Detailed Environment Representation for the SLAM Problem

Juan I. Nieto

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

Australian Centre for Field Robotics
Department of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

March 2005

Declaration

This thesis is submitted to the University of Sydney in fulfillment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Juan I. Nieto

March, 2005

Abstract

Juan I. Nieto
The University of Sydney

Doctor of Philosophy
March 2005

Detailed Environment Representation for the SLAM Problem

This thesis addresses the environment representation problem for Simultaneous Localisation and Mapping (SLAM) algorithms.

One of the main problems of SLAM is how to interpret and synthesize external sensory information into a representation of the environment that can be used by the mobile robot to operate autonomously. Traditionally, SLAM algorithms have relied on sparse environment representations. However, for autonomous navigation, a more detailed representation of the environment is necessary, and the classic feature-based representation fails to provide a robot with sufficient information. While a dense representation is desirable, it has not been possible for SLAM paradigms.

This thesis deals with the problem of obtaining detailed environment representations for the Simultaneous Localisation and Mapping problem. The thesis introduces the concept of *DenseSLAM*, which is defined as the process of simultaneous localisation and *dense* mapping. A solution for DenseSLAM named *Hybrid Metric Maps* (HYMMs) is presented. HYMMs combine feature maps with other dense metric representations such as occupancy grids. The global feature map is partitioned into a set of connected local regions, which provide a reference for a detailed multi-dimensional description of the environment. The combination of dense metric maps with feature maps has complementary properties. The stochastic feature map ensures the dense map remains consistent, and the dense map provides auxiliary information to assist navigational capabilities, such as data association, landmark extraction and path planning.

In addition to maintaining dense environment representations, the new algorithm is able to obtain *multi-layered maps*, where each layer represents different properties of the environment, such as occupancy, traversability, elevation, etc.

The classic feature-based representation relies on simple geometric models to define the landmarks. This limits EKF-SLAM to environments suited to such models. In order to overcome this restriction this thesis presents an algorithm which, rather than using geometric models to define landmarks, uses templates of raw sensed data. Complex landmarks can be identified and extracted as they are observed from multiple vantage points. The algorithm uses scan alignment in order to create observations.

This thesis presents simulation and experimental results in outdoor environments using data collected with a vehicle equipped with odometry and laser sensors.

Acknowledgements

I would like to thank all of the people who helped and supported me during my studies at the ACFR.

I want to thank first to my supervisor Eduardo Nebot for his availability, motivation, guidance and support during my PhD studies. He was not only my PhD supervisor but someone who was always available and willing to help in any problem I could have.

To Jose Guivant, for sharing with me all his expertise and the most important his friendship.

To Tim Bailey, for all his help and the guitar lessons that hopefully I will get some day.

To the rest of the ACFR members. Special thanks to Chris Mifsud for all his help with the hardware in the UTE and with Acumine. Also to Christy Wang and Ruth Olip for their assistance with the administrative stuff.

I would like to thank to my many friends in Australia who made me feel at home with their company and support during all these years. These include Alejandra, Claudette, Florencia, Olga, Noelia, Ricardo, Felipe, Gerold, Ollie, Stewart, Fabio, Trevor, Favio, Kim and many others that I met during all these years in Australia and I may have forgotten now, thanks!.

To Konstanze, for her support and love and for bringing me happiness.

Thank you to Stefan Williams, Patric Jensfelt and Simon Lacroix for their review of this thesis. Their comments were greatly appreciated.

Finally, my greatest thanks for my family. To my parents, for their support and love and for believing in me. To my brother and sisters, for their company, entertainment and love during many years.

To my family, with love.
A mi familia, con amor.

Contents

List of Figures

Chapter 1

Introduction

> *Simultaneous Localisation and Map building (SLAM) algorithms investigate the problem of rendering an autonomous vehicle in an unknown location in an unknown environment, and then to incrementally build a map of this environment while simultaneously using this map to compute absolute vehicle location.*
> *Paul Newman [59]*

A solution to the SLAM problem is necessary to make a robot truly autonomous. For this reason, SLAM has been one of the main research topics in robotics, especially during the last ten years.

Despite recent big achievements in SLAM systems, very little has been investigated concerning possible environment representations for SLAM algorithms, apart from the classic feature-based form. In many cases, the sparse representation maintained by the classic feature-based SLAM algorithms is enough for localisation. But if the goal is autonomous navigation, a sparse representation fails to provide sufficient information.

A detailed environment representation is not only necessary to plan the vehicle trajectory or make decisions, but it can also reduce problems involved with other SLAM issues, such as data association and error accumulation due to non-linearities. The development of a more comprehensive description of the environment is a prerequisite for autonomous navigation.

The aim of this thesis is to develop algorithms that allow a mobile robot to obtain detailed environment descriptions for the Simultaneous Localisation and Mapping problem.

1.1 A Brief History of Navigation

Navigation is the science of effecting safe and precise traversal of a vehicle, such as a ship, aircraft, or spacecraft, from one place to another; especially the method of determining position, course, and distance travelled.

Today, reliable navigation can be achieved with the use of satellites and sensors such as radars and lasers. However, our ancestors did not count on satellites or radars, but that did not stop them from exploring “New Worlds”.

The Egyptian Harkuf is thought to be the first documented explorer around 2300 BC [2, 75][1]. He led expeditions up the Nile to the land of Yam.

Around 1000 BC, the Polynesians were exploring islands in the Pacific Ocean. They were famous for their navigation skills. As part of their navigational techniques, they used stars, currents of the waves and migratory seabirds to find the course through thousands of miles.

The greeks were also excellent navigators. Around 340 BC, Pytheas led an expedition, which led him through the Straits of Gibraltar. He invented an accurate method of calculating latitude using a calibrated sundial and theorized the relationship between tides and the phases of the moon.

Before the invention of the magnetized compass, seafarers from Greece to China relied on winds to locate their orientation. The Greeks named eight different winds, while the Chinese devised a system of 24 seasonal winds and measured them with vanes and kites.

Some of the most extraordinary seafarers were the Vikings. Around the year 1000 AD they discovered a coast that they named Vinland for its abundance vines and berries. These areas are now part of Canada’s Labrador. The Vikings had discovered America, five centuries before Columbus. They did not have a compass, but they relied on tools from nature to assist them in their navigation. Observations of the sun and Pole Star, currents, winds and migratory whales and birds were clues for direction. They used sun compasses and shadow boards to work out their latitude.

