes these methods scale badly to larger environments. The appearance based methods make the coupling between a certain robot platform and the map quite tight, for example, the reference scans acquired in a cluttered room at different heights will be drastically different. The hypothesis in the thesis is that the feature based method provides the best way to achieve a representation that allows for efficient use of the computational resources. The features should be large scale structures that can be utilized by as many different robot platforms as possible and be robust over time.

Table 2.1 summarizes some of the characteristics of the different representations used when estimating the pose of the robot. Dead-reckoning is not part of the table, since this method is not a valid option when the application is a service robot that will operate for hours, performing various tasks. Given that the map is represented by large scale non-unique features, the purely topological localization scheme can also be disregarded. Left are four approaches to

repr.	Accuracy	Complexity	multi-modal
Gaussian PDF	high	low	no
Gaussian sum	high	medium	yes
Pose grid	medium	high	yes
Sample set	high	medium	yes
Topological	low	low	yes

Table 2.1: *Summary of characteristics for the pose estimation techniques.*

pose estimation. Clearly, if the goal is a low-complexity algorithm, the Gaussian PDF in combination with the extended Kalman filter provides the best setting. To facilitate such a solution the pose uncertainty must be kept small though, which is typically not the case during (re-)initialization. To solve the global localization problem a multi-modal pose distribution must be utilized. The position probability grid and the sample based approach are closely related, but the grid technique suffers from a level of complexity that is not compatible with the goal of scalability. When deciding between the sample based and the Gaussian sum representation the decision is not obvious. The sample based technique is clearly more general, not being restricted by the Gaussian assumption, but there are no large real world examples of the Gaussian sum approximation applied to mobile robot localization.

As a course of action the following is therefore proposed. At first investigate the use of the single Gaussian PDF and the extended Kalman filter in a setting where the initial pose is known (Chapter 3). After that, investigate both the Gaussian sum approximation (Chapter 4) and the sample based representation (Chapter 5). While the topological map representation was ruled out at the lowest level, the advantages regarding the scalability can still be made use of at a higher level. As a final step, investigate methods for building the map with the complexity constraint enforced (Chapter 6) by taking the best from both worlds, i.e. accurate metric information from the feature based representation and scalability from the topological.

2.4 The Minimalistic Model

Much effort is spent in the literature to make more and more complicated models, capturing finer and finer details. In contrast to this the thesis investigates what is termed *minimalistic models*. With this approach the question is not how detailed, but rather how simple can the model be. The idea with a minimalistic model is that it should capture the large scale structure of the environment and that such a description is robust over time. Thinking about a typical indoor environment, the most dominant large scale features are the walls that define the border of the rooms. The walls are not likely to move with time, and if they do it is not by chance. The primary feature in the work to follow is therefore *lines*. Another benefit with walls is that they will appear in a floor plan and can therefore be extracted directly from there. A typical room will in this framework be described by four walls, forming a rectangle, but any polygonal shape can be handled.

Thinking once again of a typical indoor environment and of the rectangular room model it is clear that using only these large scale lines will not be enough in all situations. If a robot needs to find its position in a quadratic room, there is no way to tell the four walls apart. Four equally likely hypotheses results. One remedy is to to allow all detectable lines in the environment to be put into the map. However, this does not comply with the idea of having a minimalistic description of the world. Furthermore the features in the map are no longer robust over time. Having a floor plan still in mind, another feature type that can extracted is the *door*. If door refers to the actual opening, it is also stable over time.

It is evident that using the minimalistic models put high requirements on the sensor being used. To extract the sparse map information a sensor is needed with high angular resolution. Due to the wide beam of the sonar sensor, the natural choice falls on a laser scanner, which is able to detect parts of walls even when they are partly covered by clutter. Vision does not fit well with the minimalistic model and is sensitive to, for example, lighting conditions. The laser sensor will therefore be used as the main sensor in the thesis.

As discussed in Appendix B the laser sensor does not always detect all materials. To rely completely on the laser sensor is therefore not wise. In (Wijk et al. 1998, Wijk & Christensen 2000) a scheme based on triangulation is presented for extracting natural sonar point landmarks. These point landmarks do not fully comply with the minimalistic model as the point landmarks often correspond to movable objects such as chairs, bookshelves and tables. Extensive experiments have shown though that in many environments enough point landmarks do in fact remain unchanged to allow for successful localization (Wijk et al. 1998, Wijk & Christensen 2000).

Figure 2.15 shows the map with all features. The doors are marked as thick dark lines. The walls modeled as lines are also marked. As can be seen the rooms are modeled as rectangles. The points are shown as small dots, typically placed along the walls where there are book shelves, tables and door frames that give good sonar reflections.

2.4.1 Lines

As was explained above, the lines used in the minimalistic environmental model are assumed to correspond to large scale structures, typically resulting in four lines per room. With a strict definition of a line, the features used in the thesis should be called line segments as they are of finite length. However, line will be used in the meaning of line segment if nothing else is said.

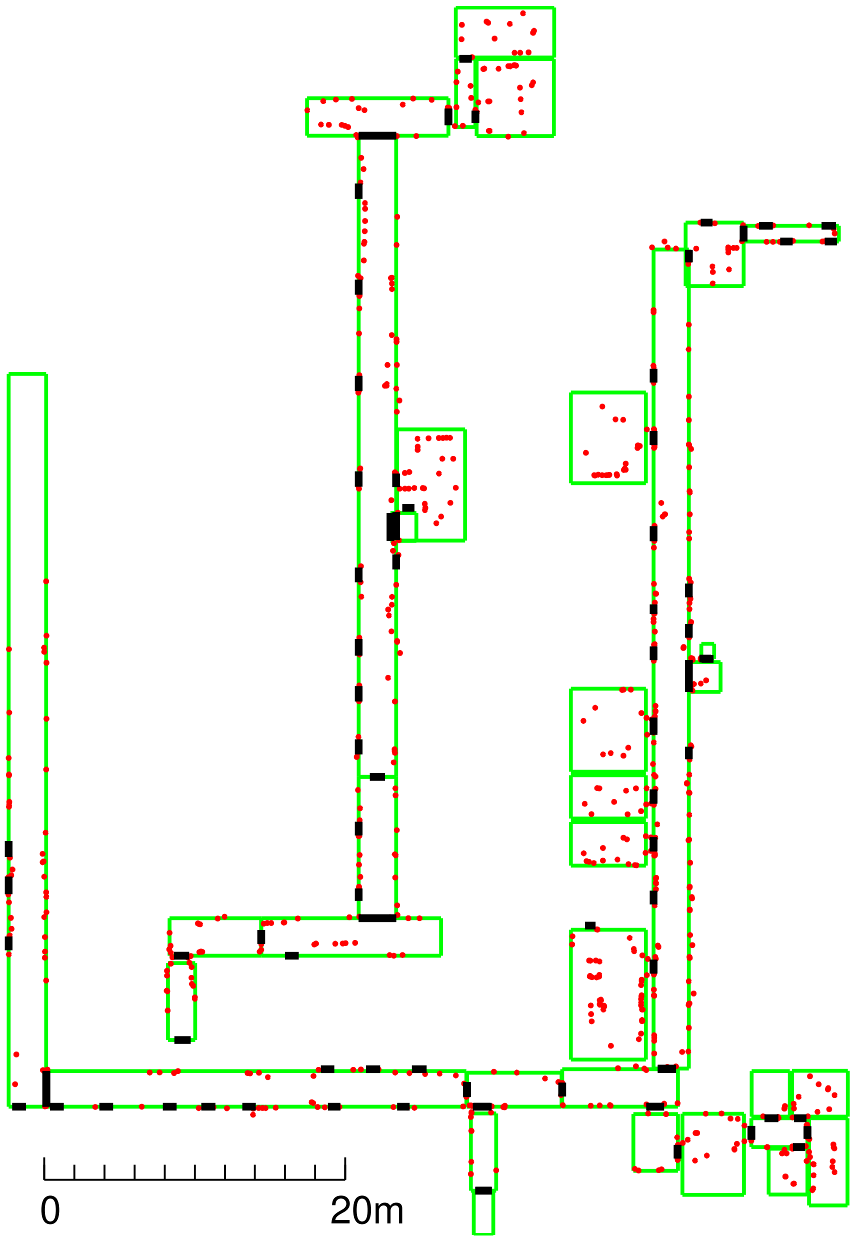


Figure 2.15: *The feature map $\mathcal{M}$. The map consists of two floors connected by an elevator. Note that the rooms are modeled as rectangles. The doors are marked as thick lines and the sonar point landmarks as small dots.*

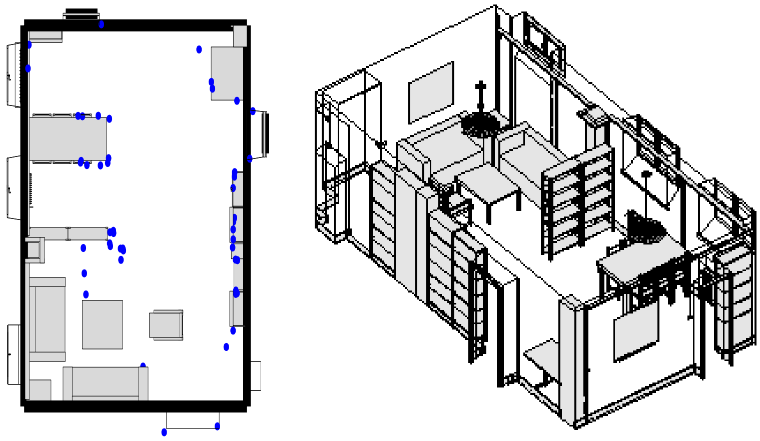


Figure 2.16: *Left: Three different feature types shown in the living-room. The four walls define the rectangular line model. There are two doors. The point landmarks are marked with dots and can be seen close to corner like structures, such the table and the shelves. Right: 3D model of the living-room.*

In the so-called living-room in the mobile robot lab at CAS, the four lines form a rectangle. The living-room is shown from above with the map features superimposed in the left subfigure of Figure 2.16. In the right subfigure the living-room is shown in much greater detail. It is evident that some of the lines are difficult to detect. This is especially true for platforms where the laser sensor is placed close to the floor.

The lines in the map are all measured by hand using a tape measure or a floor plan of the building in question. The total number of lines is 136, distributed among 34 rooms.

2.4.2 Doors

Doors are difficult to extract reliably using 2D data only. In this work only open doors are considered, as detecting closed doors is especially hard. The method for extracting door features is based on a set of templates, which captures many of the encountered door configurations, but far from all. The templates are shown in Figure 2.17. The dotted lines with dashed arrows indicate that all points between the end point of the line must lie on the other side of the line for the door detector to acknowledge it as a door. The wall segment marked as solid has to be detected, whereas the dashed ones are only shown to make the scene more clear. In all templates the width of the door has to be between a minimum and maximum value.

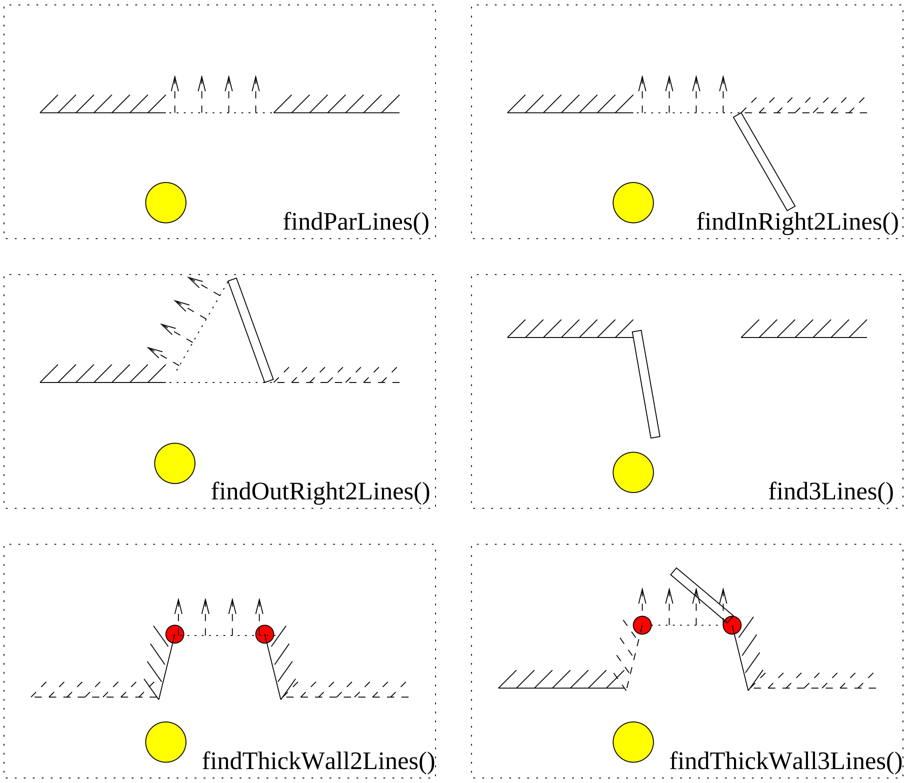


Figure 2.17: *The templates used when detecting door features.*

The goal is to construct a door extractor that rather not detects a door than reports one where there is none, i.e. a low rate of false positives at the cost of a higher rate of false negatives.

The building in which the mobile robot lab is situated used to be a hospital, and has thick concrete walls which makes detecting doors hard. When observed from the side it is not possible to see through a door. Seeing through a door is a cue common to all templates in (Figure 2.17).

The model includes 72 physical doors. However, half of these doors lead to rooms that are not in the model and can thus only been seen from one side. This is taken care of by introducing a "door" feature for each side the robot can detect the door. With this definition of a door feature, there are 108 door features in the map.

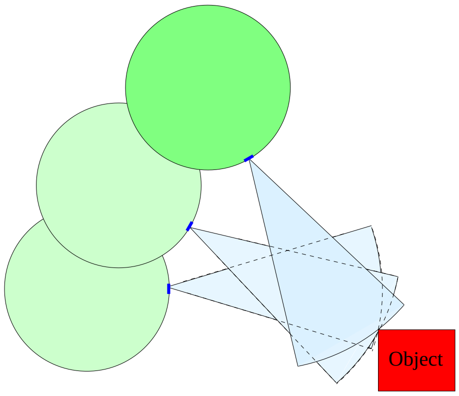


Figure 2.18: *The position of the reflecting object can be found by calculating the intersection of the the circular arcs. However, before doing so, a data association problem need to be addressed using for instance a voting scheme (Wijk 2001).*

2.4.3 Points

The basic idea is to use triangulation to improve the quality of the sonar data. Sonar data typically suffers from, for example, specular reflections and cross talk. 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 (2D assumption about the world). 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 2.18.

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. 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 originates from a true target. The crux is to know which readings to use for the triangulation. A scheme called Triangulation Based Fusion (TBF) is developed to solve this problem, see (Wijk et al. 1998, Wijk 1998, Wijk & Christensen 2000) for more details. The total number of point features in the map is 738.

Chapter 3

Pose Tracking

> "*Tracking in a cluttered environment is characterized by uncertainty in the origin of the measurements*"
>
> (Bar-Shalom 1989)

When the initial pose of the robot is known, it is enough to track the pose over time. This is the case, for example, when the robot starts at a re-charging station or the user supplies it with the pose upon starting the system. The odometric system gives information about the relative motion of the robot. This information can be integrated over time to give an estimate of the robot pose which is valid when moving over short distances. Over longer distances, however, the errors in the odometric information will accumulate and result in an unbounded pose estimation error. For an autonomous mobile robot which operates for hours performing various tasks, external sensors must be utilized to bound the pose estimation error. As explained in Chapter 2 the main problem in estimating the pose of the robot is to determine the origin of the measurements. That is, which parts of the environment produced the measurement. If the origin of the measurement is known, i.e. the correspondence problem is solved, the pose estimation problem becomes an exercise in mathematics. However, in the real world few decisions are made without uncertainty. This is true for mobile robot localization as well. Every time a measurement is processed there is a risk that a mistake is made.

In most implementations of feature based localization the update step of Figure 2.8 can be divided into three parts according to Figure 3.1. First features are extracted from senor data to create higher level information from the raw data. Using the *a priori* pose estimate and the environmental model, a prediction can then be made about the location of the features. In the matching step the extracted features are compared to the predicted features. How well they match is often judged by evaluating the Mahalanobis distance (see

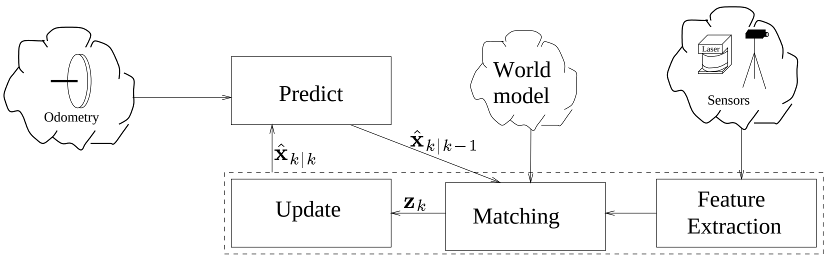


Figure 3.1: *Feature based pose estimation.*

Section 2.2.5). Finally pairs of extracted and predicted features considered as matched are used to update the pose estimate.

Extraction of features from raw sensor data can be a difficult task in a cluttered environment. Although the minimalistic environmental model provides robust features that remain in the same position over time they are sometimes hard to detect in the presence of clutter, e.g. in Figure 2.16 it is evident that all walls are partly occluded by non-modeled objects.

The minimalistic model contains three features; lines, doors and sonar point landmarks. The line feature will be used in this chapter as it provides the best conditions for being extracted even in the presence clutter using a laser sensor. Thus, this chapter presents a pose tracking method based on the line features, and it is shown that the large scale line structures provide enough information even in cluttered areas.

Section 3.1 defines the problem and discusses ideas of how to extract features from the minimalistic model. How the features are described and more in detail how they are extracted is presented in Section 3.2. Some implementation details are given in Section 3.3 followed by real world experiments in Section 3.4. A summary and discussion is given in Section 3.5.

3.1 Theory and Background

Here it is assumed that the probability density function for the robot pose can be represented by a uni-modal distribution. To justify this assumption the data association problem must be solved, i.e. measurements must be matched to the map without ambiguity. It is also assumed that the state of the system can be completely described by the pose of the robot, $\mathbf{x} = (x^{(W)}, y^{(W)}, \theta)^T$. The superscript (W) indicates world coordinates. Appendix C and, in particular Figure C.1, explain this further. It is implicitly assumed that the world is stationary, or at least that everything that is not stationary can be considered as noise. For simplicity (x, y) is henceforth used to denote $(x^{(W)}, y^{(W)})$ if nothing else is said.

Given the initial pose of the robot, an estimate thereof can be calculated at every time instant using odometric information. Knowing the approximate pose of the robot also simplifies the data association problem considerably (see Section 2.2.1). Sensor data that is likely to belong to the features in the map can be extracted effectively, and be used for updating the pose estimate and thus bound the estimation error.

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. To incorporate the information from the sensors to calculate the best possible estimate of the pose, it is necessary to specify what is meant by “best”. Often, ”best” is defined in the least square sense and the same definition is used here.

To perform the least square error estimate of the robot pose in real-time, a recursive algorithm is needed. The alternative is to store all acquired information and do a complete calculation at every time instant, in which case the computation cost grows with time. The price for real-time performance is that once a piece of information is used it is lost forever. There is no way of going back to reconsider a decision regarding, for example the data association and it is thus of vital importance that the number of errors is kept to a minimum. Hybrid methods found in between these two extremes exist. By keeping a buffer of information gathered over the last iterations it is possible to go back and reconsider a decision as long as it was made within the span of the buffer. Such a scheme is presented in (Engelson & McDermott 1992) in the context of map building. The problem with such an approach is that it often takes considerable amount of time before a mistake is discovered. The size of the buffer becomes large and the computation effort required to redo the calculations can no longer be handled in real-time.

The Kalman filter(Kalman 1960) framework has been used by numerous researchers for pose tracking and proven to be a good solution for sensor fusion (Leonard & Durrant-Whyte 1991*a*, Leonard et al. 1992, Forsberg, Larsson, Åhman & Wernersson 1993, Rencken 1994, Crowley 1989, Crowley et al. 1998, Gutmann et al. 1999, Arras & Tomatis 1999). The Kalman filter gives the optimal least square error estimate of the pose given the information at hand, assuming that the model of the system is linear and that all sources of noise are white and Gaussian (see Section 2.2.5). Most real world systems are non-linear in nature, and therefore the extended Kalman filter (EKF) must be utilized. A description of the EKF was given in Section 2.2.5. To underline the dependency on the map the measurement equation (2.15) is rewritten as

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathcal{M}, \mathbf{v}_k). \tag{3.1}$$

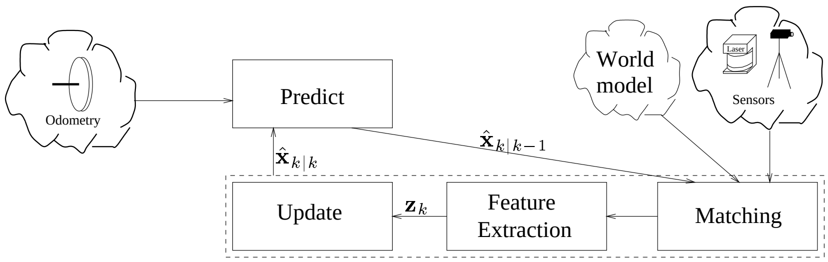


Figure 3.2: *Feature based pose tracking when the feature extraction is performed on sensor data that are matched point wise to predicted features.*

3.1.1 Line Extraction

As discussed above there is no return once a measurement is incorporated into a non-buffered recursive algorithm for state estimation. This can have devastating consequences for the tracking performance. In the worst case it can result in divergence. The covariance matrix, P_k, is used as a measure of the uncertainty in the pose estimate. Using an erroneous measurement can shift the estimate away from the true value but still reduce the covariance matrix.

In the general feature based tracking scheme, features are extracted from raw data and then matched to the map. Detecting modeled lines partially occluded by clutter is difficult. Assuming an estimate of the robot pose is given, the position of measurement points from the laser can be predicted using (3.1). The feature extraction can be summarized in two steps;

1. classify each point as belonging to a particular model features or as an outlier
2. estimate the parameters of the feature.

Figure 3.2 shows the proposed scheme as opposed to the general feature based pose estimation scheme from Figure 3.1.

3.2 Algorithm

This section describes the measurement model used in the EKF framework and how the line features are represented and extracted. It is clear that the success of the method rests on its ability to reliably extract the few lines that are represented in the minimalistic model.

3.2.1 Feature Description

To describe the extracted line features, the parameter vector (ρ, α, l) is used. This is part of the representation used in (Crowley et al. 1992). Here ρ is the perpendicular distance to the line, α is the line orientation and l is the length of the line (see Figure 3.3). The lines in the world model are augmented with the coordinates of the start and the end point, which are used when calculating visibility constraints. The features f in (2.1) are thus given by

$$f = (\rho, \alpha, l, x_s, y_s, x_e, y_e). \tag{3.2}$$

By definition a line can be seen if the start point is to the right of the end point when looking at the line.

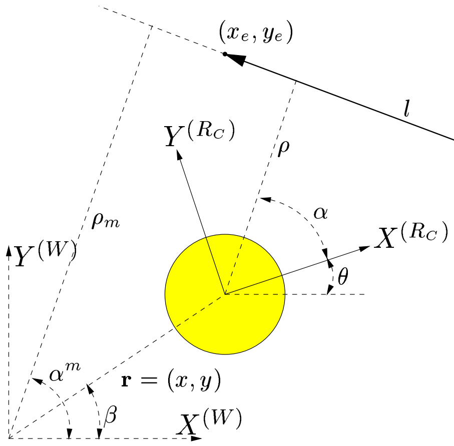


Figure 3.3: *A line is defined by the perpendicular distance ρ and the orientation α. In the map these parameters are measured relative to the world coordinate system and are denoted (ρ_m, α_m). The start point (x_s, y_s) and the end point (x_e, y_e) are used to define the position of the line and its direction. The pose of the robot can be defined by (x, y, θ) in Cartesian coordinates or alternatively (r, β, θ) in polar coordinates.*

3.2.2 Measurement Model

A line segment can in principal constrain all three degrees of freedom of a mobile robot. It gives reliable evidence about the perpendicular distance to line and the relative angle. The position along the map line can also be constrained if the length of the line segment is taken into account. This latter information is typically more unreliable though as detection of end points in the presence of clutter is difficult. A reliable indication that an end point is found is if the

the corner between two lines can be identified. Using these two lines together constrains all degrees of freedom even if the individual lines only provide perpendicular distance and relative angle. Therefore only ρ and α are considered to be measurements in EKF equations. Assume F_k lines are detected and are passed on as measurements at time k and let

$$\mathbf{z}_k^i = \begin{pmatrix} \rho^i \\ \alpha^i \end{pmatrix} = \mathbf{h}^j(\mathbf{x}_k, \mathcal{M}, \mathbf{v}_k^i), \qquad i = 1, \ldots, F_k \tag{3.3}$$

denote the i:th measurement generated by the j:th feature of $\mathcal{M}$, where $\mathbf{v}_k^i$ is the measurement noise. Using the notation from Figure 3.3 the function $\mathbf{h}^j$ can be written

$$\mathbf{h}^j(\mathbf{x}_k, \mathcal{M}) = \begin{pmatrix} \rho_m^j - \sqrt{x_k^2 + y_k^2}\cos(\beta_k - \alpha_m^j) \\ \alpha_m^j - \theta \end{pmatrix}, \tag{3.4}$$

where ρ_m^j is the distance to the j:th modeled line from the origin of the world coordinate system and α_m^j is the corresponding angle. The parameters β_k and r_k are defined in Figure 3.3. The Jacobian of $\mathbf{h}$ with respect to the state $\mathbf{x}_k$ (needed in the EKF) is given by

$$H^j = \begin{pmatrix} H_{11}^j & H_{12}^j & 0 \\ 0 & 0 & -1 \end{pmatrix}, \tag{3.5}$$

where

$$H_{11}^j = -\frac{x_k}{r_k}\cos(\beta_k - \alpha_m^j) - \frac{y_k}{r_k}\sin(\beta_k - \alpha_m^j) \tag{3.6}$$

$$H_{12}^j = -\frac{y_k}{r_k}\cos(\beta_k - \alpha_m^j) + \frac{x_k}{r_k}\sin(\beta_k - \alpha_m^j) \tag{3.7}$$

$$r_k = \sqrt{x_k^2 + y_k^2} \tag{3.8}$$

$$\beta_k = \arctan(y_k, x_k). \tag{3.9}$$

For certain robot-line configurations (3.4) predicts a negative distance ρ^j. This is taken care of by changing the sign of ρ^j and adding π to α^j in these cases. The measurement equation is then changed to

$$\mathbf{h}^j(\mathbf{x}_k, \mathcal{M}) = \begin{pmatrix} \sqrt{x_k^2 + y_k^2}\cos(\beta_k - \alpha_m^j) - \rho_m^j \\ \alpha_m^j - \theta + \pi \end{pmatrix}, \tag{3.10}$$

with Jacobian

$$H^j = \begin{pmatrix} -H_{11}^j & -H_{12}^j & 0 \\ 0 & 0 & -1 \end{pmatrix}. \tag{3.11}$$

The overall measurement vector $\mathbf{z}_k$ is given by stacking the individual measurements. Let $j(i)$ denote the index of the map feature that corresponds to the i:th measurement. With this notation the measurement equation can be written

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathcal{M}) = \begin{pmatrix} \mathbf{h}^{j(1)}(\mathbf{x}_k, \mathcal{M}) \\ \vdots \\ \mathbf{h}^{j(M)}(\mathbf{x}_k, \mathcal{M}) \end{pmatrix}. \tag{3.12}$$

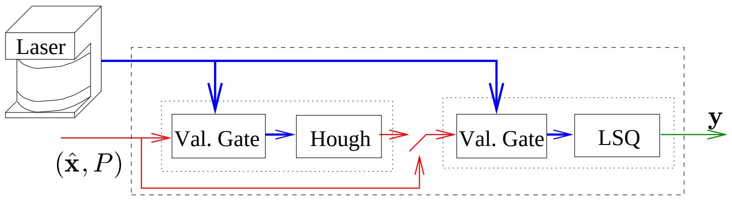


Figure 3.4: *Using the predicted pose of the robot in combination with map information, validation gates can be defined. The first step contains a local Range Weighted Hough Transform driven by data via a validation gate. The second step is a least square method using data from a second validation gate.*

3.2.3 Line Extraction

Extracting features from noisy data is an example of the more general parameter estimation problem, for which many algorithms exist. Most line fitting algorithms used in robotics have their origin in computer vision. The iterative end-point fit algorithm is one such example, developed for fitting lines to edge points in images (Duda & Hart 1973). This method is used in (Crowley 1985*a*) to extract lines from densely sampled sonar data. In (Castellanos & Tardós 1999) a similar technique divides the set of points into homogeneous regions (the lines) and spurious regions. In both of these cases the data are rather clean. Applying these algorithms on severely cluttered data will at best result in many small line segments, some of which may originate from the modeled line. An example of a method that handles large amount of clutter is the Hough transform (Hough 1962, Illingworth & Kittler 1988). For range sensor data the Range Weighted Hough Transform (RWHT) is more appropriate (Forsberg, Åhman & Wernersson 1993).

To extract the modeled lines from severely cluttered data a two step approach is proposed, which uses two different line extraction algorithms in combination with validation gates (see Figure 3.4). The first line extraction algorithm is robust against outliers, but provides only limited accuracy. The second step provides accuracy assuming that the input data is “clean”. When the pose uncertainty of the robot is small, the first step can be bypassed. In the following subsections the different parts of the line extraction algorithm are described in detail.

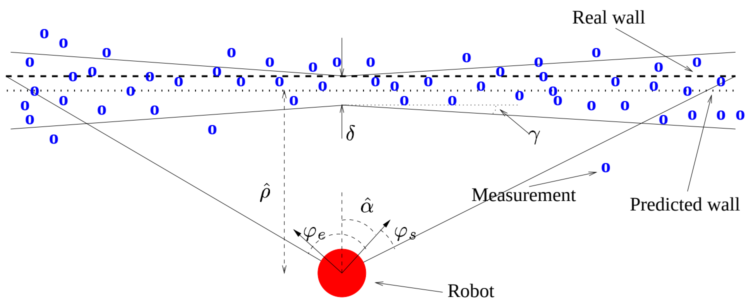


Figure 3.5: *Parameters that define the validation gate, where $\hat{\rho}$ and $\hat{\alpha}$ give the position of the gate and δ and γ the size. The parameters φ_s and φ_e are used for visibility constraints.*

Validation Gates

In this chapter the validation gates are not defined by the Mahalanobis distance as in Section 2.2.5. Instead the validation region is described by the six-tuple

$$\mathcal{G} = (\hat{\rho}, \hat{\alpha}, \delta, \gamma, \varphi_s, \varphi_e)^T. \tag{3.13}$$

Figure 3.5 shows an illustration of the parameters that define the location and size of the validation gate. Here $\hat{\rho}$ is the predicted distance to the line feature and $\hat{\alpha}$ is the predicted angle of the normal to the line. These two entities define the pose of the gate. The smallest width of the gate δ, and the opening angle, γ, define the size of the gate. The parameters φ_s and φ_e constitute the *visibility constraints*, which prevent the algorithm from attempting to detect a wall that is not in the field of view of the sensor.

Range Weighted Hough Transform

The RWHT is computationally expensive when applied directly. To keep down the computation cost a local version of the RWHT is used with a limited Hough Space, centered around the expected values, $\hat{\rho}$ and $\hat{\alpha}$. The RWHT aims at providing the second stage of the filtering algorithm with a better estimate of the line, thus paving the way for a tighter validation gate and cleaner data.

The main purpose of this first filtering step is to handle situations where $\hat{\rho}$ and $\hat{\alpha}$ are only rough estimates of the line parameters. That is, when the modeled line does not fall in the middle of the gate. Standard least squares algorithms are sensitive to outliers, which calls for a narrow gate, whereas the RWHT allows the validation gates to be more open.

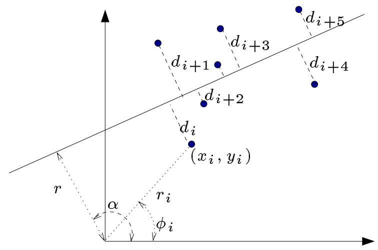


Figure 3.6: *The least square estimate minimizes the sum of squared perpendicular distances d_i.*

Least Square Line Fitting Algorithm

A point (x_i, y_i) on a line defined by the parameters (ρ, α) satisfies

$$x_i \cos\alpha + y_i \sin\alpha = \rho. \tag{3.14}$$

If, instead, the point is given in polar coordinates (r_i, φ_i) it satisfies

$$r_i \cos(\varphi_i - \alpha) = \rho. \tag{3.15}$$

When fitting a line to real data, the points do not line up perfectly and (3.15) is not satisfied exactly. The perpendicular distance, d_i, between the line and a point is given by

$$d_i = r_i \cos(\phi_i - \alpha) - \rho. \tag{3.16}$$

In a typical least squares line fitting algorithm the sum of the squared d_i's is minimized (see Figure 3.6), i.e.

$$\underset{(\rho,\alpha)}{\arg\min} \sum_i \left(r_i \cos(\phi_i - \alpha) - \rho\right)^2 . \tag{3.17}$$

The solution is given by (Deriche et al. 1992)

$$\alpha^* = \frac{1}{2}\arctan\left(\frac{b}{a-c}\right) - \frac{\pi}{2} \tag{3.18}$$

$$\rho^* = \bar{x}\cos\alpha^* + \bar{y}\sin\alpha^*, \tag{3.19}$$

where

$$\bar{x} = \frac{1}{N}\sum x_i \tag{3.20}$$

$$\bar{y} = \frac{1}{N}\sum y_i \tag{3.21}$$

$$a = \sum(x_i - \bar{x})^2 \tag{3.22}$$

$$b = 2\sum(x_i - \bar{x})(y_i - \bar{y}) \tag{3.23}$$

$$c = \sum(y_i - \bar{y})^2. \tag{3.24}$$

$$\tag{3.25}$$

With the simplifying assumption that each data point has the same Cartesian uncertainty, the measurement covariance matrix, R, can be calculated according to (Deriche et al. 1992)

$$\begin{aligned} R = &\frac{a\sigma_{yy}^2 - b\sigma_{xy}^2 + c\sigma_{xx}^2}{(a-c)^2 + b^2}\begin{pmatrix} 1 & -e \\ -e & e^2 \end{pmatrix} \\ &+ \begin{pmatrix} 0 & 0 \\ 0 & \frac{\sigma_{yy}^2 \cos^2\varphi + \sigma_{xx}^2 \sin^2\varphi - 2\sigma_{xy}^2 \sin\varphi\cos\varphi}{N} \end{pmatrix}, \end{aligned} \tag{3.26}$$

where

$$e = \bar{y}\cos\alpha^* - \bar{x}\sin\alpha^* \tag{3.27}$$

$$\varphi = (\alpha^* + \frac{\pi}{2}). \tag{3.28}$$

3.2.4 Sequential Measurement Update

As all lines are extracted from the same laser scan, the measurements are correlated. To handle the measurements correctly this correlation should be accounted for. However, to reduce the computational burden, it is neglected. By doing so the measurement update can be performed sequentially, treating each detected feature as a new measurement in the EKF. The complexity of the EKF update is linear in the number of measurements. The algorithm as a whole is also linear in the number of lines that are visible in each step.

3.3 Implementation

In this section implementation specific details are discussed, map structure, feature selection, etc. From here on, it is also assumed that a laser scanner is used with a 180° field of view.

3.3.1 Map Structure

To make the task of building the map easier, the map is broken down into smaller areas mostly corresponding to rooms. Each such area is given a local coordinate system. All features within an area are expressed in local coordinates. This means that locally the map can be made quite accurate. When combining all areas into the complete map, the transformation between a global coordinate system and each area is applied and tracking is done in global coordinates. The line features are stored on a room-by-room basis which allows fast access of the features for a particular area, as there is no need to loop through the entire map to find the features.

3.3.2 Line Length Threshold

The length of an extracted line, l, is used to reduce the risk of making errors in the data association. Only line measurements that satisfy

$$l \geq l_{min} \tag{3.29}$$

are used for updating. The shorter a line is, the larger the likelihood that it is not a part of the map $\mathcal{M}$. Short line segments have a much higher risk of originating from some line like structure which happens to be close to the modeled line. To avoid this the threshold l_{min} is chosen as

$$l_{min} = 1 \text{ m}. \tag{3.30}$$

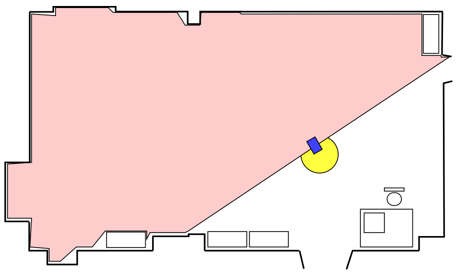


Figure 3.7: *From most places in a rectangular room, two or three walls satisfy (3.29).*

3.3.3 Feature Selection

The laser scanner is sensitive to occlusions as all laser beams originate from a single point and thus an obstacle placed directly in front of the sensor renders it blind. Using lines only from the one room makes the system sensitive, as only two or three walls satisfy (3.29) from most positions (see Figure 3.7). It is also difficult to detected the end walls in corridors when the robot is far away from them. Therefore, visible features in adjacent rooms are also considered for tracking purposes.

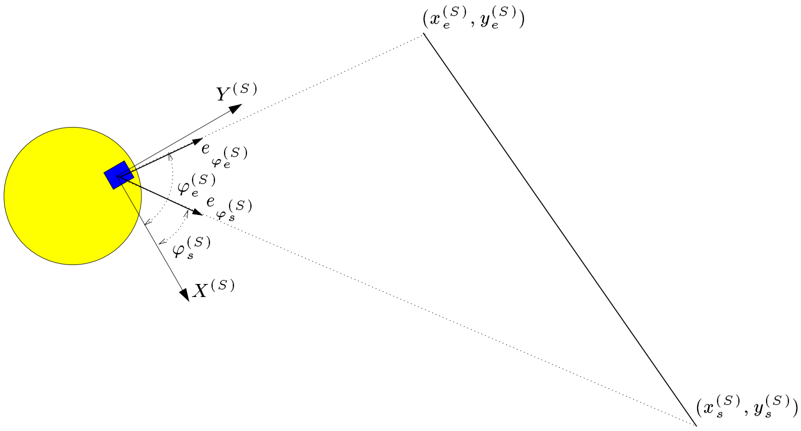


Figure 3.8: *The visibility constraints for seeing a line in a room is given by (3.31) and (3.32).*

Visibility constraints are used to determine which lines can be detected. These constraints are easiest to express in sensor coordinates (marked by superindex (S)). A line in the current room is visible if it satisfies the following criteria:

$$\text{Direction condition:} \qquad \arccos(\bar{e}_{\varphi_e^{(S)}} \cdot \bar{e}_{\varphi_s^{(S)}}) > 0 \tag{3.31}$$

$$\text{In field of view:} \qquad (0 \leq \varphi_s^{(S)} \leq \pi) \vee (0 \leq \varphi_e^{(S)} \leq \pi) \tag{3.32}$$

where $\varphi_s^{(S)}$ and $\varphi_e^{(S)}$ are the angles to the start and end point of the line respectively and $\bar{e}_{\varphi_s^{(S)}}$ and $\bar{e}_{\varphi_e^{(S)}}$ the corresponding unit vectors (see Figure 3.8).

A line in an adjacent room is visible if it can be seen through the door leading to that room, i.e.

$$\arccos(\bar{e}_{\varphi_e^{(S)}} \cdot \bar{e}_{\varphi_{dr}^{(S)}}) > 0 \quad \wedge \quad \arccos(\bar{e}_{\varphi_{dl}^{(S)}} \cdot \bar{e}_{\varphi_s^{(S)}}) > 0, \tag{3.33}$$

where $\varphi_{dl}^{(S)}$ and $\varphi_{dr}^{(S)}$ are the angles to the left and right door post, respectively (see Figure 3.9). Additional constraints have to added to handle non-convex polygonal line models.

3.3.4 Lower Bound on State Covariance Matrix

There is an explicit assumption in the EKF that the measurements are independent. When this is not the case the result is an overly optimistic estimate of the error covariance P. The size of the Kalman filter gain, K_k in (2.21), depends on the size of the covariance matrix, P. When P is small the Kalman gain is small. A small gain means that the measurements do not influence the pose estimate much, which makes the EKF more susceptible to unmodeled disturbances.

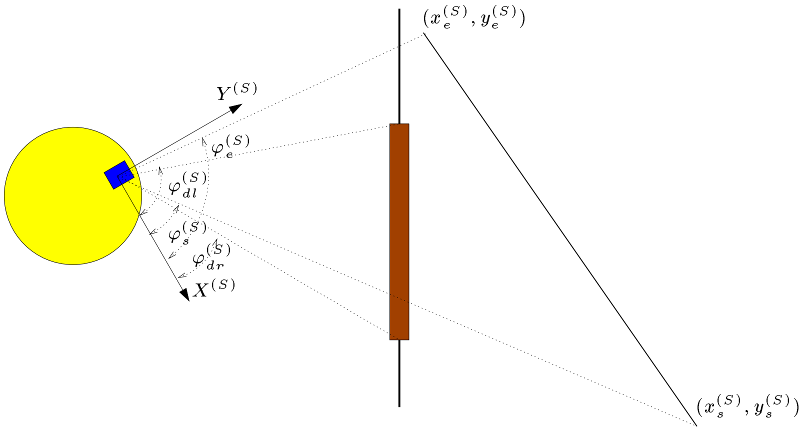


Figure 3.9: *A line in a neighboring room must satisfy (3.33) in addition to (3.31) and (3.32).*

A simple, but effective, way to deal with this is to put a lower bound on the uncertainty when the robot is moving. The lower limit for the diagonal elements of P are chosen as

$$
\begin{aligned}
P_{xx,min} &\geq (30\ \text{mm})^2 \\
P_{yy,min} &\geq (30\ \text{mm})^2 \\
P_{\theta\theta,min} &\geq (0.065\frac{\pi}{180}\ \text{rad})^2.
\end{aligned}
$$

Keeping the diagonal elements above the lower limits can be realized by adding a diagonal matrix with only positive or zero elemements to P, which preserves the positive definiteness of P.

3.3.5 Changing Room

Another situation when special care needs to be taken is when the robot passes between two rooms, i.e. when the robot goes from one local coordinate system to another. In this situation it is difficult to correctly specify the transformation between the two local room coordinate systems. This is dealt with by inducing noise into the system covariance matrix P when passing from one room to another according to

$$P := P + E, \tag{3.34}$$

where

$$E = \begin{pmatrix} 100^2\ \text{mm}^2 & 0 & 0 \\ 0 & 100^2\ \text{mm}^2 & 0 \\ 0 & 0 & (2\frac{\pi}{180})^2\ \text{rad}^2 \end{pmatrix}. \tag{3.35}$$

3.3.6 Parameters

There are a few parameters that have to be specified. These parameters are given rough values based on simple reasoning. Most of the parameters have been chosen once and never been changed after that. This means that they do not have an optimal value with respect to robustness, accuracy or what ever metric is chosen. This is intentional as an algorithm that requires tuning of parameters is sensitive to changes. Not having to tune parameters can be seen as an sign of robustness. A sensitive algorithm typically performs badly when a different environment is considered or when the parameters are not carefully tuned.

3.4 Experiments

The pose tracking algorithm presented in this chapter is the backbone of the localization system in the ongoing Intelligent Service Robot (ISR) project and as such is tested every time the system is used. In most experiments the pose tracking algorithm is run along with the rest of the system that provides the capability to go from any point to any other point in the map while avoiding obstacles. No active control of the sensing direction is performed. The experiments are performed during normal working hours, when people are walking around and doors are opened and closed, i.e. a dynamic environment.

Most of the experiments are performed on the lower floor of the mobile robot lab at the Centre for Autonomous Systems (CAS) with a Nomad200 robot platform (see Appendix D). However, results are also presented from different environments which strengthen the claim that the minimalistic environmental model is sufficient in many indoor environments.

3.4.1 Evaluation in Cluttered Environments

The minimalistic model aims to represent features which are robust over time. The question is if they can be reliably extracted from sensor data. The aim of this experiment is to show that even in severely cluttered areas the algorithm is able to find and use the few points which originate from the modeled lines. Before this can be done however clutter must be defined. To make the definition easy, everything that is not modeled can be considered as clutter. There are different types of clutter though, a distinction is here made between *structured* and *unstructured* clutter.

Unstructured clutter:
Individual objects in the environment that do not contain large line structures, e.g. table legs and flower pots.

Structured clutter:
Individual objects in the environment that contain large line structures such as a cupboard, or a collection of individual objects that line up to a, not necessarily connected, line structure.

Figure 3.10: *Two views from a severely cluttered room. Most of the clutter is unstructured. The validation gates effectively remove all but the data that is close to the predicted position of the map lines.*

Unstructured clutter is typically harmless unless its occluding effect gets too large, whereas structured clutter can distract the line extraction and cause loss of track. The latter depends on the spatial separation between the clutter and modeled features. A cupboard placed in the middle of a room cause less problems than if it is close to a wall. The allowed spatial separation depends on the uncertainty in the pose estimate. Clutter that does not enter the validation gates does not cause data association errors.

Unstructured Clutter

In Figure 3.10 parts of a severely cluttered room is viewed. Figure 3.11 shows one laser scan used as input to the pose tracking algorithm. The dark dots denote laser data that have passed the second validation gate (see Section 3.2.3). The brighter dots represent the laser data that fall outside the validation gates and are therefore rejected. As can be seen, none of the modeled features (the walls) are fully visible. The left and lower walls are barely detectable. They occupy approximately $65°$ and $60°$ field of view, respectively. For the lower wall only 17 points are validated or less than 15% of the points. For the left wall the situation is somewhat better with 18% of the potential points visible. As long as the robot is able to track two non-parallel walls in the room the uncertainty is kept small and the association problem can be solved. Looking at the data points that are rejected, the left wall is first to cause problems when the uncertainty increase. The computer monitors shown in the left subfigure of Figure 3.10 form structured clutter. If the corresponding data points are let through the first validation gate, the Hough transform selects this as the line to track. The three other walls are the most dominant line structures in the corresponding directions and can thus tolerate a larger uncertainty.

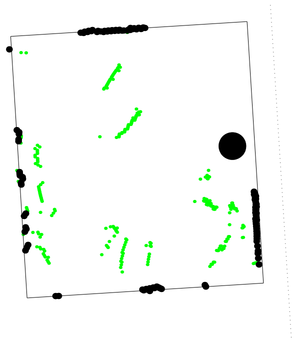


Figure 3.11: *The dark dots represent laser data that are validated to belong to modeled lines and the brighter are rejected data. The solid lines satisfy (3.31) and (3.32), i.e. they are in field of view. The dashed line is not visible as (3.31) is not satisfied, as the robot is on the wrong side of the line. The large filled circle is the estimate robot position. The pictures in Figure 3.10 are captured from approximately this position.*

The conclusion is thus that any amount of clutter can be handled as long as it does not enter the validation gate and there is data from modeled lines in the gates. When the clutter enters the validation gate the modeled line must have the strongest response in the local version of the RWHT. This is often true for unstructured clutter.

Structured Clutter

Figure 3.12 shows a different situation. The pictures are taken from the corridor intersection in the lower left corner in Figure 2.15. The left picture shows the view when approaching from the right. A large unmodeled wall cupboard occupies most of the field of view. This situation is different from the cluttered room in the previous section as the cupboard has the same orientation as the wall behind, and furthermore cuts off almost all sight of it. Besides the unmodeled wall cupboard this scene is free from clutter.

It is clear that the uncertainty must be kept small enough to keep the cupboard outside the validation gate. The cupboard is located 0.39 m in front of the wall. The size of the validation gate depends on the level of uncertainty. In Figure 3.13 the result for different levels of uncertainty is given. The left subfigure illustrates how the cupboard is classified as the wall when the gate is artificially made wider. As a result the estimate of the robot pose becomes biased by 0.39 m. The covariance matrix does not give any indication that something is wrong and the result is a biased EKF-estimate. The right subfigure shows the result when the gate is small enough to keep the cupboard outside. The data points from the cupboard are then rejected as clutter and the EKF-estimate is un-biased.

Figure 3.12: *Corridor intersection with a large wall cupboard. When approaching the intersection from the side shown in the left subfigure the cupboard acts as structured clutter if it is not modeled.*

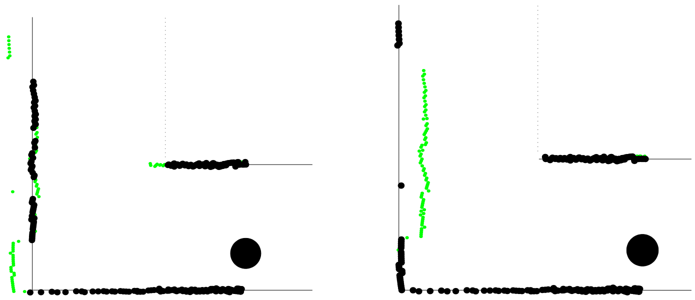


Figure 3.13: *Left: If the uncertainty is too large when approaching the intersection the cupboard falls inside the validation gate and is falsely associated with the modeled wall. Right: The data association problem is eliminated if the uncertainty is small enough to keep the cupboard outside the validation gate.*

The cupboard hides most of the wall from view when the robot approaches from the right, as shown in Figure 3.12. The short piece of wall that is visible to the left of the cupboard in the left subfigure of Figure 3.12 is not long enough to meet the minimum length requirement, (3.29). The robot cannot update the position along the corridor until it gets close to the intersection where points on both sides of the cupboard can be detected. Under normal circumstances the validation gates are able to separate the wall and cupboard since the spatial separation is relatively large. To increase robustness however, the wall cupboard can be entered into the map. The simplest change to the model is to add a single line where the cupboard is and let the wall behind start to the right of the cupboard (right subfigure of Figure 3.12). Figure 3.14 shows how the failure from the left subfigure of Figure 3.13 is changed to a correct classification of the data with the same initial uncertainty.