Around the year 1100 AD, the Chinese introduced the magnetic compass. Navigators would take an iron needle, rub its point on the lodestone and then place it in a bowl of water.

For a course to be worked out, sailors needed two pieces of information: their position on the north-south axis (latitude) and on the east-west axis (longitude). While the latitude could be estimated using the sun, moon and stars, using instruments like the astrolabe, the backstaff and the octant which came into more common use in the 18th century, there was a lack of a reliable method to estimate longitude. To measure longitude, they needed to know the time difference between where they were standing, and some other known fixed point.

The 15th century saw the beginning of European exploration to the New World. During the 15th and 16th centuries, more and more ships set out to conquer and explore new territories. The power of seafaring nations came to depend on accurate navigation. However,

[1] This section was composed with paragraphs and information extracted from [2, 75].

accurate navigation based on charts and precise timekeeping eluded even the great explorers. Christopher Columbus's first voyage in 1492 took him to the Bahamas, Cuba and Hispanola. His objective was to reach the East Indies by sailing through the Atlantic, but his lack of knowledge about longitude made him to make bad calculations. He thought 2,278 miles separated Asia from the west of Europe, a quarter of the actual distance.

During the 17th century, the quest for finding an accurate method of longitude became critical. Latitude could be figured out by determining the angular of position from the Pole Star, but longitude was still a problem. Navigators knew that being able to keep track of time was important for finding longitude, but an accurate time piece had still not been developed. Other measurement methods were attempted, such as lunar-distance or magnetic variation, but none of these proved practical.

The lack of an accurate, reliable and practical method of determining longitude had serious consequences for sailors. Maritime disasters were numerous. The British government was desperate, and they offered a prize to whomever could come up with a method for determining longitude. A marine timekeeper would have to be accurate, but given the state of clock-making technology at this time, this seemed nearly impossible to achieve. It was not until, John Harrison, who spent nearly fifty years to finally obtain an accurate timepiece he called chronometer.

During his second voyage to the Pacific from 1772-1775, James Cook carried with him a copy of John Harrison's H4 chronometer which he was able to use to estimate his longitude.

1.2 Navigation Today

Two hundred years later to Cook's second voyage to the Pacific, the US launched the first Global Positioning System (GPS) Block I satellites, Navstars 1-4 (1978). Today, the fully operational GPS includes 28 satellites.

A second configuration for global positioning called GLONASS was launched by the former Soviet Union some years later. GLONASS uses 24 satellites. Both, GPS and GLONASS have evolved from dedicated military systems into true dual-systems for military and civilian applications. Today, satellite navigation technology is utilised in numerous civilian and military applications, ranging from golf to spacecraft aviation [25]. In the future, mobile robots will be involved in different tasks, ranging from service robotics areas such as logistics, human assistance, cleaning and health care to field robotics applications such as mining and defence. Any of these tasks will require a reliable navigation system that allows the robot to operate autonomously without a human operator.

Unfortunately, satellite positioning is not available anytime, anywhere. Applications in indoor or underwater environments for example, cannot rely on GPS. Alternate means of localisation use maps of the environment. By fitting the vehicle with sensors capable of taking observations of the vehicle's surrounding, they are able to estimate the vehicle position in the given map. The next sections explain briefly the problems of localisation, and simultaneous localisation and mapping when absolute position information is not available and the vehicle only relies on observations taken by on-board sensors.

1.2.1 Localisation

Dead-reckoning is the most basic form of localisation. Dead-reckoning consists in the integration of a vehicle motion model over time. A motion model calculates the vehicle motion based on some sensory information such as velocity or acceleration. The problem with dead-reckoning is that each time the motion model is integrated, the error in the control inputs and model is integrated as well. Eventually, the estimated pose will become so uncertain that it will not be useful for any purpose, and other mechanisms for localisation are required.

One mechanism to estimate the vehicle pose with bounded uncertainty is through the use of an external map of the environment. A map of the environment could be defined by the position of distinctive landmarks. Using a sensor that gives relative information about its position with respect to these landmarks, the vehicle can estimate its position. This is the localisation problem.

More challenging is the problem of global localisation [36, 50], where a robot is not told about its initial pose but instead has to determine it from scratch. Consequently, the robot should be able to handle multiple hypothesis about the initial pose.

The problem with localisation using an *a priori* map is the necessity of exploring previously the area and building a map of the environment, something that is not possible in many circumstances, where the robot is sent to an unknown environment and is in an unknown location. Simultaneous localisation and mapping algorithms present a solution for this problem.

1.2.2 Simultaneous Localisation and Mapping (SLAM)

Map building is the process of obtaining a representation of the environment, using external sensory information. A mobile robot can use a wide variety of sensors to obtain information about the environment. Typical sensors are cameras, lasers, radars and sonars. Since the observations are taken from sensors on-board the robot, an estimate of the robot pose is necessary in order to build the map.

Building a representation of the environment with an autonomous robot, given the robot pose [14, 18, 41] is considered a problem solved today. However, the robot pose will in general be unknown, and the mapping process will have to be done simultaneously with the robot localisation.

Simultaneous localisation and mapping algorithms provide a mean to localise the robot taken external observations of the environment. Since SLAM algorithms build a map simultaneously with the localisation process, they do not require an a priori map. The SLAM problem was introduced eighteen years ago [73]. This seminal paper explained the need for concurrently estimating the vehicle pose and map position, and presented a stochastic solution based on the use of an Extended Kalman Filter (EKF), which foundations have served as the basis for most of the current solutions. The vehicle takes observations of the environment, by the use of external sensors such as range lasers, and identifies features that are incorporated in the map as landmarks. EKF-SLAM maintains a state vector with the robot pose and landmark positions. The map is then constructed as an on-line data fusion problem and an estimate of uncertainties in the robot pose and landmark locations are maintained.

One of the main issues in SLAM algorithms is the computational cost. It will be shown that the computations required grow quadratically with the number of landmarks in the map $O(n^2)$. To overcome this problem, efficient solutions appeared [26, 39, 86]. By creating local maps, only the landmarks in a local region have to be updated. The computational cost is then quadratic with the number of landmarks in the vehicle's vicinity. The local maps are periodically fused with the global map. Sub-optimal SLAM algorithms that are able to reduce the computational cost to constant time $O(1)$ have also appeared [1, 6].

An important issue with stochastic SLAM is environment modelling. SLAM maps are typically sparse and are built up of isolated landmarks observed in the environment. This makes the representation appropriate only for localisation purposes.

In autonomous navigation, the environment representation is actually required for several different purposes; localisation, path planning, detection of particular properties in the environment [42]. A sparse representation formed only by isolated landmarks will clearly not fulfill the necessities of an autonomous vehicle, and a more detailed representation will be needed. Furthermore, not only a dense representation of the environment is required, but also an algorithm that is able to obtain *multi-layered maps*, where each layer represents different properties of the environment, such as occupancy, traversability, detection of water, elevation, etc.

In summary, the development of algorithms able to obtain *multi-layered dense* maps is a prerequisite for fully autonomous navigation.

1.3 Contributions

This thesis addresses issues related to the problem of obtaining detailed environment representation for SLAM algorithms. The thesis has several contributions.

- A novel algorithm which allows the construction of detailed descriptions of the environment. The new algorithm referred as *Hybrid Metric Maps* (HYMMs) combines feature maps with other dense metric sensory representation. The global feature map is partitioned into a set of connected Local Triangular Regions (LTRs), which provide a reference for a detailed multi-dimensional description of the environment. The HYMM framework permits the combination of efficient feature-based SLAM algorithms for localisation with, for example, occupancy grid (OG) maps. This fusion of feature and grid maps has several complementary properties; for example, grid maps can assist data association and can facilitate the extraction and incorporation of new landmarks as they become identified from multiple vantage points.

- EKF-SLAM is currently the most popular filter used to solve stochastic SLAM. An important issue with EKF-SLAM is that it requires sensory information to be modelled as geometric shapes. A new algorithm is presented which rather than relying on geometric features to define the landmarks, it uses templates of raw sensed data, and utilises scan correlation algorithms to produce landmark observations.

- A more comprehensive environment representation will allow to improve the vehicle localisation process. For example, the vehicle could use the dense representation to extract and incorporate complex landmarks as they become identified. Ambiguities in association could also be eliminated by the incorporation of more information into the landmark definition. Dynamic objects could be detected by differencing maps obtained at different time. It will be shown how the dense maps can be used to improve the overall vehicle navigation process.

- Every chapter presents simulation and experimental results in outdoor environments. The simulation results are important since is possible to compare the estimates with the ground truth. The experimental results allow to verify the algorithms presented in practical environments.

1.4 Thesis Overview

Chapter 2 presents the Simultaneous Localisation and Mapping problem. The chapter also examines some of the solutions presented recently. It will be seen that while most of these solutions tackle the computational problem of SLAM, they are able to obtain only sparse environment representations. To cover the lack of research in environment representation, **Chapter 3** introduces the concept of DenseSLAM, defined as the process of *Simultaneous Localisation and Dense Mapping*. This chapter also presents a solution for the DenseSLAM problem, named *Hybrid Metric Maps* (HYMMs). The HYYMs is an algorithm that unlike traditional SLAM approaches, allows to obtain *dense* environment representations. In addition to richer environment representations, the HYMMs permit to obtain several map layers, each layer representing the same sensed data in different form or different environment properties. In order to show the flexibility of the algorithm to the use of different representations for the sensory information, **Chapter 4** presents simulation and experimental results of DenseSLAM using three different paradigms.

DenseSLAM improves the environment representation, however, similar to the classic SLAM algorithms, it is based on a features map. Feature-based SLAM approaches rely on geometric models to define the landmarks. To overcome this restriction, **Chapter 5** presents an algorithm which rather than relying on geometric models to define the landmarks, uses raw sensor information to define landmark templates. The advantages of fusing the non-geometric landmarks algorithm with DenseSLAM are explained in **Chapter 6**. This chapter also examines how a dense environment representation can help to improve the vehicle localisation process. Finally, **Chapter 7** presents conclusions and areas of future research.

Chapter 2

Simultaneous Localisation and Mapping

The problem of simultaneous localisation and mapping (SLAM) has received significant attention over the past two decades in the robotics community. As a consequence, different solutions to this problem have appeared especially over the last ten years. The reason for the extensive amount of research in SLAM is because a solution to this problem is necessary to allow autonomous navigation. The number of problems involved in finding an algorithm to concurrently localise the robot and obtain a representation of its surroundings, make SLAM a challenging area of research.

The SLAM problem was formally introduced fifteen years ago in a seminal paper by Smith *et al.* [73]. This work presented a stochastic solution, the foundations of which have served as the basis for most of the current SLAM algorithms, EKF-SLAM being the most popular one. EKF-SLAM presents a recursive solution which allows to incrementally fuse the information gathered by the sensors robot in a representation whilst using this representation to localise the robot.

While most of the solutions proposed are based on the basis of EKF-SLAM [1, 6, 10, 26, 46, 86], alternative solutions have also appeared especially in the last five years. For example, Rao-Blackwellised particle filter based SLAM (FastSLAM) [33, 54], presents an alternative solution that uses particles to represent the vehicle pose distribution and Gaussian representation for the map. Information filters have also been used to approach the SLAM problem. The sparse extended information filters (SEIF) algorithm is presented in [81]. SEIFs represent maps by graphical networks of features that are locally interconnected, where links represent relative information between pairs of nearby features, as well as information about the robots pose relative to the map. Approaches using graphical methods were also presented in [21, 66].

Batch approaches have also appeared, which, instead of estimating the map in a recursive fashion, process all the data collected in a batch manner to obtain a representation of the area explored by the robot [47].

This chapter introduces the SLAM problem and explains the main difficulties that arise when trying to find a solution. A review of some of the optimal solutions that have appeared in the last years is presented, in particular, the EKF solution is thoroughly analysed, as it is the most popular.

The chapter is organised as follows. Section 2.1 introduces the stochastic SLAM problem. Section 2.2 presents the Bayesian solution for SLAM and Section 2.3 explains the classic EKF-SLAM approach. The data association problem is presented in Section 2.4 and the importance of the correlations in SLAM is explained in Section 2.5. The computational complexity of SLAM is discussed in Section 2.6. A review of some of the most promising SLAM approaches introduced in the last years is presented in Section 2.7. Section 2.8 presents a review of the most widely used mapping algorithms. Finally, simulation and outdoor SLAM results are presented.