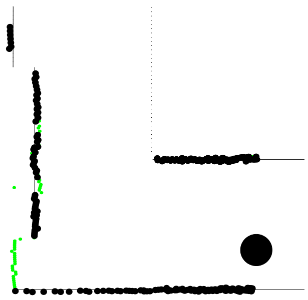


Figure 3.14: *Augmenting the model with the large wall cupboard.*

3.4.2 Accuracy

The previous section demonstrated that the algorithm is robust to large amounts of clutter. Accuracy is another performance criterion for a pose tracking algorithm. It is secondary to robustness but still important in some applications. Where robustness can be tested by making many experiments, accuracy is harder to measure. A true measurement of accuracy requires ground truth to compare with. The position accuracy of the robot can be measured by hand when the robot is standing still. However, this only gives a measure of the accuracy when the EKF has converged. If the robot can see two non-parallel walls the estimation error is the result of inaccuracies in the environmental model and sensor alignment more than anything else. If no wall is visible the accuracy is also affected by the quality of the predictions based on odometry. When the robot is moving measuring the pose by hand is not feasible. Another way is to store all sensor data and manually align it with the environmental model to get the ground truth, but this approach is too time consuming.

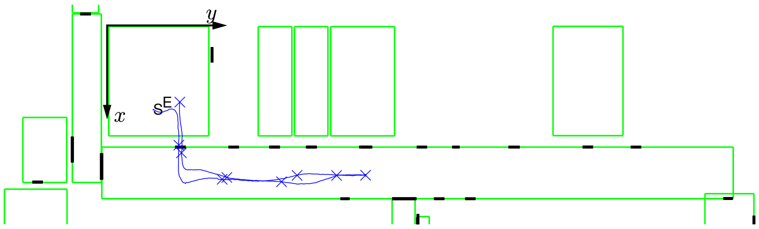


Figure 3.15: *The trajectory followed by the robot for innovation evaluation. The distance between two crosses corresponds to 20 seconds of traveling.*

A third alternative is to look at the innovation sequence, ν_k. The innovation expresses the difference between the measurement and the prediction. Assume that the association problem is solved, i.e., the origin of the data points are known. Then a small innovation is a strong indication that the pose estimate is good. A large variance in the innovation sequence indicates high noise levels for the process or the measurement noise. When the innovation has a bias it is a sign of a systematic error. The source of such errors can, for example, be an erroneous environmental model. A small variance does not imply that the pose estimate is good if an incorrect data association is made at some stage, which Figure 3.13 is an example of. Using the innovations as a measure of the accuracy requires that the data associations are correct. It therefore limits its use for on-line monitoring of the filter performance. Figure 3.15 shows the trajectory followed by the robot in a small experiment to measure the accuracy. The robot starts and stops in the same room and travels in the corridor in between. The experiment lasted 188 seconds during which time the robot traveled 40 meters. The crosses mark 20 second intervals along the trajectory. Each iteration in the run is studied manually to see which lines are detected and the data associations are manually verified.

Each modeled line has an associated innovation. The innovation is a two component vector. One component for the perpendicular distance to the line, ρ, and one regarding the orientation, α. No line in the map can be seen during the whole experiment. However, the modeled lines are all parallel or orthogonal. For this experiment, the walls are therefore divided into two groups. One group of lines that are parallel to the global y-axis and one parallel to the x-axis. The lines parallel to the y-axis give evidence about the x-coordinate of the robot pose and vice versa. The first available innovation from each of these groups are plotted as a function of time in Figure 3.16. The dashed curves correspond to the standard deviation in the x and y directions respectively. There are two major characteristics of the figure that warrant further explaining. The

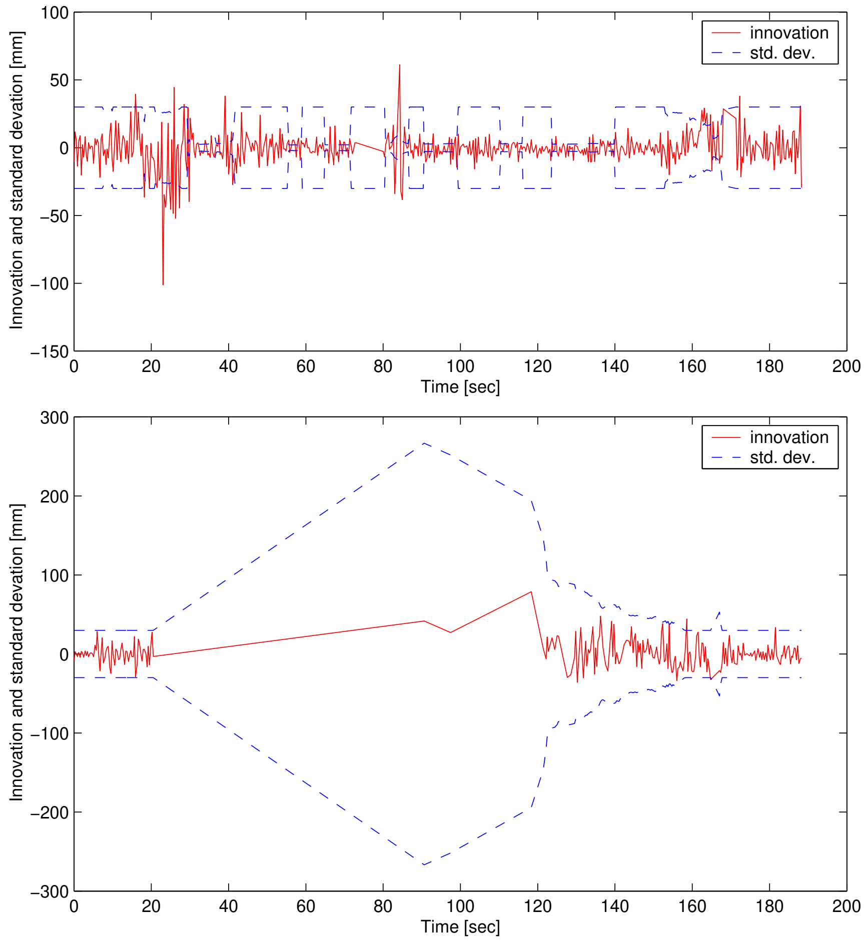


Figure 3.16: *The innovations in the x and the y directions over the trajectory in Figure 3.15. The upper figure is x and the dashed curve is the estimated standard deviation ($\sqrt{P_x}$). The lower figure shows the corresponding values in the y direction. The bias in the x-error after approximately 20 seconds and again after 160 seconds occurs when the robot changes room and can detect lines both inside the room and in the corridor. The standard deviation of the innovation is 12.4 mm in x and 16.2 mm in y, with mean values of -0.8 mm and -1.4 mm, respectively.*

first one occurs roughly 20 seconds after the start. Looking at Figure 3.15 the robot is passing the door at this point in time. When the robot is close to the door it can see lines both inside the room and in the corridor at the same

	mean	std
x	-0.8 mm	12.4 mm
y	-1.4 mm	16.2 mm

Table 3.1: *The mean and the standard deviations of the innovations in the x and y directions. This is a possible measure of the accuracy of the pose tracking method.*

time. These measurements might not agree, as the transformation between the corridor and the room is difficult to measure, causing the filter to average and hence creating a bias. The same effect is seen after 160 seconds when going back into the initial room. Roughly 80 seconds into the run the innovation jitters as well. This corresponds to the robot being in the middle of the corridor where the walls are 0.15 m further apart than modeled (in order to keep the model simple). In a corridor the uncertainty is typically small sideways but larger along the corridor. The robot is able to reject the unmodeled wall as long as it can detect the wall on the other side to keep the uncertainty down.

Possible measures of accuracy based on the innovations is the mean and the standard deviation. Table 3.1 shows these values for the experiment presented above. The bias is most likely caused by modeling errors. The width of the corridor is, for example, not constant with local variations on the order of 10-50 mm. The width is a function not only of the position in the xy-plane but also on the height it is measured at. Most measurements for the map are acquired at floor height.

The accuracy is not the same everywhere. This was seen in Figure 3.16. The lower subfigure shows that there are few updates of the y coordinate between 20 and 120 seconds, when the robot travels in the corridor, either with the sensor not facing the short wall or too far away from it. Figure 3.17 shows how the standard deviation changes when traveling along the corridor from the position marked with an 'S' to the position marked with an 'E'. The uncertainty ellipses are magnified a factor of four to make them easier to examine. The figure is also stretched perpendicular to the corridor for the same reason. The overlaid curve shows the standard deviation in y, i.e., along the corridor, as a function of the position. From the figure it is clear that the robot is unable to detect any lines along the corridor until having traveled approximately one quarter of the corridor length. At this point the robot detects a line in the nearby room (perpendicular to the corridor) and the uncertainty drops drastically. After this the uncertainty increases again until the center section is reached. Here a niche on the left hand side provides the features. Two more updates shortly after this are the result of detecting lines through the doors on the side again. The final slow decrease in uncertainty is due to detecting the short corridor end wall. Relatively few points are initially accumulated by the end wall which makes the measurement more noisy and hence does not reduce the uncertainty as much. The uncertainty perpendicular remains more or less constant during the passage, as the long walls are always visible.

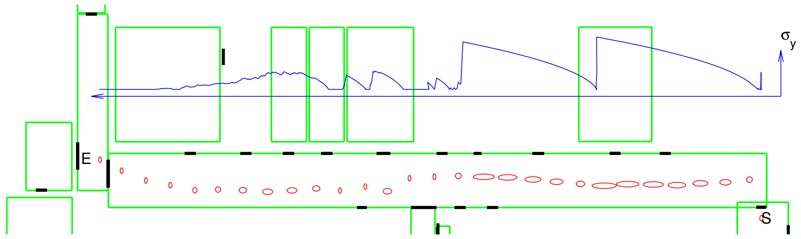


Figure 3.17: *Uncertainty when traveling along the corridor. The ellipses are magnified by a factor of four to make them easier to examine. The width of the corridor is somewhat stretched. The curve overlaid shows the pose uncertainty along the corridor. The decrease in uncertainty when traveling in the corridor comes from detecting lines in neighboring rooms.*

3.4.3 Long Term Experiment

Tracking the position in a small area is not a good measure of the robustness of a pose tracking algorithm, as the odometric error does not have much effect. Typically it is the drift in orientation that causes the largest errors. Furthermore, each area is mapped separately and it can be expected that the quality of the local map is quite good. The transformations between the rooms are more uncertain. Some of the rooms has a threshold which must be traversed to enter the room. These can cause large unmodeled odometric errors. Moving the robot throughout the environment tests the influence of all these sources of errors.

Figure 3.18 shows the path followed by the robot in a longer experiment. The total distance traveled is 740 m, the average speed is 0.14 m/s and the total time is 90 minutes. A considerable amount of time is spent on passing between rooms as the robot must slow down to do so. Along the track, different types of environments are encountered. Starting from the living-room (Figure 3.19) the robot moves through the corridor to the adjacent offices. These rooms are so small that the allowable movement of the robot is limited (see Figure 3.20). The small offices are divided into two parts, leaving limited sight of two of the walls. No problems are found in these rooms during any of the runs.

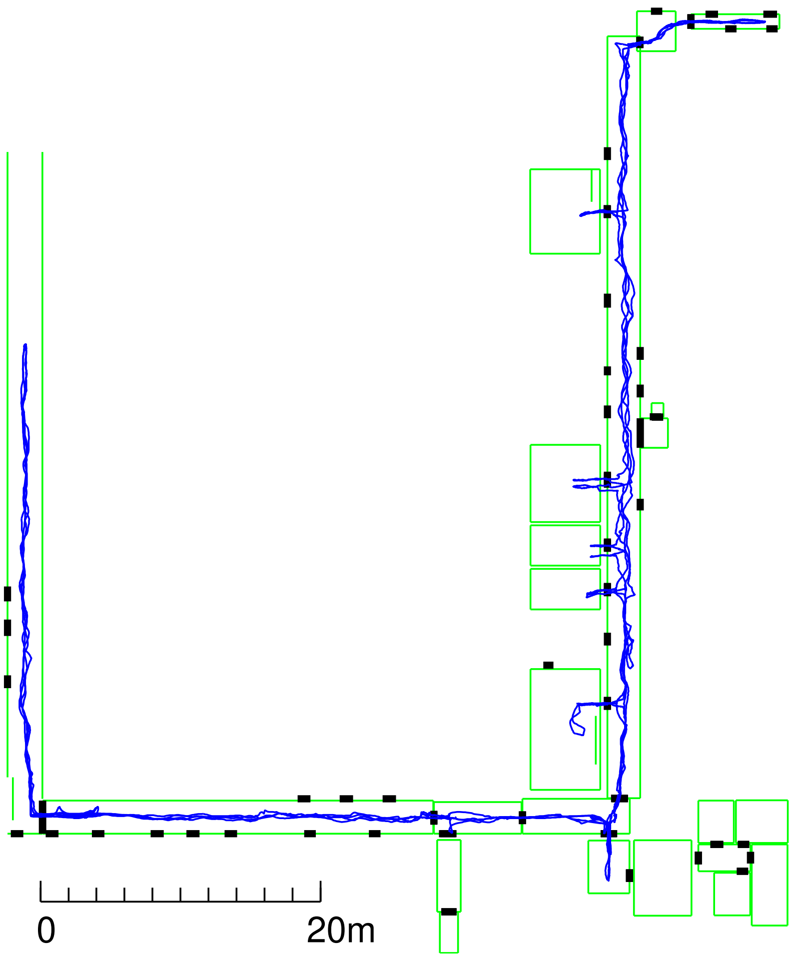


Figure 3.18: *The track followed by the robot during the long-term experiment, two loops around the lower floor of at CAS. This trajectory is 740 m long which took 90 minutes for the robot.*

The corridor at the ground floor of the laboratory is approximately 55 m long and the width is about 2.3 m (see Figure 3.21). It is obvious that the problem in the corridor is to maintain a good estimate of the position along the direction of the corridor. When moving far away ($\approx 15 - 20$ m) from the short walls, they can no longer be used reliably for an EKF update, because too few data points are accumulated. The detection of the short walls are made even more difficult if there is clutter in front of them. Because lines from neighboring rooms are used as well, a good estimate of the position along the corridor can still be maintained if some of the doors are open, providing a view of the walls inside. The standard deviation in the position estimate reached a maximum value of about 0.25 m in the corridors.

Figure 3.19: *The living-room.*

Figure 3.20: *A typical office at CAS.*

Figure 3.21: *The corridor outside the living-room.*

Tracking is maintained for the duration of the run, the limiting factor in this experiment being the battery capacity and not the tracking algorithm.

3.4.4 Different Environments

To test the robustness of the approach even further it is also evaluated in other environments. The important thing to show with this experiment is that the approach is not tailored to one specific environment and that it can also be run on different types of platforms.

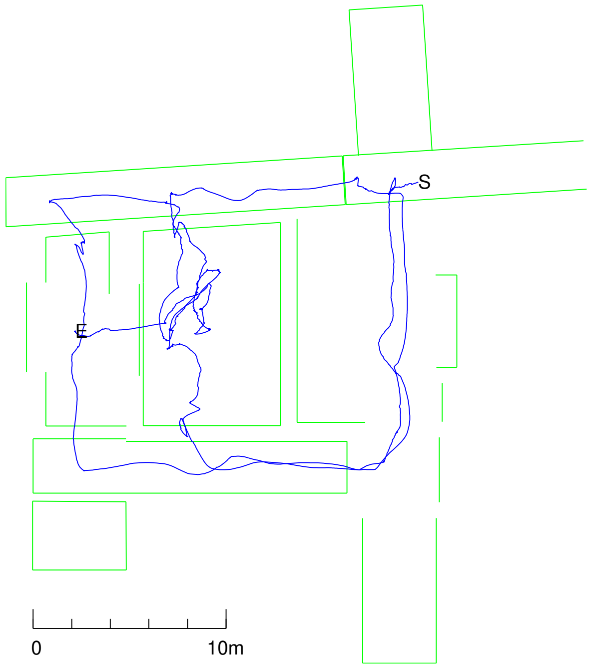


Figure 3.22: *The trajectory followed by the robot in one of the tour sessions during a demonstration in the area around the atrium in the main building at KTH. The jerky trajectory is caused by people blocking the path of the robot.*

Student Tour Guide in the Atrium

One experiment is performed during a demonstration around the atrium in the main building on the KTH campus. The robot acts as a tour guide for students on visit. The area in which the robot moves is 20×20 meters. Approximately 100 people are constantly moving around in the area. Many of them are curious and block the way of the robot. A model of the environment was acquired by hand, measuring the most dominant walls in the environment. A Nomad SuperScout platform (see Appendix D) is used in this demonstration as it is small and therefore more appropriate for demos made outside the lab as it can easily be carried.

The trajectory from one of the tour guide sessions is shown in Figure 3.22. This trajectory is about 143 meters and 33 minutes long and the robot is at all times surrounded by curious high school students. The robot keeps the track well during this tour and all other tours except in one case where the robot after 20 minutes of motion drives into something that makes the orientation change suddenly without the odometry detecting it. After this the track is lost and the robot can no longer perform its tour guiding duties. This incident highlights some important things. The EKF cannot handle large unmodeled disturbances in the odometry. Had the same change in orientation occurred gradually over some distance, track could have been maintained. According to the process model it is only by robot motion that the uncertainty in the pose estimate increases. Disturbances that occurs when the robot is moving slowly or is standing still is therefore particularly nasty. When the pose is lost another mechanism must be activated for re-localizing the robot. Such mechanisms are discussed in Chapters 4 and 5.

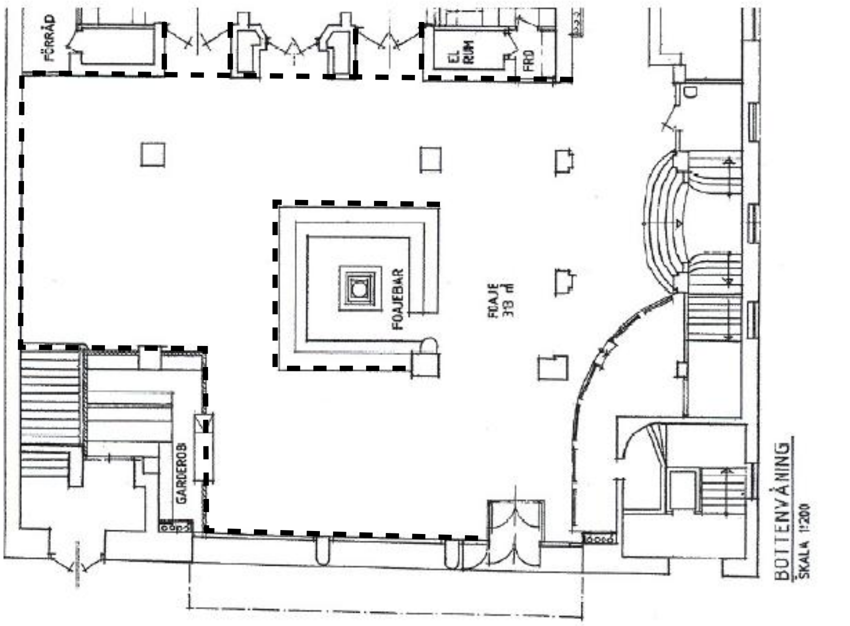


Figure 3.23: *Floor plan of the bar area at Heaven with the minimalistic model overlaid with dashed lines.*

Tour Guide At Heaven

During another demonstration the Nomad SuperScout is once again used. In this demo the area is quite small, but the number of people is overwhelming. A floor plan of the area is shown in Figure 3.23, where the minimalistic line model is overlaid in the figure (the dashed lines). Note how the four horizontal lines at the top of the floor plan are lumped together into one line and that the bar in the middle is included in the model for extra line support.

Figure 3.24: *There are 200-400 people in the bar area. The robot can only move on the lower side of the bar, and then just barely for all the people.*

When the demonstration takes place there are between 200 and 400 people in this area. People are moving in and out of the immediate bar area in groups which makes the actual number of people hard to determine. Figure 3.24 shows parts of the crowd. It is only on the lower side of the bar and a bit on the left of it that it is possible for the robot to move at all. Just like in the other demonstration people are curious. As the Scout robot only reached to the knees, some people almost stumbled over it. In a situation like this it is thus not enough to passively avoid obstacles. The robot must have the ability to foresee the motion of people and actively avoid them. At present the robot unfortunately lacks this skill.

Much of the time is spent blocked by people. Once in a while the crowd opens up to let the robot pass. Figure 3.25 shows photos of the robot negotiating peoples legs.

With so many people so close, a clear view of a wall is rare. However, most of the time small pieces of the walls are visible. These few data points are enough to update the pose estimate. This is a good illustration of one of the benefits of using a laser scanner versus the sonar sensor. In a dense crowd, the sonar sensor cannot detect anything but the surrounding legs. Figure 3.26 shows two snapshots taken from the laser.

Figure 3.25: *The SuperScout robot navigates between peoples legs.*

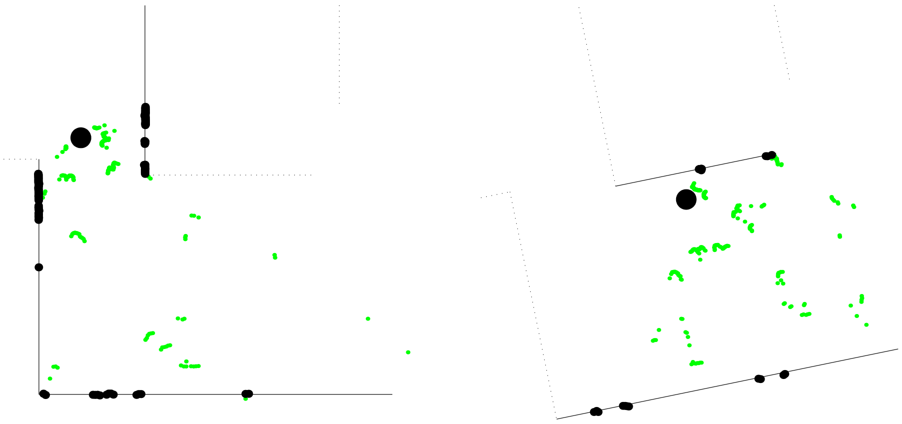


Figure 3.26: *Bits and pieces of the walls are visible most of the time. The closer people come to the robot, the more the sensor is occluded.*

Several experiments are performed during the day and the conclusion is that tracking in this dense crowd is not a problem except for one thing. When the robot is standing still or moving slowly it is sensitive to disturbances in the odometric information as discussed in the previous section. When the crowd is dense, contact with people is inevitable. Currently there is no way to detect these contacts.

3.4.5 Lower Update Rate Limit

The faster the algorithm is run the better the prediction can be made about the robot pose. If robustness is the only thing that counts, the faster the better. In an integrated system each process should try to consume only as much as it needs and not be too greedy. It is therefore of interest to investigate how fast the pose tracking algorithm must be run to provide reliable tracking.

Simply put, the question does not have an answer if the environmental model is not specified. It all comes down to being able to solve the data association problem. In an environment with many line like structures that are parallel and close, the uncertainty must be kept small to distinguish between them. If there are only a few easily distinguishable features, a large uncertainty does not pose a problem and a low update rate can be allowed.

For this experiment the data from the 90 minutes run presented in Section 3.4.3 is used. It is necessary to have a large data set so that as many different situations as possible are encountered. For each update rate the whole data set passed through. The performance is judged by visual inspection. For each iteration all points that pass the first and the second gate are examined along with the number of lines that are detected. When the update rate is lowered it is expected that at some point the track is lost.

Even with an update rate of as low as 0.05 Hz the robot tracks the pose. The uncertainty now becomes larger, but not large enough to cause erroneous data associations. Somewhere around 0.04 Hz it breaks down, when passing from the living-room out into the corridor. An update rate of 0.04 Hz means that the robot is forced to rely on only odometric information for 25 seconds, during which it can move a significant distance. Updating at this rate does not reduce the uncertainty in the pose enough to maintain tracking, i.e. the diffusion in prediction step cannot be compensated for enough by measurements. This experiment shows that the update rate under normal operating conditions can be much lower than 2-3 Hz which is used by default. The particular result is dependent on the platform being used, especially the odometric performance, the environment and the speed of the robot.

3.5 Discussion

In this chapter a low complexity, robust and accurate pose tracking algorithm based the EKF framework and the minimalistic environmental is presented. The minimalistic environmental model provides robustness using only the large scale structures, such as the four dominating walls of a room. Such features are likely to be robust over time and are relatively easy to extract due to their size, allowing for low computational complexity. The low complexity of the approach is particularly important in an integrated system with limited computational resources.

Experiments show that the method can handle a high density of clutter and is able to track the position for long periods of time. The limiting factor being the capacity of the batteries and not the method. The battery issue is currently being addressed by the construction of a re-charging station, suitable for automatic docking.

Experiments also show that a simple model for the uncertainty of the line parameters is enough and that the EKF can be run at low frequency and still maintain tracking. It is important to note that updating with low frequency increases the risk of losing track when something which is not captured by the odometric model occurs. Example of such events are slippage on thresholds or collisions with people as in Section 3.4.4. A topic for future research is therefore to augment the pose tracker with means to detect when these events occur, e.g. by comparing relative motion information from a gyro and the odometry as in (Borenstein & Feng 1996*a*).

Chapter 4

Multiple Hypothesis Localization

The method presented in Chapter 3 made the assumption that the initial pose was known. This chapter deals with the problem of finding the robot pose using no prior pose information, but the map of the world is assumed to be known. This problem is referred to as global localization or pose initialization. A typical scenario when global localization is needed is when the power is turned on and the robot must find its pose in the world. Having only the ability to track the pose over time as provided by the previous chapter implies that the user must supply the initial pose. This could be circumvented by always letting the robot start at the same pose, for example, the place where it recharges. However, if the robot loses track during a mission and is in need of re-initialization the user might not be there to help. An autonomous mobile robot thus needs global localization skills.

The results from Chapter 3 show that a high degree of robustness and accuracy can be achieved using a Gaussian representation for the robot pose uncertainty. However, this representation is only a good model if the data association problem can be solved. This means that the uncertainty must be kept small relative to the spatial separation of mapped as well as unmapped environmental features. In global localization this criterion is not met and the Gaussian representation cannot be used. The main limitation is the uni-modality of the distribution.

The same criticism is often brought forth as an argument for using non-continuous pose representations, such as the topological (Simmons & Koenig 1995, Cassandra et al. 1996) and the grid (Burgard et al. 1996). However, combinations of multiple uni-modal distributions can be used to represent the overall PDF. In (Sorenson & Alspach 1971) it is shown that a sum of Gaussians distributions can approximate any PDF arbitrarily well. The condition

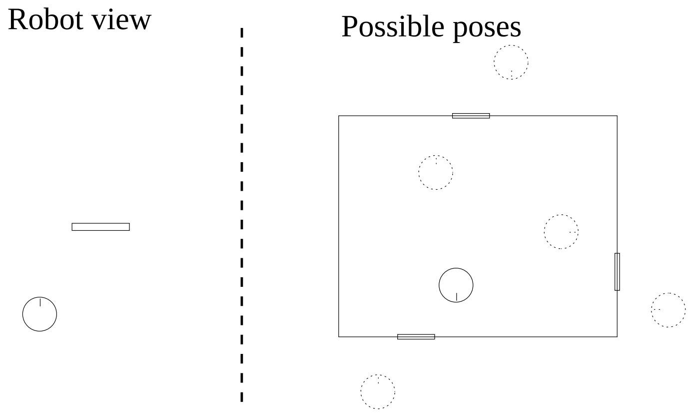


Figure 4.1: *The idea behind the Multiple Hypothesis algorithm. Detecting a feature, a door in this case, infer possible robot poses given that there is a true correspondence in the map.*

for convergence is however that the covariance of each Gaussian approaches zero and that the number of Gaussians tend to infinity. Using infinitely many Gaussian PDFs is not possible and thus, as is often the case, approximations must be considered. This chapter investigates the use of a small number of Gaussians to approximate the PDF. The Gaussian distributions ignore the fact that the probability for the robot being in certain poses should be zero, for example, where there is a wall. However, here the working hypothesis is that the Gaussian distribution gives a sufficient approximation. The desire is after all to find the robot pose and not to make the best possible approximation of the true PDF.

Consider Figure 4.1 and assume that the world consists of four walls and three doors and that the robot observes a door. Assume furthermore that the observation has a true correspondence in the map. Matching the detected door against the map yields six possible robot poses, one on each side of the three doors. A weighted sum of six uni-modal PDFs is then a good approximation for the overall PDF. Each of these can be viewed as one hypothesis about the pose of the robot.

The remainder of the chapter is divided into five main sections where the Multiple Hypothesis Localization (MHL) approach is described. Section 4.1 defines the problem and introduces the representation in more detail, while Section 4.2 describes the algorithm. Some implementation details are given in Section 4.3. Real world experiments are presented in Section 4.4 that demonstrates the applicability of MHL. The chapter closes with a summary and a discussion.

4.1 Theory and Background

The global localization problem studied in this chapter can be formulated as follows. Estimate the state of the robot, $\mathbf{x}$, using information from internal (e.g. odometry, gyros) and external sensors (e.g. sonar, laser) and a map of the environment. The state of the robot is given by its pose $\mathbf{x} = \left(x^{(W)}, y^{(W)}, \theta\right)^T$ (see Appendix C), which for ease of notation is denoted $\mathbf{x} = (x, y, \theta)$. The state is modeled to evolve according to

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}), \tag{4.1}$$

where $\mathbf{u}_k$ is the relative movement according to the odometric system and $\mathbf{w}_k$ is the process noise accounting for errors in the model. The process noise is assumed to be independent of the state $\mathbf{x}$.

Features are extracted out of raw sensor data from the external sensors. The map $\mathcal{M}$, which the robot uses for localization, is a set of features, i.e.,

$$\mathcal{M} = \{f_j \mid j = 1, \dots, M\}, \tag{4.2}$$

where M denotes the number of features in $\mathcal{M}$ and the features, f_j, are given in world coordinates. The map can be divided into T subsets where T is the number of different feature types,

$$\mathcal{M} = \left\{\mathcal{M}^t \mid t = 1, \dots, T\right\}. \tag{4.3}$$

The map is not complete, that is, not all features in the environment are represented in $\mathcal{M}$. This is in part due to the minimalistic model, but it is also unrealistic to assume that the map is complete as real world environments are never truly static. The external measurements are modeled as

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathcal{M}, \mathbf{v}_k), \tag{4.4}$$

where $\mathbf{z}_k$ is a vector of feature measurements and $\mathbf{v}_k$ is measurement noise.

Contrary to the previous chapter it is not possible to assume that the origin of the measurements can be determined unambiguously, i.e. the pairing between $\mathbf{z}_k$ and $\mathcal{M}$ is not known. In the previous chapter the problem was alleviated by keeping the uncertainty small enough to allow straightforward matching. However, global localization is the process of initializing the pose estimate, i.e. finding the pose without prior knowledge. In general the data association problem cannot be solved directly unless all features are unique. Figure 4.1 shows one such situation, where a detected door gives several possible poses for the robot. The problem of finding the pose of the robot can equivalently be formulated as the problem of associating to each measured feature a corresponding map feature. When the correspondences are determined the pose can be estimated based on the known position of the map features.

The problem presented here is closely related to the problem of multi-target tracking encountered in, for example, air surveillance applications. The tracked targets in these applications are aircrafts of different kinds. In global localization of a mobile robot, as the problem is posed in this chapter, each hypothesis about the robot pose is tracked. As pointed out in (Mazor et al. 1998) the mathematical structure of the optimal tracking algorithm is well known. However, the complexity of an optimal tracking algorithm (the Multiple Hypothesis Tracker (MHT), also known as the Track Split Filter (TSF)), is too high to be realizable in practice. The high computational complexity comes from the exponential growth in the number of hypotheses that have to be maintained, due to the fact that a hypothesis is split every time a data association is ambiguous. The original hypothesis splits into one hypothesis for each possible association. The resulting hypotheses are then split when new associations are ambiguous and so on. The result is a tree structure where each leaf representing one possible choice of the ambiguous data associations. Given finite computational resources, only a suboptimal solution can actually be implemented.

To complicate matters further, the map is not complete and the feature detectors are not perfect. Therefore each feature measurement is a result of one of the following events:

- Measurement of a map feature.
- Measurement of a feature in the environment that is not in $\mathcal{M}$.
- False positive, i.e. no feature correspondence in the environment, mapped or unmapped (an outlier).

All matches are thus ambiguous because there is always some probability that the measurement is a false positive or an unmodeled feature. The probability of each of the events depends on the quality of the feature detector, the robot pose and the map. A more complete description uses the entire state of the world, incorporating for instance moving people, and not only the map and the robot pose. This, however, is not feasible in practice.

4.1.1 Hypothesis Representation

Let $H^{(i)}$ denote the i:th hypothesis about the robot pose $\mathbf{x}$ given some assumptions about the data associations. Each hypothesis, $H^{(i)}$, is defined by a robot pose estimate, $\mathbf{x}^{(i)}$ and a corresponding covariance matrix, $P^{(i)}$. The covariance matrix defines how the probability mass is distributed (see Section 2.2.5). Each hypothesis also has a weight attached to it which measures the probability that the hypothesis is correct, $\pi^{(i)} = \Pr(H^{(i)})$. The hypothesis $H^{(i)}$ is hence represented by

$$H^{(i)} = \left\{ \mathbf{x}^{(i)}, P^{(i)}, \pi^{(i)} \right\}. \tag{4.5}$$

Assuming that the weight of the hypotheses sum to one is equivalent to assuming that one of the hypotheses is correct. Such a model is only possible if the map is complete and the feature detectors are perfect. In all other cases there is some probability that all hypotheses are wrong. To account for this possibility a zero hypothesis, $H^{(0)}$, is introduced. The probability mass of $H^{(0)}$ is $\pi_k^{(0)} = \Pr(H^{(0)})$ and specifies the probability that no hypothesis, $H_k^{(i)}$, $i = 1, \ldots, N_k$, is correct at a given time step k.

Combining the hypotheses into a Gaussian sum, the PDF of the robot pose given measurements $\mathbf{z}_0, \ldots, \mathbf{z}_k$ can be approximated by

$$p(\mathbf{x}_k|\mathbf{z}_0, \ldots, \mathbf{z}_k) \approx \pi_k^{(0)} + \sum_{i=1}^{N_k} \pi_k^{(i)} \mathcal{N}(\mathbf{x}_k^{(i)}, P^{(i)}), \tag{4.6}$$

where N_k is the number of hypotheses at time k and the weights satisfy

$$\pi_k^{(0)} + \sum_{i=1}^{N_k} \pi_k^{(i)} = 1. \tag{4.7}$$

The zero hypothesis can be viewed as a uniform distribution assigning equal probability to all possible robot states. The introduction of the zero hypothesis has the additional advantage that the initial uniform PDF can be modeled by $\pi_0^{(0)} = 1$. When new hypotheses are created, the probability mass is taken from the zero hypothesis. The stronger the evidence is for a hypothesis, the more probability mass is taken from $H^{(0)}$. A perfect detection of a unique feature results in all probability mass being transferred to one hypothesis. In a real application integration over time is needed to gather enough evidence to single out the true hypothesis.

4.1.2 Pose Hypothesis Update

If measurements can be associated with the different hypotheses, the EKF used in the previous chapter can be applied to update the pose estimate of the hypotheses. The EKF only updates the pose estimates of the different hypotheses and not their probability of corresponding to the true pose. Using Bayes' rule the probability of hypothesis $H_k^{(i)}$, after incorporating the information in $\mathbf{z}_k$, is proportional to

$$\Pr(H_k^{(i)}|\mathbf{z}_k) \propto \Pr(\mathbf{z}_k|H_k^{(i)}) \Pr(H_k^{(i)}), \; i = 1, \ldots, N_k. \tag{4.8}$$

Normalization is performed by enforcing (4.7).

In (4.8) the term $\Pr(\mathbf{z}_k|H_k^{(i)})$ expresses the probability of observing $\mathbf{z}_k$ given that the robot is at the pose suggested by hypothesis $H_k^{(i)}$. This quantity is estimated based on the map and models of the feature detectors. Assuming that $\mathbf{z}_k$ is a feature of type t, the probability $\Pr(\mathbf{z}_k|H_k^{(i)})$ can be expanded using the mutually exclusive events that the measurement is triggered by a real feature or a phantom feature (false positive). Introduce V^t as the event that a feature of type t is detectable by the sensors and the complement $\bar{V}^t$ that it is a phantom feature. The expression for $\Pr(\mathbf{z}_k|H_k^{(i)})$ is then given by

$$\Pr(\mathbf{z}_k|H_k^{(i)}) = \Pr(\mathbf{z}_k|V^t)\Pr(V^t|H_k^{(i)}) + \Pr(\mathbf{z}_k|\bar{V}^t)\Pr(\bar{V}^t|H_k^{(i)}). \tag{4.9}$$

Here $\Pr(\mathbf{z}_k|V^t)$ denotes the probability of making the observation $\mathbf{z}_k$ given that a feature of type t is visible to the sensors. In other words, $\Pr(\mathbf{z}_k|V^t)$ is a model of the reliability of the detector extracting a visible feature of type t from sensor data. The second factor, $\Pr(V^t|H_k^{(i)})$, gives the probability of a feature of type t being visible from the pose given by hypothesis $H_k^{(i)}$. The second term expresses the probability of making the observation even though such a feature cannot be seen, i.e. the probability of a false positive. As the map is not complete a distinction must be made between detecting a modeled or an unmodeled feature. The unmodeled features do not provide any information. Introducing the notation $V^t \in \mathcal{M}^t$ as the event that there is a modeled and detectable feature of type t, the probability $\Pr(V^t|H_k^{(i)})$ can be divided into

$$\Pr(V^t|H_k^{(i)}) = \Pr(V^t \in \mathcal{M}^t|H_k^{(i)}) + \Pr(V^t \notin \mathcal{M}^t|H_k^{(i)}) \tag{4.10}$$

as a feature cannot be modeled and unmodeled at the same time. With this (4.9) expands to

$$\begin{aligned}\Pr(\mathbf{z}_k|H_k^{(i)}) = &\Pr(\mathbf{z}_k|V^t)\Pr(V^t \in \mathcal{M}^t|H_k^{(i)}) + \\ &\Pr(\mathbf{z}_k|V^t)\Pr(V^t \notin \mathcal{M}^t|H_k^{(i)}) + \\ &\Pr(\mathbf{z}_k|\bar{V}^t)\Pr(\bar{V}^t|H_k^{(i)}).\end{aligned} \tag{4.11}$$

From (4.11) it is clear how devastating it is to assume a complete map and perfect detection when such is not the case. Such an assumption is equivalent to the last two terms in (4.11) being equal to zero. Detecting an unmodeled feature or getting a false positive thus makes $\Pr(\mathbf{z}_k|H_k^{(i)})$ evaluate to zero and hence hypothesis $H_k^{(i)}$ is incorrectly rejected.

4.1.3 Exploration

Due to the uncertain nature of any sensor data, it is only by integrating information over time that any level of certainty can be reached. For successful localization it is thus important that the robot is exposed to a rich set of features, through efficient exploration of the environment. For best results the exploration must be connected to the localization, i.e. the information gathered

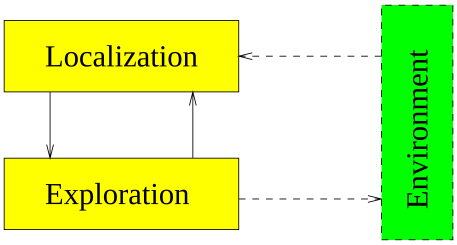


Figure 4.2: *An active exploration strategy uses the information gathered by the localization procedure to select an action. Feedback can be given from the exploration module directly or indirectly through the environment.*

by the localization procedure must influence the robot motion. Such an exploration strategy is referred to as *active*. Initially, before the localization system has gathered any information, the exploration has to be passive. The coupling between the exploration and the localization can be either *direct* or *indirect* through the environment. The indirect connection between exploration and localization is always present, since new features come into view as the robot moves. An active strategy can be seen as closing the loop between localization and exploration (see Figure 4.2).

4.2 Algorithm

To simplify matters it is assumed that the feature measurements can be treated independently. This allows for a sequential treatment of the detected features. Without loss of generality it is henceforth assumed that $\mathbf{z}_k$ is a single feature observation.

4.2.1 Feature Detectors

The features are extracted from sensor data using so called feature detectors. Each detector is specialized in detecting a certain feature type. They can be viewed as matched filters, tuned to certain patterns in the sensor data. In this thesis all features are defined by the user, but in principal any feature could be used. Figure 4.3 gives a simple illustration of the setup.

To ensure as much independence as possible the detectors keep a list of all features detected so far. Quoting (Simmons & Koenig 1995) "*repeatedly using the same sensor in the same location will usually produce highly correlated results*". The positions of all detected features are stored in odometric coordinates, which is a natural choice as the true pose is unknown. All extracted features are matched to the list of already detected features. Only

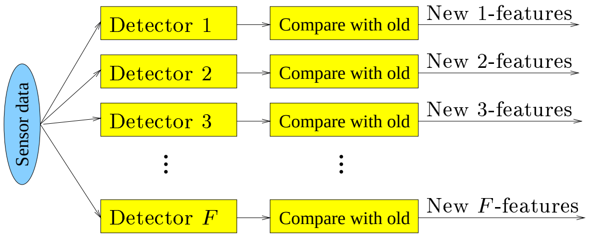


Figure 4.3: *Each feature type has a dedicated detector. A list of already detected features is kept to ensure that only new features are reported and used for localization.*

those features that do not match any in the list are reported to the localization algorithm and thus used to find the pose of the robot. In the matching with already detected features no compensation is done for odometric drift. Hence, when the robot has drifted a certain amount old feature may be reported again. At this point the assumption about independent measurements is not as strong.

4.2.2 Pose Candidates

It is clear that the number of hypotheses influence the computational resources required to run MHL. How hypotheses are initiated is therefore a key issue. In (Fortmann et al. 1983) it is pointed out that with large amounts of false measurements, it is not possible to initiate a new hypothesis based on each new measurement. In the target tracking literature it is therefore common to initiate new targets when a few consecutive measurements support their existence.

Each detected feature generates "guesses" about the robot pose, when combined with the map (compare Figure 4.1). In MHL these are approximated by a sum of Gaussian PDFs. If $\mathbf{z}_k$ is an observation of a feature, $f_j \in \mathcal{M}$ a Gaussian sum approximation of $p(\mathbf{x}_k|\mathbf{z}_k)$ results in one Gaussian being placed at the true robot pose (solid robot icon in Figure 4.1). Here the Gaussians in the approximation of $p(\mathbf{x}_k|\mathbf{z}_k)$ are referred to as pose candidates, and are denoted $C_k^{(j)}$. Each pose candidate is defined by a pose, $\mathbf{y}_k^{(j)}$, and a corresponding covariance matrix, $R_k^{(j)}$, just like a hypothesis. Hence

$$C_k^{(j)} = \left\{ \mathbf{y}_k^{(j)}, R_k^{(j)}, f_j \right\}, \tag{4.12}$$

where f_j is the map feature to which the observation is matched.

The candidates act as measurements of the hypotheses. If candidate $C^{(j)}$ is a measurement of hypothesis $H^{(i)}$ then the measurement equation is modeled as

$$\mathbf{y}_k^{(j)} = \mathbf{x}_k^{(i)} + \mathbf{v}_k^{(j)}, \tag{4.13}$$

where $\mathbf{v}_k^{(j)}$ has covariance matrix $R_k^{(j)}$.

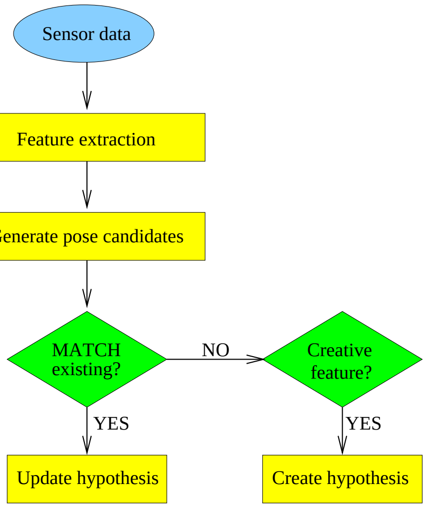


Figure 4.4: *Detected features are used to generate pose candidates, which are matched to the hypotheses and then used to update the corresponding hypothesis. Unmatched candidates from creative features initiate new hypotheses.*

Localization is realized by updating the hypotheses with matching candidates. Turning every unmatched candidate into a hypothesis soon exhaust the computational power of any computer, therefore a mechanism is needed for selecting which candidates to turn into hypotheses. The characteristics of the different features and their corresponding detectors are used to determine how candidates become hypotheses. To this end, two different types of features are distinguished between, *creative* ($\mathcal{C}$) and *supportive* ($\mathcal{S}$). The creative features carry enough strength to initiate new hypotheses whereas the supportive features can only give support to existing hypotheses. Characteristic for creative features is that they produce a relatively small number of candidates and that the detector has a low rate of false positives. The first criterion stops the number of hypotheses from growing too fast whereas the second ensures that the hypotheses have a reasonable probability of being true. A supportive feature is typically easy to detect and is encountered frequently in the environment.

4.2.3 Matching Candidates to Hypotheses

Matching candidates to existing hypotheses is an important step in MHL. Errors in this step can have devastating effects on the overall performance. There are several techniques suggested for the data association. Usually, each hypothesis has a validation gate defined around it in which a measurement/candidate must fall (see Section 2.2.5). To test whether candidate $C_k^{(j)}$ matches hypothesis $H_k^{(i)}$ the residual

$$\nu_k^{i,j} = \mathbf{y}_k^{(j)} - \mathbf{x}_k^{(i)}. \tag{4.14}$$

is defined. The Mahalanobis distance, $\rho_k^{i,j}$, between $C^{(j)}$ and $H^{(i)}$ at time k can be expressed as (2.22)

$$\rho_k^{i,j} = \nu_k^{i,j}(S_k^{i,j})^{-1}(\nu_k^{i,j})^T, \tag{4.15}$$

where $S_k^{i,j}$ is given by (2.9) and (4.13)

$$S_k^{i,j} = P_k^{(i)} + R_k^{(j)}. \tag{4.16}$$

The minimalistic environmental model provides a sparse representation of the environment. Candidates are typically clearly separated as long as they are generated by modeled features.

4.2.4 Hypotheses Initiation and $\pi_k^{(0)}$

The unmatched creative candidates are used to initiate new hypotheses. Making a hypothesis out of a candidate is easy since they have the same representation (compare (4.5) and (4.12)). The weight that is assigned to a new hypothesis depends on the weight of the zero hypothesis, the map and the quality of the feature detector in question. A correct hypothesis is generated as soon as a modeled creative feature is detected. The probability that no correct hypothesis exists is thus the probability that only unmodeled and false positives are detected so far. The probability that $\mathbf{z}_k$ is the result of an unmodeled feature or a false positive conditioned on $H^{(0)}$ is given by the two last terms in (4.11)

$$\Pr(\mathbf{z}_k|V^t)\Pr(V \notin \mathcal{M}^t|H^{(0)}) + \Pr(\mathbf{z}_k|\bar{V}^t)\Pr(\bar{V}^t|H^{(0)}). \tag{4.17}$$

Put differently, this is the probability that $H^{(0)}$ remains true. The complement is the probability that the true pose is found among the new hypotheses. Assuming that all U unmatched candidates are just as likely each of the new hypotheses are initialized with probability

$$\frac{1}{U}\Delta\pi_k^{(0)}, \tag{4.18}$$

where

$$\begin{aligned}\Delta\pi_k^{(0)} = \pi_k^{(0)}\,(1 - [\ &\Pr(\mathbf{z}_k|V^t)\Pr(V^t \notin \mathcal{M}^t|H^{(0)}) + \\ &\Pr(\mathbf{z}_k|\bar{V}^t)\Pr(\bar{V}^t|H^{(0)})\big]\big)\end{aligned} \tag{4.19}$$

is the probability mass subtracted from the zero hypothesis.

Initiating New Hypotheses

$\pi_k^{(0)} := \pi_k^{(0)} - \Delta\pi_k^{(0)}$
$\Delta\pi_k^{(0)} := \pi_k^{(0)} \left(1 - \left[\Pr(\mathbf{z}_k|V^t)\Pr(V^t \notin \mathcal{M}^t|H^{(0)}) \;+\; \Pr(\mathbf{z}_k|\bar{V}^t)\Pr(\bar{V}^t|H^{(0)})\right]\right)$
$i := N_k$
`For all unmatched candidates` $C_k^{(j)},\ j \in \mathcal{U}$
  $i := i + 1$
  $\mathbf{x}_k^{(i)} = \mathbf{y}_k^{(j)}$
  $P_k^{(i)} = R_k^{(j)}$
  $\pi_k^{(i)} = \frac{1}{U}\Delta\pi_k^{(0)}$
$N_k := i$
$pi_k^{(0)} := pi_k^{(0)} - \Delta\pi_k^{(0)}$

Figure 4.5: *The U unmatched candidates are turned into hypotheses.*

4.2.5 Pruning the Hypotheses Tree

To save computation time, the hypothesis tree must be pruned, i.e. the number of hypotheses must be reduced.