2.1 Stochastic SLAM

To obtain a representation of the world, robots must possess sensors able to acquire external information. Typical sensors are cameras, sonars, lasers, etc. In addition, a robot usually has sensors to predict its position such as inertial sensors or wheel encoders. Unfortunately, all sensors are subject to errors (sensor noise). Thus any solution to the SLAM problem will need to incorporate the sensor's noise into the models to be able to fuse data taken at different times.

Probabilistic methods are the common factor among SLAM algorithms available today [77]. This is due to the fact that they are able to incorporate the sensor's noise into the models. Furthermore, because the observations of the external world are taken relative to the vehicle position, the uncertainty accumulated over time in the robot pose due to the sensor's noise will be propagated to the map. So the errors in the vehicle position will be correlated with the map. Probabilistic methods are also capable of modelling the correlations between vehicle pose and the map.

Finding a solution to the stochastic SLAM problem implies evaluating the following probability distribution.

$$P(\mathbf{x}_k|\mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \tag{2.1}$$

Where $\mathbf{x}_k$ depicts the joint state vector representing the vehicle pose $\mathbf{x}_k^v$ and the map position $\mathbf{x}_k^m$ at time k.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^v \\ \mathbf{x}_k^m \end{bmatrix} \tag{2.2}$$

$\mathbf{Z}^k$ represents the set of observations received until time k.

$$\mathbf{Z}^k = [\mathbf{z_0}, \mathbf{z_1}, ..., \mathbf{z_k}]^T \tag{2.3}$$

$\mathbf{U}^k$ represents the set of control inputs received until time k.

$$\mathbf{U}^k = [\mathbf{u_0}, \mathbf{u_1}, ..., \mathbf{u_k}]^T \tag{2.4}$$

and $\mathbf{x}_0$ represents the initial conditions.

In order to evaluate Equation 2.1 it is necessary to define probabilistic motion and observation models.

2.1.1 Motion Model

The motion model calculates the vehicle motion between the past states $\mathbf{x}_{k-1}^v$ and the current states $\mathbf{x}_k^v$. The next equation presents the form of a probabilistic motion model.

$$\mathbf{x}_k^v = \mathbf{f}(\mathbf{x}_{k-1}^v, \mathbf{u}_k, \mathbf{v}_k) \tag{2.5}$$

In this general model, the function $\mathbf{f}$ relates the estimated vehicle pose at time k with the previously estimated robot pose at time $k-1$ and the vehicle control inputs $\mathbf{u}_k$, which as mentioned before possess noise, noted as $\mathbf{v}_k$. The estimation of the vehicle position without the aid of external information is usually called dead-reckoning estimation.

The next example shows a classic motion model, the bicycle model.

Example 2.1

Let $\mathbf{x}_k^v = [x_k^v, y_k^v, \phi_k^v]^T$ be the vehicle pose moving in a plane at time k, where (x_k^v, y_k^v) represent the vehicle position in cartesian coordinates and ϕ_k^v the vehicle heading. Let $\mathbf{u}_k = [v_k, \alpha_k]^T$ be the control inputs wheel velocity and steering angle respectively. Finally $\mathbf{v} = [v_x, v_y, v_\alpha]^T$ represents the motion model noise, also called stabilisation noise [1].

Equation 2.6 expresses the bicycle model.

[1] As mentioned, the control inputs are also subject to noise, however for simplicity in the notation the control noise is omitted at this point and only the noise in the motion model is included.

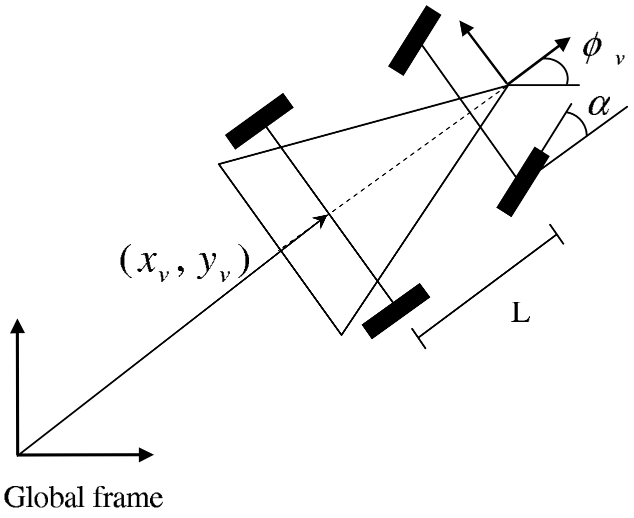


Figure 2.1: Classic bicycle model.

$$\begin{bmatrix} x_k^v \\ y_k^v \\ \phi_k^v \end{bmatrix} = \mathbf{f}(\mathbf{x}_{k-1}^v, \mathbf{u}_k, \mathbf{v}_k) = \begin{bmatrix} x_{k-1}^v + \Delta T v_k cos(\phi_{k-1}^v) \\ y_{k-1}^v + \Delta T v_k sin(\phi_{k-1}^k) \\ \phi_{k-1}^k + \Delta T \frac{v_k}{L} tan(\alpha_k) \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ v_\alpha \end{bmatrix} \tag{2.6}$$

where L is the vehicle length and $\Delta T = t_k - t_{k-1}$ is the integration time interval. Figure 2.1 illustrates the model. Appendix A shows the motion model employed for the experimental vehicle used in this thesis.

2.1.2 Observation Model

Dead-reckoning is the most basic form of localisation. It consists of the integration of Equation 2.5 over time. Every time the motion model is integrated, the noise in the control inputs is integrated as well, hence the uncertainty in the pose estimate increases monotonically. To bound the accumulation of uncertainty, a sensor able to incorporate external information is necessary.

As with the vehicle motion, a probabilistic model is used to model the uncertainty in the observations taken by the vehicle's sensors. The following equation represents a general observation model.

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{w}_k) \tag{2.7}$$

where $\mathbf{h}$ represents a function that relates the observation taken at time k $\mathbf{z}_k$ with the state vector $\mathbf{x}_k$ and the sensor noise vector $\mathbf{w}_k$. Thus, the observation model is a functional relation between the states being observed and the observations noise.

In the context of SLAM, the map is usually represented by point feature landmarks. Algorithms that use the point feature representation fall into the category of feature-based SLAM. The next example illustrates the classic observation model used for feature-base SLAM for a range-bearing sensor and a vehicle moving in a plane.

Example 2.2

Let the environment be represented by two dimensional point landmarks $\mathbf{x}_L = [x_L, y_L]^T$ and the vehicle pose represented using position and heading as shown in the previous section. The next equation shows the observation model in polar coordinates for a two dimensional landmark observed with a range-bearing sensor.

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_L - x_v)^2 + (y_L - y_v)^2} \\ atan(\frac{y_L - y_v}{x_L - x_v}) - \phi_v \end{bmatrix} + \begin{bmatrix} w_r \\ w_\theta \end{bmatrix} \tag{2.8}$$

where $\mathbf{w} = [w_r, w_\theta]^T$ represents the observation sensor noise.

Figure 2.2 illustrates the range-bearing observation model. Appendix A shows the observation model employed for the experimental vehicle used in this thesis.

Equation 2.8 can also be expressed in cartesian coordinates (see [26], Appendix C).

Provided a motion and observation model, a probabilistic SLAM solution can be derived. The next two sections undertake the SLAM problem using both, Bayesian and Gaussian estimation approaches.

2.2 SLAM: Bayesian Solution

The application of Bayesian estimation to the SLAM problem is presented in this section (Appendix B presents a general description of the Bayesian recursive algorithm).

In probabilistic terms, the SLAM problem is a Markov process of order one. This means that the state at time $k-1$ embodies all the necessary information to propagate the system states to time k. This Markov property is at the core of all SLAM solutions. In particular, robot motion is usually considered a Markov process. Its probability distribution is described as

$$P(\mathbf{x}_k^v | \mathbf{x}_{k-1}^v, \mathbf{u}_k) \tag{2.9}$$

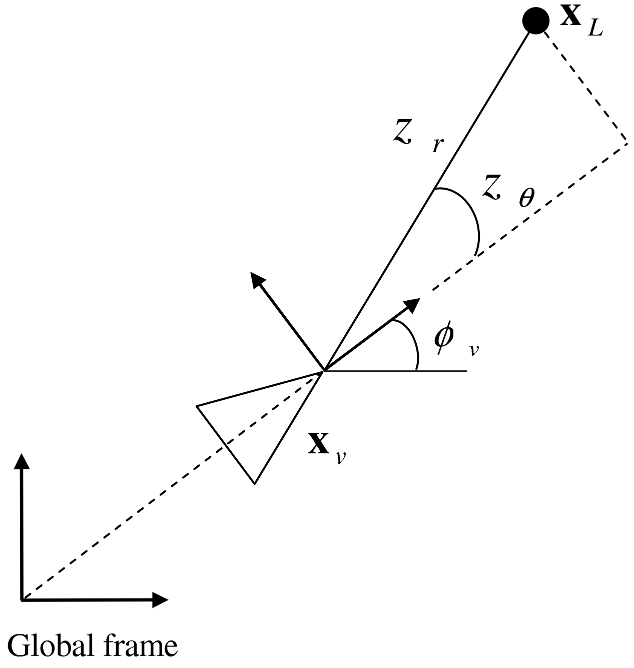


Figure 2.2: Observation model for a range and bearing sensor.

Moreover, the observation model, which relates observations to the state of the world, is described by a probabilistic model that adheres to the Markov property:

$$P(\mathbf{z}_k|\mathbf{x}_k^v, \mathbf{x}_k^m) \tag{2.10}$$

SLAM is then achieved by applying Bayes filtering to solve

$$P(\mathbf{x}_k|\mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \tag{2.11}$$

for all k, where $\mathbf{x}_k = [(\mathbf{x}_k^v)^T, (\mathbf{x}_k^m)^T]^T$ are the states representing the vehicle pose augmented with the states representing the map position. For two dimensional landmarks, $\mathbf{x}^{m_i} = [x_{L_i}, y_{L_i}]^T$.

$$\mathbf{x}^m = [x_{L_1}, y_{L_1}, ..., x_{L_i}, y_{L_i}, ..., x_{L_n}, y_{L_n}]^T \tag{2.12}$$

Using Equations 2.9 and 2.10, Equation 2.11 can be solved using Bayes theory. The resulting filter involves a convolution for the prediction step and a multiplication for the measurement update. The prediction step for Bayesian-SLAM involves solving the following convolution (see Section B.1 for a derivation of this equation).

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(\mathbf{x}_k^v,\mathbf{x}_k^m|\mathbf{x}_{k-1}^v,\mathbf{x}_{k-1}^m,\mathbf{u}_k)P(\mathbf{x}_{k-1}^v,\mathbf{x}_{k-1}^m|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)d\mathbf{x}_{k-1}^v d\mathbf{x}_{k-1}^m \tag{2.13}$$

where the first factor on the right term represents the motion model for the vehicle and map, and the second factor the information collected a priori.

As can been seen the prediction step is an integral over the vehicle pose and map at $k-1$. However, most mapping algorithms assume the world is static, i.e. they do not include dynamic objects into the map representation. This assumption avoids the necessity of having a motion model for the map. Thus, during the prediction step, the map is updated in the next form:

$$\mathbf{x}_k^m = \mathbf{x}_{k-1}^m \tag{2.14}$$

By applying Equation 2.14 to Equation 2.13 the prediction step is simplified to:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(\mathbf{x}_k^v|\mathbf{x}_{k-1}^v,\mathbf{x}_{k-1}^m,\mathbf{u}_k)P(\mathbf{x}_{k-1}^v,\mathbf{x}_{k-1}^m|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)d\mathbf{x}_{k-1}^v d\mathbf{x}_{k-1}^m \tag{2.15}$$

Furthermore the robot motion is independent of the map, thus Equation 2.15 can be expressed as:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) = \int_{-\infty}^{\infty} P(\mathbf{x}_k^v|\mathbf{x}_{k-1}^v,\mathbf{u}_k)P(\mathbf{x}_{k-1}^v,\mathbf{x}_{k-1}^m|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)d\mathbf{x}_{k-1}^v \tag{2.16}$$

The update step for Bayesian-SLAM can be deduced simply by applying Bayes rules (Equation B.9):

$$P(\mathbf{x}_k|\mathbf{Z}^k,\mathbf{U}^k,\mathbf{x}_0) = K \cdot P(\mathbf{z}_k|\mathbf{x}_k^v,\mathbf{x}_k^m)P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) \tag{2.17}$$

where K is the normalisation factor, the middle term represents the observation model and the last term the a priori distribution obtained using Equation 2.16.