Probability Thresholding

A straightforward way of discarding hypotheses is to use a lower threshold on the allowed probability of a hypothesis. Let η be this lower limit. Hypotheses that are kept must thus satisfy

$$\pi_k^{(i)} \geq \eta,\ k = 1, \ldots, N_k. \tag{4.20}$$

The probability mass of a removed, too weak, hypothesis is returned to $H^{(0)}$.

Outside the Known Environment

Even though the map is not complete it can still be assumed that the environment is bounded and that these bounds are known. Hypotheses that are significantly outside the boundaries can thus be discarded. The corresponding probability mass is not returned to the zero hypothesis but is instead redistributed among the other hypotheses through normalization according to (4.7).

Occupied Areas

Similar to being outside the map is when a hypothesis is in an area which is known to be occupied. Examples of such areas are at the positions where walls are located. Monitoring for this criterion is difficult as the hypotheses have spatial uncertainty. Removing a hypothesis on false grounds must be avoided as it could be the true hypothesis. Applying this criterion in practice is considered to be associated with too much risk and is therefore not used.

Merging Hypotheses

Yet another way to reduce the number of hypotheses is to match and merge hypotheses. If two hypothesis satisfy

$$(\mathbf{x}_k^{(i)} - \mathbf{x}_k^{(j)})(P_k^{(i)} + P_k^{(j)})^{-1}(\mathbf{x}_k^{(i)} - \mathbf{x}_k^{(j)})^T < \gamma, \tag{4.21}$$

where $\pi_k^{(i)} \geq \pi_k^{(j)}$ is assumed without loss of generality, hypothesis $H_k^{(j)}$ is deleted and

$$\pi_k^{(i)} := \pi_k^{(i)} + \pi_k^{(j)}. \tag{4.22}$$

The overall MHL algorithm is summarized in Figure 4.6.

Algorithm: Multiple Hypothesis Localization (MHL)

```
Wait for feature
Generate pose candidates
Predict new pose for all hypotheses using (2.18)
N_{k+1} := N_k
Loop over all candidates C_k^(j)
    Loop over all hypotheses H_k^(i),  i = 1,...,N_k
       if (C_k^(j) matches H_k^(i)) according to (2.23)
          Update H_k^(i) using (2.21) and (4.11)
       else (C_k^(j) ∈ C)
          Initiate new hypothesis as in Figure 4.5
H_{k+1}^(i) := H_k^(i),  i = 1,...,N_{k+1}
Normalize π_{k+1}^(i),  i = 1,...,N_{k+1} using 4.7
Prune hypothesis tree according to Section 4.2.5
Re-normalize
```

Figure 4.6: *The overall MHL algorithm.*

4.2.6 Exploration

In (Burgard, Fox & Thrun 1997) the exploration strategy is based on minimizing the entropy of $p(\mathbf{x}_k|\mathbf{z}_0,\ldots,\mathbf{z}_k)$. The concept of entropy comes originally from thermodynamics. The kind of entropy used in (Burgard, Fox & Thrun 1997) is called logical entropy and measures the amount of disorder. The entropy is maximized with total disorder. Minimizing the entropy is the

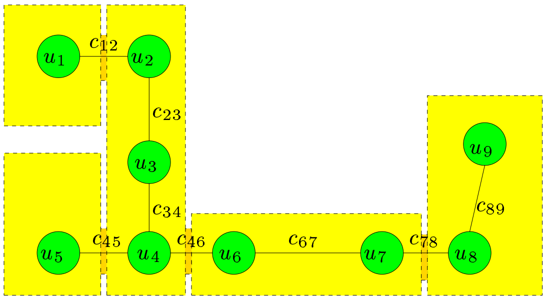


Figure 4.7: *An example of a graph used for exploration planning*

same as gathering probability mass in one place. The exploration strategy is thus directly related to the aim of localization.

In (Cassandra et al. 1996) partially observable Markov decision processes (POMDPs) are used as the basis for the decisions in a topological localization scheme. Various heuristics are presented to make the POMDPs computationally tractable.

An efficient exploration strategy avoids visiting the same place twice and strives to lead the robot to areas where there are many (hopefully discriminative) features. As the exploration is not the focus in this chapter a simple strategy based on a topological description of the environment is used, where the edges indicate straight and obstacle free paths. Each node and edge is given utility and cost values respectively. Let n_i denote the i:th node. The utility value, u_i, of node n_i measures the information content that can be acquired in its neighborhood. The cost, c_{ij}, of an edge connecting nodes n_i and n_j measures the cost of traveling between the two nodes. Figure 4.7 shows a small environment with a corresponding topological graph. The cost, c_{ij} associated with an edge is given by its length,

$$c_{ij} = \text{dist}(n_i, n_j). \tag{4.23}$$

The one exception to this rule concerns edges going through a door. Passing through a door takes longer time than driving within a room as the robot has to slow down through a narrow passage. This is accounted for by adding an extra cost, c_{door}, to these edges

$$c_{ij} = \begin{cases} \text{dist}(n_i, n_j) + c_{door}, & n_i \text{ and } n_j \text{ connected by door} \\ \text{dist}(n_i, n_j), & \text{otherwise.} \end{cases} \tag{4.24}$$

The utility of a node n_i is given as the sum of the utility of the features visible from n_i. For simplicity the utility is assumed to be constant for all features of the same type. Let u^t denote the utility of a feature of type t. An extension to using individual utility values is straightforward. The utility, u_i, of a node, n_i, is now given by

$$u_i = \sum_{t=1}^{T} |\mathcal{V}^t(n_i)| u^t, \tag{4.25}$$

where $|\mathcal{V}^t(n_i)|$ is the number of visible feature of type t from node n_i.

Given the utility of the nodes and the cost of the edges the overall exploration strategy is divided into two phases which are explained below.

1. Initialization

The aim of the first phase is to generate a set of hypotheses. At startup, when no hypotheses exist, the active exploration strategy has no input and cannot yet be used. During this time the robot moves in one direction as long as there is free space and then selects a new direction based on the current sensor data. This simple strategy does not guarantee good coverage, but has proven to be sufficient in most cases. Upon seeing the first creative feature, hypotheses are created with equal weights. It is not until more features are detected that the hypotheses start to differ in weight. The simple exploration behavior is active until the probability of the best hypothesis reaches a certain threshold, π_{thres}, i.e. switching is done when

$$\max_{i=1,\ldots,N_k} \pi_k^{(i)} \geq \pi_{thres}. \tag{4.26}$$

2. Selection

The aim of the second phase is to single out the true hypothesis. To achieve this the robot must move to new locations to detect new features. Using a topological representation this corresponds to visiting new nodes. Assuming that the best hypothesis is the true hypothesis, an estimate of the robot pose is given. The first decision taken by the exploration behavior is to get the robot to the closest node according to the pose estimate. Once on the graph the strategy is to follow the edges between the nodes. The next node to visit is selected by maximizing the expected utility using a one step horizon, i.e., only the closest nodes that have not yet been visited are considered. This is an example of greedy/myopic planning.

When the robot fails to move to the next selected node, it is indication that the pose estimate is wrong. Ruling out a hypothesis completely based on such a failure is not possible as the map is not complete and the edges cannot be guaranteed to be free from obstacles. However, the probability of the hypothesis is reduced,

$$\pi_{after}^{(i)} := \pi_{before}^{(i)} \Pr(\text{move fail}|H^{(i)}). \tag{4.27}$$

The larger π_{thres} is chosen, the longer the first strategy is active. In an environment that is both feature rich and easy to explore using random walk, this results in few and strong hypotheses. If π_{thres} is chosen low on the other hand, the second stage typically starts with many hypotheses that are approximately equally probable. The parameter π_{thres} has to be chosen low enough to be applicable in all parts of the environment.

4.3 Implementation

So far the description of MHL has been general. This section describes some of the implementation details. This includes, for example, presenting the features being used and a motivation for how to categorize them into being creative or only supportive.

	line	door	point pair
$\Pr(\mathbf{z}\vert V^t)$	0.7	0.5	0.5
$\Pr(\mathbf{z}\vert \bar{V}^t)$	0.15	0.25	0.05

Table 4.1: *Observed detector performance for different features.*

4.3.1 Feature Detection Models

In this implementation of MHL three different feature types are considered: lines, doors and points. As illustrated by Figure 4.3, each feature has a dedicated detector. The performance of these detectors determine the quantities $\Pr(\mathbf{z}|V^t)$ and $\Pr(\mathbf{z}|\bar{V}^t)$ in (4.11), which are shown in Table 4.1 for the different feature types. The values of $\Pr(\mathbf{z}|V^t)$ and $\Pr(\mathbf{z}|\bar{V}^t)$ are given by evaluating the detectors over a large set of data. To fully specify (4.11),

$$\Pr(V^t|H^{(i)})$$

is also needed for all feature types t. Assume that candidate $C_k^{(j)}$ matches hypothesis $H_k^{(i)}$ and is generated by $V^t \in \mathcal{M}^t$. It is then natural to set

$$\Pr(V^t \notin \mathcal{M}^t|H^{(i)}) = 0 \tag{4.28}$$

as the candidate would not have been created without the feature being in the map, which gives

$$\Pr(V^t|H^{(i)}) = \Pr(V^t \in \mathcal{M}^t|H^{(i)}) \tag{4.29}$$

according to (4.10).

In (Larsson et al. 1994) an expression is presented for the probability of $C_k^{(j)}$ being an observation of $V^t \in \mathcal{M}^t$ given $H_k^{(i)}$

$$p(V^t \in \mathcal{M}^t | H_k^{(i)}) = \frac{1}{\sqrt{(2\pi)^3 |S_k^{i,j}|}} e^{-\frac{\rho_k^{i,j}}{2}}, \tag{4.30}$$

where $\rho_k^{i,j}$ is given by (4.15). The probability $\Pr(V^t \in \mathcal{M}^t | H_k^{(i)})$ is estimated as

$$\Pr(V^t \in \mathcal{M}^t | H_k^{(i)}) = e^{-\frac{\rho_k^{i,j}}{2}}. \tag{4.31}$$

In the same way $\Pr(V^t | H_k^{(i)})$ is taken to be $\Pr(V^t \notin \mathcal{M}^t | H_k^{(i)})$ when $H_k^{(i)}$ and $C_k^{(j)}$ do not match, where $\Pr(V^t \notin \mathcal{M}^t | H_k^{(i)})$ is modeled as a global constant for each feature type t, representing the quality of the map.

The following subsections describe how candidates are generated for the different features. Since the platform moves when the features are detected and it is difficult to determine exactly when the sensor data was acquired, extra uncertainty is added to the measurement covariance matrix R_k for all candidates

$$R_k^{motion} = \begin{pmatrix} \sigma_{xy}^{motion} & 0 & 0 \\ 0 & \sigma_{xy}^{motion} & 0 \\ 0 & 0 & \sigma_{\alpha}^{motion} \end{pmatrix}. \tag{4.32}$$

Lines

The line feature is characterized by its ability to give accurate information about the perpendicular distance and relative angle to the robot. When matching a line segment to the map only those two degrees of freedom have strong constraints. Figure 4.8 illustrates how a candidate is created by matching the line segment to a map line feature. The larger the difference in length is, the larger the uncertainty of the pose candidate is along the line. A match is assumed possible if the detected line is shorter than or equal in length to the map line feature, i.e.

$$l \leq L, \tag{4.33}$$

where l and L are defined in Figure 4.8.

The covariance matrix, $R_k^{(j)}$, is built from several contributing factors. For simplicity it is assumed that $R_k^{(j)}$ is block diagonal, so that the uncertainty in position and orientation can be treated independently. The position uncertainty is easiest to express in a coordinate system with one axis (y) parallel to the map line feature. In this coordinate system the position covariance is assumed to be diagonal. In the x-direction, i.e., perpendicular to the line, the variance is modeled as

$$\sigma_{x_{line}}^2 = \sigma_c^2 + d^2\sigma_\alpha^2. \tag{4.34}$$

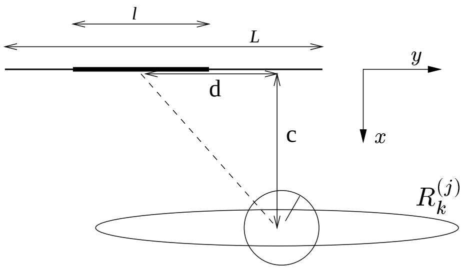


Figure 4.8: *Each match between a detected line feature and a map line feature generates one pose candidate. A line feature gives limited information about the pose along the map line, hence the elongated uncertainty ellipse.*

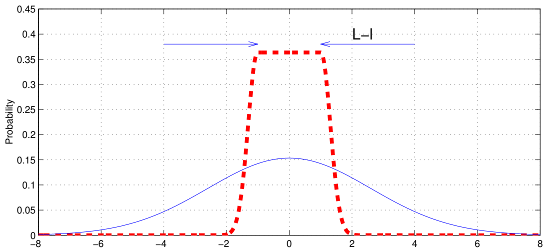


Figure 4.9: *The Gaussian distribution is a rough estimate of the PDF for the match along a map line feature. The probability should be the same for a match all along the line as the thick dashed curve shows. To approximate this, the Gaussian must be given a large variance. The difference in length in this figure is $L - l = 2$ m.*

Here σ_c^2 is the variance in the perpendicular direction, σ_α^2 is the variance in orientation of the line and the parameter d is defined in Figure 4.8. The uncertainty along the y-direction of the map line depends on the length difference, $L - l$. The Gaussian assumption is a rather poor approximation in this direction as the probability of matching the detected line to any part of the map line feature should in fact be the same (the thick dashed curve in Figure 4.9). To achieve this the variance along the line must be made artificially high. In practice this has the effect that no, or very little, information is left in this direction.

The variance is set to

$$\sigma^2_{y_{line}} = 4((L - l) + \Delta L_{min})^2, \tag{4.35}$$

where ΔL_{min} keeps the variance from becoming too small when the detected line and the map line feature have the same length. In this implementation $\Delta L_{min} = 0.3$ m. An alternative approach is to use a Gaussian sum approximation for correspondences where $L - l$ is large, i.e. to divide the elongated candidates into several smaller ones.

The covariance matrix for the position uncertainty of the candidate must be expressed in the world coordinate system. This is achieved by applying a rotation matrix to it. Let ψ be the direction of the map line feature. The rotation matrix between the line coordinate system and the world coordinate system is then given by

$$\Theta = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}, \tag{4.36}$$

and the position covariance matrix can then be written

$$R_k^{(j),xy} = \Theta \begin{pmatrix} \sigma^2_{x_{line}} & 0 \\ 0 & \sigma^2_{y_{line}} \end{pmatrix} \Theta^T. \tag{4.37}$$

The total pose covariance of a line candidate can now be expressed in world coordinates as

$$R_k^{(j)} = \begin{pmatrix} R_k^{(j),xy} & \mathbf{0} \\ \mathbf{0} & \sigma^2_\alpha \end{pmatrix} + R_k^{motion}, \tag{4.38}$$

where R_k^{motion} is defined in (4.32).

In Figure 4.10, the true robot position is marked by the black dot. A line segment is detected at the lower corner of that room. The candidates resulting from the detected feature are shown as dashed ellipses. As the detected line segment is only three meters long, candidates resulting from matches with the corridor lines are quite elongated. The longer the detected line is the fewer the matches are.

There are relatively few line features in the map ($M^{line} = 136$). From this perspective the line feature is a creative feature. However, it fails on not properly constraining all three degrees of freedom. Therefore the line feature is only used as a supportive feature.

Doors

The door feature constrains all degrees of freedom of the robot, but can be treated very much like the line feature. Some doors can only be seen from one side as they lead to unmapped areas. This is handled in the model by assigning a door to each direction that a physical door can be traversed. The benefit of

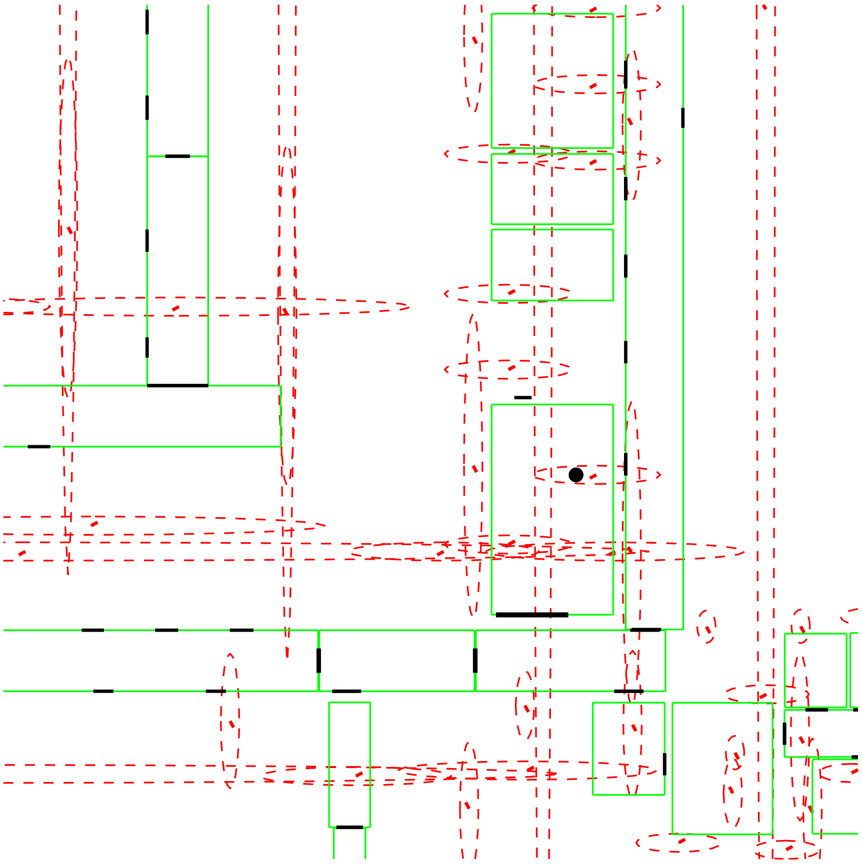


Figure 4.10: *The true robot pose is given by the black dot in the center room. Detecting the the line feature in the lower corner of that room results in the candidates marked as dashed ellipses.*

this is that each match with a map door feature only gives one candidate. Figure 4.11 shows the uncertainty ellipse that results from detecting a door. The door feature constrains the position along the door better than a line does, but the uncertainty in orientation and perpendicular distance is larger. This is an effect of the thick walls (≈ 0.4 m) in the building where the experiments are performed.

For the door feature there are two coordinate systems in which the different contributions to the overall covariance matrix are easiest expressed. The first

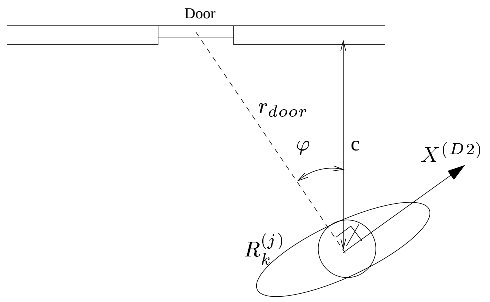


Figure 4.11: *A door feature constrains all degrees of freedom of the robot. The angle of the door and the perpendicular distance to it are rather uncertain because of the thick walls.*

one is the same as for the line feature

$$\sigma^2_{x_{door}} = \sigma^2_c \tag{4.39}$$

$$\sigma^2_{y_{door}} = ((W - w) + \Delta W_{min})^2, \tag{4.40}$$

where W is the width of the map door feature and w is the width of the detected door. The parameter ΔW_{min} has the same purpose as ΔL_{min} for the line feature and is here 0.2 m. The perpendicular uncertainty σ_c is 0.4 m, to account for the thick walls. To get this part of the covariance matrix into world coordinates the rotation matrix (4.36) is used

$$R_k^{(j),1} = \Theta \begin{pmatrix} \sigma^2_{x_{door}} & 0 \\ 0 & \sigma^2_{y_{door}} \end{pmatrix} \Theta^T. \tag{4.41}$$

The second coordinate system to consider is the coordinate system defined by the direction of observation, φ. The x-axis of this coordinate system is marked with $D2$ in Figure 4.11,

$$\sigma^2_{x_{D2}} = r^2_{door} \cos^2(\sigma_\alpha) \tag{4.42}$$

$$\sigma^2_{y_{D2}} = r^2_{door} \sin^2(\sigma_\alpha). \tag{4.43}$$

where r_{door} is the distance from the robot to the door. The rotation matrix that relates the coordinate system with the world coordinate system is given by

$$\Theta_{D2} = \begin{pmatrix} \cos(\psi + \varphi) & -\sin(\psi + \varphi) \\ \sin(\psi + \varphi) & \cos(\psi + \varphi) \end{pmatrix}. \tag{4.44}$$

Using Θ_{D2} the second contribution to the position covariance can be expressed as

$$R_k^{(j),2} = \Theta_{D2} \begin{pmatrix} \sigma^2_{x_{D2}} & 0 \\ 0 & \sigma^2_{y_{D2}} \end{pmatrix} \Theta^T_{D2}. \tag{4.45}$$

This results in the total covariance matrix in world coordinates becoming

$$R_k^{(j)} = \begin{pmatrix} R_k^{(j),1} + R_k^{(j),2} & \mathbf{0} \\ \mathbf{0} & \sigma_\alpha^2 \end{pmatrix} + R_k^{motion}. \tag{4.46}$$

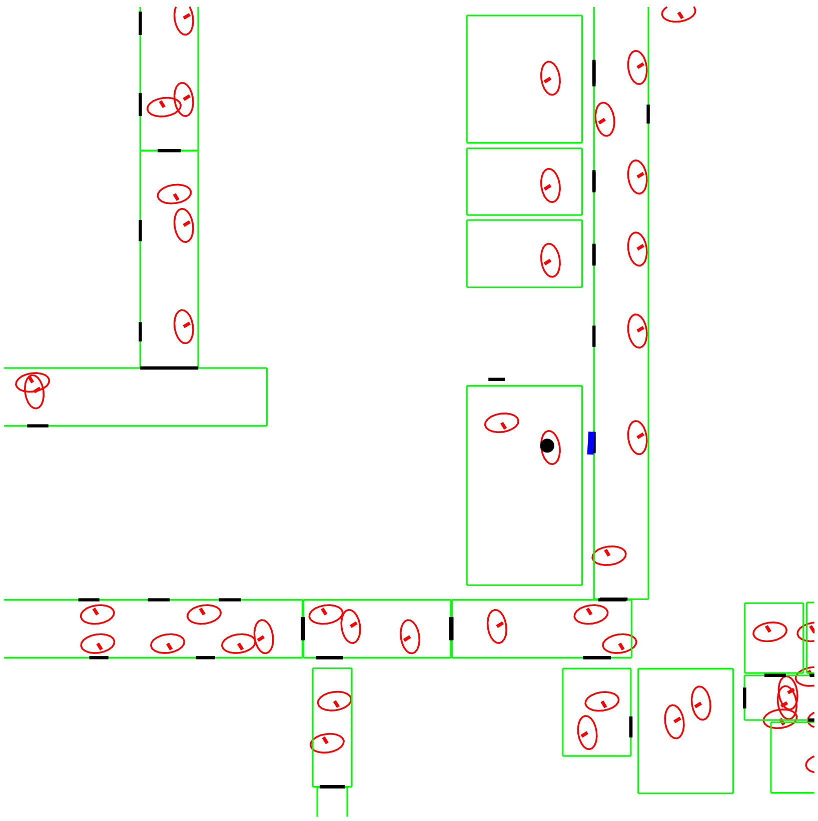


Figure 4.12: *The candidates that are generated from a detected door. The true robot pose is marked by the black dot and the detected door is to the right of the robot. One candidate is generated for each side a door can be detected.*

Figure 4.12 shows the candidates when detecting a door feature. As before the true robot position is marked by the black dot. The door that is detected is the one to the right, marked with the thick line.

In summary the door feature is a logic choice for being a creative feature. There are only 108 door features, therefore relatively few candidates are created. Each correspondence between map and observation constrains all three degrees of freedom which means that the candidates have a relatively well concentrated probability mass.

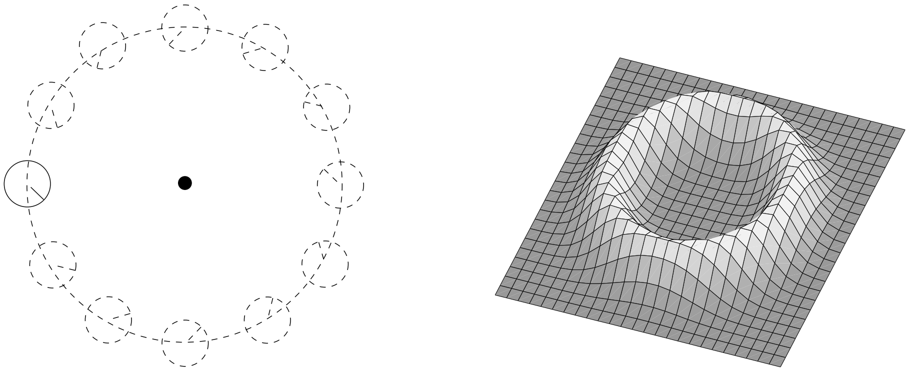


Figure 4.13: *Every match between a detected point and a map point feature constrains the robot pose to be on a circle.*

Pairs of Points

A point feature is not well suited to create pose candidates from. Even if the correspondence problem is solved, i.e. the identity of the map feature is known, the possible robot poses are along a circle with the radius equal to the distance to the point (see left subfigure of Figure 4.13). The right subfigure of Figure 4.13 shows the xy part of $p(\mathbf{x}|\mathbf{z})$. It is clear that it is not possible to approximate this with a Gaussian PDF.

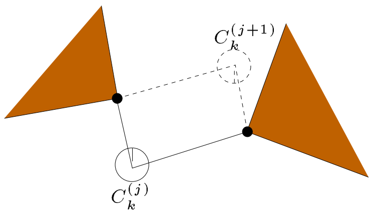


Figure 4.14: *Matching a point pair to the map gives two pose candidates. The points cannot be distinguished between and can thus be combined with the map pair in two ways.*

Another approach is to combine points to form pairs. Figure 4.14 shows a situation where the robot has detected two points marked as black dots. Since the points are indistinguishable the matching can be done in two ways. Hence, every detected pair of points that are matched to a map pair generates two candidates. These candidates are marked with small circles in Figure 4.14 and are better approximated by Gaussian PDFs. The uncertainty is assumed to be equal in the x and y directions and uncorrelated, yielding a diagonal covariance matrix.

The map contains 738 point features. Combining every point with every other point gives $\binom{738}{2} = 263,175$ possible pairs. Clearly, many of these pairs are never observed. Therefore only point combinations where the points are closer than 10 m and further apart than 1 m are considered. A further constraint is that the points are in the same room. With these constraints 9,578 pairs can be formed.

The distance between the points in a detected pair can be used to prune the number of possible map matches. A correspondence is assumed possible only if

$$|\,\mathrm{dist}(p_1, p_2) - \mathrm{dist}(f_1^{point}, f_2^{point})| < d_{pp}. \tag{4.47}$$

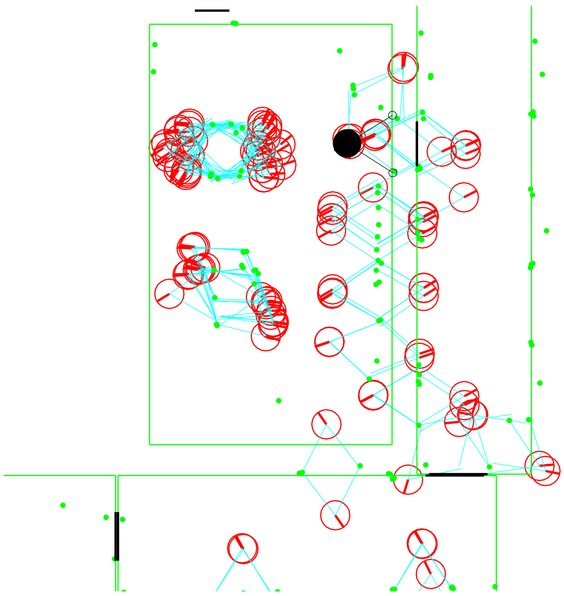


Figure 4.15: *The true robot position is marked by the black circle. Two point features are detected to the right of the robot (dark circles). The candidates are marked with dashed ellipses. The two map point features are connected with the corresponding candidate.*

Figure 4.15 illustrates how the candidates are distributed when two points are combined to form a pair and matched to the map. Two points are detected to the right of the robot, one on each side of the door opening. There is a true correspondence in the map for this point pair and thus a candidate is generated at the true robot pose. Criterion (4.47) effectively narrows the number of possible candidates down to 640 in this case. Still, as can be seen, there is quite a dense collection of candidates in some areas where clusters of points have approximately the right distance between them.

In summary, pair features formed by two point features constrain all three degrees of freedom and give candidates with a comparably small distribution of probability mass. The downside of the pair feature is that there are so many of them. In a typical situation between 300–700 matches are found for each detected point pair. Each pair match can be combined in two ways which often gives in excess of a thousand candidates. This influx of hypotheses soon exhausts the computer. The pair feature must therefore be considered as supportive, and not be given the power to create new hypotheses. In (Wijk 2001) a method is developed that exploits the visibility constraint on the points in the map. With such a scheme the number of matches can be reduced by more than an order of magnitude. Then the pair feature is better suited for being a creative feature.

4.3.2 Hypotheses Splitting

Applying hypothesis splitting in full scale as in the optimal MHT filter is not possible, but the mechanism of splitting hypotheses is still of interest. In this implementation a split creates two identical copies, one which is updated using the matching candidate (accounts for the possibility that it was a true match) and one which is not. To limit the exponential growth, only creative candidates are give the ability to cause a split and the hypotheses are merged again after the update if they then match. When merging two split hypotheses, the probability is added and the pose estimate of the updated hypothesis is used.

4.3.3 Pruning the Hypotheses Tree

As described in Section 4.2.5 hypotheses that are too weak are joined with the zero hypothesis to reduce the computational demand. The parameter η in (4.20) is chosen as 0.001. Section 4.2.5 also presents a way to discard hypotheses by checking if the corresponding pose is outside the known environment, which is implemented by testing each pose hypothesis against the rectangular room models in the minimalistic map. The uncertainty of the pose estimates are not explicitly accounted for. Poses that are more than one meter outside any rectangle is assumed to be outside and hence the corresponding hypothesis is discarded. Experiments show that this is a sufficient approximation. Care has to be taken when the uncertainty of the hypotheses become large though.

4.3.4 Exploration

The topological graph that is needed for the exploration strategy is automatically generated. It is based on so called goal points that are manually distributed throughout the environment. These goal points represent positions to which the robot can be commanded and are part of the already existing

feature	**Utility [m]**
line	1
door	5
point pair	0.5

Table 4.2: *Utilities value for different features*

system. The procedure to turn the set of goal points into a topological graph is given by

1. Let each goal point be a node.
2. Connect nodes in a room with edges. Keep the number of edges down by not allowing the connections of a node to have directions that are to close.
3. Add nodes in front of every door connect pairs of nodes through the corresponding doors.
4. Connect the door nodes with the closest node in the room.
5. Calculate the utility value for each node.
6. Attach a cost to each edge.

The resulting graph is shown in Figure 4.16 where each node is marked as a dot and the edges are marked as lines. The thick lines mark edges going through a door. The dark nodes are goal points and the lighter ones are nodes before the doors.

The switching between the two steps of the exploration strategy presented in Section 4.2.6 is done when (4.26) is fulfilled. Here $\pi_{thres} = 0.02$ is used.

To complete the description of the exploration, the cost and utility values must be specified. Each feature is given a utility value, which is expressed in distance (see Table 4.2). The creative door is given the highest utility value and the point pair the lowest. The selection of utility value is based purely on intuition and no optimality claim is made. A more detailed implementation takes into account which features have already been detected, as these should not be allowed to provide the same amount of utility to nodes any more.

As already stated in Section 4.2.6 the cost of traversing an edge is given by the distance between the nodes. Having a high cost for passing a door, as suggested by (4.24), means that the robot prefers to explore as much as possible without passing a door. In highly symmetric environments, passing through doors is often the only way to resolve the symmetries. In such cases a negative cost can be assigned to the door, effectively making it a utility. In the experiment below the cost for passing a door is set to $c_{door} = 10$. The choice is motivated by the fact that the door passage algorithm is rather slow.

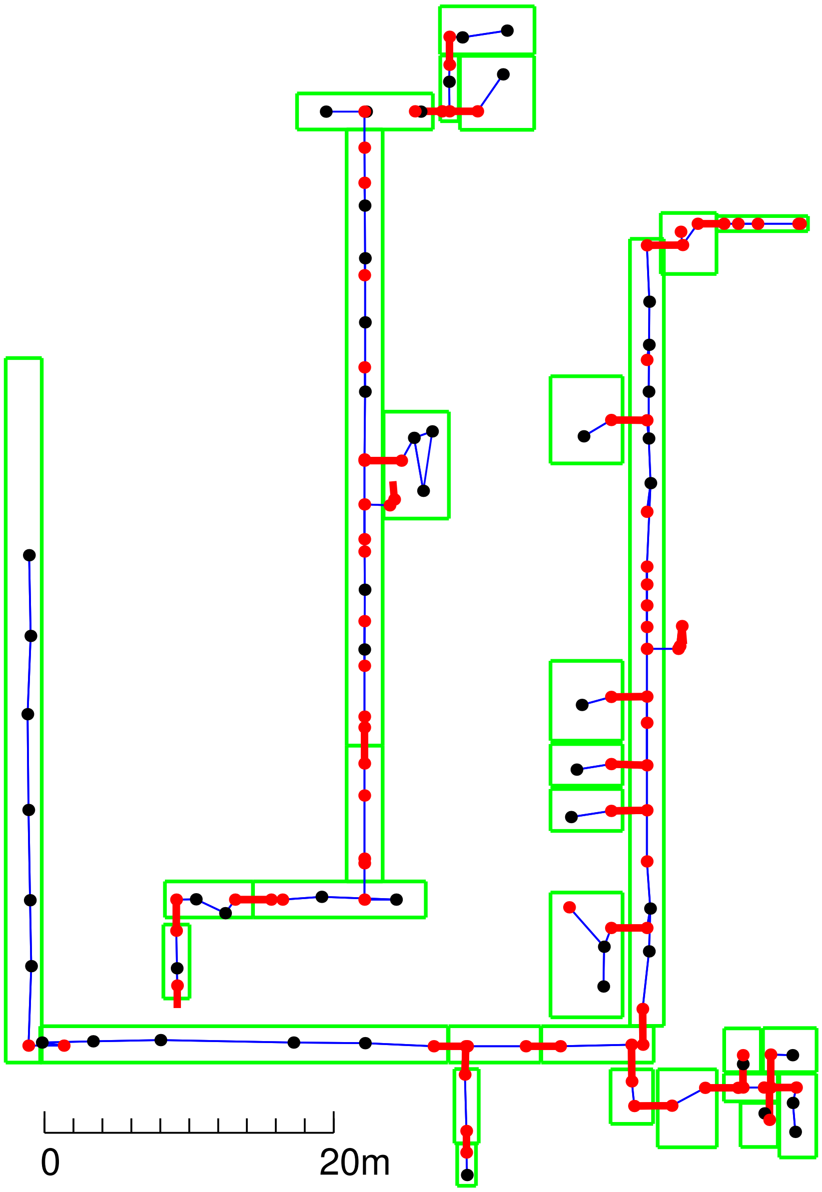


Figure 4.16: *The topological graph used for exploration. The nodes are marked as dots, dark(normal) and light(doors). The edges are marked as lines where the thick lines mark edges that go through a door.*

Failing to execute a motion command to go along an edge is used as evidence that the corresponding hypothesis is incorrect and the probability of the hypothesis is reduced by a factor $\Pr(\text{move fail}|H^{(i)})$. The goal points on which the graph is based are designed so that the robot is able to go between them. As Figure 4.16 shows, most of the edges are in corridors which by definition must be passable. The penalty for a motion command failure is thus set high, $\Pr(\text{move fail}|H^{(i)}) = 0.2$.

4.4 Experiments

MHL is tested on two different robots in two different environments. One is at CAS and the other is DaimlerChrysler Research and Technology in Berlin. The results presented below are from CAS. For the result from DaimlerChrylser please refer to (Jensfelt & Kristensen 2001).

4.4.1 Experimental Design and Execution

To evaluate the performance of any global localization scheme it is important to use many different initial conditions. If testing is performed in a small area, the results do not necessarily generalize. Good results are just as likely to be due to tuning as a good algorithm. Therefore the robot must be started from different initial configurations and in as many different areas as possible. Here, 20 different start positions throughout the environment are used to evaluate MHL.

To get a measure of the performance the pose tracking algorithm described in Chapter 3 is used to get the ground truth during the experiments. The experiments are terminated when the probability of the strongest hypothesis is above a certain threshold,

$$\max_{i=1,\dots,N_k} \pi_k^{(i)} \geq 0.9. \tag{4.48}$$

Figure 4.17 shows the trajectories followed by the robot during the active exploration. Some trajectories are omitted to keep the plot intelligible. The start and end points are labelled with ‘S’ and an ‘E’, respectively. The numbers by the arrows indicate the trajectory numbers, used later for easy referencing.

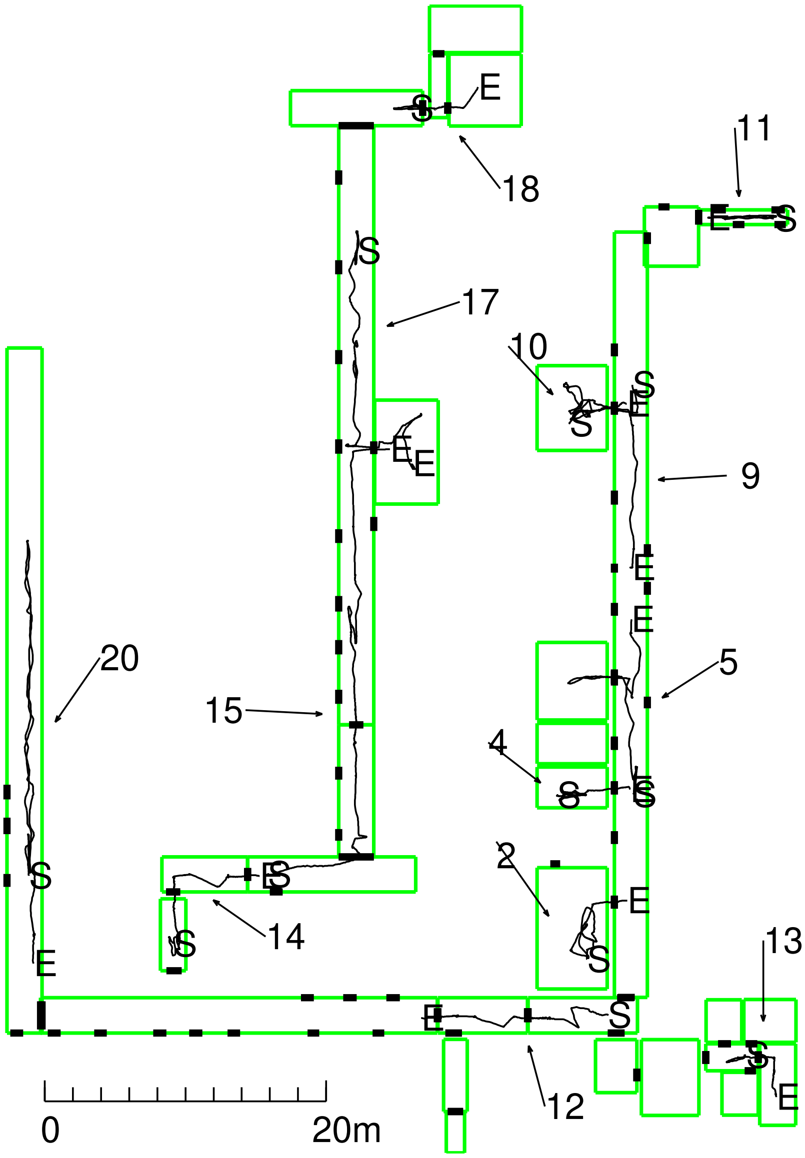


Figure 4.17: *The true trajectories of the robot for the experiments. Some trajectories are omitted to keep the plot intelligible. The start and end points are labelled with 'S' and an 'E', respectively.*

Run	Δx [m]	ΔD [m]	time [s]	M_{pair}	M_{line}	M_{door}
1	3.79	7.84	129.5	7(9)	13(13)	2(2)
2	4.88	15.66	238.1	11(20)	8(8)	3(3)
3	10.40	15.99	244.0	24(40)	13(14)	3(4)
4	4.96	9.62	354.4	5(18)	8(8)	2(5)
5	12.05	22.25	293.6	11(34)	19(19)	4(4)
6	6.44	20.01	409.0	6(32)	11(12)	3(3)
7	4.60	29.56	395.9	19(38)	24(24)	2(2)
8	5.21	6.74	121.5	14(21)	12(13)	3(5)
9	12.93	22.22	280.5	19(40)	35(37)	4(5)
F 10	4.07	12.77	287.1	3(20)	12(14)	1(3)
11	4.68	13.14	274.1	7(21)	10(11)	2(3)
12	13.08	14.25	246.5	7(26)	15(21)	5(5)
13	3.61	7.19	183.6	9(18)	6(6)	3(3)
14	7.59	13.56	252.6	22(35)	6(11)	3(3)
15	31.42	42.78	536.1	19(53)	26(29)	3(7)
16	5.85	15.90	236.1	7(31)	11(14)	2(5)
17	15.83	30.74	473.1	13(58)	21(26)	3(3)
18	5.02	7.74	168.6	8(21)	8(10)	5(6)
19	4.73	10.47	224.6	3(18)	9(10)	3(3)
20	6.87	58.17	362.5	27(54)	77(79)	2(2)
avg.	8.4	18.8	285.6	12(30)	17(19)	3(4)

Table 4.3: *Results from experiments at CAS. An F in the first column denotes that the localization failed. Δx is the straight path distance between start and end point of the exploration, whereas ΔD is the total distance traveled. Runs 6, 7, and 20 illustrate that the difference between these two distance measure can be quite large. The fourth column gives the time from start until (4.48) is reached. The last three columns contain the number of features that are matched to the winning hypothesis versus the ones that are detected (in parentheses).*

4.4.2 Evaluation of Experiments

Table 4.3 summarizes the results from the experiments at CAS, where the robot localizes successfully in 19 of the 20 runs. The table provides various statistics from the different runs. The straight path distance between the start and the end point of the exploration is given by Δx, whereas ΔD is the total distance traveled. The time until (4.48) is fulfilled is given in the fourth column. A failure is indicated by a 'F' in the first column. The last three columns contain the number of features that were matched to the winning hypothesis versus the total number of detected features (in parentheses).

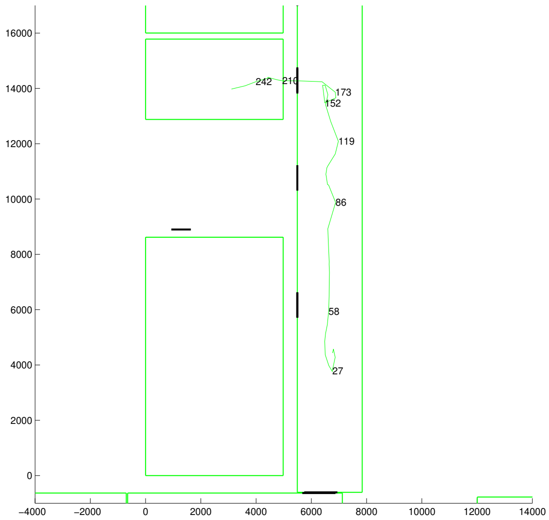


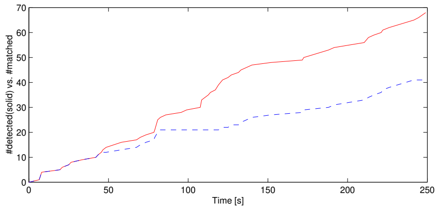
Figure 4.18: *The trajectory followed by the robot in run no. 3. The numbers along the trajectory give the time since the run started (seconds).*

To further analyze the experimental data some of the runs are discussed in more detail.

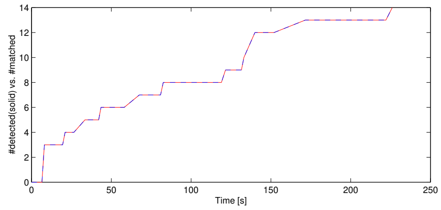
Run no. 3

In the third run the robot starts in the corridor. The complete trajectory is shown in Figure 4.18, where the numbers along the trajectory give the time since the experiment started. The first creative feature that is detected is the door leading to the living-room on the left of the starting point. The door, in combination with lines and pairs, results in the best hypothesis being the correct one when the second step of the exploration strategy starts. The robot starts by going to the closest node in the corridor and then follows the graph along the corridor and into an office.

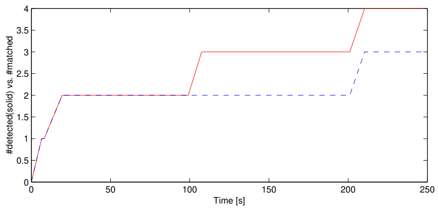
Figure 4.19 shows the matching statistics for the best hypothesis as a function of time. The solid curves are the number of detected features and the dashed curves are the number of features that are successfully matched to the best hypothesis. The weight of the best hypothesis as a function of time is shown in Figure 4.20 along with the number of hypotheses as a function of time. The decrease in weight after about 200 s is caused by a split of the best hypothesis upon seeing a door. Figures 4.19 and 4.20 are based on an off-line experiment performed at a later stage with the data collected during run no. 3 and do therefore not exactly comply with the statistics in Table 4.3.



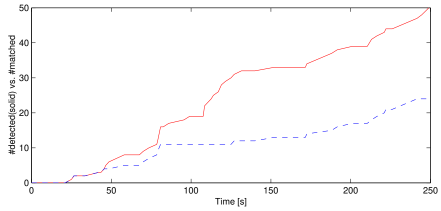
(a) all features



(b) line features



(c) door features



(d) pair features

Figure 4.19: *The matching statistics for the best hypothesis (estimated off-line) as a function of time since the start of the experiment. The solid curves denotes the total number of detected features and the dashed line is the number of those that was successfully matched to the best hypothesis.*

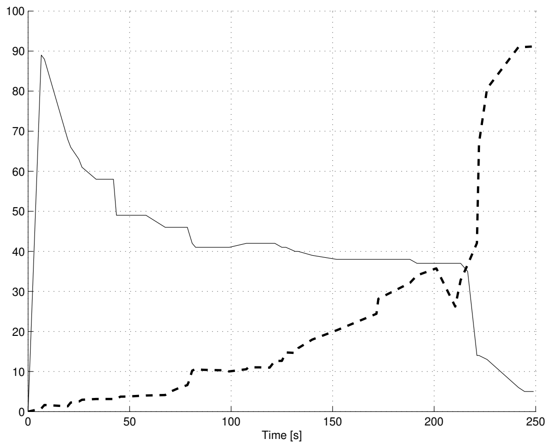


Figure 4.20: *The number of hypotheses (solid line) versus the weight (scaled by a 100) of the best hypothesis (dashed line), plotted as function of time since the experiment started.*

Run no. 4

In run no. 4 the robot was placed in a small office on the lower floor (see Figure 4.17), which is cluttered, leaving the robot limited room to maneuver, and thus making the initial exploration slow. To disambiguate the office from the office next to it, the robot must reach the corridor, which it does after 330 seconds. Once in the corridor and having detected the door to the other office the robot is localized after another 25 s (approx.).

Run no. 10

Run no. 10 starts in the isolated room on the upper region on the lower floor. This room is identical in size to the room a bit further down in the map. In this experiment the robot did not localize correctly. Instead of the room it is in, it reports being in the similar room further down. The winning hypothesis matched 3 out of 20 pairs, 12 out of 14 lines and 1 out of 3 doors. The matching ratio for the true hypothesis was 0 pairs out of 20, 6 lines out of 14 and 2 doors out of 3. The reason for the bad matching result is that this room has been completely restructured since the map was constructed. The furniture has been rearranged completely which effects the point landmark pairs. A wall previously completely covered with cupboards is now uncovered, resulting in one of the sides in the rectangular line model not matching anymore.

Run no. 15

In run no. 15 the robot starts on the lower end in the map on the upper floor and traveled upwards. The robot is not able to detect any of the (at the time) closed doors on the left hand side of the corridor. It is not until the robot reaches the kitchen in the middle of the corridor that the true pose can be singled out. The total time until successfully localized is 536 seconds, i.e. nearly ten minutes. This is a result of the exploration strategy switching between different best hypotheses in the corridor. Waiting for a motion command to fail is a time consuming way to rule out a hypothesis.

Run no. 20

In the last experiment, run no. 20, the robot is placed in the almost feature-less left hand side corridor on the lower floor, where it is only able to detect one of the doors in this corridor as the others are closed. When the second phase of the exploration starts the robot travels up the corridor. The same map line features are observed from all nodes up to the end of the corridor. As the utility of the nodes is not updated based on which features are detected, the robot continues up the corridor expecting to detect new features. It is not until reaching the last node that the robot travels back. When coming to a region where new features can be seen the robot is localized. The 10 strongest hypotheses all matched the same number of lines and doors, highlighting the high degree of symmetry. However, the point landmark pairs eventually singled out the true hypothesis. Had the robot initially traveled down towards the other corridor, it would have localized faster and with less ambiguity.