Using Equations 2.16 and 2.17 a recursive Bayesian solution for SLAM can be obtained [77].

Optimal SLAM techniques based on Bayesian filtering with non-Gaussian assumptions are only a conceptual solution. Due to the high dimensionality of the problem, the Bayesian solution cannot be determined analytically. This fact constrains the solution to the use of approximations or suboptimal Bayesian solutions. The most common solutions adopted by the SLAM community assume Gaussian noises and "linearisable" models and use Extended Kalman Filters (EKFs) to obtain recursive algorithms.

Nevertheless, some Bayesian solutions for the SLAM problem solutions have appeared. A Bayesian-SLAM algorithm using Sum of Gaussians representations was presented in [49]. However the representation used is inappropriate for Bayesian fusion and the SLAM will produce incorrect results. This work will be reviewed in Section 4.3. Perhaps the most promising Bayesian non-linear solution for SLAM is FastSLAM. FastSLAM addresses the real time implementation of the SLAM problem from a Bayesian point of view [54]. The authors apply Rao-Blackwellised particle filters [16] to obtain a recursive algorithm for SLAM. The next subsection gives an overview of FastSLAM.

2.2.1 FastSLAM

FastSLAM is an algorithm that applies Rao-Blackwellised particle filters [16] to the SLAM problem. It is based on a straightforward factorisation. This factorisation is based on the observation that if one knew the true path of the robot, the individual landmark localisation problems are mutually independent. In other words, if there is no uncertainty about the robot pose and the observations are independent, the estimation of each landmark in the map is independent of the rest of the map. In practice, the path is of course unknown. However, the conditional independence enables the filter to estimate the posterior (Equation 2.17) in the following factored form:

$$\begin{aligned}
&P(\mathbf{x}_k^v, \mathbf{x}_k^m | \mathbf{Z}^k, \mathbf{U}^k, x_0) \\
&\quad = P(\mathbf{x}_k^m | \mathbf{x}_k^v, \mathbf{Z}^k, \mathbf{U}^k, x_0) P(\mathbf{x}_k^v | \mathbf{Z}^k, \mathbf{U}^k, x_0) \\
&\quad = \prod_{i=1}^{N} P(\mathbf{x}_k^{m_i} | \mathbf{x}_k^v, \mathbf{Z}^k, \mathbf{U}^k, x_0) P(\mathbf{x}_k^v | \mathbf{Z}^k, \mathbf{U}^k, x_0)
\end{aligned} \tag{2.18}$$

The first factor in Equation 2.18 (third line) is the product of the N landmarks estimation given the robot pose is known, and the second factor is the estimation of the robot pose. This factorisation is the fundamental idea behind FastSLAM: FastSLAM decomposes the SLAM problem into a localisation and N landmark position estimation problems. Furthermore, FastSLAM relies on a particle filter to estimate the robot posterior

$$P(\mathbf{x}_k^v | Z^k, U^k, x_0) \tag{2.19}$$

This particle filter can be updated in constant time for each particle in the filter. To update the map, the algorithm relies on N *independent* Kalman filters for the N landmark estimates

$$P(\mathbf{x}_k^{m_i}|\mathbf{x}_k^v, Z^k, U^k, x_0) \tag{2.20}$$

As shown in [54], the entire filter can be updated in time logarithmic in the number of landmarks N. While other solutions of similar efficiency exist [27, 1, 6], FastSLAM can also handle non-linear robot motion models.

The efficacy of the algorithm relies on the fact that there always has to be a number of particles 'close enough' to the actual robot pose so the independent assumption is always maintained. One of the problems of the algorithm is particle depletion. Every time the particles representing the robot pose distribution are resampled, a number of them are lost and because each particle has an estimate of the whole map, part of the map history is lost as well. FastSLAM 2.0 [53] improves the particle impoverishment problem by improving the proposal distribution used to weight the particles. However, even though the particle depletion problem is attenuated, the influence of this problem in the performance and convergence of the filter, especially when working with large vehicle uncertainty is still an open question.

2.3 SLAM: The Extended Kalman Filter (EKF) Solution

The standard Bayesian solution (Equation 2.17) can be intractable even for maps with only a small number of landmarks. The EKF approach presents a tractable solution assuming that the noise is Gaussian and independent over time.

The EKF algorithm yields an estimate obtained by minimising the mean square estimation error (MMSE) of the underlying states $\mathbf{x}$ at time i based on a given set of observations up to time j. The MMSE solution is the conditional mean of the states given the observations [51]:

$$\hat{\mathbf{x}}_{i|j} \triangleq E[\mathbf{x}_i|\mathbf{Z}^{\mathbf{j}}] \tag{2.21}$$

The EKF solution assumes the noises are Gaussian, temporally uncorrelated and have zero mean:

$$E[v_k] = E[w_k] = 0, \forall k \tag{2.22}$$

with covariances matrix:

$$
\begin{aligned}
E[v_i v_j^T] &= \delta_{ij}\mathbf{Q_i} \\
E[w_i w_j^T] &= \delta_{ij}\mathbf{R_i}
\end{aligned}
\tag{2.23}
$$

and the process and observation noises uncorrelated:

$$E[v_i w_j^T] = 0, \forall i, j \tag{2.24}$$

2.3.1 SLAM State Vector

For the SLAM problem, the state vector is formed by the vehicle pose and the landmarks position.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^v \\ \mathbf{x}_k^m \end{bmatrix} \tag{2.25}$$

where $\mathbf{x}_k^v$ represents the vehicle states and $\mathbf{x}_k^m$ the landmarks map position at time k.

For the vehicle moving in a plane:

$$\mathbf{x}^v = [\hat{x}_v, \hat{y}_v, \hat{\phi}_v]^T \tag{2.26}$$

$$\mathbf{x}^m = [\hat{x}_{L_1}, \hat{y}_{L_1}, \ldots, \hat{x}_{L_i}, \hat{y}_{L_i}, \ldots, \hat{x}_{L_n}, \hat{y}_{L_n}]^T \tag{2.27}$$

In the context of EKF filtering, $\mathbf{x}^v$ describes the mean of the marginal Gaussian distribution representing the vehicle pose and $\mathbf{x}^m$ the mean of the marginal Gaussian distribution representing the map.

The covariance matrix for a variable $\mathbf{x}_i$ with mean $\bar{\mathbf{x}}_i$ is defined as:

$$\mathbf{P_{ii}} = E[(\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T] \tag{2.28}$$

Applying this definition to Equation 2.25, the covariance matrix representing the mean square errors and correlations for the vehicle pose and map is obtained.

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{v1} & \ldots & \mathbf{P}_{vn} \\ \mathbf{P}_{v1}^T & \mathbf{P}_{11} & \ldots & \mathbf{P}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{vn}^T & \mathbf{P}_{1n}^T & \ldots & \mathbf{P}_{nn} \end{bmatrix} \tag{2.29}$$

where

$$\mathbf{P_{ij}} = E[(\mathbf{x}_i - \hat{\mathbf{x}}_i)(\mathbf{x}_j - \hat{\mathbf{x}}_j)^T|\mathbf{Z}] \tag{2.30}$$

represents the cross-covariance between the variables i and j.

Equation 2.29 can be written in a more compact manner as:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm} \\ \mathbf{P}_{vm}^T & \mathbf{P}_{mm} \end{bmatrix} \tag{2.31}$$

Equations 2.25 and 2.31 represent the first and second moment estimates of the vehicle pose and map position for the Gaussian SLAM problem.

The next two sections present the prediction and update step for the classic EKF-SLAM.

2.3.2 Prediction Stage

Every time a new control input is received, the algorithm can obtain a new prediction of the underlying states based on the motion models and the new inputs. This is done using the motion model for the robot and the map.

$$\mathbf{x}_{k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}^v_{k|k-1} \\ \hat{\mathbf{x}}^m_{k|k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\hat{\mathbf{x}}^v_{k-1|k-1}, \mathbf{u}_k) \\ \hat{\mathbf{x}}^m_{k-1|k-1} \end{bmatrix} \tag{2.32}$$

The subscript $b|a$ means “the estimated at time b given observations up to time a”. The function $\mathbf{f}$ represents the relation between the past estimated robot pose and the predicted pose according to the new control inputs received $\mathbf{u}$. This model could be, for example, the one presented in Example 2.1. As pointed out before, the map is assumed to be static (Equation 2.14), thus the map position does not change during the prediction step and $\hat{\mathbf{x}}^m_{k|k-1} = \hat{\mathbf{x}}^m_{k-1|k-1}$.

To propagate the covariance matrix, the EKF linearises the motion model around the previous estimated states. Then, the covariance matrix for the prediction step will be:

$$\mathbf{P}_{k|k-1} = \nabla\mathbf{f_x}\mathbf{P}_{k-1|k-1}\nabla\mathbf{f_x}^T + \nabla\mathbf{f_u}\mathbf{Q}_k\nabla\mathbf{f_u}^T \tag{2.33}$$

where

$$\nabla \mathbf{f_x} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)} = \begin{bmatrix} \nabla \mathbf{f}_{\mathbf{x}_v n_v \times n_v} & 0_{n_v \times n_m} \\ 0^T_{n_v \times n_m} & \mathbf{I}_{n_v \times n_v} \end{bmatrix} \tag{2.34}$$

$$\nabla \mathbf{f_u} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)} \tag{2.35}$$

n_v is the number of dimensions needed to represent the vehicle pose, n_m the dimensions of the map and thus $n = n_v + n_m$ the total state vector dimensions. **U** represents the covariance matrix of the control inputs noise **u**, which is zero mean and uncorrelated in time. In this thesis the noise considered in the motion model during the prediction step is the noise in the control inputs since this is considerably bigger than the noise in the models. To avoid confusion in the notation, the inputs noise covariance will be noted from now on as **Q** instead of **U** which is also used for the control input vector[2].

Since the map is considered stationary, the evaluation of the covariance matrix shown in Equation 2.33 can be simplified. This can be observed in the spareness property of the Jacobian shown in Equation 2.34. This property of SLAM algorithms, allows a reduction of the computational complexity involved in the prediction step. The next equation shows the evaluation of the covariance matrix using the sparseness property just explained.

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \mathbf{P}^{vv}_{k|k-1} & \mathbf{P}^{vm}_{k|k-1} \\ (\mathbf{P}^{vm}_{k|k-1})^T & \mathbf{P}^{mm} \end{bmatrix} = \begin{bmatrix} \nabla \mathbf{f_x} \mathbf{P}^{vv}_{k-1|k-1} \nabla \mathbf{f}^T_{\mathbf{x}} + \nabla \mathbf{f}_{\mathbf{u}_k} \mathbf{Q} \nabla \mathbf{f}^T_{\mathbf{u}} & \nabla \mathbf{f_x} \mathbf{P}^{vm}_{k-1|k-1} \\ (\nabla \mathbf{f_x} \mathbf{P}^{vm}_{k-1|k-1})^T & \mathbf{P}^{mm} \end{bmatrix} \tag{2.36}$$

Equations 2.32 and 2.36 form the states and covariance evaluation for the prediction stage of the EKF-SLAM.

2.3.3 Update Stage

In order to fuse the new observations gathered by the sensors, the first step is to predict the observations based on the observation model **h** and the predicted states $\hat{\mathbf{x}}_{k|k-1}$.

$$\hat{\mathbf{z}}_i = \mathbf{h}_i(\hat{\mathbf{x}}_{k|k-1}) \tag{2.37}$$

The index i indicates that the observation corresponds to the i-th landmark in the map.

[2] In Kalman filter notation **Q** is usually used for the motion model noise, also called stabilisation noise. However in this thesis the noise considered in the motion prediction process will be only the noise in the control inputs, and **Q** will be used to represent the control signals covariance matrix.

Next, the innovation sequences are evaluated as the difference between the actual and the predicted observation.

$$\nu_i = \mathbf{z}_i - \hat{\mathbf{z}}_i \tag{2.38}$$

Based on Equations 2.37 and 2.38 the innovation covariance matrix can be evaluated as:

$$\mathbf{S} = \nabla\mathbf{h}_{\mathbf{x}}\mathbf{P}_{k|k-1}\nabla\mathbf{h}_{\mathbf{x}} + \mathbf{R} \tag{2.39}$$

where

$$\nabla\mathbf{h}_{\mathbf{x}} = \left.\frac{\partial\mathbf{h}}{\partial\hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \tag{2.40}$$

and **R** is the sensor noise covariance matrix.