Summary of Experiments

To summarize the experiments presented in this chapter and the ones presented in (Jensfelt & Kristensen 2001) from DaimlerChrysler, successful localization is achieved in 19 out of 20 runs and 18 out of 20 runs respectively. When the localization fails or is inefficient it can be traced back to one of the following properties of the current implementation

- No hypotheses are generated and thus localization is not possible unless a modeled creative feature is detected (compare run no. 3 at DaimlerChrysler in (Jensfelt & Kristensen 2001)).
- The greedy-one step exploration strategy is sometimes unable to present the robot with new features. This is in part caused by not updating the utility values of the different nodes to account for features that are already detected. A clear example of this is run no. 20 in this chapter.
- If the map differs significantly from the environment it models, localization might also fail as illustrated by run no. 10 above. However, this is true for any localization algorithm and not a specific property of MHL.

4.5 Summary

This chapter has presented the first thorough investigation of how Multi Hypothesis Tracking techniques can be applied to the problem of mobile robot localization. Using a Gaussian sum approximation of the pose distribution allows for the use of the extended Kalman filter, providing a computationally efficient way of fusing information. It also allows for high accuracy without having to trade it for high memory consumption. Bayesian probability theory is used for evidence fusion, which permits explicit modeling of feature detector and map uncertainty. The result is the so-called Multiple Hypothesis Localization (MHL) scheme.

Results from extensive tests in real world experiments in two different environments and with two different platforms show that MHL work well. This is underlined by 19 out of 20 successful localizations and 18 out of 20, respectively. The cause of the failures are identified, and are explained by the simple exploration strategy, only initiating hypothesis when detecting doors and an outdated map in some areas.

Although the results are very good, it is only a first attempt at using Multiple Hypothesis techniques for large scale mobile robot localization. Future research include investigating more advances exploration strategies and more elaborate data association schemes, see e.g. (Uhlmann 1995). Further, different pruning and merging schemes in combination with other hypotheses splitting procedures are also interesting for investigation.

Chapter 5

Monte Carlo Localization

The Bayesian filtering problem, i.e. finding $p(\mathbf{x}_k|\mathbf{z}_0,\ldots,\mathbf{z}_k)$ is in general a non-linear filtering problem. Analytic approximations are presented in Chapters 3 and 4, where the PDF is represented by a single Gaussian or a sum thereof respectively. Although the results are good, it is inherently true that the success rests on how well the true PDF can be approximated. The single Gaussian is shown to be good for tracking applications but is insufficient as soon as the PDF is multi-modal or has a wide tail. A sum of Gaussians can handle larger uncertainties but still rests on a Gaussian assumption and suffers from an exponential growth in the number of Gaussian terms needed. Discretizing the state space into a mesh suffers from an extreme computational burden. In this light it is natural to look for yet another way to represent the PDF. Somewhere between the Gaussian sum approximation and the mesh approach the Monte Carlo methods are found. Here the idea is that instead of an analytic approximation of the PDF or a discretization of the whole state space a set of samples is kept. Under the right conditions these samples can then represent any given PDF assuming that enough samples are used. The theory behind these methods is as old as the Kalman filter.

The outline of the chapter is as follows. Section 5.1 gives a rather detailed description of the the theory behind the Monte Carlo methods to underline some issues that must be considered in a real-time implementation. In Section 5.2, a Monte Carlo method applied to mobile robot localization is presented along with two suggested approaches for improvements. Details on how these are implemented in the special case of feature based global localization is given in Section 5.3. Extensive experiments showing a significant improvement using the two suggested approaches are presented in Section 5.4 accompanied with a comparison between using different combinations of features. Finally a summary and discussion is given.

5.1 Theory

Before describing the theory behind the Monte Carlo methods, the global localization problem is repeated to make the discussion more concrete.

Problem Statement Given a set of measurements $Z_k = \{\mathbf{z}_0, \ldots, \mathbf{z}_k\}$ acquired at time steps $0, 1, \ldots, k$ estimate the state $\mathbf{x}_k$. Here is the pose of the mobile robot, i.e. $\mathbf{x}_k = (x_k, y_k, \theta_k)$. The evolution of the system can be described by

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \tag{5.1}$$

where $\mathbf{u}_k$ is odometric data and $\mathbf{w}_k$ is the process noise and accounts for the approximation error in $\mathbf{f}$. Measurements are generated by

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathcal{M}, \mathbf{v}_k) \tag{5.2}$$

where $\mathcal{M}$ is the map and $\mathbf{v}_k$ is the measurement noise. Estimation of $\mathbf{x}_k$ is achieved by estimating the PDF $p(\mathbf{x}_k|Z_k)$ from which any function of $\mathbf{x}_k$ conditioned on the measurements can be calculated.

Using Bayes' theorem, $p(\mathbf{x}_k|Z_k)$ can be rewritten as (Handschin & Mayne 1969)

$$p(\mathbf{x}_k|Z_k) = \frac{\int p(\mathbf{x}_{k-1}|Z_{k-1})p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{z}_k|\mathbf{x}_k)d\mathbf{x}_{k-1}}{\int\int p(\mathbf{x}_{k-1}|Z_{k-1})p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{z}_k|\mathbf{x}_k)d\mathbf{x}_{k-1}d\mathbf{x}_k} \tag{5.3}$$

Expanding this further by recursive use of Bayes' theorem yields a k-dimensional expression of integrals. Monte Carlo methods, or simulation based methods as they are also known as, are used for evaluating the high dimensional expression numerically in an efficient way by recursive propagation of the PDF in the form of a sample set. Let $s_k^{(i)} = (\mathbf{x}_k^{(i)}, \pi_k^{(i)})$ denote the i:th sample with $\mathbf{x}_k^{(i)}$ denoting the corresponding robot pose and $\pi_k^{(i)}$ the weight of the sample. The sample set at time k is thus given by

$$S_k = \{s_k^{(1)}, \ldots, s_k^{(N_k)}\}, \tag{5.4}$$

where N_k is the number of samples. Note that the sample set size, in general, is time dependent.

5.1.1 Evaluating Function Expectations

It is often not the density function itself that is of interest but rather the expected value of some function. If $p(\mathbf{x}_k|Z_k)$ is sampled with equally *weighted samples* an approximation of the expected value for any function $r(\mathbf{x}_k)$ conditioned on the measurements Z_k can be calculated as

$$E_{p(\cdot|Z_k)}[r(\mathbf{x}_k)] \approx \frac{1}{N_k}\sum_{i=1}^{N_k} r(\mathbf{x}_k^{(i)}). \tag{5.5}$$

The strong law of large numbers gives that the approximation can be made arbitrarily good in the limit $N_k \to \infty$ (Doucet 1998). An estimate of the robot pose can be acquired by letting $r(\mathbf{x}_k) = \mathbf{x}_k$. That this might not always be the best way to estimate the pose is discussed later in the chapter (see Section 5.3.5).

5.1.2 Bayesian Importance Sampling

As it is hard in general to sample from any function and in particular from an unknown function such as $p(\mathbf{x}_k|Z_k)$, other methods must be used to generate a sample set that represents the density function $p(\mathbf{x}_k|Z_k)$. Importance sampling (IS) is such a method. The simple idea behind the method is best illustrated by rewriting the expression for the expectation value of some function $r(\mathbf{x}_k)$,

$$\begin{aligned} E_{p(\cdot|Z_k)}\left[r(\mathbf{x}_k)\right] &= \int r(\mathbf{x}_k) p(\mathbf{x}_k|Z_k) d\mathbf{x}_k \\ &= \int r(\mathbf{x}_k) \frac{p(\mathbf{x}_k|Z_k)}{q(\mathbf{x}_k|Z_k)} q(\mathbf{x}_k|Z_k) d\mathbf{x}_k. \end{aligned}$$

Introducing $\pi_k(\mathbf{x}_k) = \frac{p(\mathbf{x}_k|Z_k)}{q(\mathbf{x}_k|Z_k)}$ yields

$$E_{p(\cdot|Z_k)}\left[r(\mathbf{x}_k)\right] = \int r(\mathbf{x}_k)\pi_k(\mathbf{x}_k) q(\mathbf{x}_k|Z_k) d\mathbf{x}_k \tag{5.6}$$

$$= E_{q(\cdot|Z_k)}\left[r(\mathbf{x}_k)\pi_k(\mathbf{x}_k)\right] \tag{5.7}$$

where it is assumed that $q > 0$ if $p > 0$. The evaluation of the expected value with respect to p has thus shifted to evaluating the expected value with respect to q weighted with π. If the function $q(\mathbf{x}_k|Z_k)$ can be sampled and the samples are given weights $\pi_k^{(i)} = \frac{p(\mathbf{x}_k^{(i)}|Z_k)}{q(\mathbf{x}_k^{(i)}|Z_k)}$, the weighted sample set

$$S_k = \{(\mathbf{x}_k^{(i)}, \pi_k^{(i)}) \mid i = 1, \ldots, N_k\} \tag{5.8}$$

is an approximation of $p(\mathbf{x}_k|Z_k)$. By letting $N_k \to \infty$ the density approximation can be made arbitrarily good and the expectation value can be expressed as

$$E_{p(\cdot|Z_k)}\left[r(\mathbf{x}_k)\right] \approx \frac{1}{N}\sum_{i=1}^{N} r(\mathbf{x}_k^{(i)})\pi_k^{(i)}. \tag{5.9}$$

The distribution q is called the *importance function* (Doucet 1998) or the *sampler density* (Liu & Chen 1998).

For this method to work in the current form it must be possible to calculate $\pi_k(\mathbf{x}_k)$ and hence evaluate $p(\mathbf{x}_k|Z_k)$. The distribution $p(\mathbf{x}_k|Z_k)$ is unknown but it can be shown (Doucet 1998) that $E_{p(\cdot|Z_k)}[r(\mathbf{x}_k)]$ can be approximated by

$$E_{p(\cdot|Z_k)}\left[r(\mathbf{x}_k)\right] \approx \frac{\sum_{i=1}^{N_k} r(\mathbf{x}_k^{(i)})\pi_k^{(i)}}{\sum_{i=1}^{N_k} \pi_k^{(i)}} \tag{5.10}$$

where $\pi_k^{(i)}(\mathbf{x}_k)$ is redefined as

$$\pi_k^{(i)}(\mathbf{x}_k^{(i)}) = \frac{p(Z_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)})}{q(\mathbf{x}_k^{(i)}|Z_k)}. \tag{5.11}$$

By normalizing the weights according to

$$\bar{\pi}_k^{(i)} := \frac{\pi_k^{(i)}}{\sum_{i=1}^{N_k} \pi_k^{(i)}}. \tag{5.12}$$

the expectation value can now be expressed as

$$E_{p(\cdot|Z_k)}\left[r(\mathbf{x}_k)\right] \approx \sum_{i=1}^{N_k} r(\mathbf{x}_k^{(i)})\bar{\pi}_k^{(i)}. \tag{5.13}$$

In (Liu & Chen 1998) a sample set $\{(\mathbf{x}_k^{(i)}, \pi_k^{(i)}) \mid i = 1, \ldots, N_k\}$ drawn from a distribution q is said to be *properly weighted* with respect to the the distribution p if for any integrable function r

$$\lim_{N_k \to \infty} \frac{\sum_{i=1}^{N_k} r(\mathbf{x}_k^{(i)})\pi_k^{(i)}}{\sum_{i=1}^{N_k} \pi_k^{(i)}} = E_{p(\cdot)}\left[r(\mathbf{x}_k)\right]. \tag{5.14}$$

5.1.3 Sequential Importance Sampling (SIS)

For an online implementation it is important to be able to calculate the weights recursively. Assuming that the importance function can be decomposed as

$$q(\mathbf{x}_k|Z_k) = q(\mathbf{x}_0|Z_0) \prod_{l=1}^{k} q(\mathbf{x}_l|\mathbf{x}_{l-1}, Z_l) \tag{5.15}$$

the weights can be recursively calculated as (Krishnamurthy 2000)

$$\pi_k^{(i)} = \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, Z_k)}\pi_{k-1}^{(i)}. \tag{5.16}$$

This method is referred to as Sequential Importance Sampling (SIS) and was proposed already in the 1960's (Handschin & Mayne 1969, Handschin 1970). The initial sample set is created by drawing from the prior $p(\mathbf{x}_0)$. This function encodes all *a priori* knowledge about the state. In a global localization scenario the initial sample set is given by sampling the pose space uniformly.

When computing the weight $\pi_k^{(i)}$, $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^{(i)})$ can be calculated from the system equation (5.1) as (Gordon et al. 1993)

$$p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}) = \int p(\mathbf{w}_{k-1})\delta(\mathbf{x}_k^{(i)} - \mathbf{f}(\mathbf{x}_{k-1}^{(i)}, \mathbf{u}_k, \mathbf{w}_{k-1}))d\mathbf{w}_{k-1}. \tag{5.17}$$

This is a prediction of how the samples moves based on the odometric information and the known characteristics of the system noise $\mathbf{w}_k$. The function $p(\mathbf{z}_k|\mathbf{x}_k^{(i)})$ in (5.16) gives the probability of making observation $\mathbf{z}_k$ from the pose $\mathbf{x}_k^{(i)}$ and can be calculated from (5.2) as (Gordon et al. 1993)

$$p(\mathbf{z}_k|\mathbf{x}_k^{(i)}) = \int p(\mathbf{v}_k)\delta(\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k))d\mathbf{v}_k. \tag{5.18}$$

5.1.4 Selection of Importance Function

Taking a step back, the aim is to be able to sample from the unknown distribution $p(\mathbf{x}_k|Z_k)$. The problem has now been transfered to being able to sample from some other distribution $q(\mathbf{x}_k|Z_k)$. Many different suggestions have been made for q in the literature. The simplest choice is to make it a constant, only dependent on the state and not the observations, i.e. $q(\mathbf{x}_k^{(i)}|Z_k) = q(\mathbf{x}_k^{(i)})$. The downside of this choice is that the information at hand in the form of (5.1) and (5.2) is not used. When deciding on an importance function it is worth keeping in mind that high-dimensional distributions often have the probability mass concentrated to a small region (Mackay 1996). If the localization procedure is successful that region should correspond to the region close to the true robot pose. With a fixed importance function this fact cannot be exploited. The region where the probability mass is concentrated is called the *typical set*, T, of the distribution. In (Handschin & Mayne 1969, Handschin 1970) a more intuitive importance function is chosen, namely

$$q(\mathbf{x}_k^{(i)}|Z_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{5.19}$$

With this choice of importance function (5.16) simplifies to

$$\pi_k^{(i)} = \pi_{k-1}^{(i)} p(\mathbf{z}_k|\mathbf{x}_k^{(i)}). \tag{5.20}$$

Now the system model is used in the design of the importance function, but not the observations.

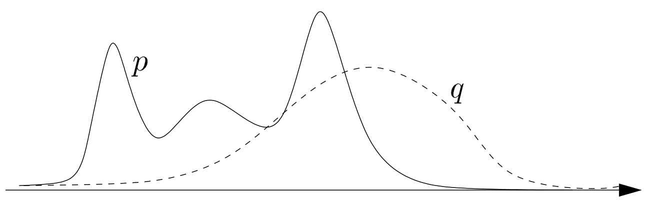


Figure 5.1: *When the importance function approximates the true PDF well, fewer samples are needed for a good approximation.*

5.1.5 Sampling/Importance Resampling (SIR)

Using Sequential Importance Sampling in practice often leads to a huge range in the magnitude of the weights. Samples with low weight contributes very little to the approximation of $p(\mathbf{x}_k|Z_k)$. The closer the importance function is to $p(\mathbf{x}_k|Z_k)$, the smaller the variations are between the weights and the fewer samples, N_k, are needed to make the sample set a good approximation of the distribution $p(\mathbf{x}_k|Z_k)$ (Smith & Gelfand 1992). Figure 5.1 illustrates the problem. To the left where p is large q is small. The samples are drawn from q and thus many samples are needed to get good sample support where q is small. On the right side, q is large and many samples are generated, but p is small and the weights are low. The function p can still be approximated arbitrarily well but the number of samples that is needed is large. For reasons of efficiency it is thus of interest to redistribute the resources, i.e. the samples. To this end an intermediate step where the sample set is resampled is introduced (Rubin 1987). The idea is that sample support should be shifted from regions where the sample weights are low to regions where the weights are higher. Resampling increases the chance for good samples to amplify themselves and in doing so give more weight to the importance function in that region and provide better future estimates. Resampling is achieved by drawing samples from the sample set and letting the probability of drawing a particular sample be proportional to its weight. Let $\{(\tilde{\mathbf{x}}_k^{(i)}, \tilde{\pi}_k^{(i)}) \mid i = 1, \ldots, N_k\}$ denote the sample set after the normal importance sampling step and normalization of the weights. The resampling step generates a new sample set $\{(\mathbf{x}_k^{(i)}, \pi_k^{(i)}) \mid i = 1, \ldots, N_k\}$ by drawing samples from the old set such that

$$p(\mathbf{x}_k^{(j)} = \tilde{\mathbf{x}}_k^{(i)}) = \tilde{\pi}_k^{(i)} \quad \text{for any} \quad i, j \tag{5.21}$$

This method is presented in (Smith & Gelfand 1992, Gordon et al. 1993) under the name Bootstrap filter, (Liu & Chen 1998) call it SIS with resampling (SISR) and (Rubin 1987) refer to it as Sampling/Importance Resampling (SIR).

5.1.6 Discussion

The available resources on a mobile robot, and the requirement of online calculations, put a limit on how many samples that can be used. This means that the samples that are available must be used in the best possible way. When using SIR the samples typically cluster in regions corresponding to poses that best support the observations. However, unless the importance function is a good approximation of $p(\mathbf{x}_k|Z_k)$, there might be very few samples in the typical set T of $p(\mathbf{x}_k|Z_k)$. Samples in regions where $p(\mathbf{x}_k|Z_k)$ is small has little effect of on the PDF approximation. With a finite number of samples and an importance function with low support in T the algorithm might collapse. In (Gordon et al. 1993) it is said "*Through this process, the representation of the PDF may become most inadequate within a few steps. Indeed if there is no* *system noise, all the N samples may rapidly collapse to a single value.*" Two methods to overcome the problem are suggested in (Gordon et al. 1993). One is called *roughening* and the other *prior editing*. Roughening adds an independent jitter to each sample after the prediction step. This is similar to *simulated annealing* discussed in e.g. (Neal 1993, Mackay 1996). The word annealing comes from the process of first heating and then cooling down metal or glass to make it less brittle. Heating the metal gives the molecules more energy to move and find "their right place" in the material. In simulated annealing the heating corresponds to increasing the jitter so that the samples are allowed to adjust and provide a better estimate of the PDF. With time the temperature is lowered, i.e. the jitter is decreased.

Getting back to the methods proposed by (Gordon et al. 1993), prior editing adds an acceptance test for the samples being resampled. By evaluating (5.2) at the pose corresponding to a sample a prediction is given for the measurement. If the measurement is too far from the prediction the sample is rejected and another one is drawn. Nothing is said about how to handle outliers. In the case where only a few samples fulfill the test all samples are shifted to the corresponding poses. If there are no samples that fulfill the acceptance test the algorithm apparently deadlocks. In either case the estimation is destroyed. A better alternative is to reject the samples with some probability that depends on the reliability of the measurement. This avoids the deadlocks and leave room for accounting for the possibility of an outlier.

The success of all the methods described so far rests on the assumption that the initial pose distribution, $p(\mathbf{x}_0)$, is known and can be well represented by the initial sample set. When $N_k \to \infty$ this is not a problem, but in every real situation $N_k < \infty$. It is in most cases reasonable to assume that the environment surrounding the robot is bounded. Still, in a large environment the initial sample set is sparse and there is no guarantee that there is a sample at the correct pose. To guarantee sample support everywhere $N_k \to \infty$ is needed which is only the dream of a mathematician and not the reality of a roboticist.

There is no such thing as a perfect feature detector and unique features do not grow on trees either. When starting from a uniform prior, several iterations are needed before the probability mass is shifted to the typical set when using a realistic sensor model. The approximation is thus vulnerable to outliers during the first iterations. If the first observation is an outlier the samples close to the true pose are likely to have low weights since they do not match the erroneous observation. Chances are then high that they are removed from the sample set. This in turn means that unless many samples are used, a global localization algorithm based on the SIR method might not be successful. Two methods for improving the sample support are presented in the next section.

5.2 Algorithm

In this section the SIR algorithm applied to mobile robot localization is described in more detail. Two ways of improving the performance by modifying the standard SIR algorithm are also suggested. All algorithms presented below are initialized in the same way (see Figure 5.2). Any knowledge of the initial pose of the robot is encoded in $p(\mathbf{x}_0)$ which is assumed known.

Initialization

`Draw` N_0 `samples from the known` $p(\mathbf{x}_0)$

`Give the samples equal weights` $\pi_0^{(i)} = \frac{1}{N_0}$

$S_0 = \left\{(\mathbf{x}_0^{(i)}, \pi_0^{(i)}) | i = 1, \ldots, N_0\right\}$ `approximates` $p(\mathbf{x}_0)$

Figure 5.2: *Initialization of Monte Carlo Localization algorithm*

5.2.1 Monte Carlo Localization

In robotics the SIR algorithm was first applied in vision based tracking (Isard & Blake 1996, Isard & Blake 1998) under the name CONDENSATION. The first application of SIR on mobile robot localization is presented in (Dellaert, Fox, Burgard & Thrun 1999, Fox, Burgard, Dellaert & Thrun 1999) where it is called Monte Carlo Localization (MCL). The latter name is used in the thesis as it gives a fair connection to the origin of the method.

Monte Carlo Localization can be described by a three step algorithm (see Figure 5.3). The first two steps correspond to the two steps given by Figure 2.8 and the last one is the resampling step in which the samples are redistributed to reduce the variance in the sample weights.

5.2.2 Suggestion 1: Random Sampling

Using $N_k < \infty$ leads to a situation where there might not be any samples close to the true pose. The outcome of MCL is thus dependent on the initial sample set. Samples can only be generated from other samples. Small perturbations are induced in the sample set by the noise term $\mathbf{w}_k$ in (5.1) used in the prediction step. These perturbations help to catch small discrepancies between the estimated and true state given a good initial estimate. However, they are not in general able to populate regions in the pose space which lack sample support. The possibility that the initial sample set does not populate the typical set T of the true PDF must thus be taken into account. If T is not populated, the sample set needs to be reinitialized as it does not contain any information about the true state. Re-initialization corresponds to setting the

Algorithm: MCL

Prediction Step

`For each` $i = 1, \ldots, N_{k-1}$:

  `Draw` $\bar{\mathbf{x}}_k^{(i)}$ `from (5.17)`

  $\bar{\pi}_k^{(i)} = \frac{1}{N_{k-1}}$

$\bar{S}_k = \left\{ (\bar{\mathbf{x}}_k^{(i)}, \bar{\pi}_k^{(i)}) \mid i = 1, \ldots, N_k \right\}$ `approximates` $p(\mathbf{x}_k | Z_{k-1})$

Measurement Update Step

`For each` $i = 1, \ldots, N_{k-1}$:

  $\tilde{\mathbf{x}}_k^{(i)} = \bar{\mathbf{x}}_k^{(i)}$

  $\tilde{\pi}_k^{(i)} = \bar{\pi}_k^{(i)} p(\mathbf{z}_k | \tilde{\mathbf{x}}_k^{(i)})$ `(5.16)`

`Normalize all weights through division by` $\sum_{i=0}^{N_{k-1}} \tilde{\pi}_k^{(i)}$

$\tilde{S}_k = \left\{ (\tilde{\mathbf{x}}_k^{(i)}, \tilde{\pi}_k^{(i)}) \mid i = 1, \ldots, N_k \right\}$ `approximates` $p(\mathbf{x}_k | Z_k)$

Resampling Step

`For each` $i = 1, \ldots, N_k$:

  `Draw` j `from` $\{1, \ldots, N_{k-1}\}$ `such that` $\Pr(j = l) = \tilde{\pi}_k^{(l)}$

  $\mathbf{x}_k^{(i)} = \tilde{\mathbf{x}}_k^{(j)}$

  $\pi_k^{(i)} = \frac{1}{N_k}$

$S_k = \left\{ (\mathbf{x}_k^{(i)}, \pi_k^{(i)}) \mid i = 1, \ldots, N_k \right\}$ `also approximates` $p(\mathbf{x}_k | Z_k)$

Figure 5.3: *The Monte Carlo Localization algorithm. The sample sets $\tilde{S}_k$ and S_k approximate the same distribution, $p(\mathbf{x}_k|Z_k)$, but with different sample poses.*

prior $p(\mathbf{x}_{k-1})$ to be a uniform distribution, i.e. every state is just as likely. If $p(\mathbf{x}_{k-1})$ is uniform then so is the importance function, $p(\mathbf{x}_k|\mathbf{x}_{k-1})$.

Algorithm: MCL with Random Sampling

Prediction Step

For each $i = 1, \dots, N^r_{k-1}$:

Draw $\bar{\mathbf{x}}^{(i)}_k$ from a uniform distribution

For each $i = (N^r_{k-1} + 1), \dots, N_{k-1}$:

Draw $\mathbf{w}^{(i)}_k$ from $p(\mathbf{w}_k)$ and let $\bar{\mathbf{x}}^{(i)}_k = \mathbf{f}(\mathbf{x}^{(i)}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}^{(i)}_{k-1})$

$\bar{\pi}^{(i)}_k = \frac{1}{N_k} \;\forall\; i = 1, \dots, N_{k-1}$

Measurement Update Step

Same as in Figure 5.3

Resampling Step

Same as in Figure 5.3

Figure 5.4: *Adding samples from a uniform distribution.*

Introduce r_r as the ratio of samples that are drawn from a uniform prior instead of the standard importance function, where r_r can be viewed as the probability that the typical set is not populated and re-initialization is needed. The MCL algorithm with random sampling is shown in Figure 5.4.

The idea of sampling from a uniform distribution can also be found in (Fox, Burgard, Dellaert & Thrun 1999, Jensfelt, Wijk, Austin & Andersson 2000, Lenser & Veloso 2000).

5.2.3 Suggestion 2: Planned Sampling

Drawing samples from a uniform distribution is a rather ineffective way of exploring the state space. Unless $r_r N$ is large, the probability is still small to draw a sample inside the typical set of the true PDF. The importance function (5.19) does not depend on the last observations. Assuming the measurement $\mathbf{z}_k$ has a correspondence in the environmental model, $p(\mathbf{x}_k|\mathbf{z}_k)$ has support in T. With this quality it could be used to sample from, i.e. be used as an importance function,

$$q(\mathbf{x}^{(i)}_k|Z_k) = p(\mathbf{x}_k|\mathbf{z}_k). \tag{5.22}$$

Using (5.22) as importance function corresponds to creating new hypotheses from $p(\mathbf{x}_k|\mathbf{z}_k)$ in Chapter 4. This approach is here called *planned sampling* as samples are generated in a planned way.

Due to imperfect sensors and an imperfect environmental model, $p(\mathbf{x}_k|\mathbf{z}_k)$ might lack support in some regions just like (5.19). Combining the two importance functions is likely to give better total support and thus in the end a better approximation of $p(\mathbf{x}_k|Z_k)$. Let r_p denote the ratio of samples to draw from the alternative importance sampling function $p(\mathbf{x}_k|\mathbf{z}_k)$. When calculating the

Algorithm: MCL with Planned Sampling

Prediction Step

For each $i = 1, \ldots, N^p_{k-1}$:

Draw $\bar{\mathbf{x}}_k^{(i)}$ from $p(\mathbf{x}_k|\mathbf{z}_k)$

$\bar{\pi}_k^{(i)} = \alpha \frac{1}{N_k}$ where α is defined by (5.23)

For each $i = (N^p_{k-1}+1), \ldots, N_{k-1}$:

Draw $\mathbf{w}_k^{(i)}$ from $p(\mathbf{w}_k)$ and let $\bar{\mathbf{x}}_k^{(i)} = \mathbf{f}(\mathbf{x}_{k-1}^{(i)}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}^{(i)})$

$\bar{\pi}_k^{(i)} = \frac{1}{N_k}$

Measurement Update Step

Same as in Figure 5.3

Resampling Step

Same as in Figure 5.3

Figure 5.5: *Algorithm for MCL with planned sampling*

weights of the samples with this importance function (5.16) does not simplify to (5.20). To get a properly weighted sample set, the full form of (5.16) must be used, i.e.

$$\pi_k^{(i)} = \underbrace{\frac{p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{p(\mathbf{x}_k^{(i)}|\mathbf{z}_k)}}_{\alpha} p(\mathbf{z}_k|\mathbf{x}_k^{(i)})\pi_{k-1}^{(i)} = \alpha p(\mathbf{z}_k|\mathbf{x}_k^{(i)})\pi_{k-1}^{(i)}. \tag{5.23}$$

The approach suggested in this section (previously published in (Jensfelt, Austin, Wijk & Andersson 2000, Jensfelt, Wijk, Austin & Andersson 2000)) is similar to the work recently published in (Lenser & Veloso 2000), but was developed independently of these results. In (Thrun, Fox & Burgard 2000) different methods for obtaining a properly weighted sample set are further investigated.

5.3 Implementation

So far in this chapter the measurements have been kept general, but henceforth the measurements are assumed to be features. Several features can typically be detected from a single sensor scan. To make the calculations more tractable it is assumed that even if multiple features are detected from the same scan they are independent, and also that the features in the measurement vector, $\mathbf{z}_k$, are sorted in ascending order with respect to time.

Implementation: MCL

Prediction and Measurement Update Step

```
    x̃_k^(i) := x_{k-1}^(i)
    π̃_k^(i) := π_{k-1}^(i)
    For each feature measurement z_k^l:
        For each i = 1,...,N_{k-1}:
            x̃_k^(i) := f(x̃_k^(i), u_{k-1,l}, w),    w ∈ p(w_{k-1})
            π̃_k^(i) := π̃_k^(i) p(z_k^l | x̃_k^(i))
    Normalize all weights through division by Σ_{i=0}^{N_{k-1}} π̃_k^(i)
```

$\tilde{\mathbf{x}}_k^{(i)} := \mathbf{x}_{k-1}^{(i)}$

$\tilde{\pi}_k^{(i)} := \pi_{k-1}^{(i)}$

For each feature measurement $\mathbf{z}_k^l$:

For each $i = 1, \ldots, N_{k-1}$:

$\tilde{\mathbf{x}}_k^{(i)} := \mathbf{f}(\tilde{\mathbf{x}}_k^{(i)}, \mathbf{u}_{k-1,l}, \mathbf{w}), \quad \mathbf{w} \in p(\mathbf{w}_{k-1})$

$\tilde{\pi}_k^{(i)} := \tilde{\pi}_k^{(i)} p(\mathbf{z}_k^l | \tilde{\mathbf{x}}_k^{(i)})$

Normalize all weights through division by $\sum_{i=0}^{N_{k-1}} \tilde{\pi}_k^{(i)}$

Resampling Step

Same as in Figure 5.3

Figure 5.6: *Implementation of the MCL algorithm for feature based localization.*

5.3.1 Monte Carlo Localization

The first two steps of the MCL algorithm are iterated over the set of extracted features $\mathbf{z}_k$. The resampling step is performed once at the end. The reasoning behind this is that performing the resampling step after individual measurements makes the overall algorithm more sensitive to outliers. As previously discussed a false measurement is likely to give samples close to the true state low weights and hence make the chances of them surviving the resampling step smaller. Fusing several measurements makes the system less sensitive, as correct measurements compensates for possible outliers. The MHL technique from the previous chapter has an effective way of representing wide spread clusters of probability mass. To achieve the same with a sample set, many samples are needed, typically with low weight. It is then increasingly more likely that the samples will be removed in the resampling step even though the overall cluster has a large probability.

The implementation is depicted in Figure 5.6 where $\mathbf{u}_{k,l}$ denotes the motion reported by the odometry between extraction of features $\mathbf{z}_k^l$ and $\mathbf{z}_k^{l-1}$.

5.3.2 Planned Sampling

Drawing samples from $p(\mathbf{x}_k|\mathbf{z}_k)$ is nontrivial as $\mathbf{z}_k$ could be any combination of features. The choice of $p(\mathbf{x}_k|\mathbf{z}_k)$ as an importance function was based on the observation that it is likely to approximate $p(\mathbf{x}_k|Z_k)$ well. Sampling from individual $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is easier than sampling from $p(\mathbf{x}_k|\mathbf{z}_k)$ and the support is likely to be approximately the same.

Implementation: MCL with Planned Sampling

Prediction and Measurement Update Step

$\tilde{\mathbf{x}}_k^{(i)} := \mathbf{x}_{k-1}^{(i)}$
$\tilde{\pi}_k^{(i)} := \pi_{k-1}^{(i)}$
`For each` $l = 1, \ldots, F_k$:
  `For each` $i = 1, \ldots, \sum_{j=1}^{l-1} n_k^p(\mathbf{z}_k, j)$:
    $\tilde{\mathbf{x}}_k^{(i)} := \mathbf{f}(\tilde{\mathbf{x}}_k^{(i)}, \mathbf{u}_{k-1,l}, \mathbf{w}), \quad \mathbf{w} \in p(\mathbf{w}_{k-1})$
  `For each` $i = (1 + \sum_{j=1}^{l-1} n_k^p(\mathbf{z}_k, j)), \ldots, \sum_{j=1}^{l} n_k^p(\mathbf{z}_k, j))$:
    `Draw` $\tilde{\mathbf{x}}_k^{(i)}$ `from` $p(\mathbf{x}_k|\mathbf{z}_k^{(j)})$
  `For each` $i = (1 + \sum_{j=1}^{l-1} n_k^p(\mathbf{z}_k, j)), \ldots, N_{k-1}$:
    $\tilde{\mathbf{x}}_k^{(i)} := \mathbf{f}(\tilde{\mathbf{x}}_k^{(i)}, \mathbf{u}_{k-1,l}, \mathbf{w}), \quad \mathbf{w} \in p(\mathbf{w}_{k-1})$
  $\tilde{\pi}_{k-1,l}^{(i)} := \tilde{\pi}_k^{(i)} p(\mathbf{z}_k^l|\tilde{\mathbf{x}}_k^{(i)}) \;\; \forall \;\; i$

Resampling Step

`Same as in Figure 5.3`

Figure 5.7: *Implementation of MCL with planned sampling for feature based localization. Note that part of the sample set is replaced for each detected feature.*

This means that drawing N_k^p samples from $p(\mathbf{x}_k|\mathbf{z}_k)$ is replaced by drawing $n_k^p(\mathbf{z}_k^j)$ samples from $p(\mathbf{x}_k|\mathbf{z}_k^j)$, where

$$N_k^p = \sum_{j=1}^{N_k^p} n_k^p(\mathbf{z}_k, j). \tag{5.24}$$

In this implementation $n_k^p(\mathbf{z}_k, j)$ depends only on a quality measure, $q(t)$, which in turn only depends on the feature type t

$$n_k^p(\mathbf{z}_k, j) = \frac{q(\text{type}(\mathbf{z}_k^{(j)}))}{\sum_{l=1}^{F_k} q(\text{type}(\mathbf{z}_k^{(l)}))} N_k^p \tag{5.25}$$

where F_k is the number of detected features in step k.

To get a properly weighted sample set, the planned samples must be weighted according to (5.23). At first glance it seems like a trivial task as $p(\bar{\mathbf{x}}_k^{(i)}|\bar{\mathbf{x}}_{k-1}^{(i)})$ is given by (5.17). This is however not true when $\bar{\mathbf{x}}_k^{(i)}$ is drawn from $p(\mathbf{x}_k^{(i)}|\mathbf{z}_k^j)$. In this case $p(\bar{\mathbf{x}}_k^{(i)}|\bar{\mathbf{x}}_{k-1}^{(i)})$ is most likely a very small number. To avoid this a heuristic is applied and α is set to 1. It is immediately noted that this choice of α gives too much weight to samples that match a single measurement but does not necessarily agree with previous measurements. The resulting sample set does not converge to $p(\mathbf{x}_k|Z_k)$ as $N \to \infty$. If it can be shown that the performance is enhanced by using planned sampling, the natural next step is to weigh the sample set properly.

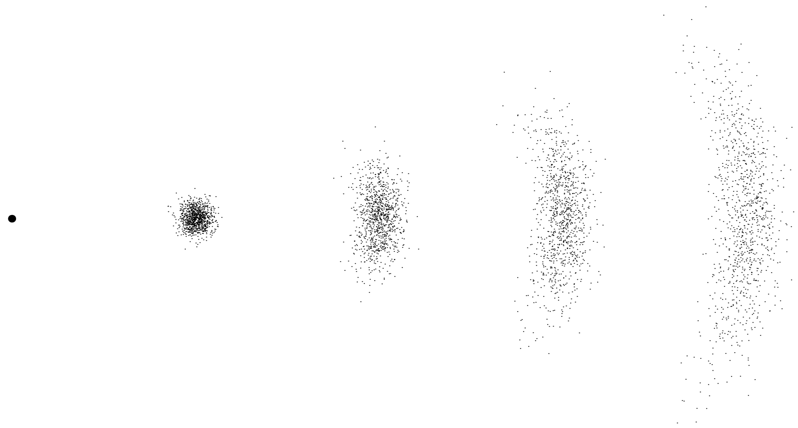


Figure 5.8: *This plot shows how 1,000 samples are propagated along a straight line path using the motion model (5.26).*

5.3.3 Motion Model

When generating samples from $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ in the prediction step of the MCL algorithms the following model is used

$$\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$
$$= \begin{pmatrix} x_{k-1} + D_{k-1}(1 + w^d_{k-1})\cos(\theta_{k-1} + \chi_{k-1}) \\ y_{k-1} + D_{k-1}(1 + w^d_{k-1})\sin(\theta_{k-1} + \chi_{k-1}) \\ \theta_{k-1} + D_{k-1} w^\theta_{k-1} \end{pmatrix}, \tag{5.26}$$

where

$$\begin{aligned} \mathbf{w}_k &= \left(w^d_k \; w^\theta_k\right)^T \\ \mathbf{u}_k &= (\Delta x_k \; \Delta y_k)^T \\ D_k &= \sqrt{(\Delta x^{(R)}_k)^2 + (\Delta y^{(R)}_k)^2} \\ \chi_k &= \arctan(\Delta y^{(R)}_k, \Delta x^{(R)}_k) \in (-\pi, \pi]. \end{aligned}$$

The variables w^d_k and w^θ_k are assumed to be independent, zero-mean and normally distributed variables, according to

$$w^d_k \sim \mathcal{N}(0, \sigma_d) \qquad w^\theta_k \sim \mathcal{N}(0, \sigma_\theta). \tag{5.27}$$

The input signal, $\mathbf{u}_k$, to the motion model (5.26) is the relative motion predicted by the odometric system between time steps $k-1$ and k. The larger σ_d and σ_θ

are, the more diffusion is added in the prediction step. Standard MCL requires robot motion for the sample set to converge to the true PDF, because the diffusion is the only process that allows the samples to change to new poses. For convergence it is also important that the sensor data provide more information than the diffusion caused by the motion dissipates. Figure 5.8 shows how an initially compact sample set diffuses when using uncertain odometric data.

5.3.4 Features

The minimalistic environmental model described in Section 2.4 consists of three different features; lines, door and points. The lines and the doors are extracted from laser data and the points from sonar data. Just like in Chapter 4 the combination of two points into a pair is also considered as a feature type.

The feature detectors are not perfect and neither is the map. Only features in the map can be used to provide information about the absolute pose of the robot. Introduce C as the class of measurements that originate from unmodeled features or phantom measurements and C^t a as subclass of C consisting of measurements of feature of type t. Let $\Pr(\mathbf{z}_k^j \in C^t|\mathcal{M}, \mathbf{x})$ be the probability of detecting a feature from class C^t from pose $\mathbf{x}$ given the map $\mathcal{M}$. To simplify matter this probability is assumed to depend only on the feature type, i.e.

$$\Pr(\mathbf{z}_k^j \in C^t|\mathcal{M}, \mathbf{x}) = \Pr(\mathbf{z}_k \in C^t) = const. \tag{5.28}$$

The two main functions to be described for each feature are $p(\mathbf{z}_k^j|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{z}_k^j)$, where $p(\mathbf{z}_k^j|\mathbf{x}_k)$ is needed in the measurement update step to update the weights of the samples and $p(\mathbf{x}_k|\mathbf{z}_k^j)$ when generating planned samples. Let $p(f_i^t|\mathbf{x}_k)$ be the probability of detecting the i:th feature of type t from pose $\mathbf{x}_k$ where t is the same type as $\mathbf{z}_k^j$. The probability $p(\mathbf{z}_k^j|\mathbf{x}_k)$ is then given by

$$p(\mathbf{z}_k^j|\mathbf{x}_k) \propto \Pr(\mathbf{z}_k^j \in C^t) + (1 - \Pr(\mathbf{z}_k^j \in C^t)) \max_{i=1,\ldots,|\mathcal{M}^t|} p(f_i^t|\mathbf{x}_k). \tag{5.29}$$

When determining the quality factor, q, for each feature type, t, the ability of the features to produce planned samples close to the true robot pose are considered. Here the quality is defined as the probability to draw a sample within 0.5 m from the true robot pose.

To achieve good performance of MCL it is necessary to use a pessimistic noise model for the feature detectors, i.e. the extracted features must be given an artificially high level of uncertainty. Low feature noise means that $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is very peaky and few samples are thus given high weight. The peakier $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is, the more samples have to be used. This problem is addressed in (Thrun, Fox & Burgard 2000).

Lines

Lines are detected using the Hough Transform. Given a detected line segment it can be matched against the map and the most likely robot poses can be calculated (compare with the pose candidates in Chapter 4). Sampling from the corresponding distribution is performed by repeating the following for each planned sample

1. Randomly select a map feature that satisfies (4.33).
2. Draw a sample along the $L-l$ long middle part of the center line (parallel to the wall) of the ellipse shown in Figure 4.8.

To save computations sampling is thus only done at the most likely distance from the map line feature. The function $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is modeled as

$$p(f_i^{line}|\mathbf{x}_k) = \begin{cases} e^{-\frac{1}{2}\left((\frac{\nu^\rho}{\sigma_d^{line}})^2+(\frac{\nu^\alpha}{\sigma_\alpha^{line}})^2+(\frac{h}{\sigma_h^{line}})^2\right)} & (4.33) \text{ holds} \\ 0 & (4.33) \text{ does not hold} \end{cases}$$

where ν^ρ and ν^α corresponds to the innovations in the Kalman filter in Chapter 3, and h is the distance outside the flat region in Figure 4.9 (inside the flat region $h = 0$).

Approximately 50% of the lines that are extracted with the Hough Transform corresponds to lines that are in the map when evaluated over a large part of the environment, which leads to

$$\Pr(\mathbf{z}_k \in C^{line}) = 0.5.$$

On average an extracted line segment matches 70 map line features and the average value for $L - l$ in (4.8) is 7 m due to the many long corridor lines. Using the strategy described above to generate planned samples, the total length along which samples are drawn is 70x7=490 m. The probability of the measurement having a true correspondence in the map is given by

$$1 - \Pr(\mathbf{z}_k \in C^{line}) \tag{5.30}$$

and hence the probability to draw a sample within 0.5 m from the true pose is given by

$$q(line) = \frac{0.5 \text{ m} \cdot (1 - \Pr(\mathbf{z}_k \in C^{line}))}{490 \text{ m}} \approx 5 \cdot 10^{-4}.$$

Doors

Door are extracted from laser data in the same way as described in Sections 2.4.2. The treatment of doors is the same as for lines with the exception of having larger uncertainty in the direction perpendicular to the door. The extra uncertainty comes from the large wall thickness of the building.

The performance of the door detector and the quality of the map is given by

$$\Pr(\mathbf{z}_k \in C^{door}) = 0.3. \tag{5.31}$$

in the environment under consideration.

The door is a powerful feature, both because there are relatively few doors in the environment ($M^{door} = 108$), but also because there is a distinct peak in $p(\mathbf{x}_k|\mathbf{z}^{door})$ for each door. The quality factor for doors is

$$q(door) = \frac{1 - \Pr(\mathbf{z}_k \in C^{door})}{108} = 6 \cdot 10^{-3}.$$

Points

The total number of points in the map is $M^{point} = 738$. A brute force implementation of the maximization in (5.29) involves looping through all features in the map. The point features are typically detected at short distances (< 5 m), hence only the points that are close to the current sample need to be taken into account. The points are therefore stored in a grid (5×5 m^2 cells) over the environment to allow fast access of the points that are close to a certain position. Through this construction the size of the map no longer affects the computation cost directly, only indirectly through the number of samples that has to be used.

The function $p(f_i^{point}|\mathbf{x}_k)$ is modeled as

$$p(f_i^{point}|\mathbf{x}_k) \propto e^{-\frac{1}{2}\left((\frac{\nu^d}{\sigma_d^{point}})^2 + (\frac{\nu^\alpha}{\sigma_\alpha^{point}})^2\right)} \tag{5.32}$$

where ν^d and ν^α are the innovations in distance and angle to the point feature respectively.

Sampling is done by randomly selecting one matching map feature and then randomly select one position along the corresponding possible circle (see Figure 4.13).

Normally on 30% of the detected points are in C^{point}, but in some parts the performance drops to 50%. Therefore the conservative probability

$$\Pr(\mathbf{z}_k \in C^{point}) = 0.5,$$

is chosen.

A detected point feature carry little value, as a point feature always matches all map features of that type and the support area of $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is rather large. A detected point feature is on average 1.5 m away from the robot, which gives

$$q(point)\frac{(1 - \Pr(\mathbf{z}_k \in C^{point})) \cdot 0.5}{738 \cdot (2\pi 1.5)} \approx 3 \cdot 10^{-5}.$$

Pairs

Just like for the point features the pairs are stored in a grid structure to reduce the access time. Matching a pair feature to the map gives rise to two possible robot poses (see Figure 4.14). The likelihood function for detecting a pair feature from pose $\mathbf{x}_k$ is modeled as

$$p(f_i^{pair}|\mathbf{x}_k) \propto e^{-\frac{1}{2}\left((\frac{\Delta d}{\sigma_d^{pair}})^2+(\frac{\Delta \alpha}{\sigma_\alpha^{pair}})^2\right)} \tag{5.33}$$

where Δd and $\Delta\alpha$ are the distance and angle innovations respectively.

Sampling from $p(\mathbf{x}_k|\mathbf{z}_k^j)$ is performed by first randomly select a one of the matching pair features and than place one sample in the center of one of the two circles from Figure 4.15. To save computations, all samples are thus placed at one of the the peaks of the distribution.

As a pair is constructed from two points, the probability of at least one point being incorrect (and hence the pair) is

$$\Pr(\mathbf{z}_k \in C^{point}) = 1 - (1 - \Pr(\mathbf{z}_k \in C^{point}))^2 = 0.75.$$

On average a pair is matched with 650 map features which yields the quality measure (remember that pairs can be matched in two ways)

$$q(pair) = \frac{1 - \Pr(\mathbf{z}_k \in C^{point})}{2 \cdot 650} \approx 2 \cdot 10^{-4}.$$

The assumption of independence between measurements do not hold if both points and pairs are used at the same time. The quality values for the two features hint that the pairs provide a more effective importance function. On the other hand $p(\mathbf{z}_k|\mathbf{x}_k)$ is less of an approximation for the points than for the pairs. For these reasons the best of both worlds are used, i.e. planned samples are generated from the pairs but the points are used to update the weight of the samples.

A more thorough description of the use of sonar point features applied to MCL is found in (Wijk 2001), where a more effective scheme is developed to take advantage of the fact that points can typically only be observed from certain directions.

5.3.5 From Sample Set to State Estimate

The sample set approximates $p(\mathbf{x}_k|Z_k)$, but in most cases the information that is sought is the best possible estimate of the robot pose. One way to define this is to use the conditional expectation value

$$E_{p(\cdot|Z_k)}[\mathbf{x}_k] = \int \mathbf{x}_k p(\mathbf{x}_k|Z_k) d\mathbf{x}_k \tag{5.34}$$

or when using the sample set approximation

$$E_{p(|Z_k)}[\mathbf{x}_k] \approx \sum_{i=1}^{N_k} \mathbf{x}_k^{(i)} \pi_k^{(i)}. \tag{5.35}$$

In Chapter 3 this method is used. There is however one major difference, the distribution is uni-modal as the task is pose tracking. Here on the other hand the distribution is by nature multi-modal. For example if there are two peaks that have accumulated most of the probability mass the expectation value is somewhere in between the two. An alternative is to use the maximum a posteriori estimate of the robot pose, i.e.

$$\hat{\mathbf{x}}_k = \underset{\hat{\mathbf{x}}_k}{\operatorname{argmax}}\, p(\hat{\mathbf{x}}_k|Z_k). \tag{5.36}$$

In this context it is tempting to use the sample with the highest weight before the resampling step as the estimate. This however results in an estimate that represent the pose corresponding to the sample that best fit the observations in a given iteration and not necessarily the peak of $p(\mathbf{x}_k|Z_k)$. In the thesis a grid is used to find an estimate of the robot pose. The grid consists of 1×1 m^2 cells covering the entire known environment, and each grid cell is given the total weight of all samples that fall within this cell. The estimate is then found by looking for the cell with the highest weight.

5.4 Experiments

Two different experiments are performed. The first one is intended to evaluate the performance increase gained by using random and planned sampling with the standard MCL algorithm. The second experiment demonstrates the strength that fusion of features give compared to using them one by one. Fixed sized samples sets are used in the experiments. An adaptive scheme is proposed in for example (Fox, Burgard, Dellaert & Thrun 1999). In the experiments presented below the algorithm is run once every 5 seconds if any new features have been reported. This low frequency is chosen to be sure that the results can in fact be achieved in real time.

5.4.1 Collecting Evaluation Data

To evaluate the performance of the standard MCL algorithm and the two extensions it is important that they be tested under the same conditions. Furthermore as the methods rely on drawing samples at random from various distributions the performance is not the same even if the same algorithm is run twice on the same set of data. To reduce the influence of chance, the algorithms are run ten times on several sets of data and the average performance measured.

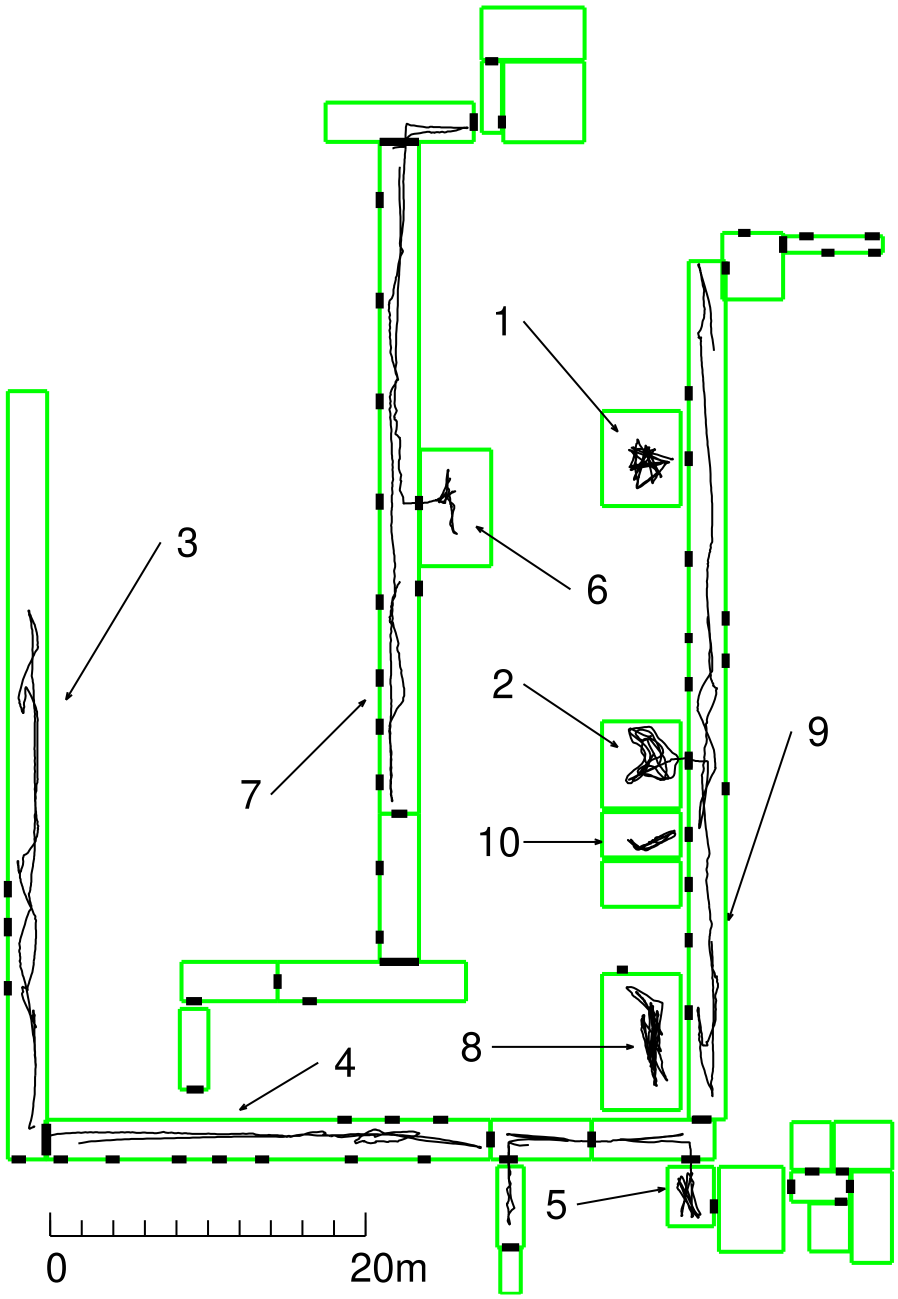


Figure 5.9: *The different trajectories for the MCL experiments at CAS are marked with numbers. The point features are omitted to make the figure less cluttered.*

The data sets are collected using a Nomad200 robot (see Appendix D). Ten different areas are covered. These areas represent different types of environments that are encountered at CAS. The trajectories followed by the robot are shown in Figure 5.9. Each trajectory is given a number.

The data is collected using a simple exploration behavior. The behavior is designed to take the robot to open spaces. It has no memory of where the robot has been before. Each trajectory consists of ten minutes of robot motion. The distance covered by the robot is to a large extent a function of the area it is in. In small, cluttered, offices the amount of motion that is allowed is limited. A corridor area on the other hand allows for long distances of motion. It should be emphasized that no active exploration strategy is used. Such a strategy makes the evaluation much more difficult as the motion of the robot becomes a function of the localization algorithm.

To judge success of localization the true trajectory followed by the robot is recorded using the pose tracking algorithm of Chapter 3. The localization experiment is terminated when either one of two conditions is satisfied. The first condition is that the probability of the best pose estimate is larger than

$$p_{th} = 0.9(1 - r_r - r_p). \tag{5.37}$$

The reasoning behind decreasing the threshold when using random sampling is that a fraction r_r of the samples is drawn from a uniform distribution and hence the threshold is in effect defined on the probability mass that is left. For planned sampling a fraction r_p of the samples are drawn from the alternative importance function and due to the lack of proper weighting this probability mass is also removed. The second condition upon which an experiment is terminated is at the end of the collected data set, i.e. when no convergence is detected after approximately ten minutes.

5.4.2 Evaluation: MCL

With $N \to \infty$ the sample set approximates $p(\mathbf{x}_k|Z_k)$ arbitrarily well. In a real application a finite sample set size must be used. For real time performance with this implementation the limitation in the number of samples that can be used is approximately 20,000 on a Pentium III 450 MHz. This is not a large number of samples considering that the environment is approximately 900 m^2, only 22 samples per square meter if the environment is sampled uniformly. When considering that the angle must be estimated as well, the sparseness is even clearer. To test the influence of the sample set size, global localization is performed along the ten trajectories with different N. Table 5.1 summarizes the results for four different sample set sizes. For each trajectory-N-pair the number of successful localizations over the ten runs is presented along with the average number of iterations.

Just as is expected the number of successful localizations increases as the number of samples is increased. It is also clear that the local characteristics

MCL								
Exp	$N = 10,000$		$20,000$		$50,000$		$100,000$	
1	2	(21)	4	(30)	6	(45)	7	(33)
2	0	(-)	6	(38)	8	(32)	10	(35)
3	0	(-)	0	(-)	0	(-)	0	(-)
4	0	(-)	2	(62)	1	(46)	4	(71)
5	1	(44)	3	(27)	5	(23)	4	(27)
6	4	(33)	1	(19)	9	(33)	10	(31)
7	0	(-)	1	(56)	3	(64)	4	(67)
8	5	(35)	9	(18)	1	(16)	10	(14)
9	2	(34)	4	(46)	6	(38)	9	(26)
10	1	(42)	0	(-)	2	(20)	3	(18)
Avg	15%	(34)	30%	(33)	50%	(33)	61%	(33)

Table 5.1: *Result when running the MCL algorithm over varying sample set sizes N. Each column give number of successful localization out of ten. The average number of iterations until (5.37) is satisfied is given in parentheses.*

of the environment influences the performance heavily. A clear example of this is trajectory 3 for which no successful localization is given. Trajectory 3 is in a corridor with very few features (compare run no. 20 in Section 4.4). There are three doors in this corridor but they are not detected during the experiment. Bare corridor walls result in very few point landmarks as well. The same explanation can be given for the poor results for trajectories 4 and 5. The experiment performed in the room marked by 10 shows bad results as well. This room is identical to the room beneath it in the map when looking only at door and line features. Due to the large amount of clutter it is not possible to see all four walls in the room. The small size of the room limits the motion and few features are detected.

In the room marked by 8 and the corridor marked by 9 localization is almost completely successful when using the largest sample set size ($N = 100,000$). Room 8 is unique in size and all walls can be detected using the Hough transform. With the help of the door and the point features localization is both reliable and fast in this room. The corridor marked with 9 also give good results. The main difference between this corridor and corridors 3 and 4 is that most of the doors are open. This provides door features, but also views of the rooms inside from which valuable line features can be extracted.

5.4.3 Evaluation: Random Sampling

The poor results with the standard MCL algorithm when used in conjunction with the minimalistic environmental model can to some degree be explained

Random sampling with $N = 50,000$								
Exp	$r_r = 0.01$		0.1		0.2		0.4	
1	2	(16)	5	(40)	6	(38)	10	(55)
2	9	(36)	9	(34)	10	(34)	10	(52)
3	0	(-)	0	(-)	0	(-)	0	(-)
4	2	(42)	4	(55)	3	(57)	4	(65)
5	3	(20)	3	(40)	6	(36)	10	(48)
6	8	(24)	10	(37)	9	(59)	0	(-)
7	3	(78)	4	(99)	7	(82)	2	(88)
8	10	(16)	10	(17)	10	(16)	10	(19)
9	6	(31)	8	(20)	9	(43)	9	(67)
10	2	(24)	7	(41)	9	(39)	8	(24)
Avg	45	(30)	60	(38)	69%	(43)	63%	(48)

Table 5.2: *Results when varying the ratio r_r of random sampling when $N = 50,000$. A clear improvement is seen when comparing the best column in this table ($r_r = 0.2$) with the $N = 50,000$ column in table 5.1.*

by the fact that each detected feature does not in itself give much information about the pose. Seeing a point is next to worthless and so is seeing a single line. The minimalistic model contains only very few features and so unless all features in an area are seen it is hard to tell the areas apart. As an example all rooms to the left of corridor 9 have the same width. This means that unless the upper and lower walls are both detected all rooms match the measurements. In (Fox, Burgard, Dellaert & Thrun 1999) raw sensor data is used as input to the MCL algorithm. The raw sensor data is richer in its description of the environment.

Table 5.2 summarizes the results when experimenting with different values for r_r, the ratio of samples that should be drawn from a uniform distribution in each step. The sample set size N is chosen as 50,000. It is not at all surprising to see that $r_r = 0.01$ does not change much. 500 samples spread uniformly over 900 m^2 has little effect. The increase in performance for higher values of r_r shows that adding samples from a uniform distribution have a positive effect. The best performance is achieved with $r_r = 0.2$ out of the tested values for r_r. Noticeable is also that the number of iterations needed for the algorithm to converge increase with increasing r_r. When r_r becomes larger the chance that samples are removed from the typical set, T, is also increased. In the end of the convergence most of the samples are gathered in a small region of the state space. The random samples are thus mostly taken from this region and therefore slows down the convergence. Convergence speed although being an interesting characteristic of a localization algorithm is secondary to reliability. It must therefore be concluded that drawing some fraction of the sample set from a uniform distribution increases the performance of the MCL algorithm.

Planned sampling with $N = 10,000$								
Exp	$r_p = 0.01$		0.02		0.05		0.1	
1	5	(47)	8	(32)	10	(40)	10	(42)
2	10	(31)	10	(27)	10	(30)	10	(36)
3	0	(-)	0	(-)	0	(-)	0	(-)
4	5	(64)	7	(64)	9	(68)	0	(-)
5	8	(23)	10	(28)	10	(24)	9	(27)
6	10	(29)	10	(31)	6	(33)	0	(-)
7	8	(93)	10	(82)	8	(122)	0	(-)
8	10	(9)	10	(8)	10	(10)	10	(22)
9	10	(25)	10	(18)	10	(19)	10	(29)
10	7	(34)	10	(36)	10	(23)	10	(27)
Avg	73%	(37)	85%	(36)	83%	(40)	59%	(31)

Table 5.3: *Result when varying the ratio r_p of samples drawn from the alternative importance function. The best column in this table ($r_p = 0.02$) is superior to any column in tables 5.1 and 5.2 although the sample set size N is only 10,000.*

5.4.4 Evaluation: Planned Sampling

Random sampling improves the performance but is not able to achieve convincing reliability. With $r_r = 0.2$ the percentage of successful localizations is still only 69%. In addition this result is achieved with a sample set size which cannot be handled in real time. The main reason for the failure of MCL and random sampling is lack of sample support at the true pose. Planned sampling is designed to alleviate this problem. To guarantee real time performance the experiments are performed with $N = 10,000$. With this sample set size the standard MCL algorithm resulted in only 15% convergence. The corresponding number for random sampling is approximately 40% convergence. Table 5.3 shows the results of varying the ratio, r_p, of samples drawn from the alternative importance function. Far superior results are achieved in all these experiments compared to the ones in Sections 5.4.2 and 5.4.3. Ratios 0.01, 0.02, 0.05 and 0.1 are evaluated. Best results are given with $r_p = 0.02$, i.e. when 2% of the samples are drawn from the alternative importance function. The performance drops when increasing r_p too much, because the improperly weighted planned samples create strong sample support in regions that currently matches the data well, irrespective of the previous samples.

5.4.5 Comparison of Algorithms

Figure 5.10 shows a comparison of standard MCL against adding random and planned sampling for different sample set sizes N. The curves give the per-

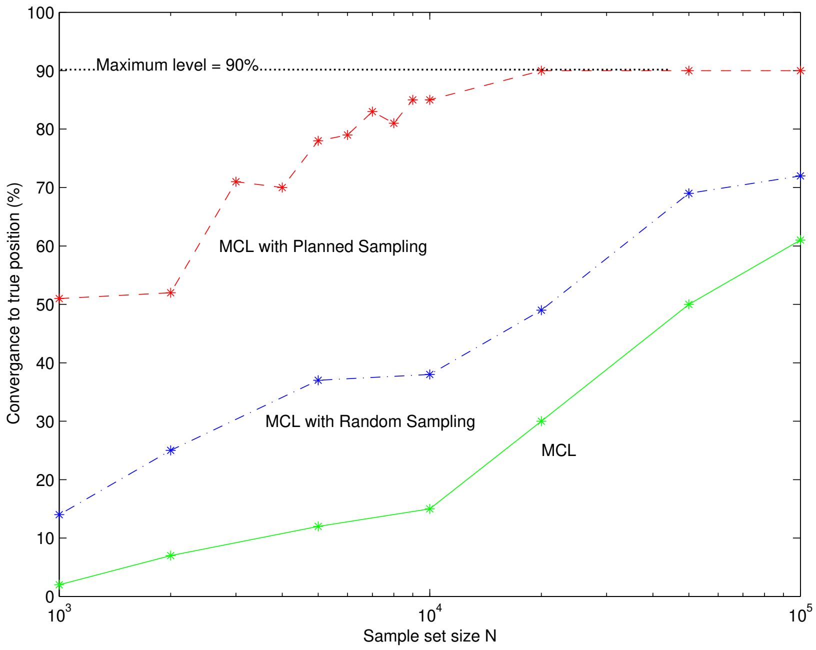


Figure 5.10: *Convergence to true position (%) vs. sample set size N. The solid line represents MCL, the dashed-dotted line represents random sampling and the dashed line represents planned sampling. The maximum level that could be reached was 90% because one of the ten data sets contained too few detectable features. Each point in the graph corresponds to 100 experiments!*

formance averaged over the ten different trajectories. The random sampling algorithm is evaluated with $r_r = 0.2$ which gave best results in the experiments in Section 5.4.3, and planned sampling is evaluated with $r_p = 0.02$.

That planned sampling is superior to the two other algorithms is beyond doubt. Random sampling is also significantly better than the standard MCL algorithm. Fixing a certain level of performance, the three different algorithms require drastically different sample set sizes and thus computational resources. Standard MCL requires between 2 and 2.5 times more samples than with random sampling and 40 to 50 times more samples than with planned sampling.

5.4.6 Combining Different Features

Four different features are used in the experiments presented so far, lines, doors, points and pairs. In the following experiment the strengths and weaknesses of the individual features are investigated as well as the benefit of combining them.

The best results in the previous experiments with planned sampling are achieved using $r_p = 0.02$. In all experiments presented below this is the setting being used, together with $N = 10,000$ to ensure real-time performance.

The accuracy which a feature provides for localization is to a large degree a question of how many dimensions it can establish constraints on. Consider a wall in a corridor, this line feature on its own does only provide information about the perpendicular distance to the wall and the orientation of the robot, but with high accuracy though. Using line features in environments where non-parallel lines can be detected provide high accuracy. Doors do not provide good accuracy because of the thick walls in the old hospital building. Points can provide excellent accuracy if the matching problem is solved, i.e. the correct map feature is matched to the detected feature, but this is difficult since the point map features have very little separation in some areas.

Planned sampling with $N = 10,000$, $r_p = 0.02$								
Traj\feat	**L**		**D**		**T**		**P**	
1	1	(29)	0	(-)	1	(60)	0	(-)
2	10	(60)	0	(-)	3	(52)	0	(-)
3	0	(-)	0	(-)	0	(-)	0	(-)
4	0	(-)	0	(-)	0	(-)	0	(-)
5	8	(44)	0	(-)	1	(89)	0	(-)
6	0	(-)	0	(-)	0	(-)	0	(-)
7	6	(118)	0	(-)	1	(71)	0	(-)
8	10	(47)	0	(-)	4	(78)	2	(64)
9	2	(32)	0	(-)	0	(-)	0	(-)
10	1	(38)	0	(-)	0	(-)	0	(-)
avg.	38%	(60)	0%	(-)	10%	(69)	2%	(64)

Table 5.4: *Summary of running 10 experiments for each trajectory and feature. Character codes: L -* Line, *D -* Door, *T -* Point *and P -* Pair.

Single Feature Using only one feature give poor results. Table 5.4 shows the result for the four different features used alone. The line feature gives the best result with 38 successful localizations out of the 100 experiments. The second best single feature is the point feature. The doors and the pairs are next to useless when used alone. The point and the line feature are both easy to detect. The line feature is easiest of all features to detect and the number of lines in the map is relatively small. Doors are hard to detect. The pair feature is more frequent than the doors but still a rare feature compared to lines and points. Figure 5.11 shows the localization performance along trajectory 9 for the different features. The probability mass within 1 m from the true robot pose is plotted versus time. It is evident that the point and the line feature are the only features that provide enough information to get the algorithm to start converging. No clear signs of convergence is visible for the pair and the door features. A feature that is rare is not able to overcome the diffusion caused by the uncertainty in the odometric information. Even if sample support exist at the true state, the information dissipates too fast.

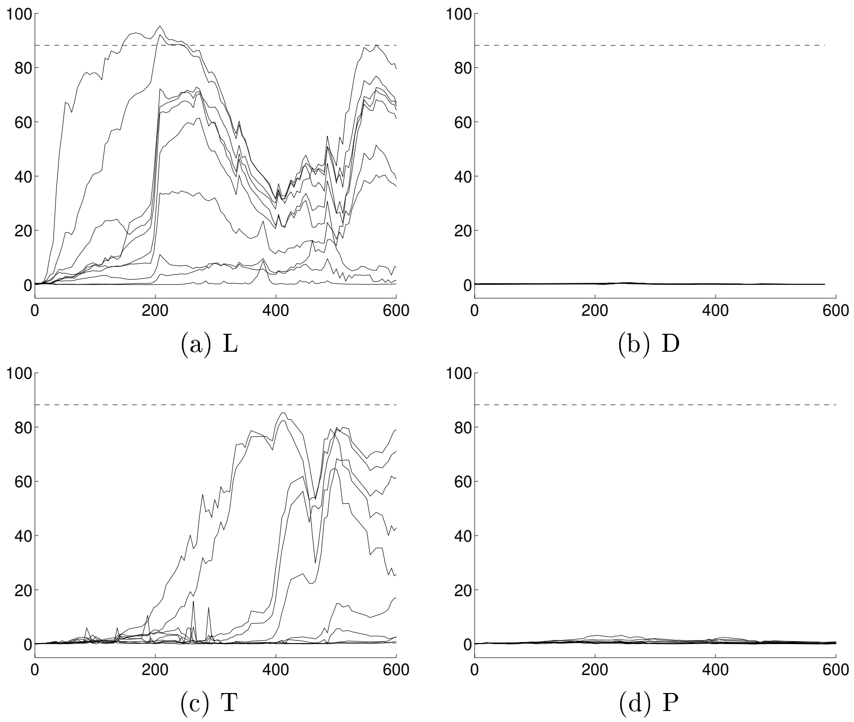


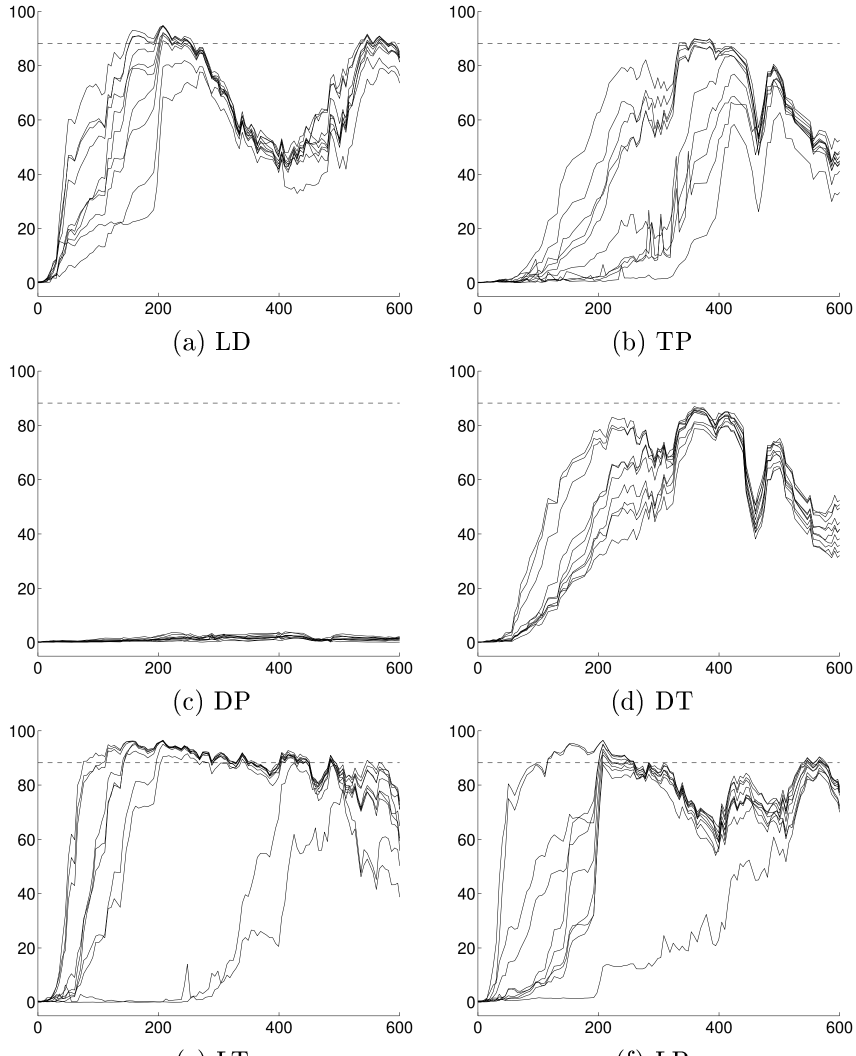
Figure 5.11: *Each sub-figure shows the result of 10 attempts to perform global localization along trajectory 9 using different features. For each experiment the percentage of the total weight of the samples within 1 m of the true pose is shown on the vertical axis. The dashed lines mark the threshold used for a successful localization. The same scale is used in all figures to highlight the differences.*

Planned sampling with $N = 10,000$, $r_p = 0.02$						
Traj\feat	**LD**	**TP**	**DP**	**DT**	**LT**	**LP**
1	1 (15)	3 (57)	0 (-)	2 (53)	10 (38)	0 (-)
2	10 (47)	10 (42)	0 (-)	9 (34)	9 (45)	10 (42)
3	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
4	0 (-)	0 (-)	0 (-)	0 (-)	7 (54)	4 (78)
5	7 (47)	7 (87)	0 (-)	0 (-)	10 (22)	10 (36)
6	3 (54)	0 (-)	0 (-)	0 (-)	10 (42)	4 (55)
7	3 (119)	2 (67)	0 (-)	0 (-)	10 (90)	9 (117)
8	4 (29)	10 (66)	0 (-)	0 (-)	10 (9)	10 (13)
9	9 (41)	3 (57)	0 (-)	0 (-)	9 (35)	8 (34)
10	1 (24)	0 (-)	0 (-)	2 (65)	7 (22)	3 (51)
avg.	38% (49)	35% (62)	0% (-)	13% (42)	82% (40)	57% (51)

Table 5.5: *Summary of running 10 experiments for each trajectory and feature combination. Character codes: L -* Line, *D -* Door, *T -* Point *and P -* Pair.

Combining Two Features Table 5.5 shows the results of combining two features. As in Table 5.4 the number of successful localizations is shown along with the average number of iterations for convergence. Figure 5.12 shows the performance along trajectory 9. The conclusions induced by the experiments with single features are here strengthened. The infrequent pair and door features do not even combined provide enough information to get convergence. Combining any of these two features with the line or the point feature boost the performance though. A comparison of Figures 5.11(a) and 5.12(a) illustrate the effect that the door feature has. By sampling from the alternative importance function based on observations of doors, sample support around the true pose is quickly established and the result is made less dependent on the initial sample set. Combining the door feature with the point feature has a similar effect as shown by Figures 5.11(c) and 5.12(d). Note that neither of the runs shown in Figure 5.12(d) reach the required threshold for being classified as a successful localization and thus do not show up in Table 5.5. The performance is greatly improved though.

The pair feature is similar to the door in its effect on the performance. When combined with a frequently observed feature that can more than compensate for the diffusion caused by motion, the performance is greatly improved. The improvement is not as large as when using the door feature, but still significant. Looking at Table 5.5 the most significant improvement is achieved when combining lines and points. In 82 cases out of 100 a successful localization is obtained. The only information not contained in the line and the point features combined is the information from the doors. Since doors are infrequently detected this loss does not cost much in performance.

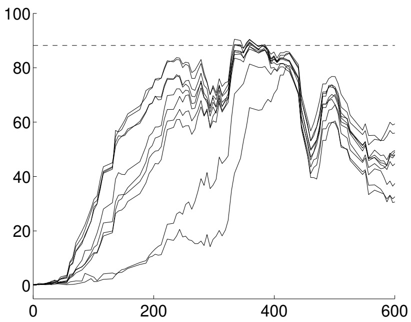


(a) LD (b) TP

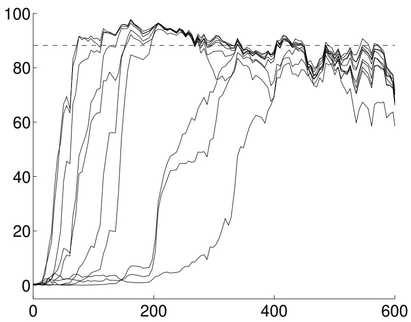
(c) DP (d) DT

(e) LT (f) LP

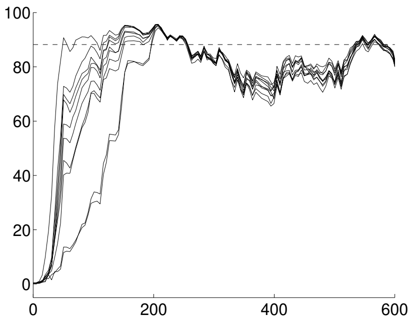
Figure 5.12: *Each subfigure shows the result of 10 attempts to perform global localization along trajectory 9 when combining two features. For each experiment the percentage of the total weight of the samples within 1 m of the true pose is shown on the vertical axis. The dashed lines mark the threshold used for a successful localization.*



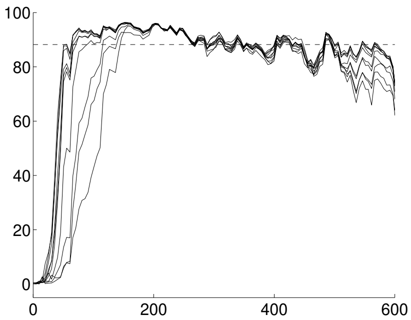
(a) DTP



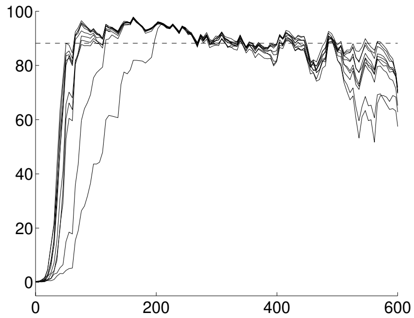
(b) LTP



(c) LDP



(d) LDT



(e) LDTP

Figure 5.13: *Each subfigure shows the result of 10 attempts to perform global localization along trajectory 9 when combining three or all features. For each experiment the percentage of the total weight of the samples within 1 m of the true pose is shown on the vertical axis. The dashed lines mark the threshold used for a successful localization.*

Planned sampling with $N = 10,000$, $r_p = 0.02$					
Traj\feat	**DTP**	**LTP**	**LDP**	**LDT**	**LDTP**
1	2 (59)	80 (26)	1 (12)	9 (37)	8 (32)
2	8 (32)	10 (33)	10 (37)	10 (27)	10 (27)
3	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
4	0 (-)	7 (59)	7 (84)	8 (59)	7 (64)
5	0 (-)	10 (22)	10 (30)	10 (23)	10 (28)
6	1 (70)	10 (36)	9 (54)	10 (34)	10 (31)
7	2 (80)	9 (108)	7 (116)	8 (88)	10 (82)
8	4 (84)	10 (10)	10 (12)	10 (9)	10 (8)
9	6 (58)	9 (36)	10 (27)	10 (19)	10 (18)
10	2 (61)	10 (16)	7 (30)	8 (19)	10 (36)
avg.	25% (57)	83% (38)	71% (45)	83% (34)	85% (36)

Table 5.6: *Summary of running 10 experiments for each trajectory and feature combination. Character codes:* L - Line, D - Door, T - Point *and* P - Pair.

Combining Three and All Features Having used features alone and in combinations of two, most of the characteristics of the different features have already been acquired. Combing three or all of them thus only act as a verification of the conclusion drawn so far. Points and pairs are derived from the same sensor data and therefore contain the same information. Hence, combining these two feature with a third is not much different from using only one of the two in combination with the third. The pair feature is somewhat better as a basis for the alternative importance function and somewhat faster convergence is to be expected. Comparing Tables 5.5 and 5.6 shows that the line and point combination on average needs 40 iterations for the 82 successful localizations. Adding the pairs feature changes this to 38 iterations and 83 successful localizations which is an insignificant improvement.

5.5 Summary

In this chapter the Monte Carlo technique are outlined in general and the application of the Sampling/Importance Resampling (SIR) algorithm to mobile robot global localization are discussed in more detail. SIR is known as Monte Carlo Localization (MCL) in the localization literature and was applied to the problem in (Dellaert, Fox, Burgard & Thrun 1999). Due to limitations in the computational resources a finite sample set size must be used. In a large environment this leads to a sparse sample set and the probability of having sample support in areas where the conditional PDF $p(\mathbf{x}_k|Z_k)$ is significant becomes increasingly smaller. Two approaches for improving the performance of the standard MCL algorithm are presented. By drawing samples from a uniform distribution in each iteration the performance increases. For similar performance the required sample set size is cut in half and with that the computational demand. The observation that $p(\mathbf{x}_k|\mathbf{z}_k)$ is likely to be significant where $p(\mathbf{x}_k|Z_k)$ is significant led to the introduction of an alternative importance function that boosted the performance significantly. Using this second approach lowers the required sample set size by a factor of 40-50 for similar levels of performance compared with the standard MCL algorithm.

A thorough investigation of the effects of combining different features is also presented. Four different features are considered; lines, doors, points and pairs. It is concluded that the features can roughly be divided into two groups. Those that are infrequent and whose main purpose is to act as a good basis for the alternative importance function and those that are frequent and makes sure that the algorithm converges. The pair and the door feature belong to the first group and the point and the line to the second. These two groups are similar to the division in Chapter 4 in terms of creative and supportive features. The line and the door feature are extracted from laser data whereas the point and pair come from sonar data. Using only one sensor modality is found inefficient, however, combining both laser and sonar yields a significant improvement in performance.

Chapter 6

Hierachical Simultaneous Localization and Mapping

> "*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*"
>
> (Elfes 1989)

The localization methods presented in Chapters 3-5 have one thing in common, they all assume the existence of a map of the environment. As already mentioned in Chapter 1, a service robot should be able to construct the map on its own. It was also concluded that building a map requires that localization is performed simultaneously. This process is known as simultaneous localization and mapping (SLAM). This chapter investigates feature based SLAM with a focus on methods to deal with the scaling issue that standard SLAM algorithms suffer from. The aim is to construct a minimalistic description of the environment. That is, not to make the most detailed map possible, but rather try to capture the large scale and robust features that are successfully used in the previous chapters.

6.1 Background

Many methods for SLAM have been suggested in the literature. Not surprisingly there is a tight coupling between the map representation being used and the approach taken to SLAM. Brooks mentions two desirable properties of a map (Brooks 1984):

- "*It should be stable. Small variations in observations should almost everywhere lead to structurally isomorphic representations.*"

- "*It should be mostly monotonic. Usually when additional observations are made the representation of the world should be augmented rather than being restructured. This need not always be the case.*"

Three main directions can be identified in the literature: topological, grid-based and feature-based approaches. As the thesis deals with feature-based localization emphasis is placed on these methods.

In topological techniques the environment is modeled as a graph, in the extreme case completely without geometric information. Localization is achieved by recognizing places/nodes. A subway map is an example of a topological map. Here the nodes corresponds to stations and are identified by their name. One advantage with this is that: "*movement errors do not accumulate globally in topological maps as they do in maps with a global coordinate system since the robot only navigates locally, between the places*" (Kortenkamp & Weymouth 1994). Topological mapping scales well to large environments since the amount of information that is stored is limited to the description of the places/nodes. One of the major disadvantages with topological SLAM is that it typically is quite difficult to reliably recognize a place. The problem of determining if a node has been visited before is referred to as the *am-I-there-yet*-question in (Brooks 1984, Brooks 1985), or alternatively put, have we been here before?

Ever since the introduction of the occupancy grid by Moravec and Elfes (Moravec & Elfes 1985), the grid based mapping techniques are widely used for mapping and localization. As discussed in Chapter 2 each grid cell holds a value that represents the probability that it is occupied by an object. A typical implementation of grid-based SLAM is to keep a local and a global grid. The global grid is where the overall map is stored and the local map is used to update it. By matching the local map to the global map a measurement is given of the position of the robot. The local map can also be used to improve the global map. An inherent problem with grid based methods is that they are computationally expensive and consume much memory.

In (Thrun, Burgard & Fox 2000) an approach to SLAM is presented that fits somewhere between the feature based and the grid based techniques (compare appearance based methods in Section 2.1.5). It uses scan-matching to build the map and a sample based method to perform localization.

6.1.1 Feature-Based SLAM

Most of the work on feature-based SLAM can be traced back to (Smith et al. 1987), where *stochastic mapping* is presented, which is an extended Kalman filter based approach to SLAM. The robot pose and the location of all map features are collected in one large state vector. Both the robot pose and the location of the map features are updated when mapped features are re-observed. In essence, localization is performed within the current map, and when the robot enters new areas the state vector is augmented with new features. The two main steps of stochastic mapping are prediction and update. In the prediction step the control signals to the robot or odometric information is used to predict the state at the next time step. In the update step measurements of features are used to update the robot pose and the mapped features.

The EKF gives only an approximative solution. A linearization must be performed around the predicted state before the update can be done. Estimation errors give linearization errors which eventually can cause divergence. Moutarlier and Chatila address this problem in (Moutarlier & Chatila 1990) where the *relocation-fusion* approach is presented. The update step from (Smith et al. 1987) is divided in two steps. In the first one, relocation, only the robot pose estimate is updated using the measurements. Linearization is then done around the updated robot pose before the features are updated in a fusion step.

A direct implementation of stochastic mapping has a $\mathcal{O}(N^3)$ complexity, where N is the number of mapped features. The problem is referred to as the map scaling problem. It is the correlation terms that result in the high complexity. An attempt to reduce the correlations to a level where they can be neglected is presented in (Leonard & Durrant-Whyte 1991*b*). The reduction is achieved by noticing that the correlations are the result of updating the robot pose with an uncertain map features or updating the map features with uncertain robot pose. Neglecting the correlations are motivated by only performing an update with confirmed robot poses and confirmed features. Confirmed is defined by the covariance being smaller than some threshold in which case it is approximated to zero. As soon as the robot starts moving the robot pose is uncertain and it is therefore necessary to acquire the first feature before motion has started.

The map scaling issue is approached in (Uhlmann 1995) where Covariance Intersection (CI) (also known as Gaussian Intersection) is applied. CI is applicable even when the correlations are unknown and provides a conservative estimate of the covariance. The overly optimistic estimate that typically results when neglecting the correlation terms in standard stochastic mapping, as pointed out in e.g. (Castellanos et al. 1997), is here replaced by an overly pessimistic estimate.

Another approach to the cross-correlation problem is presented in (Csorba & Durrant-Whyte 1997, Csorba et al. 1997), where a relative frame of reference is used. That is, the positions of the features are given only in relation to other features and not to some absolute frame of reference. One disadvantage, which is pointed out in (Csorba et al. 1997), 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.

Important contributions to feature-based SLAM are also given in (Castellanos & Tardós 1999). The main contribution here is the combination of many features. It is typically assumed that a feature can only match a feature of the same type. This limitation is alleviated by the introduction of so-called binding matrices that relate different features. In this way a plane can be used to update an edge for example.

In (Dissanayake et al. 2000) yet another approximation technique to full SLAM is proposed. It is noted that the removal of features from the state vector does not lead to any inconsistencies, only loss of information. By carefully selecting which features to remove performance is only slightly reduced. The scheme for removing features is based on only keeping the best features that are no longer visible. That way the robot can still re-localize when it returns to a position it has already been to, and the number of features that have to be maintained is reduced.

Cox and Leonard only consider map building and not the full SLAM problem in (Cox & Leonard 1994). The mapping problem is cast as a multiple hypothesis problem, where each hypothesis represents one way of interpreting the measurements, i.e. one map. The position of the robot is assumed to be precisely known. Applying this technique on SLAM is a true challenge as each hypothesis on its own suffers from the map scaling problem.

6.1.2 Hierarchical SLAM

Here, a hierarchical approach to SLAM is argued for, i.e. H-SLAM. Neglecting the correlations between features altogether has been shown to cause failure. On the other hand, even though the state vector is carefully pruned, the complexity still limits the size of the environment that can be handled. At some point the environment has to be divided into smaller units.

A two-layer hierarchy is here considered. As the environment becomes large the number of levels in the hierarchy can be increased, for example, the lowest levels could be rooms, the second floors and the third level in the hierarchy could correspond to buildings. The idea of a hierarchical map structure is not new.

In (Hébert et al. 1996) the environment is divided into local maps. The division is here driven by the desire to create local maps that are independent of the odometric errors. This is possible by measuring the relative position of the features.

Chong and Kleeman also use a local map strategy in combination with an advanced sonar sensor array which is capable of localizing and classifying simple indoor features (Chong & Kleeman 1997). The division into submaps is here driven by a desire to reduce the memory and the processing requirements. Furthermore the strategy addresses the problem of divergence caused by error accumulation from, for example, inaccurate odometric models and linearizations. Each submap maintains full correlation information. A new submap is created when the uncertainty within the current submap exceeds a certain threshold.

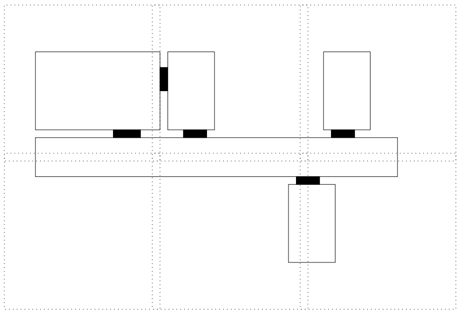


Figure 6.1: *Using square submaps to build a map. The borders between the submaps often end up in inconvenient locations. Here the intersection between the upper and lower row of submaps are in the middle of a corridor which leads to a large feature overlap and unnecessary switching between the submaps.*

This chapter is mainly based on the Decoupled Stochastic Mapping (DSM) technique presented in (Feder 1999), where, like in (Chong & Kleeman 1997), each submap maintains full correlation information, but correlations between submaps are only given through the use of a common coordinate system. The submaps are fixed in size and are distributed in a grid structure (see Figure 6.1). The crucial step when dividing the world into submaps is to correctly handle the transition between submaps. Care has to be taken to ensure a consistent error estimate, but at the same time global convergence is desired so that the uncertainty must not be excessively overestimated. While the grid based submap allocation strategy is easy to implement, the intersections between the submaps often end up in inconvenient positions. In Figure 6.1 the border between the upper row of submaps and the lower is in the middle of the corridor. This gives two undesirable consequences. First, the features that can be detected when moving in the corridor are the same irrespectively of whether the robot is in the upper row of submaps or the lower. This results in a significant overlap of features between the submaps. Second, the robot is likely to change submap more often than with a better distribution of submaps.

The map used in the previous chapters is made manually. Each room is first mapped individually and then their relative positions are estimated. Guided by this, it is more natural to use one submap for each room (see Figure 6.2). Switching between submaps is now done only when going through a door. To incorporate corridors and other regions that are not really rooms, the concept of room is expanded to an area. The door concept is also expanded to *gateway*, which connects two areas.

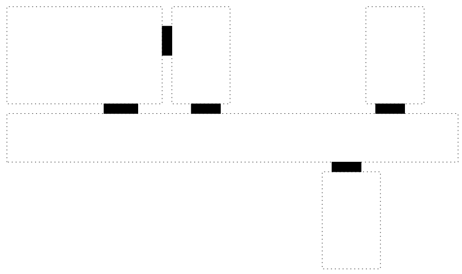


Figure 6.2: *Using one submap per room (area in general). The division between the submaps is now more natural and each area is more accurately mapped as it is contained in one submap.*

6.1.3 Outline

The remainder of this chapter is organized as follows. In Section 6.2 the theory of standard feature-based SLAM is briefly outlined, whereas Section 6.3 describes the DSM technique and some other algorithmic details. Section 6.4 compares standard SLAM with H-SLAM regarding map quality and complexity. The suggested room based map allocation strategy is also compared with the original grid based method. Furthermore, an active strategy for room based mapping in cooperation with a user is presented as a possible scenario for how mapping can be accomplished in a service robot application.

6.2 Theory

The state vector, $\mathbf{x}_k$, incorporates the location of all N mapped features, $\{\mathbf{x}_k^i,\ i = 1, \ldots, N\}$ as well as the robot pose and is thus given by

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{x}_k^r \\ \mathbf{x}_k^1 \\ \vdots \\ \mathbf{x}_k^N \end{pmatrix}. \tag{6.1}$$

The SLAM problem can be formulated as augmenting and estimating $\mathbf{x}_k$ given measurements of the environment, i.e. estimating both $\mathbf{x}_k^r$ and $\{\mathbf{x}_k^i,\ i = 1, \ldots, N\}$ and adding new features if needed.

Let $\hat{\mathbf{x}}_{k|k}$ denote the estimate of the state vector at time k. The corresponding estimation error covariance matrix can be decomposed as

$$P_{k|k} = \begin{pmatrix} P^{rr} & P^{r1} & \dots & P^{rN} \\ P^{1r} & P^{11} & \dots & P^{1N} \\ \vdots & \vdots & \ddots & \vdots \\ P^{Nr} & P^{N1} & \dots & P^{NN} \end{pmatrix}, \tag{6.2}$$

where P^{rr} is the covariance matrix of the robot pose estimate, and P^{ii} the covariance matrices for the features. The correlations between different features and the robot pose are given by the off-diagonal submatrices. The estimated state vector $\hat{\mathbf{x}}_{k|k}$ together with the covariance matrix $P_{k|k}$ is often referred to as a stochastic map. This term highlights the fact the the map is not fixed, it is being estimated as the robot moves along.

6.2.1 Prediction

The robot pose and the feature locations are given in a fixed global coordinate system. Moving the robot therefore only effects the estimate of the robot pose $\hat{\mathbf{x}}^r$. The update is based on the motion model for the robot,

$$\hat{\mathbf{x}}^r_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}^r_{k-1|k-1}, \mathbf{u}_k, 0) \tag{6.3}$$

where the zero corresponds to the best prediction of the process noise $\mathbf{w}_k$ (2.14). With the simplifying assumption that $\mathbf{w}_k$ is independent of the state, the covariance matrix $P^{rr}_{k|k-1}$ approximately evolves as

$$P^{rr}_{k|k-1} = F_{x^r} P^{rr}_{k-1|k-1} F^T_{x^r} + F_u Q_k F^T_u \tag{6.4}$$

where Q_k is the covariance matrix of the process noise and F_{x^r} and F_u are the Jacobians of $\mathbf{f}$ evaluated in $(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$. The covariance for the features do not change by moving the robot, but the correlations between the robot pose and features do. These are updated according to (Moutarlier & Chatila 1990)

$$P^{ri}_{k|k-1} = F_{x^r} P^{ri}_{k-1|k-1}. \tag{6.5}$$

6.2.2 Update

To reduce the uncertainty, features extracted from sensor data must be successfully matched to the stochastic map. The measurement vector $\mathbf{z}_k$ containing M observed features is given by the, in general non-linear, measurement equation

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) = \begin{pmatrix} \mathbf{z}^1_k \\ \vdots \\ \mathbf{z}^M_k \end{pmatrix} = \begin{pmatrix} \mathbf{h}^1(\mathbf{x}^r_k, \mathbf{x}^{o^1}_k, \mathbf{v}^1_k) \\ \vdots \\ \mathbf{h}^M(\mathbf{x}^r_k, \mathbf{x}^{o^M}_k, \mathbf{v}^M_k) \end{pmatrix}. \tag{6.6}$$

It is here assumed that the l:th measurement $\mathbf{z}_k^l$ corresponds to the $o(l)$:th mapped feature $\mathbf{x}_k^{o(l)}$ where $o(l) \in [1, \ldots, N]$. Using the standard equations for the extended Kalman filter yields (2.21) the update equations

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, 0)) \tag{6.7}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{6.8}$$

where K_k is the Kalman gain and H_k is the Jacobian of $\mathbf{h}$ with respect to $\mathbf{x}_k$ evaluated in $\hat{\mathbf{x}}_{k|k-1}$. The Kalman gain is given by

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}, \tag{6.9}$$

where R_k is the measurement covariance matrix. It is assumed that the measurements are independent, which gives that R_k is a block diagonal matrix,

$$R_k = \begin{pmatrix} R_k^1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & R_k^M \end{pmatrix}. \tag{6.10}$$

As noted in (Moutarlier & Chatila 1990) the matrix H_k is quite sparse. The larger the state vector is, the sparser H_k is. By taking this into account the computational complexity is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(MN^2)$. The matrix $(HPH^T + R)$ can be divided into submatrices given by

$$\begin{aligned}
[HPH^T + R]_{l,m} = & H_{x^r}^l P^{rr} (H_{x^r}^m)^T + H_{x^{o(l)}}^l P^{o(l)r} (H_{x^r}^m)^T + \\
& H_{x^r}^l P^{ro(m)} (H_{x^{o(m)}}^m)^T + H_{x^{o(l)}}^l P^{o(l)o(m)} (H_{x^{o(m)}}^m)^T + \delta_{lm} R^l
\end{aligned} \tag{6.11}$$

where $l, m = 1, \ldots, M$,

$$\delta_{lm} = \begin{cases} 1, & l = m \\ 0, & l \neq m \end{cases}. \tag{6.12}$$

and the time indices are omitted. The matrix (PH^T) can be divided in a similar way into

$$[PH^T]_{i,l} = P^{ir} (H_{x^r}^l)^T + P^{io(j)} (H_{x^{o(l)}}^l)^T \tag{6.13}$$

where $i = 0, \ldots, N$, $l = 1, \ldots, M$ and $\mathbf{x}_k^0 = \mathbf{x}_k^r$.

6.2.3 Augmenting the Map

When the SLAM process starts, the state vector only contains the robot pose. Without any features in the map there is no way to keep the uncertainty bounded. Extracted features that do not match any feature in the map are natural candidates for inclusion. However, every new feature in the map increases the computation cost. Therefore, care has to be taken as to which features to add.

Given a measurement $\mathbf{z}^l$, introduce the function $\mathbf{t}$ that transforms the measurement into the global reference frame of the map. The new map feature is then given by

$$\hat{\mathbf{x}}^{N+1} = \mathbf{t}(\hat{\mathbf{x}}, \mathbf{z}^l) \tag{6.14}$$

and the state vector can be augmented as

$$\hat{\mathbf{x}} := \begin{pmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{x}}^{N+1} \end{pmatrix}. \tag{6.15}$$

The covariance matrix of the new feature is, using a first order approximation, given by

$$P^{N+1N+1} = T_{x^r} P^{rr} (T_{x^r})^T + T_{z^l} R^l (T_{z^l})^T, \tag{6.16}$$

where T_{x^r} and T_{z^l} are the Jacobians of $\mathbf{t}$. This equation makes the difference between mapping, where the pose is assumed known, and SLAM explicit. The uncertainty in the robot pose estimate propagates directly to the uncertainty of the new map feature. This coupling between the robot pose and the new map feature is however not what makes SLAM so computationally intensive. This comes instead from the correlation between the robot pose and features, i.e. the off-diagonal elements of P. The correlations between the new feature and the robot pose and the old features are given by

$$P^{N+1i} = T_{x^r} P^{ri}, \; i = 0, 1, \ldots, N, \tag{6.17}$$

where once again the robot state is given index $i = 0$.

6.3 Algorithm

There are two fundamental problems with the approach presented above. One is the computational complexity and the other is the data association. Even if the spareness of H is exploited the algorithm is still $\mathcal{O}(MN^2)$. In a large enough environment this is a problem. The limit in the number of features that can be handled is highly dependent on the environment and the feature types being used. As has been the case in all chapters so far, the data association problem must be solved for successful executions of the algorithms. In the context of SLAM, it is crucial to find correct correspondences between the measurements and the map features. In the previous chapters an error in the data associations caused a loss of track. In SLAM however, the problem is more severe. An error in the data association cannot only lead to errors in estimating the robot pose, it can destroy the entire map.

6.3.1 Data Association

Data association is performed with validation gates in innovations space (see Section 2.2.5) and a nearest neighbor strategy. The innovation covariance matrix, S_k, in (2.22) must now explicitly account for the uncertainty of the map feature as well, and is given by (Feder 1999)

$$S_{i,j} = H_{x^i} \begin{pmatrix} P^{rr} & P^{ri} \\ P^{ir} & P^{ii} \end{pmatrix} (H_{x^i})^T + R^l. \tag{6.18}$$

6.3.2 Adding Features to the Map

All unmatched features are added to the state vector directly, but only as unconfirmed features. Correspondences between measurements and unconfirmed features are not exploited to update the state estimate. To reach the status of confirmed, a feature has to be supported by new observations within a certain time window. In this way the number of features do not grow too large, but new features still benefit from other observations. When traveling in a new area there is typically a set of unconfirmed features out of which some become confirmed and some are rejected. It is only by re-observing features that the uncertainty can be reduced. Constantly augmenting the map without ever re-observing features does not lead to a decrease in uncertainty. Determining when a feature is confirmed is thus of critical importance. Therefore there is a tight coupling been the ability to re-observe features and the way the robot can move when building a map.

6.3.3 Reducing the Number of Features in the Map

Reducing the number of features in the map is important. It is of interest, not only from a computational point of view, but also because of the underlying concept of the minimalistic model. As described above, features are deleted from the map when they are not confirmed fast enough. Another way to reduce the number of features is to remove features that are too close to some other feature. Too close is defined by the Mahalanobis distance just like when matching measurements to features. The error between two features is given by

$$\nu_{i,j} = \mathbf{x}^i - \mathbf{x}^j = H \begin{pmatrix} \mathbf{x}^i \\ \mathbf{x}^j \end{pmatrix}, \tag{6.19}$$

where H is

$$H = (I \quad -I)\,. \tag{6.20}$$

The error covariance in (2.22) is given by

$$S_{i,j} = H \begin{pmatrix} P^{ii} & P^{ij} \\ P^{ji} & P^{jj} \end{pmatrix} H^T = P^{ii} - P^{ij} - P^{ji} + P^{jj}. \tag{6.21}$$

6.3.4 Dividing the Map Into Submaps

The Decoupled Stochastic Mapping (DSM) technique from (Feder 1999, Leonard & Feder 1999) is an approximation technique where instead of creating one map covering the entire environment, local submaps are used. All submaps share a common coordinate system. Each submap has its own estimate of the robot pose and keeps a full covariance matrix.

The state of a submap is frozen as soon as the robot leaves the submap. Hence, the robot pose and the state of the features is only updated in the current submap, making the technique scale to larger environments. By limiting the size of the submaps the computational demand can be upper bounded, but the memory required still increases with the size of the environment. However, all but the current submap can be stored on disk.

An important part of DSM is how to move between the submaps and still guarantee a consistent estimate, i.e. that the estimation error covariance matrix are not underestimated. Three different cases of submap traversal is distinguished between:

1. moving from an existing submap to a new submap,
2. moving between two existing submaps from a newer submap to an older submap and
3. moving from an older existing submap to a newer existing submap.

Creating a New Submap

Initializing a new submap is trivial, however, when to do it is a more subtle question. Since there is a global coordinate system, the robot pose from the old submap is simply entered as the first state in the state vector of the new submap. The covariance of the pose estimate is also copied into the covariance matrix of the new submap. Let A be the old submap and B the new. The initialization of B is then given by

$$ {}^{B}\mathbf{x} = {}^{A}\mathrm{x}^{r}, \tag{6.22} $$

$$ {}^{B}\mathrm{P} = {}^{A}\mathrm{P}^{rr}. \tag{6.23} $$

Cross-Map Vehicle Relocation

If the robot leaves submap B for submap A it needs to relocate in that submap. Switching to using the old robot pose estimate from submap A is clearly not possible as it still contains the last estimate of the robot pose. There is no new information about the features in submap A, and so these feature state estimates are still the same. Since the submaps have a common coordinate system the pose estimate from submap B can be moved into submap A. It is, however, important to make sure that the state estimate remains consistent. This is achieved by the so-called cross-map vehicle relocation step,

$$^A\mathbf{x} := \begin{pmatrix} {}^B\mathbf{x}^r \\ {}^A\mathbf{x}^1 \\ \vdots \\ {}^A\mathbf{x}^{N_A} \end{pmatrix}, \tag{6.24}$$

$$^A\mathrm{P} := \begin{pmatrix} {}^A\mathrm{P}^{rr} + {}^B\mathrm{P}^{rr} & {}^A\mathrm{P}^{r1} & \dots & {}^A\mathrm{P}^{rN_A} \\ {}^A\mathrm{P}^{1r} & {}^A\mathrm{P}^{11} & \dots & {}^A\mathrm{P}^{1N_A} \\ \vdots & \vdots & \ddots & \vdots \\ {}^A\mathrm{P}^{N_Ar} & {}^A\mathrm{P}^{N_A1} & \dots & {}^A\mathrm{P}^{N_AN_A} \end{pmatrix}, \tag{6.25}$$

where the robot pose uncertainty of the old submap A is increased by the uncertainty from submap B.

Cross-Map Vehicle Update

Uncertainty propagates from submap to submap through (6.23) and (6.25). Unless there is some scheme to also let newer submaps benefit from the lower uncertainties in previous submaps, the global uncertainty can never be decreased in a submap once it is created. This is applicable when again moving back from submap A to submap B. Submap A was created before submap B and as such has a smaller global uncertainty. The so-called cross-map vehicle update step can here be applied. It is a two step process where the robot pose estimate from submap A is used as a measurement of the robot pose in submap B. Due to the correlation between the robot pose and the features in a submap such strategy cannot be applied directly as it incorrectly reduces the uncertainty of the features. For this purpose a de-correlation step is added before the update is performed.

Step 1: De-correlation In the de-correlation step the old information about the robot pose in submap B is removed by replacing it with a random pose within the submap and inflating the robot pose covariance. To counteract the incorrect reduction in feature uncertainty caused by the update, the feature uncertainties is also increased (see (Feder 1999) for a justification).

$$
{}^{B}\mathbf{x} := \begin{pmatrix} {}^{B}\phi \\ {}^{B}\mathbf{x}^{1} \\ \vdots \\ {}^{B}\mathbf{x}^{N}_{B} \end{pmatrix} \tag{6.26}
$$

$$
{}^{B}\mathrm{P} := \begin{pmatrix} {}^{B}\mathrm{P}^{rr} + {}^{B}\Phi & {}^{B}\mathrm{P}^{r1} & \dots & {}^{B}\mathrm{P}^{rN_B} \\ {}^{B}\mathrm{P}^{1r} & {}^{B}\mathrm{P}^{11} & \dots & {}^{B}\mathrm{P}^{1N_B} \\ \vdots & \vdots & \ddots & \vdots \\ {}^{B}\mathrm{P}^{N_B r} & {}^{B}\mathrm{P}^{N_B 1} & \dots & {}^{B}\mathrm{P}^{N_B N_B} \end{pmatrix} \tag{6.27}
$$

Here ${}^{B}\phi$ is a random pose in submap B and ${}^{B}\Phi$ is a covariance matrix corresponding to a level of uncertainty that surpasses the size of the submap.

Step 2: Update In the second step the robot pose from submap A is used as a measurement in submap B. The measurement equation corresponding to measuring the robot pose, i.e.

$$
\mathbf{z} = H\, {}^{B}\mathbf{x} + \mathbf{w} = {}^{B}\mathbf{x}^{r} + \mathbf{w} \tag{6.28}
$$

where $\mathbf{z} = {}^{A}\mathbf{x}^{r}$, $Q = E[\mathbf{w}\mathbf{w}^{T}] = {}^{A}\mathbf{P}^{rr}$ and $H = [I_{33} \;\; 0]$. Updating the state vector ${}^{B}\mathbf{x}$ and error covariance ${}^{B}\mathbf{P}$ is then performed using the standard EKF update formulas (2.13).

6.4 Experiments

In this section an evaluation is made of the standard SLAM algorithm versus H-SLAM with the two different approaches for allocating submaps, square grids and room based. An evaluation is also presented for using the extra degree of freedom that a pan-tilt unit provides to detect lines which otherwise are hard to detect. A minimum length of 1 m is used for the line features in the experiments.

6.4.1 A Small Illustration

First a small example is given to show how the submaps are allocated and how the features are distributed between the submaps. This comparison is based on data collected when driving a Nomad200 between the living-room and one of the nearby offices, back and forth two times. Figure 6.3 shows the resulting map, containing 54 line features when using standard SLAM. The trajectory followed by the robot is also given. By comparing the map with the real environment, errors in the data associations are clearly visible. For example, in the lower

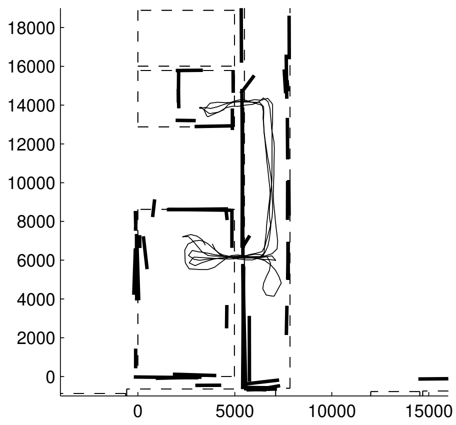


Figure 6.3: *A small scale SLAM experiment with 54 features. The robot drives back and forth between the living-room and a nearby office twice. Errors in the data associations are visible in the lower part of the corridor and the inner wall of the living-room where there are many lines close together where there should be only one line.*

corner of the corridor there are several lines where there should be only two perpendicular ones. The same is true for the inner wall in the living-room. There are no time stamps on the laser sensor data, i.e. registration between laser data and odometric information is difficult. The low bandwidth over the serial connection with the laser sensor gives a transfer time of approximately 0.2-0.3 s for each scan. The robot is not running a real-time operating system and the code is not written to guarantee fixed process scheduling. The errors in the registration can therefore be considerable in some cases. Some of these issues can be addressed by using more advanced verification schemes.

Based on the same data, the results of the H-SLAM algorithm with square grids are shown in Figure 6.4. In total, the method results in 74 features. Because of the the way the submaps are allocated there is a considerable feature overlap between the submaps. The individual submaps are shown to the right in Figure 6.4. The dotted lines mark the borders of the submaps. This example is a worst case scenario for the grid strategy as the grid-maps are allocated with 45° to the environment. In the current implementation the submaps are aligned with the odometric coordinate system when the map is initialized. To get the best out of a square grid approach, the orientation of the grids must be chosen wisely, which is hard given that the task is to map an unknown environment.

Figure 6.5 shows the result of yet again using the same data, but now with the room based submap allocation strategy. The gateways between the submaps are defined by the user in the current implementation. The submaps are now more intuitively distributed. The total number of features is 65 and there are three submaps, one for each room the robot visits. The difference in the number of features is not so large, but the alignment problem is addressed automatically by having one submap per room.

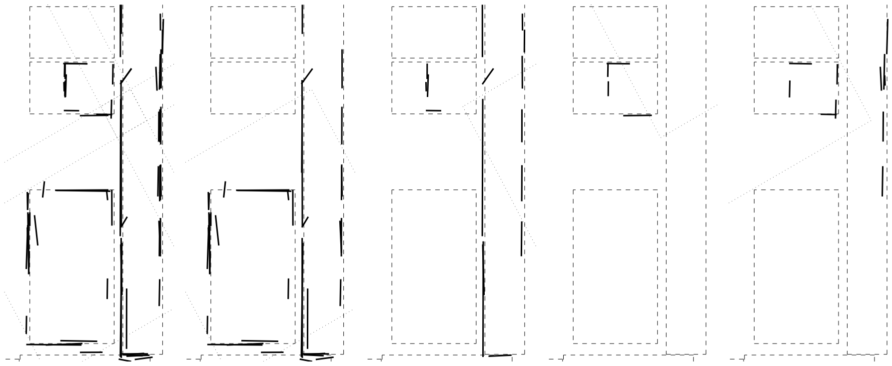


Figure 6.4: *Small example of H-SLAM with square grid submaps. The leftmost figure shows all submaps overlaid. The dotted lines mark the border of the submaps. The four other figures show the features from the different submaps. There are 74 features in total with 42, 17, 4 and 11, respectively in the four submaps.*

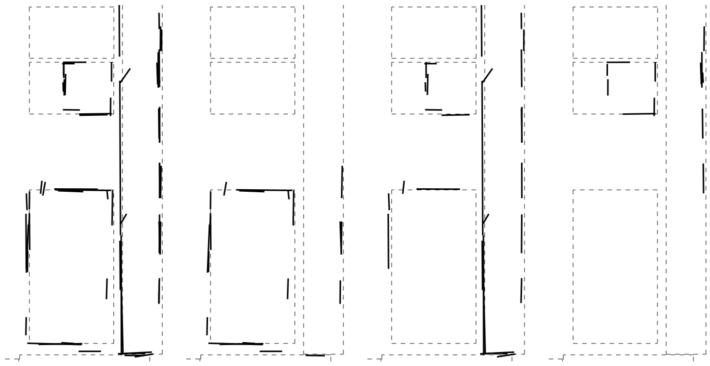


Figure 6.5: *Small example of H-SLAM with room based submaps. The leftmost figure shows all submaps overlaid with 65 features in total. The three figures on the right show the different submaps with 23, 30 and 12 features, respectively.*

6.4.2 Lower Floor at CAS

The computational complexity becomes noticeable when the environment is larger. The total computation time in the three cases in the previous example was 41 s, 34 s and 30 s, respectively. To test the ability to build a map of a larger environment, the lower floor at CAS is tested using a Nomad200 robot, controlled with a joystick. The robot travels a total distance of 388 m over a period of 44 minutes (average speed of 0.14 m/s).

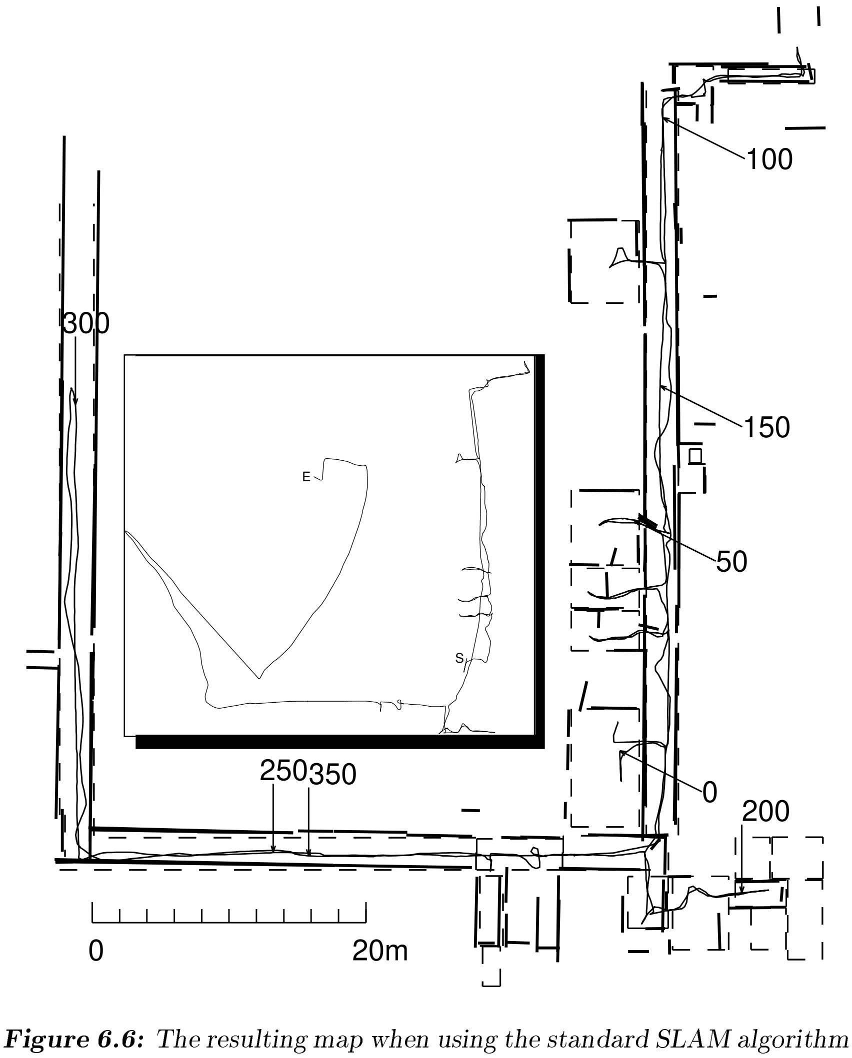


Figure 6.6: *The resulting map when using the standard SLAM algorithm. The arrows with number mark distance traveled along the trajectory. The raw odometric data is also shown (inner plot).*

The map that results when applying the standard SLAM algorithm is shown in Figure 6.6. The robot trajectory is also shown along with indications of distance traveled every 50 meters. The raw odometric pose estimate is also shown in the inner plot. The strong line support in a corridor makes the corridors become almost straight. That they are not perfectly straight is because the robot is not able to associate all small line pieces from a corridor wall to one line. There are some errors in the length of the corridors, but that is to be expected as there is little information in that direction. The map consists of 114 lines and the total computational time to build it is 340 s. This is only 12% of the total time of the experiment, but due to the complexity it becomes increasingly difficult to maintain real-time performance when the number of features increases. It is when the robot is translating and rotating fast that the problems manifest themselves as these are the cases when fast updating is most needed. In a real application this means that the robot has to discard valuable information while the old is being processed.

Figure 6.7 shows the map that resulted when using square submaps, 20 by 20 meters in size, with a 2 m overlap. The corridor outside the living-room is now somewhat more bent as it is divided into four submaps. In the middle of the corridor there is a large collection of lines from three different submaps that corresponds to the same physical wall. Information is only shared between the submaps when the robot travels between them. This means that when the robot travels along the corridor for the first time, an empty submap is created three times. All prior information about features in the corridor is left behind in the previous submap. This effect is even more clear in the leftmost corridor where a new submap is allocated almost exactly at the time when the odometry starts to drift much more rapidly (compare inner plot of Figure 6.6). The robot is able to find its way back to the living-room, but the quality of the map suffers. The total computation time for this map is 185 s and the total number of features is 166. These features are divided as 43, 17, 26, 18, 11, 19, 16, 7 and 9 between the submaps. The reduction in total computation time is 50% compared to standard SLAM. More importantly, no more than 43 features are processed at the same time which allows for some margin to the real-time threshold. By aligning the square grids with the dominant wall directions of the environment (see Figure 6.8), better results are achieved. The leftmost corridor still shows clear signs of the submap intersection, but it is not as severe as before. However, the topmost area that was previously well mapped now suffers the effects of being in the intersection between three submaps. This example illustrates the problems that are caused by using a fixed submap strategy.

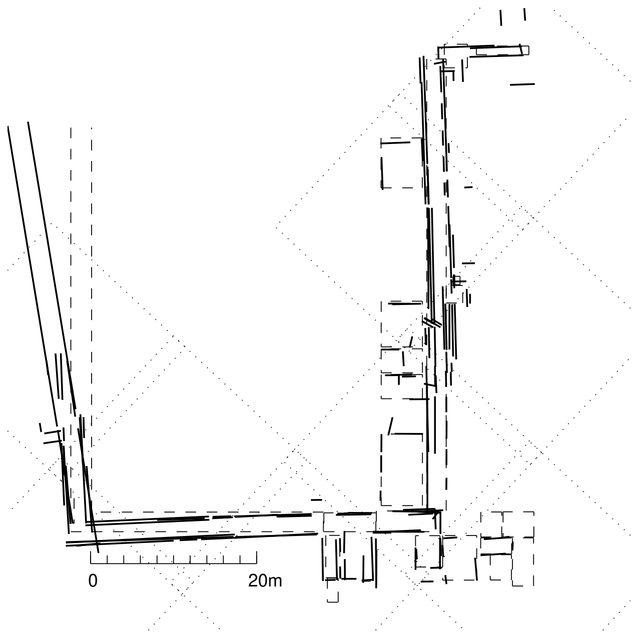


Figure 6.7: *H-SLAM with square grids on the lower floor at CAS.*

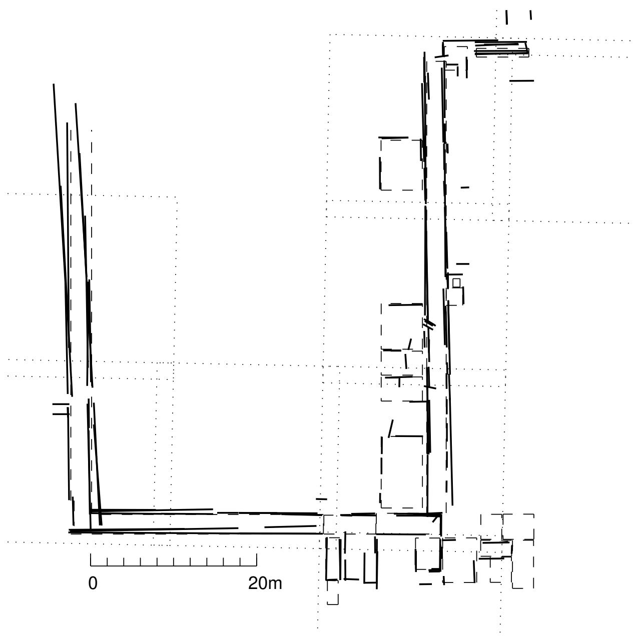


Figure 6.8: *H-SLAM with aligned square grids on the lower floor at CAS.*

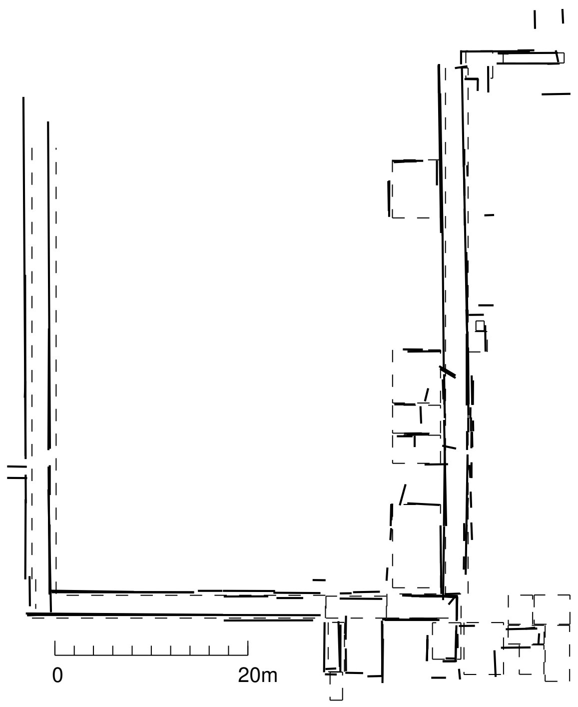


Figure 6.9: *H-SLAM with room based submaps on the lower floor at CAS.*

Figure 6.9 shows the resulting map when assigning one map to each room. Also in this case the corridor is somewhat bent. If the robot had traveled all the way up the corridor before entering the offices on the sides, the effect would not have been as obvious as the robot would stay within the same submap. The horizontal corridor in the Figure 6.9 is split by two doors and is hence divided into three different submaps. Effects of this are clearly visible in the center of the corridor where the corridor lines from the different submaps do not quite line up. The total computation time to build the map is 182 seconds 166 feature are used in total, divided as 7, 38, 6, 7, 15, 14, 8, 10, 18 and 28.

Figure 6.10 shows the estimated standard deviations for the position estimation errors in the x and y directions. The angular uncertainty results in large uncertainty in the x direction when the robot travels up the corridor (x is perpendicular to the corridor). The uncertainty in the y direction is reduced by detecting lines in the rooms on the side of the corridor. Notice how the uncertainty drops again when the robot travels back to the area outside the living-room, close to where it started. The uncertainty reaches its maximum at the top of the leftmost corridor. The H-SLAM algorithms mostly give a larger uncertainty than the standard SLAM algorithm, as is to be expected.

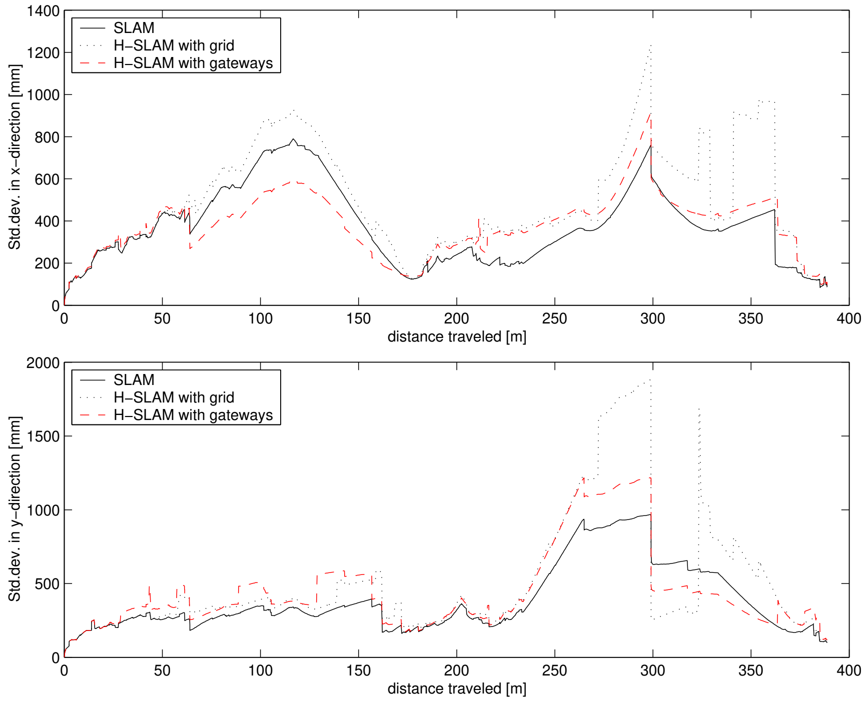


Figure 6.10: *The uncertainty in x and y direction for SLAM, H-SLAM with grids and H-SLAM with room based submaps. Notice how the uncertainty decreases when the robot reaches a position it has already been at. When traveling along the corridor to the left and then up the uncertainty reaches its maximum.*

There are instances when the estimated uncertainty is smaller though. One such example is when the robot comes out of the room above the two offices in the main corridor. This could be an indication of an inconsistency.

6.4.3 The Atrium in the Main Building

The lower floor at CAS does not contain any loops. It is when the robot travels in a loop that SLAM is really put to the test. For successful handling of such loops the robot must be able to associate new measurements to map features introduced in the map when the loop was started. For this purpose the data that was gathered during the demonstration described in Section 3.4.4 in the area around the atrium in the main building is once again used. Here a Nomadic SuperScout robot moves autonomously while giving guided tours.

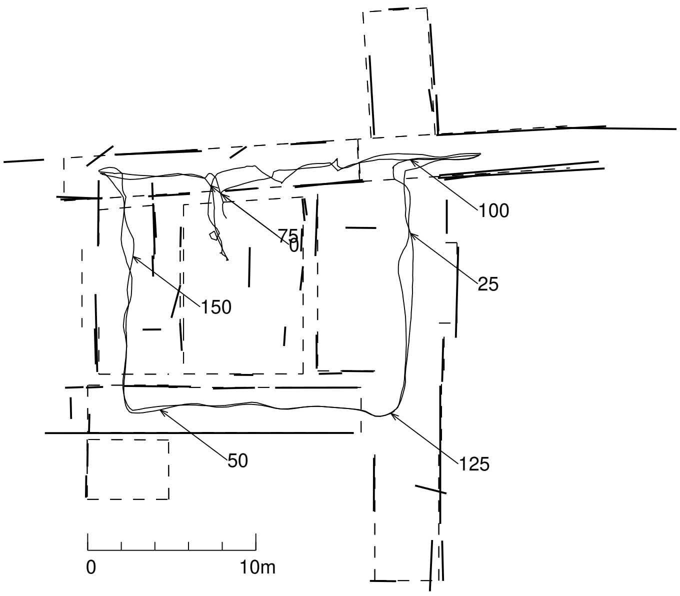


Figure 6.11: *The map built using standard SLAM. The true trajectory followed by the robot is also shown in the figure. Every 25 m is marked with an arrow to indicate the direction of motion. The loop around the atrium is successfully closed and the resulting map match the handmade map (dashed lines) quite well. 67 line features are used in total.*

There are many people around the robot and it is therefore not able to detect lines at long distances. The total distance traveled is 165 m and the total time is close to 22 minutes (average speed is 0.12 m/s).

Figure 6.11 shows the map built using the standard SLAM algorithm. Also shown in this figure is the robot trajectory during the run with marks every 25 m. The total number of lines is 67 and the total computation time is 75 s. The robot closes the outer loop in the area twice. When traveling into new areas on the first lap, the uncertainty grows steadily. When the robot revisits the area where it started it is able to associate the measurements with already stored map features. When this happens the low uncertainty associated with the initial features in the map is propagated back through the map by means of the correlation terms in the covariance matrix. Had the robot not been able to solve the association problem it would add new features to the map and the uncertainty would keep growing. Also in this map there are clear signs of errors in the data associations, for example, in the lower right corner. Three door leafs have also been added to the map, one in the lower right corner, and two in the upper left. Since the door leaf are highly dynamic they are undesirable features in the map, but in the current implementation this problem is not addressed.

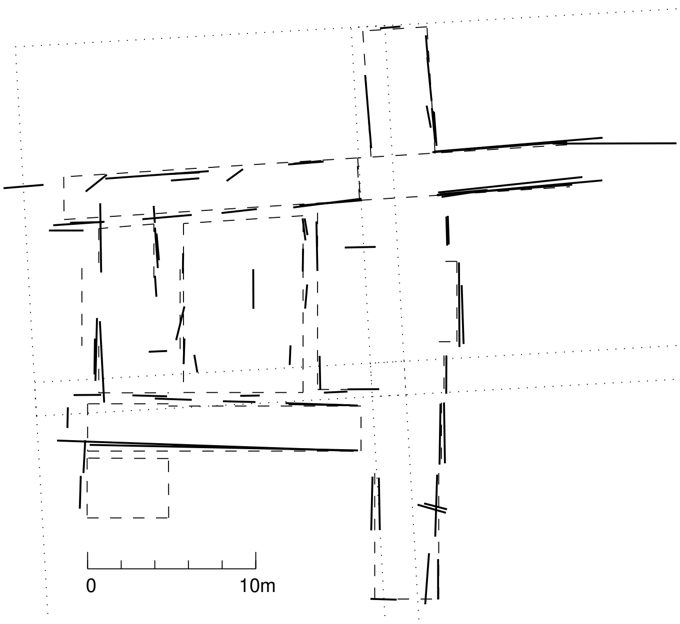


Figure 6.12: *The map resulting from H-SLAM with square grids. The map contains 82 (41, 20, 9 and 12) line features to compare with 67 for standard SLAM. The loop is successfully closed twice and the resulting map matches the handmade map (dashed lines) quite well.*

Figures 6.12 shows the resulting map for the square grid based H-SLAM approach. The environment is smaller than at CAS and so four 20 by 20 meters square grid submaps are enough. In this small environment, one submap can cover the entire robot trajectory, if the localization of it is chosen carefully. The map contains 82 line features in total, divided as 41, 20, 9 and 12 on the different submaps. The total computation time is 58 s to compare with 75 s for standard SLAM. Since the environment is smaller, the cost for feature extraction is still dominant in the total amount of computations that are spent. Comparing the generated map with the handmade one (dashed lines) shows them to be quite consistent. The only significant discrepancy is in the total vertical distance on the left side which is underestimated. Just like when mapping the lower floor at CAS, features from the different submaps do not match perfectly. However, the loop is successfully closed which is the most important part of the test.

The result of the second approach to H-SLAM, with room based submaps is shown in Figure 6.13. In total, 85 features are used and the computational time is 54 s. The features are distributed among six different submaps as 3, 33, 11, 18, 12 and 8. The somewhat smaller total computation time is most likely explained by the smaller submaps where the largest one contains 33 compared to 41 for the square grid map. The loop is also this time closed successfully. However, notice how the wall in the upper left room is modeled with many line features. The reason for this is that this wall can be seen clearly from three of the submaps. Besides from this, the room based submap technique give a more consistent map of the environment where features which are represented in several submaps are closer together.

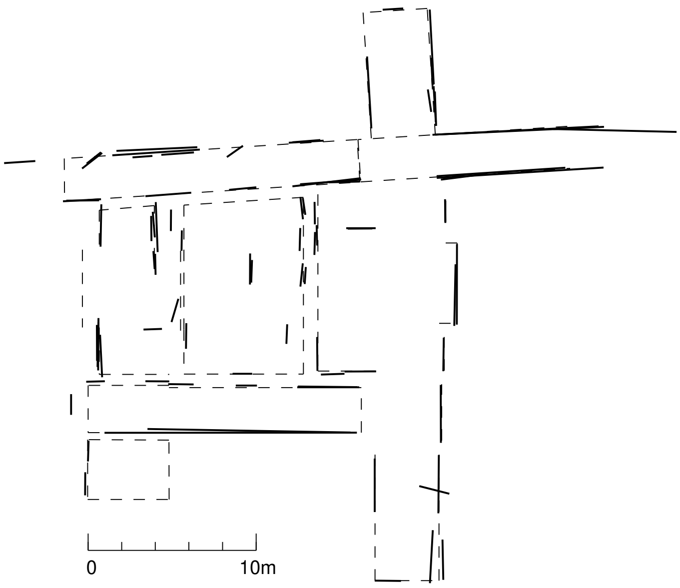


Figure 6.13: *The map resulting for H-SLAM with room submaps. The map consists of six submaps with 3, 33, 11, 18, 12 and 8 features, respectively, which makes 85 features in total. The map is consistent with the handmade map except for the two lines in the lower right corner. Errors in the data association are also visible in the upper left corner where several lines are close together.*

The estimated standard deviation of the position estimation error is shown in Figure 6.14. Until the loop is closed for the first time the uncertainty increases. Upon closing the loop there is an almost instantaneous drop in the uncertainty down to a level that is close to the initial one. The uncertainty in the x direction is reduced much later for the room based H-SLAM algorithm. This is the result of the submap division. There are simply no features with the right orientation in the corresponding submap.

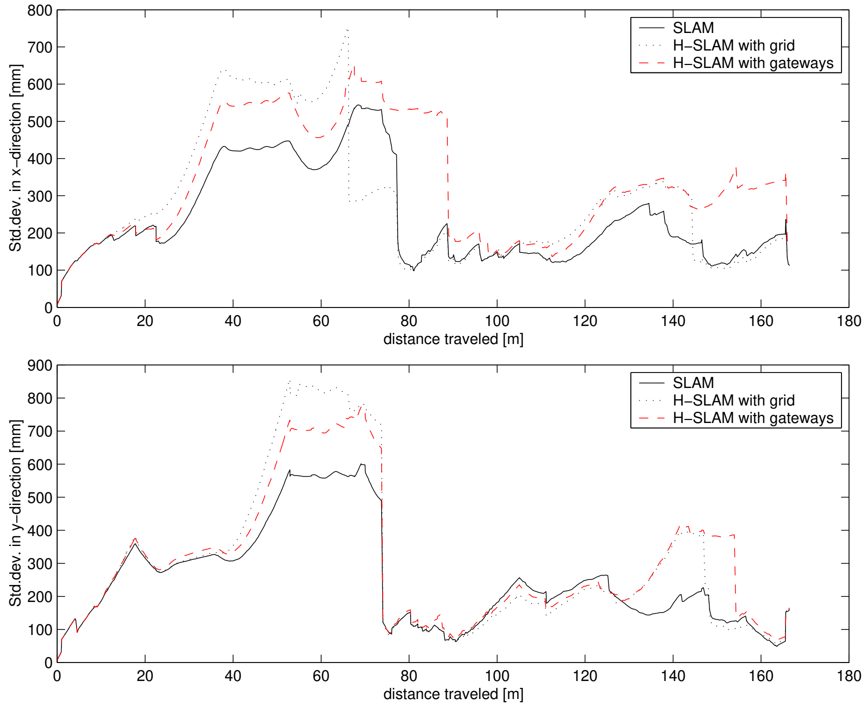


Figure 6.14: *The uncertainty in the x and y directions. The uncertainty grows almost monotonically until the first loop is closed. When this happens the uncertainty is reduced significantly. The uncertainty when traveling around the second loop is smaller because a map already exists. For every additional loop the map is improved.*

6.4.4 Active H-SLAM: "Turner & Hooch"

The experiments from Chapters 3-5 give strong indications that pose tracking and global localization can be performed using a minimalistic model of the environment. However, depending on the height of the sensor and the position of the robot, the walls can be heavily occluded, which makes minimalistic mapping difficult. This is reflected by the sparse submaps of rooms in e.g. Figure 6.5. Rooms are typically more cluttered than corridors and thus it is harder to detect lines. For this reason it is of interest to investigate methods that actively seek to improve the map in regions where it is sparse.

The target for the methods developed in the thesis is a service robot application. Although the robot must have the ability to generate a map representation from sensor data, it is not clear that it has to do so entirely autonomously. In fact a certain degree of interaction between the user and the robot might be necessary. The robot has to be taught the names of locations that the user wants the robot to be able to go to, i.e. there is a need for a common language. In this section a semi-automatic mapping method is investigated. The user presents the robot with the different areas and gives them names that can later be used for referencing. This strategy is inspired by the movie "Turner & Hooch" in which Tom Hanks' character shows a dog all the rooms of his house and instructs the dog about whether it is allowed in there or not. A typical scenario with a robot might look like:

1. User: "Robot, this is the living-room, start mapping it and let me know when you are done."
2. The robot explores the room until it is satisfied with the map.
3. Robot: "The living-room is now mapped. Guide me to the next area".
4. The user guides the robot to the door leading out in the corridor.
5. User: "Robot, you can now enter the corridor and start mapping it. Let me know when you are done here"
6. and so on ...

When a room is mapped the robot can also be shown important positions within the room, e.g. a sofa or a bookshelf. By guiding the robot to a position there is no need for the user to know anything about coordinate systems, representations, etc.

Mapping Objective

In the minimalistic framework, the objective is to build a model of each area which only captures the large scale structures. If two line features in the map can be verified to correspond to the same wall in the environment the number of map features can be reduced by one. This saves computations and brings the map closer to a minimalistic description of the environment.

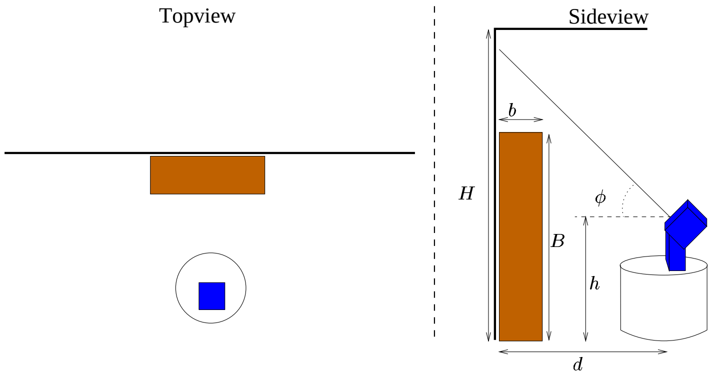


Figure 6.15: *What is occluded at sensor height is often free higher up. By mounting the laser sensor at a pan-tilt unit this can be utilized and the two line segments found on either side of the cupboard can be joined.*

Strategy

The strategy taken here for mapping a room can be divided into two steps, *hypotheses generation* and *hypotheses verification*, where hypothesis refers to a hypothesis about the existence of a wall. Let such a hypothesis be denoted by $W^{(i)}$.

As the name implies the hypotheses generation step aims at forming hypotheses about where the large scale line structures are. A brute force solution is to let the robot move around at random and extract lines, which are then combined to form wall hypotheses. However, there are more structured procedures. In (Hinkel & Knieriemen 1988) the so-called *angle histogram* is used to find the most likely wall directions. Another voting based technique is the Range Weighted Hough Transform (RWHT) (Forsberg, Åhman & Wernersson 1993) applied in Chapter 3. The disadvantage of the angle histogram is that it assumes right angles between the walls, whereas the RWHT does not. The downside of the Hough Transform is that it is computationally expensive, but this is of little concern as the hypotheses generation is a bootstrapping step and the Hough Transform is therefore used.

In the hypotheses verification step the robot attempts to actively verify the different wall hypotheses by positioning itself at a favorable position and directing the sensors towards the hypothesized wall. Most platforms are limited to turning the sensors in different directions in the plane. With this limitation concluding that two lines in fact belong to the same wall is difficult. If, however, the sensor can be tilted, verification becomes easier. Typically the walls are much less occluded higher up. By mounting a laser scanner on a pan-tilt unit it is possible to analyze the wall at different heights (see Figure 6.15).

When all wall hypotheses, $W^{(i)}$, have either been verified or rejected the map is evaluated to make a decision if it complete or not. When the map is found not to be complete further exploration is needed, otherwise the robot robots back to the user that the room is mapped.

Figure 6.16: *The result of combining two horizontal laser scans to give a 360° view of the living-room. Notice how the left and the lower walls are almost completely occluded when using a Scout robot with the sensor at 0.5 m above ground.*

Hypothesis Generation

Figure 6.16 shows an example of what the living-room looks like when combining two scans of the horizontal plane, using a SuperScout robot with the sensor at approximately 0.5 m above the floor. The left and the lower walls are here almost completely occluded, whereas the right and the upper wall create two strong wall hypotheses using the RWHT. By comparing this scene with the maps of the living-room acquired by the Nomad200 robot (see e.g. Figure 6.5), it is clear that it is much more difficult to map a room when the sensor is as low as on the SuperScout robot.

In most areas, the intersection points between the walls and the ceiling are easy to detect, and can thus be used to find the walls. Detecting these points can be achieved by mounting the laser scanner on a pan-tilt unit. Wall hypotheses can then be generated from the estimated intersection points by only considering the xy-coordinates and applying the RWHT. Figure 6.17 shows the result of tilting the laser sensor so that it scans vertically. The left subfigure shows the scan in Cartesian coordinates and the right one in polar coordinates. Clearly, in this case, the intersections between the walls and the ceiling are given by the two peaks in the polar plot. Figure 6.18 shows the location and configuration of the laser sensor with respect to the room. If the sensor is rotated to scan the ceiling the data shown in Figure 6.19 can be acquired. The estimated intersection points between the walls and the ceiling are marked as dark dots. Figure 6.20 shows the intersection points projected to the horizontal plane. Comparing this data with the data acquired at sensor height (see Figure 6.16) confirms the hypothesis that the walls are more clearly visible at ceiling height.

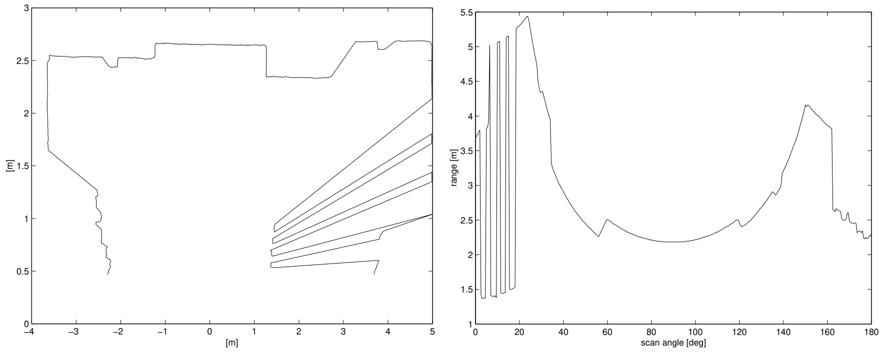


Figure 6.17: *A vertical laser scan in Cartesian (left) and polar (right) coordinates. The intersections between the ceiling and the walls are given by the two peaks in the polar plot. The negative direction is east in the Cartesian coordinate system.*

Figure 6.18: *The scene that resulted in Figure 6.17. Note how the sensor is tilted 90 backwards to give a view of the ceiling. The left photo shows the wall facing east and the right photo the west wall.*

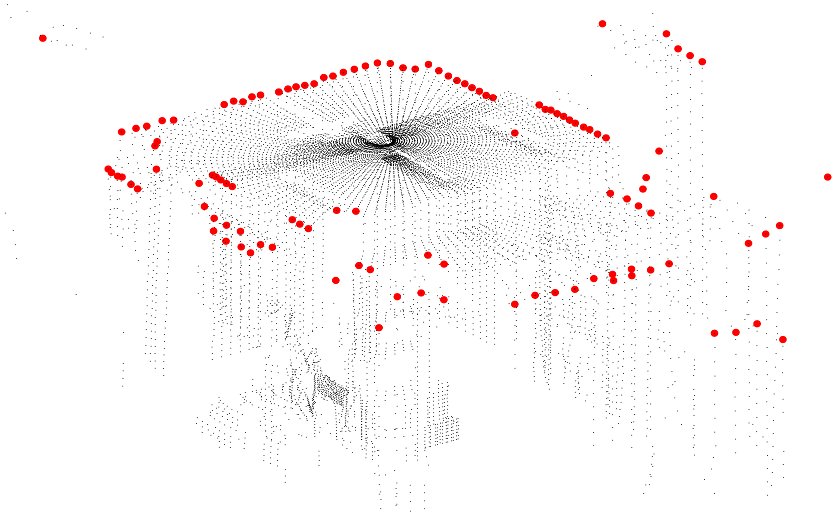


Figure 6.19: *The result of scanning with the sensor tilted 90° to give vertical scans. The estimated intersection points between the walls and the ceilings are marked with dark dots.*

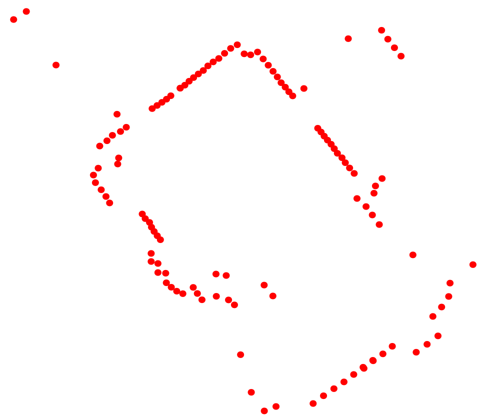


Figure 6.20: *The estimated intersection points projected on the xy-plane. The walls are now easier to detect than in a horizontal scan.*

Hypotheses Verification

To get a good view of a potential wall given by $W^{(i)}$, the robot positions itself in the middle and at a distance d from $W^{(i)}$ (see Figure 6.15). When choosing d there are two competing issues. The risk of something occluding the sensor increases with d, but the larger d is the easier it is to get a good view above an object placed in front of the wall, .e.g. a bookshelf. The distance d is chosen as a compromise where the width, b, and height, B, of typical objects are assumed to be less than 0.4 m and 2 m respectively. The ceiling height, H, is larger than 2.3 m and the height of the sensor above ground is 0.5 m. The condition to barely see the wall above an object can then be calculated as

$$\frac{d}{H-h} = \frac{b}{H-B} \Rightarrow d \approx 2 \text{ m}. \tag{6.29}$$

At this distance from a wall hypothesis, $W^{(i)}$, the pan-tilt unit is used to scan perpendicular to $W^{(i)}$ from ceiling height to ground level. The merging of line segments is left to the SLAM algorithm through the mechanism described in Section 6.3.3. If the robot fails to move to a desired vantage point it tries to scan the wall anyway from the position where the motion failure was reported.

A wall hypothesis, $W^{(i)}$, is assumed to be verified if all map lines that are close to $W^{(i)}$ add up to a length that is more than 30% of the length of $W^{(i)}$. When this condition is not met $W^{(i)}$ is rejected and assumed to be the result of for example structured clutter.

While moving between the different vantage-points lines are constantly extracted to provide input to the SLAM algorithm to reduce the position error and include all visible lines in the map.

Mapping the Living-Room

To test the above scheme an experiment is performed in the living-room, which is one of the rooms that was previously sparely modeled. This time the living-room is mapped using a SuperScout robot with the laser sensor approximately 0.5 m above ground, to compare with the Nomad200 where the laser is 0.9 m above ground. The map that results is shown in the left subfigure of Figure 6.21. Even if the SuperScout robot is driven into every corner of the room providing the best possible view of the different walls, the map is even sparser than, for example, the map in Figure 6.5. That this is the case can be understood by studying Figure 6.22 which shows photos from the room when the model was acquired. The inner wall in the living-room is almost completely occluded by sofas, a table, chairs, a bookshelf and a large robot. The line in the middle of the room comes from the table. The door leaf of the door out into the corridor is also included in the map. This is undesirable as this feature is not stable over time and methods for classifying it as such are needed. The line in the lower corner of the room comes from the niche that is visible in the right photo in Figure 3.19.

When the active scheme is used, and scans are taken at different heights the model becomes more complete (right subfigure of Figure 6.21). Due to the amount of clutter in front of the inner wall the robot is not able to verify that the to line segments in the middle belongs to the same wall. On the outer wall the robot has successfully joined the lines detected on either side of the door opening by using data from high up on the wall. Notice also how one of the door leafs have been included in this map as well.

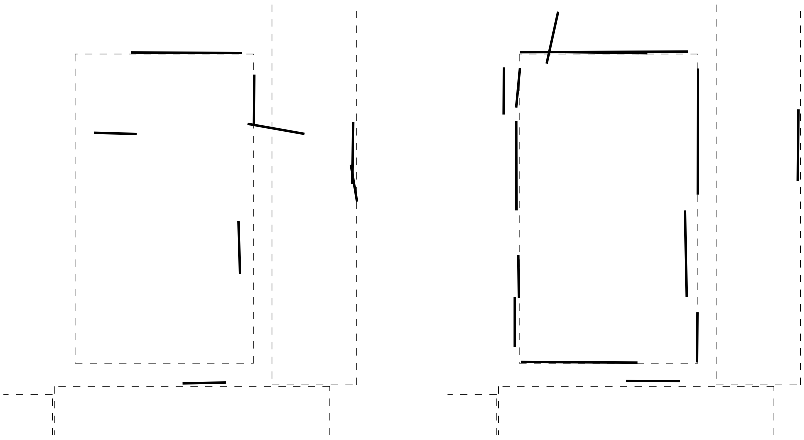


Figure 6.21: *Left: Model build when only searching for lines in the horizontal plane. Right: Active H-SLAM.*

Figure 6.22: *Photos from the living-room when map was built.*

6.5 Summary

In this chapter the SLAM problem is studied. An hierarchical approach, H-SLAM, is proposed to overcome the map scaling issue that otherwise inevitably stops SLAM from being applied in large environments. The Decoupled Stochastic Mapping (DSM) approach presented in (Feder 1999) is used as the basis for the hierarchical approach, where the environment is divided into local submaps sharing a common coordinate system. It is shown that the original submap allocation strategy sometimes leads to poor performance as it is impossible to control where the intersections between the submaps end up. Instead a room based strategy is proposed to give a more accurate description of the individual rooms, if not the entire environment. A comparison is presented both between SLAM and H-SLAM and between the two submap allocation strategies.

None of the maps from the two environments are perfect. There is plenty of room for improvements. Improving the data associations results in fewer map features. It is also important to further investigate the consistency of the DSM algorithm that H-SLAM is currently based on. The underestimation of the estimation error that is seen in Figure 6.10 indicates such a problem.

The active H-SLAM approach presented at the end of the experimental section is a first attempt to get a more complete description of the environment. The improvement gained by adding the extra degrees of a pan-tilt unit is clearly established. At the same time it gives one possible scenario for how mapping can be done in cooperation between the robot and the user.

Chapter 7

Summary and Future Research

7.1 Summary

Autonomous mobile robots in the service of mankind are getting closer to becoming a reality. There are already many examples of mobile robots, but none which are truly autonomous. In manufacturing industry, robots have until recently been synonymous with fixed robots or robots that are constrained to, for example, a gantry. In times when every possibility to reduce production costs must be considered, mobile robots are of great interest. They would allow for much faster changes to the production line. The extra flexibility comes at the price of higher complexity. In a highly constrained assembly line type of environment the robot can use its precise mechanics to alleviate many of the problems that researchers in the robotics community have dedicated their lives to. Navigation and localization take other forms in such a setting. By confining the robots to areas where humans are not allowed, the robots do not have to avoid unknown obstacles. Localization is replaced by the problem of reaching well defined positions, typically with an accuracy far below a millimeter.

There are examples of robots that do move around in factory environments, but they are typically guided by bar codes on the walls, magnetic tapes on the floor or active beacons. If a robot could localize itself using the natural environment, there would be large savings in the infrastructure and much more flexible production line could be achieved. If the horizon is widened to include all applications for mobile robots, the benefits are even higher for a system that can localize itself using only information from the natural environment. One such application is a service robot that can perform various tasks in a domestic or office environment.

The work presented in the thesis focuses on the problem of localization in the context of a service robot type of application. The environment is thus assumed to be indoor and semi-structured. The methods that have been presented do not rely on any artificial landmarks, but instead utilize the natural structures of the environment. The working hypothesis throughout the thesis has been the use of minimalistic models. Contrary to many approaches presented in the literature, the philosophy behind the minimalistic models is not to make the most detailed model possible, but rather capture the large scale structures of the environment. Three basic types of features have been used; lines, doors and points. In the minimalistic model of the environment the lines typically correspond to the dominant walls of a room. These kinds of lines assume the use of a sensor with high angular resolution or a sparse, uncluttered environment. The latter is unrealistic for most indoor office or domestic settings and thus a laser scanner has been the primary sensor modality. Doors are also dominant features in most indoor environments and they are natural to model as they act as gateways between different areas/rooms. One single sensor modality is not enough in all situations, for example, glass is transparent to the laser. To complement the laser sensor and the line and door features, point features extracted from sonar data have also been considered.

In Chapter 1 the problem of localization was roughly divided into three subgroups, pose tracking, global localization and map acquisition. This thesis presents contributions to all of these problems. Before pose tracking and global localization can be performed there has to be a map. The map was initially acquired by hand, on a room by room basis. The transformation between the rooms where then estimated. Although the minimalistic modeling approach allows for a relatively simple map construction, it is a tedious job in a large environment. In a service robot application the robot must be able to build it own representation of the environment. Partly because installation would otherwise be too expensive, but also because the environment may change over time.

When building a map without any prior knowledge and without access to the true pose of the robot, localization has to be performed at the same time as the mapping. The process is referred to as simultaneous localization and mapping (SLAM). Most existing approaches to SLAM are based on Kalman filters. While performing well in small environments, the computational complexity is too high for medium or large sized environments. Chapter 6 presented an approach to mapping that allows the user to guide the robot through the environment and introduce different areas that the robot should know about. An area is often a room, but not restricted to such. In most office or domestic environments, each such an area is small enough for SLAM to be directly applicable. The complete map is divided into submaps, each built using a standard SLAM algorithm. The mapping is done in an active way within each submap. The laser sensor is mounted on a pan-tilt unit and can therefore detect lines that would otherwise be difficult to detect. In the initial phase of the exploration the attention is directed towards the height of the ceiling as it typically provides a much less occluded view of the walls of a room. The walls detected in this way are then used to guide the robot to positions from which the walls are likely to be visible. Although being only an initial implementation, the advantages with adding the extra degrees of freedom provided by the pan-tilt unit have clearly been established.

When a map of the environment exists it is possible for the robot to perform global localization and pose tracking. Once again looking at it from an application point of view, the global localization comes before pose tracking. Starting with global uncertainty to find the pose involves finding correspondences between the sensor data and the environmental model. Under normal circumstances it is not possible to solve these correspondences unambiguously unless sensor data is integrated over time.

In a feature based setting, each possible correspondence between a detected feature and a map feature gives one candidate to the pose of the robot. In Chapter 4 hypotheses about the robot pose were created from these possible correspondences. It was assumed that each hypothesis could be modeled using a Gaussian PDF. A Kalman filter was assigned to each hypothesis to track the corresponding pose estimate. Bayesian probability theory was used to update the probabilities of hypotheses being correct, given the evidence provided by the measurements. A computationally efficient implementation was achieved by keeping the number of hypotheses small. To this end the feature types were divided into two groups, supportive and creative. The supportive features cannot create new hypotheses, only provide support for already existing ones. An active exploration strategy utilizing a topological graph description of the environment was used in the experiments. Although being a greedy algorithm, the active exploration was shown to be sufficient in most situations. Experiments were performed in two different types of environments, one old hospital building and a modern office environment. The experimental results showed that the Multiple Hypothesis Localization (MHL) was able to correctly find the pose of the robot in all but a few cases. The failures were the result of poor environmental modeling and the exploration strategy that was not able to successfully locate a creative feature. The work presented in this chapter is the first thorough investigation of the use of a multiple hypothesis approach to localization.

In Chapter 5 the Monte Carlo Localization (MCL) method, that recently have gained a lot of attention, was evaluated in the minimalistic setting. The Monte Carlo methods use a sample set to represent the pose distribution. Any PDF can then be represented assuming that enough samples are used. It was concluded that in a highly symmetric environment with non-distinct features the computational resources needed to provide reasonable performance surpasses what is available on any mobile robot platforms today. The main problem can be traced back to not having sample support at the correct pose. That is, there are no samples close to the correct robot pose. Two different improvements were investigated. In the first, new samples were introduced in each iteration from a uniform distribution. In the second, new samples were introduced similar to the way hypothesis are created in MHL, i.e. samples were added based on possible correspondences between detected feature and map features. It was shown that the first method offered a reduction of the number of samples to roughly half. The second method was shown to offer a even higher reduction and still get significantly improved performance. An evaluation was also presented of the use of different combinations of features.

Once the pose of the robot is found, either using one of the global localization strategies or supplied by the user, it is possible to rely on simpler methods for maintaining the pose estimate. Not having to perform a global search for the correspondences between the map and detected features allows for more efficient implementations. Chapter 3 presents one such pose tracking approach. While the two global localization strategies can handle any feature based representation, the pose tracking algorithm was tailored to the minimalistic model. Only the line feature was used for pose update. The doors where used to determine the visibility conditions between the different rooms. The large scale features gave way for an algorithm that was shown to be operational even in dense clutter. The key to success being the ability to single out the few data points that belong to the modeled lines. Experiments were presented from three different environments and with two different robot platforms. Two more platforms (see Section D.3) have also been successfully tested but these experiments were left out from the presentation due to space limitations.

7.2 Future Research

In this section some of open problems will be outlined:

The correspondence problem, i.e. matching measurements with either previous meaurements or with a model of some sort is still an open question. This problem is also referred to as data association. In this work and in the majority of other work gating techniques have been used for this problem. However, there are many situations, especially when the uncertainty becomes larger, that such approaches are not sufficient. Solving the data association problem would undoubtedly be one of the greatest achievements not only in robotics but in all fields of science that deal with sensor data of uncertain origin.

The problem of simulataneous localization and mapping (SLAM) is still an open problem, especially in a large scale environment. Chapter 6 and other work are only scratching the surface of large scale SLAM. Many of the problems can be traced back to the data association problem.

As seen in Chapter 3 it is of uttermost importance to have mechanisms for fault detection. An autonomous system must be able to detect failures and preferrable counteract them. The sooner the causing events are detected the easier it is to handle them. Examples are slippage when driving over a threshold or when the robot bumps into a person in a dense crowd. The introduction of inertial sensors to the system would greatly contribute to improving the performance in these situations.

The focus has been on estimating the pose without much thought to the mission at hand for the overall system. There are many situations when high accuracy is not needed. Going from one side of a corridor to the other does not justify knowing the pose of the robot with millimeter accuracy. As long as the destination at the other side of the corridor is reach, the task is solved. The underlying philosophy is thus that of task oriented localization, i.e. letting the localization depend on the task at hand.

In connection with task oriented localization, active control of the the direction of attention for various sensors is also important. Not only because it can improve performance, but also because it most likely demands less computations to extract information from where it is easiest to access. The pan-tilt unit used for the mapping in Chapter 6 provides a good setting for such experiments.

In Chapter 5 it was shown how the robustness can be increased by combining different sensor modalities. Only sonar and laser sensors have been evaluated, but sensor fusion can be taken much further than this, by combining heterogenous sensors. Vision is the sensor that offers the largest potential source of information. Despite being sensitive to for example varying lighting conditions, valuable information is still to be found. In a global localization task a door offers so much more information than just the relative position to the robot. Typically there are signs and names in the vicinity of the door giving strong evidence about the location.

When working with a large scale system, integration inevitably becomes a major issue. Even if the different modules existed and were robust there are currently no methods to combine them to create a large scale system in an systematic manner. It is thus clear that systems integration in itself constitutes one of the fundamental problems that needs to be addressed. It is only by building and testing real world system that the real issues arise.

Another point that must be brought forth is the need for really long term experiments. The long experiments presented in this thesis is a step in that direction, but the length of the tests should be measured in days and weeks and not hours. Related to this is the problem of the power supply. Currently most robots used batteries that limit their autonomy to between 2 and 5 hours.

On a more detailed level there are also some suggestions for future work:

The second approach presented in Chapter 5 provided a great improvement in the ability to find the pose of the robot. However, as was seen in some of the experiments the samples drawn from the alternative importance function distracted the pose estimation. These samples always give support to the regions that match the current measurements the best, which is due to the improper weighting. Proper weighting schemes are thus an important step for the future.

Comparing the result from Chapter 5 and 4 it is clear that an active exploration strategy offers a great advantage. The hypothesis framework used in MHL provides a perfect platform for applying various more advanced exploration techniques. Extracting hypotheses from the sample set of MCL is also possible. Initial work in this direction is given in (Seiz et al. 2000).

A more thorough comparison of Multiple Hypothesis Localization and Monte Carlo Loclaization would also be of great interest.

The interaction between the robot and the human in the map building presented in Chapter 6 can also be improved. Graphical feedback and speech input/output will contribute to a powerful demonstration of localization.

Alternatives to the decoupled stochastic mapping technique used in Chapter 6 are also of interest. Instead of having a common coordinate system, an alternative approach is to let each submap have its own coordinate system and instead estimate the transformations between submaps. In essence this corresponds to letting each submap be a feature on a higher level in the hierarchy. Standard SLAM can then be performed on all levels.

Appendix

Appendix A

Odometric Model

A.1 Introduction

Since most platforms so far in robotics research are 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 which count the number of rotations for the wheel axes and the steer axes. Typically, high resolution encoders are used for this purpose. The angle information is discretized, and the number of "ticks" are counted. The resolution is normally high though and the discretization is only problem when measuring slow rotations.

As with all sensors, a model of the odometry provides valuable information about performance and limitations. Any odometric model, however good, is at best an approximation of the true kinematics. When using odometry for pose predictions, the most critical part of the estimation is the ability to estimate the orientation of the platform. Even a small error in the orientation, θ, of the platform eventually leads to large errors in the position. By careful modeling, these systematic errors can be made small (Borenstein & Feng 1996*b*, Borenstein 1998). Non-systematic errors on the other hand cannot be captured by the model and leads to devastating errors. A common source for non-systematic errors is wheel slippage.

Indoor platforms normally have much better odometric quality than outdoor once because of the non-planar surfaces which face outdoor platforms. It is important to keep in mind the conditions under which the final product is going to operate. In many labs, the floors are smooth and the odometric system is accurate. Relying on odometry to a large extent may prove fatal if the floor surface is changed by e.g. placing a carpet on the floor.

Borenstein and Feng present a benchmark test called the University of Michigan Benchmark test (UMBmark) to evaluate the performance of the odometric system (Borenstein & Feng 1995, Borenstein & Feng 1996*b*). 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 are also developed.

In (Borenstein 1998) results are presented for the commercially available mobile robot called "OmniMate" which show that the IPEC-method (internal position error correction) described in e.g. (Borenstein et al. 1996) successfully compensates for non-systematic errors, giving a overall increase in accuracy of one order of magnitude.

Since localization is an important part of a mobile robot and as the odometric information is both cheap and easy to use it has become a central part of most localization systems. The odometry is typically highly reliable over short distances, but degrades with the distance as there is nothing that bounds the positioning error.

There are different kinematic designs of mobile robots. This design influences the performance of the odometry to a large extent. To construct a platform with high performing odometry, it is important to carefully think the kinematic design through.

The main contribution in this Appendix is the development of a model for the odometric system on a synchro drive robot. The aim is a model that can be used in an iterative update procedure, such as a Kalman filter. The model should provide an estimate of the robot motion as well as the uncertainty in this estimate. It is also desirable that the model be consistent in the sense that it should give the same result independent of how the path is segmented. In (Chong & Kleeman 1996) such a model is developed for a differential drive robot. Just like in (Chong & Kleeman 1996) the model rests on the assumption that the robot moves on circular arcs. Such an assumption is commonly made, see e.g. (Fox et al. 1997).

A.2 Synchro-Drive Robot

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 a chain or a belt.

Figure A.1 illustrates the idea that all wheels are connected and one way to make the wheel translate as well as steer. Translation is achieved by rotating the inner axis (α). The outer axis steers the wheel, giving them an angle ϕ. The platform itself does not turn when the wheels turn, meaning that the orientation, θ, of the platform with respect to a global frame of reference is ideally constant. The direction of motion in world coordinates, γ, is given by $\gamma = \theta + \phi$.

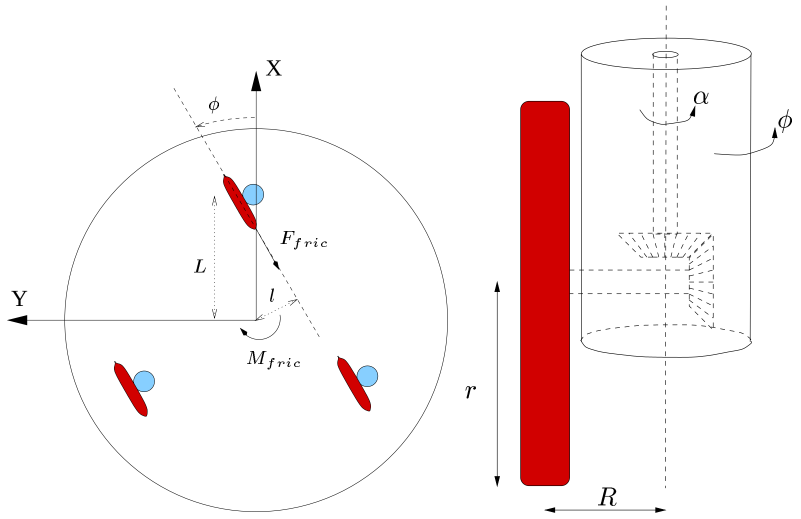


Figure A.1: *Left: Synchro drive configuration of the Nomad200 robot. Right: Example of synchro drive wheel.*

A.2.1 Odometric Model

Let the pose of the robot be $\mathbf{x} = (x^{(W)}, y^{(W)}, \theta)$, where $(x^{(W)}, y^{(W)})$ is the position in world coordinates and θ is the orientation of the robot coordinate system (see Appendix C). Henceforth the position in world coordinates is denoted (x, y) to save space. It is assumed that the robot moves along circular arcs, with radius r. This is motivated by the fact that that any path can be divided into a set of arcs. The special cases of such a motion is a straight line $(r = \infty)$ and turning on the spot $(r = 0)$.

Introduce $\Delta\gamma$ as the change in direction of motion from one time step to the next (see Figure A.2), i.e.

$$\gamma_{k+1} = \gamma_k + \Delta\gamma_k. \tag{A.1}$$

Two consecutive poses (x_k, y_k) and (x_{k+1}, y_{k+1}) are related by

$$\begin{cases} x_k = x_k^{rot} + r\cos(\gamma_k - \frac{\pi}{2}) = x_k^{rot} + r\sin\gamma_k \\ y_k = y_k^{rot} + r\sin(\gamma_k - \frac{\pi}{2}) = y_k^{rot} - r\cos\gamma_k, \end{cases} \tag{A.2}$$

and

$$\begin{cases} x_{k+1} = x_k^{rot} + r\cos(\gamma_{k+1} - \frac{\pi}{2}) = x_k^{rot} + r\sin\gamma_{k+1} \\ y_{k+1} = y_k^{rot} + r\sin(\gamma_{k+1} - \frac{\pi}{2}) = y_k^{rot} - r\cos\gamma_{k+1} \end{cases}, \tag{A.3}$$

where (x_k^{rot}, y_k^{rot}) denotes the coordinates of the center of rotation for the arc motion.

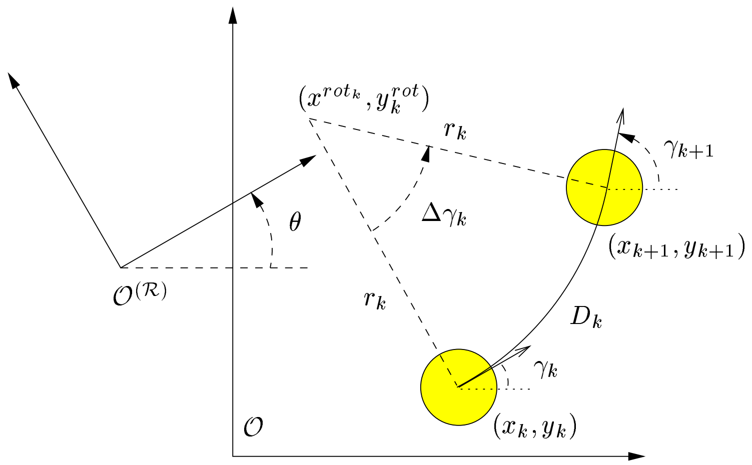


Figure A.2: *The motion of the robot is approximated to be on arcs.*

From (A.2) and (A.3), the pose of the robot at time $k+1$ is easily found as a function of the pose at time k

$$\begin{cases} x_{k+1} = x_k + r\left(\sin\gamma_{k+1} - \sin\gamma_k\right) \\ y_{k+1} = y_k - r\left(\cos\gamma_{k+1} - \cos\gamma_k\right). \end{cases} \tag{A.4}$$

Assuming the steer angle in robot coordinates can be measured without uncertainty, e.g. using high resolution encoders, the definition of the pose, $\mathbf{x}_k$, can be changed to (x, y, γ). Let $\mathbf{u}_k = (D_k, \Delta\gamma_k)$ be the input to the odometric model, where D_k is the distance traveled along the arc and $\Delta\gamma_k$ is the change in motion direction. With this notion the radius of the motion is given by

$$r_k = \frac{D_k}{\Delta\gamma_k}. \tag{A.5}$$

Note that a negative r_k corresponds to turning clock wise and a positive r_k corresponds to turning counter clock wise. The odometric model can now be written

$$\begin{aligned} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) &= \begin{pmatrix} x_k + r_k\left(\sin(\gamma_k + \Delta\gamma_k) - \sin\gamma_k\right) \\ y_k - r_k\left(\cos(\gamma_k + \Delta\gamma_k) - \cos\gamma_k\right) \\ \gamma_k + \Delta\gamma_k \end{pmatrix} \\ &= \begin{pmatrix} x_k + \frac{D_k}{\Delta\gamma_k}\left(\sin(\gamma_k + \Delta\gamma_k) - \sin\gamma_k\right) \\ y_k - \frac{D_k}{\Delta\gamma_k}\left(\cos(\gamma_k + \Delta\gamma_k) - \cos\gamma_k\right) \\ \gamma_k + \Delta\gamma_k \end{pmatrix}. \end{aligned} \tag{A.6}$$

A.2.2 Non-Systematic Errors

Let P_k denote the covariance matrix of the pose estimate at time k. Using a first order approximation, P_k may be updated according to

$$\begin{aligned} P_{k+1} &= \left(\frac{\partial f}{\partial \mathbf{x}_k}\right) P_k \left(\frac{\partial f}{\partial \mathbf{x}_k}\right)^T + \left(\frac{\partial f}{\partial \mathbf{u}_k}\right) \Sigma_k \left(\frac{\partial f}{\partial \mathbf{u}_k}\right)^T \\ &= \left(\frac{\partial f}{\partial \mathbf{x}_k}\right) P_k \left(\frac{\partial f}{\partial \mathbf{x}_k}\right)^T + Q_k, \end{aligned} \tag{A.7}$$

where Σ_k is the covariance matrix for the odometric input $\mathbf{u}_k$. The two Jacobians in (A.7) are given by

$$\frac{\partial f}{\partial \mathbf{x}_k} = \begin{pmatrix} 1 & 0 & r_k\,(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k) \\ 0 & 1 & r_k\,(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k) \\ 0 & 0 & 1 \end{pmatrix} \tag{A.8}$$

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{u}_k} &= \begin{pmatrix} \frac{1}{\Delta\gamma_k}(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k) & -\frac{D_k}{(\Delta\gamma_k)^2}(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k) + \frac{D_k}{\Delta\gamma_k}\mathrm{c}\,\gamma_{k+1} \\ -\frac{1}{\Delta\gamma_k}(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k) & \frac{D_k}{(\Delta\gamma_k)^2}(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k) + \frac{D_k}{\Delta\gamma_k}\mathrm{s}\,\gamma_{k+1} \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\Delta\gamma_k}(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k) & -\frac{r_k}{\Delta\gamma_k}(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k) + r_k\,\mathrm{c}\,\gamma_{k+1} \\ -\frac{1}{\Delta\gamma_k}(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k) & \frac{r_k}{\Delta\gamma_k}(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k) + r_k\,\mathrm{s}\,\gamma_{k+1} \\ 0 & 1 \end{pmatrix}, \end{aligned} \tag{A.9}$$

where (A.5) has been used and c and s denote cos and sin respectively. The matrix Σ_k is assumed to be diagonal

$$\Sigma_k = \begin{pmatrix} \sigma^2_{D_k} & 0 \\ 0 & \sigma^2_{\Delta\gamma_k} \end{pmatrix}, \tag{A.10}$$

i.e. the error in distance traveled is independent of the error in change of motion direction. In view of the findings in the previous section, this assumption is clearly not valid. However, using too many parameters in the odometric model has the downside though that they become difficult to identify. The error made by neglecting the correlation terms is judged not to be significant.

The variance in distance traveled is assumed to depend only on the distance traveled, i.e.

$$\sigma^2_{D_k} = k_D|D_k|, \tag{A.11}$$

which can be written

$$\sigma^2_{D_k} = k_D|r_k\Delta\gamma_k| \tag{A.12}$$

using (A.5).

The variance in change in motion direction on the other hand depends both on distance traveled, D_k, and the change in motion direction $\Delta\gamma_k$,

$$\sigma^2_{\Delta\gamma_k} = k^\gamma_\gamma|\Delta\gamma_k| + k^D_\gamma|D_k| = (k^\gamma_\gamma + k^D_\gamma|r_k|)|\Delta\gamma_k| = k_\gamma(r_k)|\Delta\gamma_k|. \tag{A.13}$$

Now divide the k:th step into N sub-steps, $i = 1, \ldots, N$, such that $\mathbf{x}^0_k = \mathbf{x}_k$ and $\mathbf{x}^N_k = \mathbf{x}_{k+1}$ as well as $P^0_k = P_k$ and $P^N_k = P_{k+1}$. With this notation (A.7) can be written as

$$P_{k+1} = P^N_k = \left(\frac{\partial f}{\partial \mathbf{x}^{N-1}_k}\right) P^{N-1}_k \left(\frac{\partial f}{\partial \mathbf{x}^{N-1}_k}\right)^T + \left(\frac{\partial f}{\partial \mathbf{u}^{N-1}_k}\right) \Sigma^{N-1}_k \left(\frac{\partial f}{\partial \mathbf{u}^{N-1}_k}\right)^T. \tag{A.14}$$

By recursively expanding P^i_k on the right hand side of (A.14) the following is obtained

$$\begin{aligned} P_{k+1} = &\left(\frac{\partial f}{\partial \mathbf{x}^{N-1}_k}\right) \left[\left(\frac{\partial f}{\partial \mathbf{x}^{N-2}_k}\right) P^{N-2}_k \left(\frac{\partial f}{\partial \mathbf{x}^{N-2}_k}\right)^T + \right.\\ &\left.\left(\frac{\partial f}{\partial \mathbf{u}^{N-2}_k}\right) \Sigma^{N-2}_k \left(\frac{\partial f}{\partial \mathbf{u}^{N-2}_k}\right)^T\right] \left(\frac{\partial f}{\partial \mathbf{x}^{N-1}_k}\right)^T + \\ &\left(\frac{\partial f}{\partial \mathbf{u}^{N-1}_k}\right) \Sigma^{N-1}_k \left(\frac{\partial f}{\partial \mathbf{u}^{N-1}_k}\right)^T \\ = &\left(\prod_{i=0}^{N-1} \frac{\partial f}{\partial \mathbf{x}^i_k}\right) P_k \left(\prod_{i=0}^{N-1} \frac{\partial f}{\partial \mathbf{x}^i_k}\right)^T + \\ &\sum_{i=0}^{N-1} \left[\left(\prod_{j=i+1}^{N-1} \frac{\partial f}{\partial \mathbf{x}^j_k}\right) \left(\frac{\partial f}{\partial \mathbf{u}^i_k}\right) \Sigma^i_k \left(\frac{\partial f}{\partial \mathbf{u}^i_k}\right)^T \left(\prod_{j=i+1}^{N-1} \frac{\partial f}{\partial \mathbf{x}^j_k}\right)^T\right]. \end{aligned} \tag{A.15}$$

The first factor in the second expression of (A.15) can, using (A.8), to be written as

$$\prod_{j=i+1}^{N-1} \frac{\partial f}{\partial \mathbf{x}^j_k} = \begin{pmatrix} 1 & 0 & r_k\left(\mathrm{c}\,\gamma^{N-1}_k - \mathrm{c}\,\gamma^{i+1}_k\right) \\ 0 & 1 & r_k\left(\mathrm{s}\,\gamma^{N-1}_k - \mathrm{s}\,\gamma^{i+1}_k\right) \\ 0 & 0 & 1 \end{pmatrix}. \tag{A.16}$$

With the help of this, the first term of (A.15) can be written (remembering that $\mathbf{x}^0_k = \mathbf{x}_k$)

$$\begin{pmatrix} 1 & 0 & r_k\left(\mathrm{c}\,\gamma^{N-1}_k - \mathrm{c}\,\gamma_k\right) \\ 0 & 1 & r_k\left(\mathrm{s}\,\gamma^{N-1}_k - \mathrm{s}\,\gamma_k\right) \\ 0 & 0 & 1 \end{pmatrix} P_k \begin{pmatrix} 1 & 0 & r_k\left(\mathrm{c}\,\gamma^{N-1}_k - \mathrm{c}\,\gamma_k\right) \\ 0 & 1 & r_k\left(\mathrm{s}\,\gamma^{N-1}_k - \mathrm{s}\,\gamma_k\right) \\ 0 & 0 & 1 \end{pmatrix}^T. \tag{A.17}$$

Introduce Q_k for the second term of (A.15), made up of a sum of Q_k^i,

$$Q_k = \sum_{i=0}^{N-1} Q_k^i. \tag{A.18}$$

To evaluate Q_k^i, start with

$$\begin{aligned}&\left(\frac{\partial f}{\partial \mathbf{u}_k^i}\right)^T \left(\prod_{j=i+1}^{N-1} \frac{\partial f}{\partial \mathbf{x}_k^j}\right)\\ &= \begin{pmatrix} \frac{1}{\Delta\gamma_k^i}\left(\mathrm{s}\,\gamma_k^{i+1} - \mathrm{s}\,\gamma_k^i\right) & r_k\left(-\frac{1}{\Delta\gamma_k^i}\left(\mathrm{s}\,\gamma_k^{i+1} - \mathrm{s}\,\gamma_k^i\right) + \mathrm{c}\,\gamma_k^{N-1}\right) \\ -\frac{1}{\Delta\gamma_k^i}\left(\mathrm{c}\,\gamma_k^{i+1} - \mathrm{c}\,\gamma_k^i\right) & r_k\left(\frac{1}{\Delta\gamma_k^i}\left(\mathrm{c}\,\gamma_k^{i+1} - \mathrm{c}\,\gamma_k^i\right) + \mathrm{s}\,\gamma_k^{N-1}\right) \\ 0 & 1 \end{pmatrix}.\end{aligned} \tag{A.19}$$

By letting $N \to \infty$ the path traveled between time k and $(k+1)$ is split in infinitesimally small segments. When $N \to \infty$, the change in orientation in each individual step $\Delta\gamma_k^i \to 0$ and thus $\gamma_k^{N-1} \to \gamma_{k+1}$. The error made by extending the sum to N instead of $N-1$ also goes to zero as $N \to \infty$. It is useful to recall the following relations

$$\lim_{\epsilon\to 0}\left(\sin(\alpha+\epsilon) - \sin\alpha\right) = \epsilon\cos\alpha \tag{A.20}$$

$$\lim_{\epsilon\to 0}\left(\cos(\alpha+\epsilon) - \cos\alpha\right) = -\epsilon\sin\alpha \tag{A.21}$$

$$\lim_{N\to\infty}\left(\frac{\partial f}{\partial \mathbf{u}_k^i}\right)^T \left(\prod_{j=i+1}^{N} \frac{\partial f}{\partial \mathbf{x}_k^j}\right) = \begin{pmatrix} \mathrm{c}\,\gamma_k^i & \left(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k^i\right) \\ \mathrm{s}\,\gamma_k^i & \left(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k^i\right) \\ 0 & 1 \end{pmatrix}. \tag{A.22}$$

Using (A.22) and (A.10) the elements of Q_k^i can be found to be:

$$Q_{k,11}^i = \sigma_{D_k^i}^2\,\mathrm{c}^2\,\gamma_k^i + \sigma_{\Delta\gamma_k^i}^2 r_k^2\left(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k^i\right)^2 \tag{A.23}$$

$$\begin{aligned}Q_{k,12}^i =&\sigma_{D_k^i}^2\,\mathrm{c}\,\gamma_k^i\,\mathrm{s}\,\gamma_k^i + \\ &\sigma_{\Delta\gamma_k^i}^2 r_k^2(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k^i)(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k^i)\end{aligned} \tag{A.24}$$

$$Q_{k,13}^i = \sigma_{\Delta\gamma_k^i}^2 r_k(\mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k^i) \tag{A.25}$$

$$Q_{k,22}^i = \sigma_{D_k^i}^2\,\mathrm{s}^2\,\gamma_k^i + \sigma_{\Delta\gamma_k^i}^2 r_k^2\left(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k^i\right)^2 \tag{A.26}$$

$$Q_{k,23}^i = \sigma_{\Delta\gamma_k^i}^2 r_k\left(\mathrm{s}\,\gamma_{k+1} - \mathrm{s}\,\gamma_k^i\right) \tag{A.27}$$

$$Q_{k,33}^i = \sigma_{\Delta\gamma_k^i}^2. \tag{A.28}$$

Using (A.12) and (A.13) and summing up the elements of Q_k^i, Q_k can be found. Note that the sum is the definition of the Riemann integral with respect to γ_k^i.

This is illustrated with the element $Q_{k,11}$

$$
\begin{aligned}
Q_{k,11} &= \lim_{N\to\infty} \sum_{i=0}^{N-1} Q^i_{k,11} \\
&= \lim_{N\to\infty} \sum_{i=0}^{N-1} \left[\mathrm{c}^2\, \gamma^i_k \sigma^2_{D^i_k} + r_k^2 \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, \gamma^i_k \right)^2 \sigma^2_{\gamma^i_k} \right] \\
&= \lim_{N\to\infty} \sum_{i=0}^{N-1} \left[\mathrm{c}^2\, \gamma^i_k k_D |r_k \Delta\gamma^i_k| + r_k^2 \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, \gamma^i_k \right)^2 k_\gamma |\Delta\gamma^i_k| \right] \\
&= \lim_{N\to\infty} \sum_{i=0}^{N-1} \left[\mathrm{c}^2\, \gamma^i_k k_D |r_k| + r_k^2 \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, \gamma^i_k \right)^2 k_\gamma \right] |\Delta\gamma^i_k| \\
&= \operatorname{sign}(\Delta\gamma_k) \lim_{N\to\infty} \sum_{i=0}^{N-1} \left[\mathrm{c}^2\, \gamma^i_k k_D |r_k| + r_k^2 \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, \gamma^i_k \right)^2 k_\gamma \right] \Delta\gamma^i_k \\
&= \operatorname{sign}(\Delta\gamma_k) \int_{\gamma_k}^{\gamma_{k+1}} \left(k_D |r_k|\, \mathrm{c}^2\, x + r_k^2 k_\gamma \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, x \right)^2 \right) dx \\
&= \left(k_D |r_k| + r_k^2 k_\gamma \right) \operatorname{sign}(\Delta\gamma_k) \int_{\gamma_k}^{\gamma_{k+1}} \mathrm{c}^2\, x dx \\
&+ r_k^2 k_\gamma \operatorname{sign}(\Delta\gamma_k) \int_{\gamma_k}^{\gamma_{k+1}} \left(\mathrm{c}^2\, \gamma_{k+1} - 2\, \mathrm{c}\, \gamma_{k+1}\, \mathrm{c}\, x \right) dx \\
&= \left\{ \cos^2 x = \frac{1}{2} \left(1 + \mathrm{c}(2x) \right) \right\} \\
&= \frac{1}{2} \left(k_D |r_k| + r_k^2 k_\gamma \right) \operatorname{sign}(\Delta\gamma_k) \left(\Delta\gamma_k + \frac{\mathrm{s}(2\gamma_{k+1}) - \mathrm{s}(2\gamma_k)}{2} \right) \\
&+ r_k^2 k_\gamma \operatorname{sign}(\Delta\gamma_k) \left(\Delta\gamma_k\, \mathrm{c}^2\, \gamma_{k+1} - 2\, \mathrm{c}\, \gamma_{k+1} \left(\mathrm{s}\, \gamma_{k+1} - \mathrm{s}\, \gamma_k \right) \right) \qquad \text{(A.29)}
\end{aligned}
$$

The rest of the elements of Q_k can be found in the same manner to be

$$
\begin{aligned}
Q_{k,12} &= -\frac{|r_k| k_D + r_k^2 k_\gamma}{4} \operatorname{sign}(\Delta\gamma_k) \left(\mathrm{c}(2\gamma_{k+1}) - \mathrm{c}(2\gamma_k) \right) \\
&+ r_k^2 k_\gamma \operatorname{sign}(\Delta\gamma_k) \left(\frac{1}{2} \Delta\gamma_k\, \mathrm{s}(2\gamma_{k+1}) + \mathrm{c}(2\gamma_{k+1}) - \mathrm{c}(\gamma_k + \gamma_{k+1}) \right) \qquad \text{(A.30)} \\
Q_{k,13} &= r_k k_\gamma \operatorname{sign}(\Delta\gamma_k) \left(\Delta\gamma_k\, \mathrm{c}\, \gamma_{k+1} - \mathrm{s}\, \gamma_{k+1} + \mathrm{s}\, \gamma_k \right) \\
&= \left\{ r_k \operatorname{sign}(\Delta\gamma_k) = |r_k| \right\} \\
&= |r_k| k_\gamma \left(\Delta\gamma_k\, \mathrm{c}\, \gamma_{k+1} - \mathrm{s}\, \gamma_{k+1} + \mathrm{s}\, \gamma_k \right) \qquad \text{(A.31)} \\
Q_{k,22} &= \frac{1}{2} \left(k_D |r_k| + r_k^2 k_\gamma \right) \operatorname{sign}(\Delta\gamma_k) \left(\Delta\gamma_k - \frac{\mathrm{s}(2\gamma_{k+1}) - \mathrm{s}(2\gamma_k)}{2} \right) \\
&+ r_k^2 k_\gamma \operatorname{sign}(\Delta\gamma_k) \left(\Delta\gamma_k\, \mathrm{s}^2\, \gamma_{k+1} + 2\, \mathrm{s}\, \gamma_{k+1} \left(\mathrm{c}\, \gamma_{k+1} - \mathrm{c}\, \gamma_k \right) \right) \qquad \text{(A.32)}
\end{aligned}
$$

$$
\begin{aligned}
Q_{k,23} &= r_k k_\gamma \operatorname{sign}(\Delta\gamma_k)\left(\Delta\gamma_k \,\mathrm{s}\,\gamma_{k+1} + \mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k\right) \\
&= |r_k| k_\gamma \left(\Delta\gamma_k \,\mathrm{s}\,\gamma_{k+1} + \mathrm{c}\,\gamma_{k+1} - \mathrm{c}\,\gamma_k\right) \qquad \text{(A.33)} \\
Q_{k,33} &= k_\gamma |\Delta\gamma_k|. \qquad \text{(A.34)}
\end{aligned}
$$

Straight Line Motion

A special case for the arc motion is when the robot moves along a straight line. This corresponds to $r_k \to \infty$ and $\Delta\gamma_k \to 0$. Keeping (A.5), (A.12) and (A.13) in mind along with (A.20), (A.21) and $\gamma_{k+1} = \gamma_k^i + \frac{N-i}{N}\Delta\gamma_k$, (A.23)-(A.28) can be written

$$
\begin{aligned}
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,11}^i &= k_D \frac{|D_k|}{N}\,\mathrm{c}^2\,\gamma_k + \\
& k_\gamma^D \frac{|D_k|}{N}\left(\frac{D_k}{\Delta\gamma_k}\right)^2 \left(-\frac{N-i}{N}\Delta\gamma_k \,\mathrm{s}\,\gamma_k\right)^2 \\
&= k_D |D_k|\,\mathrm{c}^2\,\gamma_k \frac{1}{N} + \\
& k_\gamma^D |D_k|\,(D_k)^2\,\mathrm{s}^2\,\gamma_k \frac{(N-i)^2}{N^3} \qquad \text{(A.35)} \\
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,12}^i &= \frac{1}{2} k_D |D_k|\,\mathrm{s}(2\gamma_k)\frac{1}{N} - \\
& \frac{1}{2} k_\gamma^D |D_k|\,(D_k)^2\,\mathrm{s}(2\gamma_k)\frac{(N-i)^2}{N^3} \qquad \text{(A.36)} \\
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,13}^i &= k_\gamma^D \frac{|D_k|}{N}\frac{D_k}{\Delta\gamma_k}\left(-\frac{N-i}{N}\Delta\gamma_k \,\mathrm{s}\,\gamma_k\right) \\
&= -k_\gamma^D |D_k| D_k \,\mathrm{s}\,\gamma_k \frac{N-i}{N^2} \qquad \text{(A.37)} \\
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,22}^i &= k_D |D_k|\,\mathrm{s}^2\,\gamma_k \frac{1}{N} + \\
& k_\gamma^D |D_k|\,(D_k)^2\,\mathrm{c}^2\,\gamma_k \frac{(N-i)^2}{N^3} \qquad \text{(A.38)} \\
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,23}^i &= k_\gamma^D |D_k| D_k \,\mathrm{c}\,\gamma_k \frac{N-i}{N^2} \qquad \text{(A.39)} \\
\lim_{\substack{N\to\infty,\ r_k\to\infty \\ D_k = r_k\Delta\gamma_k}} Q_{k,33}^i &= k_\gamma^D |D_k| \frac{1}{N}. \qquad \text{(A.40)}
\end{aligned}
$$

A standard handbook in mathematics provides the following information about sums

$$\sum_{x=0}^{n} x = \frac{n(n+1)}{2} \approx \frac{n^2}{2} \tag{A.41}$$

$$\sum_{x=0}^{n} x^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{n^3}{3}. \tag{A.42}$$

The elements of Q_k can now be found by evaluating the sums and letting $N \to \infty$.

$$Q_{k,11} = |D_k| \left(k_D \operatorname{c}^2 \gamma_k + \frac{1}{3} k_\gamma^D \left(D_k\right)^2 \operatorname{s}^2 \gamma_k \right) \tag{A.43}$$

$$Q_{k,12} = \frac{1}{2} |D_k| \left(k_D - \frac{1}{3} k_\gamma^D \left(D_k\right)^2 \right) \operatorname{s}(2\gamma_k) \tag{A.44}$$

$$Q_{k,13} = -\frac{1}{2} k_\gamma^D |D_k| D_k \operatorname{s} \gamma_k \tag{A.45}$$

$$Q_{k,22} = |D_k| \left(k_D \operatorname{s}^2 \gamma_k + \frac{1}{3} k_\gamma^D \left(D_k\right)^2 \operatorname{c}^2 \gamma_k \right) \tag{A.46}$$

$$Q_{k,23} = \frac{1}{2} k_\gamma^D |D_k| D_k \operatorname{c} \gamma_k \tag{A.47}$$

$$Q_{k,33} = k_\gamma^D |D_k|. \tag{A.48}$$

A.2.3 Systematic Errors

The odometric error can be divided into a systematic and a non-systematic error. By calibrating the odometric system, the systematic error can be reduced. In a probabilistic framework, the error is often assumed to be independent over time, zero-mean etc. With these assumptions it is clear that a systematic error is highly undesired as it violates the assumptions.

To determine the systematic errors on the Nomad200 platform, an experiment is devised where the robot is driven approximately 10 m in one direction. Table A.1 summarize the data from the experiments by showing the true distance traveled based on tape measurements, the distance traveled according to odometry, the ratio between real and the odometric distance, the initial robot orientation, θ_{start}, the steer angle, $\phi = \gamma - \theta$ and the change in orientation $\Delta\theta$.

Clearly the variation in the distance ratio is small and thus this parameter can be well approximated by a constant. Let this ratio, or scaling factor, be denoted by β, i.e.,

$$D = \beta D^{(R)}. \tag{A.49}$$

The mean value for the scaling factor is

$$\beta = 1.0066, \tag{A.50}$$

a 0.66% scale error in other words.

Table A.1 shows a correlation between the steer angle, ϕ, and the change in orientation, $\Delta\theta$. Figure A.3 illustrates this more clearly, by plotting $\Delta\theta$ as a function of ϕ. The wheels are mechanically linked, both in translation and rotation. If one of the wheels has a different radius it will result in slippage and hence a friction force, F_{fric}, causing a moment of rotation, M_{fric}. With the notation from Figure A.1 the rotational moment is given by

$$M_{\text{fric}} = -L \sin\phi F_{\text{fric}} \propto -\sin\phi. \tag{A.51}$$

The friction force, F_{fric}, depends on the surface properties. Compensating for this systematic rotation of the platform thus means that the friction force must be estimated on-line. This is an interesting problem on its own. When the surface material is the same throughout the environment so is the force. Overlaid on the experimental data is the curve $-2.2^\circ \sin\phi$. The drift in robot orientation per meter distance traveled in direction ϕ is therefore modeled as

$$-2.2 \sin\phi \ \text{deg}/s. \tag{A.52}$$

A.3 Summary

In summary a consistent odometric model is derived based on an assumption that the robot path can be approximated by circular arcs. This model is of interest, for example, when propagating the uncertainty in the prediction step (2.18) of an extended Kalman filter. The process noise covariance matrix Q is derived in general as well as in the special case of straight line motion. Parts of the systematic error on a Nomad200 platform are also identified.

D	$D^{(R)}$	$D/D^{(R)}$	θ_{start}	$\approx \phi$	$\Delta\theta$
[mm]	[mm]	[1]	[deg]	[deg]	[deg]
9825.0	9762.1	1.006446	10.1	79.9	-1.830
9792.6	9723.4	1.007116	42.4	47.6	-2.150
9801.1	9736.2	1.006663	70.8	19.2	-1.750
9795.1	9731.0	1.006584	100.8	-10.8	-0.570
9785.5	9723.6	1.006368	131.0	-41.0	0.940
9785.2	9739.8	1.004663	162.6	-72.6	1.980
9905.2	9842.4	1.006374	190.5	-100.5	2.170
9878.7	9813.4	1.006656	220.3	-130.3	0.780
9805.2	9740.5	1.006647	249.7	-159.7	0.800
9875.4	9804.1	1.007267	279.8	-189.8	-0.600
9830.1	9762.6	1.006910	309.9	-219.9	-1.660
9795.1	9726.5	1.007052	338.0	-248.0	-1.890
	mean	1.006562	-	-	-0.315
	std	0.000666	-	-	1.585

Table A.1: *Distance traveled measured with tape measure D and odometry $D^{(R)}$ as well as the ratio between the two along with the change in robot orientation.*

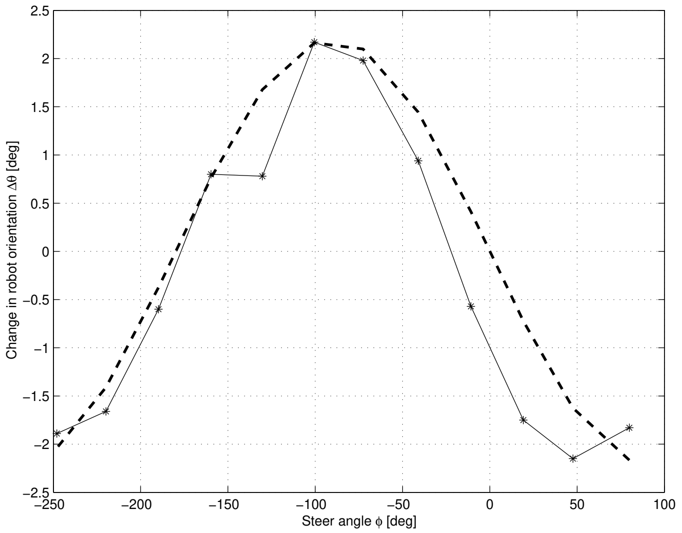


Figure A.3: *The change in robot orientation depends heavily on the motion direction in robot coordinates, ϕ. The curve $-2.2^\circ \sin\phi$ (dashed) is overlaid on the experimental data.*

Appendix B

The Laser Sensor

As the laser scanner is the main sensor used in the thesis, a short historical overview and basic description of the technology will be given for this fantastic invention called laser. The overview is based on material found in (Hecht 1987). Those familiar with the laser sensor should move directly to Section B.6 where the SICK laser scanners are investigated.

B.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 a 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 fed 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.

In July 1961, Maiman announced existence of the first operating laser. It was a pulsed solid-state ruby[1] 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 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 (Hecht 1987, 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[2]. 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.

B.2 Laser Range Finders

Having found applications in so many areas, it is not surprising that the laser is 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

$$D = \frac{1}{2}cT \tag{B.1}$$

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*. To realize such a system, a high

[1] Ruby is one of the most used laser media still today

[2] Cryogenic temperatures refer to extremely low temperatures, ranging from -150°C to the absolute zero at -273°C.

precision means for measuring time is needed. Thinking in terms of a robot application a range resolution in the order of centimeters is desirable. The speed of light is approximately $3 \cdot 10^8$ m/s. 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 (Andersen et al. 1992). This means that ranges above the wavelength of the modulator cannot be distinguished from ranges below it.

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 (Chatila 1985, Hoppen et al. 1990, Buchberger et al. 1993, Borthwick et al. 1993, Forsberg, Åhman & Wernersson 1993, Weiss & von Puttkamer 1995, Arras & Vestli 1998, Gutmann et al. 1998, Fox, Burgard & Thrun 1999).

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 (see for example (Nashashibi et al. 1992)).

B.3 Material Dependence

In (Ljunggren Klöör & Wernersson 1992*b*, Ljunggren Klöör & Wernersson 1992*a*), 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 observed that objects of type 1,2 and 4 dominate in lab scenes whereas 1,2 and 3 are the most common types outdoor. 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.

B.4 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 time very accurately.
- The range accuracy is fairly good. (Forsberg, Åhman & Wernersson 1993) reports to have a standard deviation of 20 mm and the new generation of laser scanners from SICK Electro-Optics has an accuracy better than 10 mm and an angular resolution of 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° or 0.25° depending on operation mode 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 if disregarding specular reflections and multiple reflections, but not for a camera image, which takes a lot of effort to interpret.

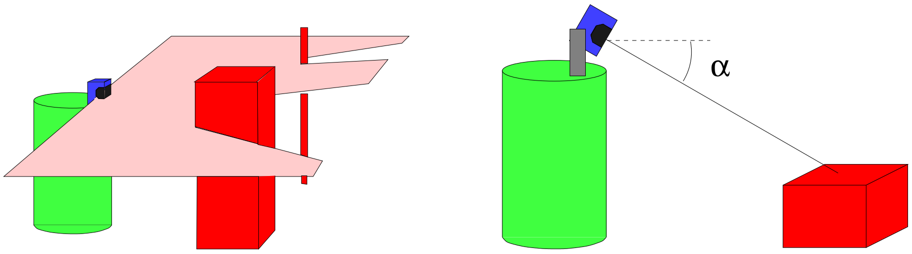


Figure B.1: *In the normal configuration, the laser scans only in a horizontal plane. By tilting the laser, information can be gathered from a larger subspace than a plane assuming the platform is moving.*

B.5 Drawbacks of 2D Laser Scanners

Among the drawbacks with 2D laser scanners are:

- The sensor provides range information limited to a plane (see Figure B.1) which means that only a 2D intersection of the 3D world can be sensed. To get information about other parts of the environment the sensor has to be mounted on a pan-tilt-unit. Another possible solution is to mount the sensor so that it intersects the world under an angle α (see Figure B.1) and instead move the platform.

- The sensor is still expensive. The price will decrease though if an application is found that motivates mass production of the sensor.

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

B.6 The SICK PLS and LMS laser scanners

Before extracting information from laser sensor data it is important to know the underlying characteristics of this data. This warrants a closer look at the particular sensor used in the experiments of the thesis.

The PLS 200 from SICK Electro-Optics is an example of a TOF laser scanner (see Section B.2). It scans the environment at a rate of 25 Hz using a rotating mirror. Each scan thus takes 40 ms. Due to the limitation of using a 38.4 kBaud serial connection the highest theoretical data rate is approximately 5 Hz if a 2 bit overhead on each byte is assumed. For most of the experiments the date rate is between 2 and 3 Hz.

The angular beam separation is 0.5°, resulting in 361 range readings, $\mathbf{z}_i = (r_i, \phi_i)$, covering a 180° field of view (see Figure B.1). Some of the experiments are performed with the more accurate SICK LMS model. In the following, the results apply to the PLS sensor if nothing else is said, but can be extended to the LMS with obvious changes.

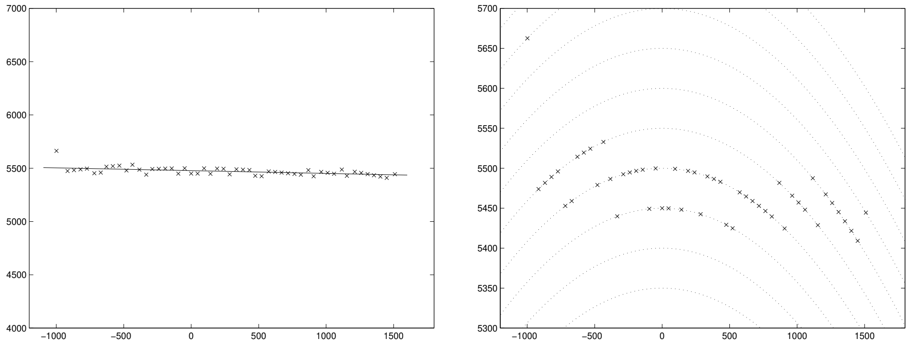


Figure B.2: *Segment of a scan taken of a wall approximately 5.5 m away from the sensor. A line marking the location of the wall is added to the left subfigure. The right subfigure shows the range readings along with arcs with 50 mm spacing, clearly showing the discretization.*

B.6.1 Quantization

The range readings of the PLS sensor are quantized in steps of size $\Delta r = 50$ mm. Figure B.2 shows a part of a scan taken against a wall approximately 5.5 m away from the sensor. The discretization is clearly visible in the right subfigure where the same data is shown with different scaling and circular arcs with 50 mm spacing. Due to the gross discretization it is not easy to get an estimate of the underlying range distribution, i.e. given that the true range is R what is the probability distribution for measuring r. To get a better estimate of the underlying range distribution, an experiment is performed where 100 range readings, $q_{i,j}$, $i = 1, \ldots, 100$, are collected at each of 50 ($j = 1, \ldots, 50$), tightly spaced positions in front of a wall. Let Δd_j, ($j = 1, \ldots, 50$) be the distance traveled towards the wall according to odometry. Figure B.3 shows a histogram over $q_{i,j} + \Delta d_j$ for the PLS sensor. Figure B.4 shows the same for the LMS sensor. Assuming the odometry to have negligible uncertainty over the short traveled distance, the spread in the figures are due to the error distribution inherent in the laser scanner. These distributions clearly resembles Gaussians.

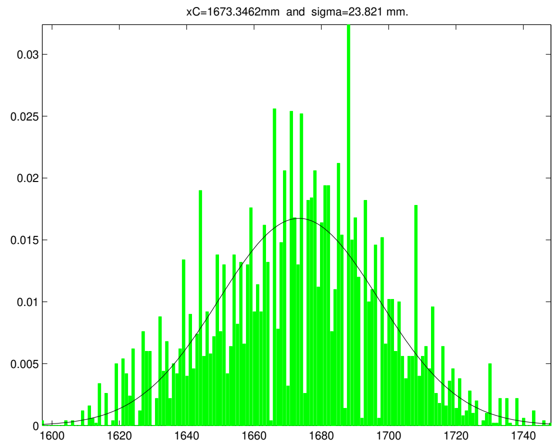


Figure B.3: *Estimated range distribution of the PLS laser sensor. The distribution is approximated as a Gaussian with a standard deviation of $\approx$ 24 mm.*

The standard deviation can be estimated to 24 mm for the PLS and 5.5 mm for the LMS. The resulting Gaussians are overlaid in the figures. As the distance measurements are quantized, only multiples of the quantization step can be measured, $n\Delta r,\ n = 0, 1, 2, \ldots$. Assuming R to be the true distance, the PDF for the measurements can be written as a sampled Gaussian,

$$p(n\Delta r|R) = \frac{e^{-\frac{1}{2}(\frac{n\Delta r - R}{\sigma_r})^2}}{\sum_m e^{-\frac{1}{2}(\frac{m\Delta r - R}{\sigma_r})^2}}. \tag{B.2}$$

The variance in a measurement error $(R - n\Delta r)$ depends on the true distance because of the quantization according to

$$V(R - n\Delta r|R) = \sum_n (R - n\Delta r)^2 p(n\Delta r|R). \tag{B.3}$$

Figure B.5 shows the corresponding standard deviation as a function of true distance, R. Here the simplifying assumption is made that the standard deviation for the measurement error of the PLS sensor is independent of the true distance and equal to the worst case, i.e. approximately 26 mm.

B.6.2 Footprint Size

The laser energy propagates in a cone, just like the ultrasonic energy used in a sonar sensor. 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° (Pol n.d.).

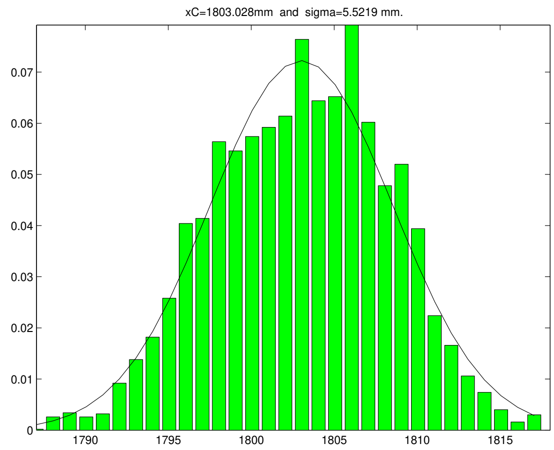


Figure B.4: *Estimated range distribution of the LMS laser sensor. The distribution is approximated as a Gaussian with a standard deviation of ≈5.5 mm.*

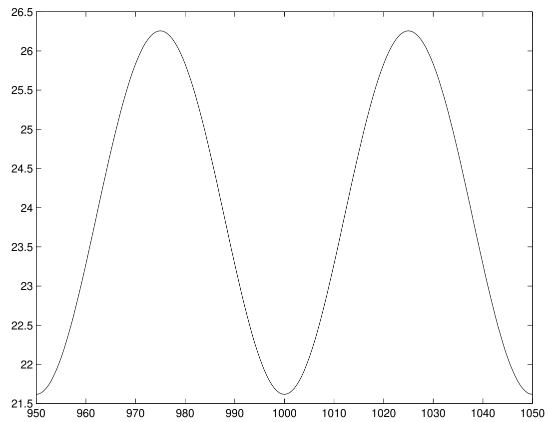


Figure B.5: *Standard deviation for measurement error as a function of true distance for the PLS sensor.*

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, a series of experiments are performed. The laser sensor is placed approximately 4.7 m away from two wooden boards. The range measured by one of the beams is studied. By sliding the boards perpendicular to that laser beam and monitoring the measurement the size of the beam can be estimated. Figure B.6 illustrates the experimental setup. When the boards are not detected, the measured distance will be the distance to the wall behind.

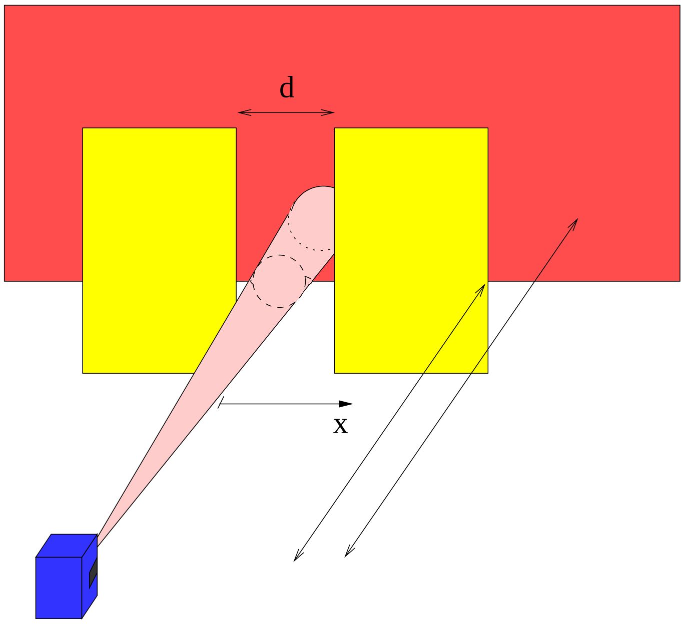


Figure B.6: *The footprint of the laser beam is determined by sliding two boards towards each other with the laser beam in between.*

In the first experiment only one of the boards is used. 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 position perpendicular to the beam, Table B.1 show the resulting measurements. As can be seen from the table the range measurements do not jump discontinuously from the wall distance to the board distance. When the board is at position $x = 4$ the range measurements jump over a range of 0.4 m, giving measurements in between 5.05 m (in between) and 5.45 m (wall). In the range $x \in [4, 11]$ the measurement are in between wall and board. This range corresponds to an angle of approximately $\frac{7}{4700} = 0.0015$ rad $= 0.085°$, i.e. using a wooden board the laser beam has a region of approximately 0.09° on each side, which generates spurious, so called phantom measurements.

x [mm]	0	1	2	3	4	5	6	7
range [m]	5.45	5.45	5.45	5.45	5.05–5.45	4.9	4.95	4.85
x [mm]	8	9	10	11	12	13	14	15
range [m]	4.8	4.8	4.75	4.75	4.7	4.7	4.7	4.7

Table B.1: *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.*

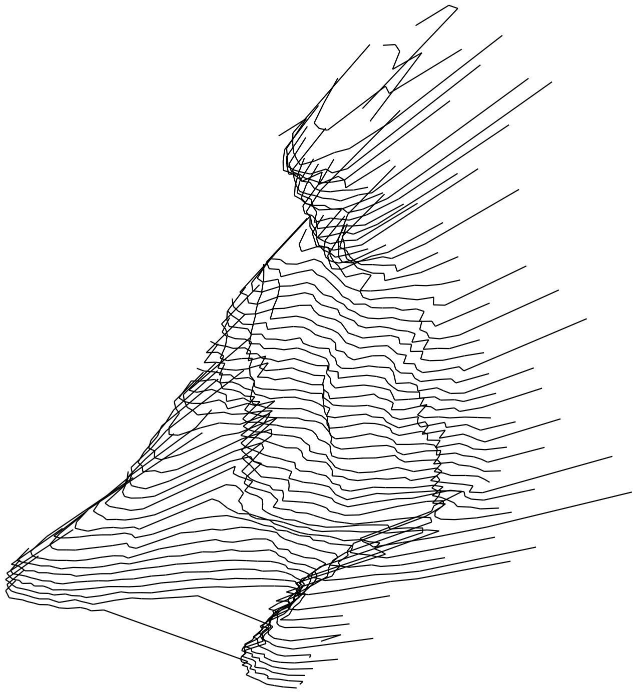


Figure B.7: *Phantom measurements are visible at the sides of the person being captured by the pan-tilt mounted laser scanner.*

To illustrate the phenomena of phantom measurements further a person is placed in the middle of a room with no other object close by. A laser sensor mounted on a pan-tilt unit scans the person and the scan points that belong to the person are plotted in Figure B.7. If there were no phantom measurements all the scan points would lie on the person. As can be seen though, many of the measurements at the border of the person end up in the middle of the air as a result of only partially being reflected by the person.

To determine the size of the beam the two boards are used, 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°. The size of the laser beam, that is strong enough to give the correct correct measurement, is thus 0.25°. 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°. As the laser beams are separated by 0.5°, there is a blind region between the beams.

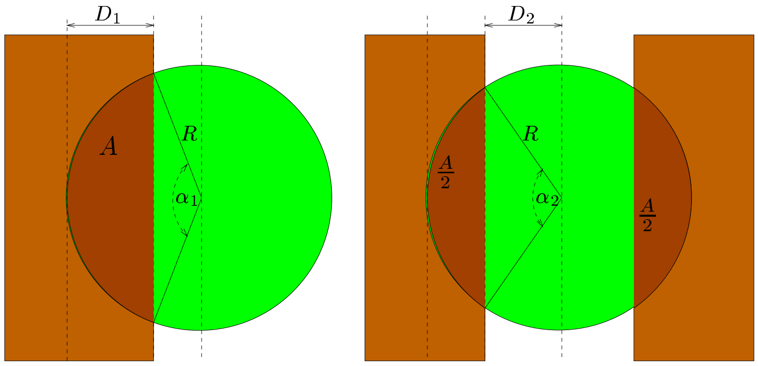


Figure B.8: *The reflecting areas are assumed to be the same and can thus be used to put up an equation to solve for the beam radius R.*

Another way to make an estimate of the foot print size is to make the simplifying assumption that the energy is equally spread over the beam. Using the fact that the beam is circular in shape an equation can be formulated for the area that is needed to reflect enough energy to get a correct range measurement. The origin of the equation can been seen in Figure B.8. Using some geometry one will find the equation

$$A = (\alpha_1 - sin(\alpha_1) = 2(\alpha_2 - sin(\alpha_2)), \tag{B.4}$$

where $\alpha_1 = 2 \arccos \frac{R - D_1}{R}$ and $\alpha_2 = 2 \arccos \frac{D_2}{R}$. Solving (B.4) using the information from the experiments above, i.e. $D_1 = 9$ mm and $D_2 = 10$ mm yields a beam radius, $R \approx 15.5$ mm. This corresponds to a beam width of $0.38°$. The difference between the two results is caused by the assumption of equally spread energy.

B.7 Summary

In this Appendix a brief description of the technology behind the laser sensor and a short historical overview has been given. Some advantages and drawbacks with 2D scanning laser sensors have been outlined. The contribution is the characterization of the two laser scanners from SICK, the PLS and the LMS. The effect of quantization and the underlying range distribution are investigated as well as the beam width. It is concluded that unless the whole beam is reflected, readings that are somewhere between two objects in depth can result, so called phantom measurements.

Appendix C

Coordinate Systems

Besides the world coordinate system denoted with super script (W), three other coordinate systems are considered. One is the odometric coordinate system, marked with super script (R). The origin of the odometric coordinate system is defined by the position at which the robot was turned on, or where the wheel encoders where zeroed. When the robot moves this coordinate system will remain fixed with respect to the world coordinate system under ideal conditions. However, due to for example wheel slippage and an imperfect odometric

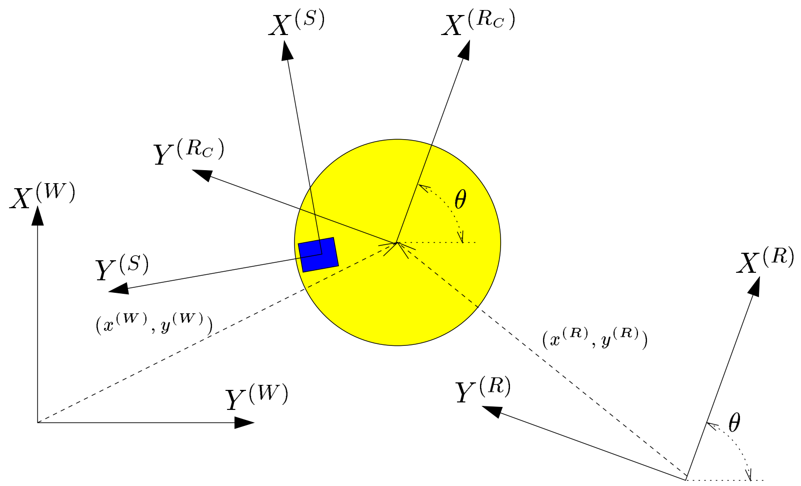


Figure C.1: *Definition of the state of the robot. The world coordinate system is not moving. The odometric information is given in the robot coordinate system. This coordinate system will drift over time due to for example wheel slippage. The sensor coordinate system is attached to the laser sensor.*

model, the odometric coordinate system will drift. The third coordinate system considered has the same orientation as the odometric coordinate system but is attached to the center of the robot (super script (R_C)). The last coordinate system under consideration is the coordinate system of the laser sensor. This coordinate system is attached to the laser sensor (S). The many different coordinate systems all have their purpose. Typically some of the relations or constrints are expressed easiest in one of the them. An example are the visibility constraints which are easiest expressed in the laser coordinate system.

Appendix D

The Experimental Platforms

The two main platforms used in the experiments in this thesis are the Nomad200 platform and the smaller SuperScout platform, both manufactured by Nomadics.

D.1 Nomad200

The Nomad200 is a synchro drive platform with three wheels, mechanically connected. It has 16 standard Polaroid ultrasonic sensors (Pol n.d.) placed around the top of the body (see Figure D.1). At the lower end of the body 16 IR sensors are placed. The IR sensor are very sensitive to the reflecting material and has a short range of operation. Therefore they provide more or less binary information, on/off, and as such are only used for obstacle avoidance. The upper body can rotate without effecting the direction of motion. This provide means to actively control the direction of attention for the sensors. A SICK PLS laser scanner is placed on top of the platform. It scans the environment approximately 93 cm above the ground, giving it a rather elevated view of its surrounding. The on-board computer is a Pentium III 450 MHz with 128 Mb of RAM.

Figure D.1: *Nomad200 platform called Asterix has 16 sonars and a PLS laser scanner. The upper body of the platform can rotate and thus the direction of the laser can be controlled.*

Figure D.2: *The SuperScout Louie with a PTU mounted LMS. Using the PTU the direction of the laser scanner can be controlled. In the upright position the laser is 52 cm above ground.*

D.2 Nomad SuperScout

One of the three Nomadics SuperScouts in the lab is shown in Figure D.2. It is equipped with a SICK LMS laser scanner mounted on a pan-tilt unit (PTU). The PTU can tilt $\pm 110°$ at a maximum angular velocity of $135°$/s. In the current configuration tilting more than $+75°$ or less than $-90°$ will damage the platform. Panning is possible in the interval $\pm 1080°$ at a maximum angular velocity of $216°$/s. The cable length sets the limit for the pan angle to approximately $\pm 90°$. There are 16 sonar sensors place equidistantly around the platform. The on-board computer is a Pentium MMX 233 MHz with 64 Mb of RAM, making it the computationally weakest platform of the four experimental platforms.

The two main wheels of the SuperScout are made of hard plastic and there is a small caster wheel in the back. Thresholds are almost impossible to traverse because of the the plastic wheels, but the odometric information is accurate as long as the surface is smooth. The caster wheel is of the same kind that is found on suitcases and it often gets blocked by, for example, small stones.

Figure D.3: *Left: The Nomadics XR4000 platform Obelix. Right: Goofy is a PeopleBot from ActiveMedia*

D.3 Other Platforms

The Nomadics XR4000 platform (left photo in Figure D.3) is by far the most complicated platform in the lab. It has three CPUs. Two of these are running standard Linux. The third is used to control the PUMA 560 manipulator and runs the real-time operating system QNX. The platform is equipped with 48 sonar and IR sensor placed in two rings, one at the top of the body and one at the bottom. The laser sensor is placed inside the main body, looking out through a thin stripe. The scan plane is 49 cm above ground. The platform has 4 wheels, each actuated with two motors in a way similar how the wheels of an ordinary office chair moves. This makes the platform close to holonomic. This comes at the cost of the by far worst odometric performance of the four experimental platform. A drift in orientation in the order of several tens of degrees over less than ten meters of motion is not uncommon.

The so-called PeopleBot is manufactured by ActiveMedia (right photo in Figure D.3). It is a differential drive robot with two rubber wheel. The rubber is quite soft which gives the platform a somewhat shaky motion. The SICK LMS laser scanner is placed only 30 cm above to the ground. Beneath the laser scanner there are 8 sonar sensors and 8 more are placed at the top, also facing forward.

Bibliography

Alspach, D. L. & Sorenson, H. W. (1972), 'Nonlinear bayesian estimation using gaussian sum approximations', *IEEE Transactions on Automatic Control* **AC-17**(4), 439–448.

Andersen, C., Madsen, C., Sorensen, J., Kirkeby, N., Jones, J. & Christensen, H. (1992), 'Navigation using range images on a mobile robot', *IEEE Transactions on Robotics and Automation* **10**(2-3), 147–160.

Anderson, B. D. O. & Moore, J. B. (1979), *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, N. J.

Arras, K. O. & Siegwart, R. Y. (1997), 'Feature extraction and scene interpretation for map-based navigation and map building', *Proc. of SPIE, Mobile Robotics XII* **3210**.

Arras, K. & Tomatis, N. (1999), Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision, *in* 'Proc. of the 3rd European Workshop on Advanced Mobile Robots (Eurobot'99)', Zürich, Switzerland, pp. 177–185.

Arras, K. & Vestli, S. (1998), Hybrid, high-precision localisation for the mail distributing mobile robot system mops, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)', Vol. 4, pp. 3129–3134.

Austin, D. & Jensfelt, P. (2000), Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 2, San Francisco, CA, USA, pp. 1036–1041.

Bar-Shalom, Y. (1989), Recursive tracking algorithms: From the kalman filter to intelligent trackers for cluttered environments, *in* 'Proc. of IEEE International Conference in Control and Applications', pp. 675–680.

Bar-Shalom, Y. & Fortmann, T. E. (1988), *Tracking and Data Association*, Academic Press, Inc.

Borenstein, J. (1998), 'Experimental results from internal odometry error correction with the omnimate mobile robot', *IEEE Transactions on Robotics and Automation* **14**(6), 963–969.

Borenstein, J., Everett, H. & Feng, L. (1996), *Navigating Mobile Robots: System and Techniques*, A K Peters, Ltd.

Borenstein, J. & Feng, L. (1995), Correction of systematic odometry errors in mobile robots, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95)', IEEE/RSJ, Pittsburgh, PA, pp. 569–574.

Borenstein, J. & Feng, L. (1996*a*), Gyrodometry: A new method for combining data from gyros and odometry in mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation', pp. 423–428.

Borenstein, J. & Feng, L. (1996*b*), 'Measurement and correction of systematic odometry errors in mobile robot', *IEEE Transactions on Robotics and Automation* **12**(6), 869–880.

Borenstein, J. & Koren, Y. (1989), 'Real-time obstacle avoidance for fast mobile robots', *IEEE Transactions on Systems, Man, and Cybernetics* **19**(5), 1179–1187.

Borenstein, J. & Koren, Y. (1991), 'Histogram in-motion mapping for mobile robot obstacle avoidance', *IEEE Transactions on Robotics and Automation* **7**(4), 535–539.

Borthwick, S., Stevens, M. & Durrant-Whyte, H. (1993), Position estimation and tracking using optical range data, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)', IEEE/RSJ, pp. 2172–2177.

Brooks, R. A. (1984), Aspects of mobile robot visual map making, *in* 'Proc. of 2nd International Symposium on Robotics Reseach', Kyoto, Japan, pp. 369–375.

Brooks, R. A. (1985), Visual map making for a mobile robot, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)', pp. 824–829.

Buchberger, M., Jörg, K. W. & von Puttkamer, E. (1993), Laserradar and sonar based world modeling and motion control for fast obstacle avoidance of the autonomous mobile robot mobot-iv, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'93)', Vol. 1, IEEE, pp. 534–40.

Burgard, W., Fox, D. & Henning, D. (1997), Fast grid-based position tracking for mobile robots, *in* 'Proc. of the German Conference on Artificial Intelligence'.

Burgard, W., Fox, D., Henning, D. & Schmidt, T. (1996), Estimating the absolute position of a mobile robot using position probability grids, *in* 'Proc. of the National Conference on Artificial Intelligence (AAAI-96)', Portland, Oregon, USA, pp. 896–901.

Burgard, W., Fox, D. & Thrun, S. (1997), Active mobile robot localization by entropy minimization, *in* 'Proc. of the 2nd Euromicro Workshop on Advanced Mobile Robots', IEEE/CS.

Cassandra, A. R., Kaelbling, L. P. & Kuiren, J. A. (1996), Acting under uncertainty: Discrete bayesian models for mobile-robot navigation, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'96)', Vol. 2, pp. 963–972.

Castellanos, J. A. & Tardós, J. D. (1999), *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*, Kluwer Academic Publishers.

Castellanos, J., Tardos, J. & Schmidt, G. (1997), Building a global map of the environment of a mobile robot: The importance of correlations, *in* 'Proc. of the IEEE International Conference on Robotics and Automation', pp. 1053–1059.

Chatila, R. (1985), Mobile robot navigation: Space modeling and decisional processes, *in* 'Proc. of 3rd International Sysmposium on Robotics Research', Gouvieux, France, pp. 373–378.

Chatila, R. & Laumond, J.-P. (1985), Position referencing and consistent world modeling for mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)', pp. 138–145.

Cho, D. W. (1990), 'Certainty grid representation for robot navigation by a bayesian method', *Robotica* **8**, 159–165.

Chong, K. S. & Kleeman, L. (1996), Accurate odometry and error modelling for a mobile robot, Technical report, Intelligent Robotics Research Centre (IRRC), Monash University, Clayton Victoria 3168, Australia.

Chong, K. S. & Kleeman, L. (1997), Feature-based mapping in real, large scale environments using an ultrasonic array, *in* 'Proc. of International Conference on Field and Service Robotics', pp. 538–545.

Christensen, H., Kirkeby, N., Kristensen, S. & Knudsen, L. (1994), 'Model-driven vision for in-door navigation', *Robotics and Autonomous Systems* **12**, 199–207.

Cox, H. (1964), 'On the estimation of state variables and parameters for noisy dynamic systems', *IEEE Transactions on Automatic Control* **AC-9**(1), 5–12.

Cox, I. J. (1989), Blanche: Position estimation for an autonomous robot vehicle, *in* 'Proc. of the International Workshop on Intelligent Robots and Systems', IEEE/RSJ, Tsukuba, Japan, pp. 432–439.

Cox, I. J. & Leonard, J. J. (1994), 'Modeling a dynamic environment using a bayesian multiple hypothesis approach', *Artificial Intelligence* **66**, 311–344.

Crowley, J. (1985*a*), Dynamic world modeling for an intelligent mobile robot using a rotating ultra-sonic ranging device, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)', Vol. 1, IEEE, pp. 128–135.

Crowley, J. (1985*b*), 'Navigation for an intelligent mobile robot', *IEEE Journal of Robotics and Automation* **1**(1), 31–41.

Crowley, J. L. (1989), World modeling and position estimation for a mobile robot using ultrasonic ranging, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'89)', pp. 674–680.

Crowley, J. L. (1995), 'Mathematical foundations of navigation and perception for an autonomous mobile robot', Workshop on Reasoning with Uncertainty in Robotics.

Crowley, J. L., Stelmaszyk, P., Skordas, T. & Puget, P. (1992), 'Measurement and integration of 3-d structures by tracking edge lines', *International Journal of Computer Vision* **8**(1), 29–52.

Crowley, J. L., Wallner, F. & Schiele, B. (1998), Position estimation using principal components of range data, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)', IEEE, Leuven, Belgium, pp. 3121–3128.

Csorba, M. & Durrant-Whyte, H. (1997), A new approach to map building using relative position estimates, *in* 'Proc. of SPIE', Vol. 3087, pp. 115–125.

Csorba, M., Uhlmann, J. K. & Durrant-Whyte, H. F. (1997), A sub optimal algorithm for automatic map building, *in* 'Proc. of the American Control Conference', Albuquerque, New Mexico, pp. 537–541.

Dellaert, F., Burgard, W., Fox, D. & Thrun, S. (1999), Using the CONDENSATION algorihm for robust, vision-based mobile robot localization, *in* 'Proc. of the IEEE Computer Society Conference of Computer Vision and Pattern Recognition', Vol. 2.

Dellaert, F., Fox, D., Burgard, W. & Thrun, S. (1999), Monte carlo localization for mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)', Vol. 2, Detroit, Michigan, USA, pp. 1322 – 1328.

Deriche, R., Vaillant, R. & Faugeras, O. (1992), *From Noisy Edges Points to 3D Reconstruction of a Scene : A Robust Approach and Its Uncertainty Analysis*, Vol. 2, World Scientific, pp. 71–79. Series in Machine Perception and Artificial Intelligence.

Dissanayake, G., Durrant-Whyte, H. & Bailey, T. (2000), A compuationally effecient solution to the simultaneous localization and map building (slam) problem, *in* 'Proc. of the IEEE International Conference on Robotics and Automation', pp. 1009–1014.

Doucet, A. (1998), On sequential simulation-based methods for bayesian filtering, Technical Report CUED/F-INFENG/TR.310, Signal Processing Group, Department of Engineering, University of Cambridge, CB2 1PZ Cambridge. http://www-sigproc.eng.cam.ac.uk/smc/smcpapers.html.

Drumheller, M. (1987), 'Mobile robot localization using sonar', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-9**(2), 325–332.

Duda, R. O. & Hart, P. E. (1973), *Pattern Calssification and Scene Analysis*, John Wiley & Sons, Stanford Research Institute, Menlo Park, California.

Durrant-Whyte, H. (1988), 'Sensor models and multisensor integration', *The International Journal of Robotics Research* **7**(6), 97–113.

Durrant-Whyte, H. (1994), 'Where am i', *Industrial Robot* **21**, 11–16.

Elfes, A. (1989), A Probabilistic Framework for Robot Perception and Navigation, PhD thesis, Carnegie-Mellon University.

Engelson, S. P. & McDermott, D. V. (1992), Error correction in mobile robot map learning, *in* 'Proc. of the IEEE/RSJ International Conference on Robotics and Automation (IROS'92)', Vol. 3, pp. 2555–2560.

Fabrizi, E. & Saffiotti, A. (2000), Extracting topology-based maps from gridmaps, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 3, San Francisco, CA, USA, pp. 2972–2978.

Faugeras, O., Ayache, N. & Faverjon, B. (1986), Building visual maps by combining noisy stereo measurements, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'86)', pp. 1433–1438.

Feder, H. J. S. (1999), Simultaneous Stochastic Mapping and Localization, PhD thesis, MIT.

Forsberg, J., Åhman, P. & Wernersson, Å. (1993), The hough transform inside the feedback loop of a mobile robot, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'93)', Vol. 1, IEEE, pp. 791–798.

Forsberg, J., Larsson, U., Åhman, P. & Wernersson, Å. (1993), Navigation in cluttered rooms using a range measuring laser and the hough transform, *in* 'Proc. of the International Conference on Intelligent Autonomous Systems', pp. 248–257.

Fortmann, T. E., Bar-Shalom, Y. & Scheffe, M. (1983), 'Sonar tracking of multiple targets using joint probabilistic data association', *IEEE Journal of Oceanic Engineering* **OE-8**(3), 173–184.

Fox, D., Burgard, W., Dellaert, F. & Thrun, S. (1999), Monte carlo localization - efficient position estimation for mobile robots, *in* 'Proc. of the National Conference on Artificial Intelligence (AAAI-99)', Orlando, Florida, USA.

Fox, D., Burgard, W. & Thrun, S. (1997), 'The dynamic window approach to collision avoidance', *IEEE Robotics & Automation Magazine* **4**(1), 23–33.

Fox, D., Burgard, W. & Thrun, S. (1998), 'Active markov localization for mobile robots', *Robotics and Autonomous Systems* **25**, 195–207.

Fox, D., Burgard, W. & Thrun, S. (1999), 'Markov localization for mobile robots in dynamic environments', *Journal of Artificial Intelligence Research* **11**, 391–427.

Gelb, A., ed. (1974), *Applied Optimal Estimation*, MIT Press.

Gordon, N., Salmond, D. & Smith, A. (1993), 'Novel approach to nonlinear/non-gaussian bayesian state estimation', *IEE PROCEEDINGS-F* **140**(2), 107–113.

Gutmann, J.-S., Burgard, W., Fox, D. & Konolige, K. (1998), An experimental comparison of localization methods, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)', Vol. 2, pp. 736–743.

Gutmann, J.-S. & Schlegel, C. (1996), Amos: comparison of scan matching approaches for self-localization in indoor environments, *in* 'Proc. of the First Euromicro on Advanced Mobile Robot', pp. 61–67.

Gutmann, J.-S., Weigel, T. & Nebel, B. (1999), Fast, accurate and robust self-localization in polygonal environments, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)', IEEE/RSJ, pp. 1412–1419.

Hager, G. & Mintz, M. (1990), Sensor modeling and robust sensor data fusion, *in* 'Proc. of the International Symposium on Robotics Research', pp. 69–74.

Handschin, J. (1970), 'Monte carlo techniques for prediction and filtering of non-linear stochastic processes', *Automatica* **6**, 555–563.

Handschin, J. & Mayne, D. (1969), 'Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering', *Internaltional Journal on Control* **9**(5), 547–559.

Hébert, P., Betgé-Brezetz, S. & Chatila, R. (1996), Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: The use of local maps, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'96)', pp. 757–764.

Hecht, E. (1987), *Optics*, 2nd edn, Addison-Wesley Publishing Company.

Hinkel, R. & Knieriemen, T. (1988), Environment perception with a laser radar in a fast moving robot, *in* 'Symposium on Robot Control', pp. 68.1–68.7.

Hoppen, P., Knieriemen, T. & von Puttkamer, E. (1990), Laser-radar based mapping and navigation for an autonomous mobile robot, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'90)', Vol. 2, IEEE, pp. 948–953.

Horswill, I. (1998), *Artificial intelligence and mobile robots: case studies of successful robot systems*, The AAAI Press/The MIT Press, 445 Burgess Drive, Menlo Park, CA 94025, chapter The Polly System, pp. 73–90.

Hough, P. (1962), A method and means for recognizing complex patterns. U.S. Patent, Number 3,069,654.

Hu, H., Brady, M. & Probert, P. (1991), Coping with uncertainty in control and planning for a mobile robot, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'91)', Vol. 2, Osaka, Japan, pp. 1025–1030.

Illingworth, J. & Kittler, J. (1988), 'A survey of the hough transform', *Computer Vision, Graphics and Image Processing* **43**, 87–116.

Isard, M. & Blake, A. (1996), Contour tracking by stochastic propagation of conditional density, *in* 'Proc. of the European Conference on Computer Vision', Cambridge, UK, pp. 343–356.

Isard, M. & Blake, A. (1998), 'CONDENSATION – conditional density propagation for visual trackoing', *International Journal on Computer Vision* **29**(1), 5–28.

Jazwinski, A. H. (1970), *Stochastic Processes and Filtering Theory*, Vol. 64 of *Mathematics in Science and Engineering*, Academic Press.

Jensfelt, P. (1999), Localization using laser scanning and minimalistic environmental models, Licentiate thesis, Automatic Control, Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

Jensfelt, P., Austin, D. & Christensen, H. I. (2000), Toward task oriented localization, *in* 'The 6th Int. Conf. on Intelligent Autonomous Systems (IAS-6)', pp. 612–619.

Jensfelt, P., Austin, D., Wijk, O. & Andersson, M. (2000), Feature based condensation for mobile robot localization, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 3, San Francisco, CA, USA.

Jensfelt, P. & Christensen, H. (1998), Laser based position acquisition and tracking in an indoor environment, *in* 'Proc. of the International Symposium on Robotics and Automation', Vol. 1, IEEE, Saltillo, Coahuila, Mexico, pp. 331–338.

Jensfelt, P. & Christensen, H. (1999), Laser based pose tracking, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)', Vol. 4, IEEE, Detroit, Michigan, USA, pp. 2994–3000.

Jensfelt, P. & Christensen, H. I. (2001), 'Pose tracking using laser scanning and minimalistic environmental models', *to appear in IEEE Transactions on Robotics and Automation* .

Jensfelt, P. & Kristensen, S. (1999), Active global localisation for a mobile robot using multiple hypothesis tracking, *in* 'Proc. of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation (avail at http://www.dsv.su.se/ijcai-99/)', Stockholm, Sweden, pp. 13–22.

Jensfelt, P. & Kristensen, S. (2001), 'Active global localisation for a mobile robot using multiple hypothesis tracking', *accepted for IEEE Transactions on Robotics and Automation* .

Jensfelt, P., Wijk, O., Austin, D. & Andersson, M. (2000), Experiments on augmenting condensation for mobile robot localization, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 3, San Francisco, CA, USA, pp. 2518–2524.

Kailath, T., Sayed, A. & Hassibi, B. (2000), *Linear Estimation*, Prentice Hall. ISBN: 0-13-022464-2.

Kalman, R. (1960), 'A new approach to linear filtering and prediction problems', *Transactions of the ASME* **82**, 35–45.

Koenig, S. & Simmons, R. G. (1998), *Artificial intelligence and mobile robots: case studies of successful robot systems*, The AAAI Press/The MIT Press, 445 Burgess Drive, Menlo Park, CA 94025, chapter Xavier: A Robot Navigation Architeture Based on Partially Observable Markov Process Models, pp. 91–122.

Kortenkamp, D. & Weymouth, T. (1994), Topological mapping for mobile robots using a combination of sonar and vision sensing, *in* 'Proc. of the National Conference on Artificial Intelligence (AAAI-94)'.

Krishnamurthy, V. (2000), Adaptive sensor based signal processing. Lecturing notes fall 2000.

Kuc, R. & Siegel, M. (1987), 'Physically based simulation model for aucoustic sensor robot navigation', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-9**(6), 766–778.

Kuipers, B. (1978), 'Modeling spatial knowledge', *Cognitive Science* **2**, 129–153.

Kuipers, B. & Byun, Y.-T. (1991), 'A robot exploration and mapping strategy based on a semantic hierachy of spatial representations', *Journal of Robotics and Autonomous Systems* **8**, 47–63.

Kuipers, B. J. (1977), Representing knowledge of large-scale space, Technical Report TR-418 (revised version of Doctoral thesis May 1977, MIT Mathematical Department), MIT Artificial Intelligence Laboratory.

Larsson, U., Forsberg, J. & Wernersson, Å. (1994), On robot navigation using identical landmarks: Interesting measurements from a time-of-flight laser, *in* 'In Proc. of Multisensor Fusion and Integration for Intelligent Systems', pp. 17–26.

Lenser, S. & Veloso, M. (2000), Sensor resetting localization for poorly modelled mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 2, pp. 1225–1232.

Leonard, J. & Durrant-Whyte, H. (1991*a*), 'Mobile robot localization by tracking geometric beacons', *IEEE Transactions on Robotics and Automation* **7**(3), 376–382.

Leonard, J. J., Durran-Whyte, H. F. & Cox, I. J. (1992), 'Dynamic map building for an autonomous mobile robot', *The International Journal of Robotics Research* **11**(4), 286–298.

Leonard, J. J. & Durrant-Whyte, H. F. (1991*b*), Simultaneous map building and localization for an autonomous mobile robot, *in* 'Proc. of the International Workshop on Intelligent Robots and Systems', Vol. 3, IEEE, Osaka, Japan, pp. 1442–1447.

Leonard, J. J. & Durrant-Whyte, H. F. (1992), *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publisher, Boston.

Leonard, J. J. & Feder, H. J. S. (1999), A computationally effecient mathod for large-scale concurrent mapping and localization, *in* 'Proc. 9th International Symposium on Robotics Research'.

Liu, J. S. & Chen, R. (1998), 'Sequential monte carlo methods for dynamic systems', *Journal of the American Statistical Association* **93**(443), 1032–1044.

Ljunggren Klöör, P. & Wernersson, Å. (1992*a*), On estimating a robot's motion from laser range measurements using the distance transform, *in* 'Proc. of the International Conference on Pattern Recognition', Vol. 1, pp. 772–76.

Ljunggren Klöör, P. & Wernersson, Å. (1992*b*), On motion estimation for a mobile robot navigating in a natural environment: Matching laser range measurements using the distance transform, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'92)', IEEE/RSJ, Raleigh, NC, pp. 1421–28.

Lu, F. & Milios, E. (1995), Optimal global pose estimation for consistent sensor data registration, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'95)', pp. 93–100.

Lu, F. & Milios, E. (1997*a*), 'Globally consistent range scan alignment for environmental mapping', *Autonomous Robots* **4**, 333–349.

Lu, F. & Milios, E. (1997*b*), 'Robot pose estimation in unknown environments by matching 2d range sacns', *Journal of Intelligent and Robotic Systems* **18**, 249–275.

Mackay, D. (1996), 'Introduction to monte carlo methods', In Proc. of Erice summer school. http://wol.ra.phy.cam.ac.uk/mackay/BayesMC.html.

Maybeck, P. S. (1979), *Stochastic models, estimation and control*, Vol. 1, Academic Press.

Mazor, E., Averbuch, A., Bar-Shalom, Y. & Dayan, J. (1998), 'Interacting multiple model methods in target tracking: A survey', *IEEE Transactions on Aerospace and Electronic Systems* **34**(1), 103–123.

Moravec, H. & Elfes, A. (1985), High resolution maps from wide angle sonar, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'85)', IEEE, pp. 116–121.

Moutarlier, P. & Chatila, R. (1990), Stochastic multisensory data fusion for mobile robot location and environmental modelling, *in* 'Proc. of the International Symposium on Robotics Research', pp. 85–94.

Murase, H. & Nayar, S. K. (1995), 'Visual learning and recognition of 3-d objects from appearance', *International Journal of Computer Vision* **14**(1), 5–24.

Nashashibi, F., Devy, M. & Fillatreau, P. (1992), Indoor scene terrain modeling using multiple range images for autonomous mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'92)', Vol. 1, IEEE, pp. 40–46.

NDC (n.d.), 'Netzler & dahlgren co ab', http://www.ndc.se/.

Neal, R. M. (1993), Probabilistic inference using markov chain monte carlo methods, Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Nourbakhsh, I. (1998), *Artificial intelligence and mobile robots: case studies of successful robot systems*, The AAAI Press/The MIT Press, 445 Burgess Drive, Menlo Park, CA 94025, chapter Dervish: An Office Navigation Robot, pp. 73–90.

Nourbakhsh, I., Powers, R. & Birchfield, S. (1995), 'Dervish: An office navigation robot', *AI Magazine* **16**(2), 53–60.

Oriolo, G., Vendittelli, M. & Ulivi, G. (1995), On-line map building and navigation for autonomous mobile robots, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'95)', Vol. 3, pp. 2900–2906.

Pitt, M. K. & Shephard, N. (1999), 'Filtering via simulation: auxiliary particle filters', *Journal of the American Statistical Association* **94**.

Pol (n.d.), *Polaroid Ultrasonic Ranging System User's Manual*.

Raschke, U. & Borenstein, J. (1990), A comparison of grid-type map-building techniques by index of performance, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'90)', Vol. 3, IEEE, pp. 1828–1832.

Rencken, W. (1993), Concurrent localization and map building for mobile robots using ultrasonic sensors, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)', IEEE/RSJ, Yokohama, Japan, pp. 2192–2197.

Rencken, W. (1994), Autonomous sonar navigation in indoor, unknown and unstructured environments, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)', Vol. 3, IEEE, pp. 431–438.

Reuter, J. (2000), Mobile robot self-localization using pdab, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 4, IEEE, pp. 3512–3518.

Roumeliotis, S. I. & Bekey, G. A. (2000), Bayesian estimation and kalman filtering: A unified framework for mobile robot localization, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 3, IEEE, pp. 2985–2992.

Rubin, D. B. (1987), 'A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The sir algorithm', *Journal of the American Statistical Association* **82**(398), 543–546.

Schiele, B. & Crowley, J. L. (1994), A comparison of position estimation techniques using occupancy grids, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'94)', Vol. 2, IEEE, pp. 1628–1634.

Seiz, M., Jensfelt, P. & Christensen, H. I. (2000), Active exploration for feature based global localization, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)"', Vol. 1, pp. 281–287.

Sim, R. & Dudek, G. (1999), Learning visual landmarks for pose estimation, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99)', Vol. 3, pp. 1972–1978.

Simmons, R. & Koenig, S. (1995), Probabilistic navigation in partially observable environments, *in* 'Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'95)', pp. 1080–1087.

Smith, A. & Gelfand, A. (1992), 'Bayesian statistics without tears: A sampling-resampling perspective', *The American Statistician* **46**(2), 84–88.

Smith, R., Self, M. & Cheeseman, P. (1987), A stochastic map for uncertain spatial relationships, *in* '4th International Symposium on Robotics Research'.

Sorenson, H. & Alspach, D. (1971), 'Recursive bayesian estimation using gaussian sums', *Automatica* **7**, 465–479.

Sorenson, H. & Stubberud, A. (1968), 'Non-linear filtering by approximation of the *a posteriori* density', *International Journal on Control* **8**(1), 33–51.

Thrun, S. (1998), Finding landmarks for mobile robot navigation, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)', Vol. 2, Leuven, Belgium, pp. 958–963.

Thrun, S. & Bücken, A. (1996), Integrating grid-based and topological maps for mobile robot navigation, *in* 'Proc. of the National Conference on Artificial Intelligence (AAAI-96)', Portland, Oregon, USA, pp. 944–950.

Thrun, S., Burgard, W. & Fox, D. (2000), A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 1, San Francisco, CA, USA, pp. 321–328.

Thrun, S., Fox, D. & Burgard, W. (2000), Monte carlo localization with mixture proposal distribution, *in* 'Proc. of the National Conference on Artificial Intelligence (AAAI)', Austin, Texas, USA.

Uhlmann, J. K. (1995), Dynamic Map Building and Localization: New Theretical Foundations, PhD thesis, University of Oxford, Robotics Research Group, Department of Engineering Science.

Ulrich, I. & Nourbakhsh, I. (2000), Appearance-based place recognition for topological localization, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'00)', Vol. 2, pp. 1023–1029.

Wallner, F. (1997), Position Estimation for a Mobile Robot from Principal Components of Laseer Data, PhD thesis, INPG.

Weckesser, P. & Dillmann, R. (1997), Navigating a mobile service-robot in a natural environment using sensor-fusion techniques, *in* 'Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)', Vol. 3, pp. 1423–1428.

Weiss, G. & von Puttkamer, E. (1995), 'A map based on laserscans without geometric interpretation', *Intelligent Autonomous Systems* pp. 403–407.

Welch, G. & Bishop, G. (2001), An introduction to the kalman filter, Technical Report TR 95-041, Department of Computer Science, University of Narth Carolina at Chapel Hill, Chapel Hill, NC 27599-3175. http://www.cs.unc.edu/ welch/kalman/.

Wijk, O. (1998), Navigation of mobile robots using natural landmarks extracted form sonar data, Licentiate thesis, Automatic Control, Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

Wijk, O. (2001), Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization, PhD thesis, Royal Institue of Technology, Stockholm, Sweden.

Wijk, O. & Christensen, H. (2000), 'Triangulation based fusion of sonar data with application in robot pose tracking', *IEEE Transaction on Robotics and Automation* **16**(6), 740–752.

Wijk, O., Jensfelt, P. & Christensen, H. (1998), Triangulation based fusion of ultrasonic sensor data, *in* 'Proc. of the IEEE International Conference on Robotics and Automation (ICRA'98)', Vol. 4, IEEE, Leuven, Belgium, pp. 3419–24.

Åström, K. (1996), Invariancy Methods for Points, Curves and Surfaces in Computational Vision, PhD thesis, Lund University, Department of Mathematics, Lund Institute of Technology, P.O. Box 118, SE-221 00 Lund, Sweden. ISBN 91-628-2022-2.