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.41}$$

For a range and bearing sensor and a map represented by two dimensional landmarks (see Example 2.2), Equation 2.40 will be:

$$\nabla\mathbf{h}_{\mathbf{x}} = \begin{bmatrix} \frac{1}{\Delta}(-\Delta_x \;\; -\Delta_y \;\; 0 \;\; 0 \;\; \dots \;\; 0 \;\; \Delta_x \;\; \Delta_y \;\; 0 \;\; \dots \;\; 0) \\ \frac{1}{\Delta^2}(-\Delta_y \;\; \Delta_x \;\; -\Delta^2 \;\; 0 \;\; \dots 0 \;\; \Delta_y \;\; -\Delta_x \;\; 0 \;\; \dots \;\; 0) \end{bmatrix} \tag{2.42}$$

where

$$\Delta_x = x_L - x_v \tag{2.43}$$

$$\Delta_y = y_L - y_v \tag{2.44}$$

$$\Delta = \sqrt{\Delta_x^2 + \Delta_y^2}$$

Finally, the first and second moment for the EKF solution are updated as:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}\nu_i \tag{2.45}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}\mathbf{S}\mathbf{W}^T \tag{2.46}$$

where $\mathbf{W}$ is the Kalman gain.

$$\mathbf{W} = \mathbf{P}_{k|k-1} \nabla \mathbf{h}_{\mathbf{x}} \mathbf{S}^{-1} \tag{2.47}$$

The sparseness property of Equation 2.42 allows efficient calculations of Equations 2.39 and 2.47. The computational complexity of EKF-SLAM will be analysed later in this chapter.

2.3.4 State augmentation

The previous section addressed the problem of fusing observations of landmarks already incorporated in the map, so the dimensions of the system remain constant. However, when a new landmark is observed for the first time, the state vector and covariance matrix have to be augmented and the new landmark has to be initialised.

Let $\hat{\mathbf{x}}$ be the current state vector at the time an observation from a new landmark is obtained[3].

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}^v \\ \hat{\mathbf{x}}^m \end{bmatrix} \tag{2.48}$$

where $\hat{\mathbf{x}}^v$ represents the vehicle states and $\hat{\mathbf{x}}^m$ the landmarks map position.

And let $\mathbf{P}$ be the current covariance matrix.

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}^{vv} & \mathbf{P}^{vm} \\ (\mathbf{P}^{vm})^T & \mathbf{P}^{mm} \end{bmatrix} \tag{2.49}$$

When a new landmark $\mathbf{x}^L$ is observed, the landmark position is initialised using the observation model.

$$\mathbf{x}^L = \mathbf{h}(\mathbf{x}^v, \mathbf{z}) \tag{2.50}$$

And the state vector is then augmented with the new landmark position.

$$\hat{\mathbf{x}}_a = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{x}}^L \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}^v \\ \hat{\mathbf{x}}^m \\ \hat{\mathbf{x}}^L \end{bmatrix} \tag{2.51}$$

[3] For brevity in the notation the k index used for time is dropped in this section.

Since the new landmark initialisation depends on the current vehicle pose, the new landmark will be correlated to the vehicle pose estimated and as a consequence to the map. Let $\mathbf{P}_a$ be the augmented covariance matrix.

$$\mathbf{P}_a = \begin{bmatrix} \mathbf{P}^{vv} & \mathbf{P}^{vm} & \mathbf{P}^{vL} \\ (\mathbf{P}^{vm})^T & \mathbf{P}^{mm} & \mathbf{P}^{mL} \\ (\mathbf{P}^{vL})^T & (\mathbf{P}^{mL})^T & \mathbf{P}^{LL} \end{bmatrix} \tag{2.52}$$

Applying the definition of covariance and cross-covariance (Equations 2.28 and 2.30) the new terms in the covariance matrix can be calculated.

$$\begin{aligned} \mathbf{P}^{vL} &= \mathbf{P}^{vv}\nabla\mathbf{h}_{\mathbf{x}_v}^T \\ \mathbf{P}^{mL} &= \mathbf{P}^{mv}\nabla\mathbf{h}_{\mathbf{x}_v}^T \\ \mathbf{P}^{LL} &= \nabla\mathbf{h}_{\mathbf{x}_v}\mathbf{P}^{vv}\nabla\mathbf{h}_{\mathbf{x}_v}^T \quad + \quad \nabla\mathbf{h}_{\mathbf{z}}\mathbf{R}\nabla\mathbf{h}_{\mathbf{z}}^T \end{aligned} \tag{2.53}$$

where

$$\nabla\mathbf{h}_{\mathbf{x}_v} = \frac{\partial\mathbf{h}}{\partial\mathbf{x}_v} \tag{2.54}$$

$$\nabla\mathbf{h}_{\mathbf{z}} = \frac{\partial\mathbf{h}}{\partial\mathbf{z}} \tag{2.55}$$

In practice the landmark initialisation can be simplified by assuming that the new landmark had been incorporated before and assigning a very large uncertainty to the landmark position, and zero to the cross-covariances terms of the new landmark with the rest of the sytem. The augmented a priori matrix will look like:

$$\mathbf{P}_a = \begin{bmatrix} \mathbf{P}^{vv} & \mathbf{P}^{vm} & 0 \\ (\mathbf{P}^{vm})^T & \mathbf{P}^{mm} & 0 \\ 0 & 0 & \mathbf{P}^{LL*} \end{bmatrix} \tag{2.56}$$

This matrix is then used as the a priori covariance matrix in the update stage (Equation 2.46).

Note: Although this form presents a practical solution to the initialisation problem, the difference in magnitude between $\mathbf{P}^{LL*}$ (which is set very large) and the rest of the terms in the covariance matrix can provoke numerical problems. For this reason it is recommended to calculate the initialisation as in Equation 2.53.

2.4 Data Association

In every SLAM application, the measurements need to be associated with the underlying states that are being observed. This is usually referred as the *data association problem.* Successful data association involves association of the measurements with the correct states, initialising new landmarks and detecting and rejecting spurious measurements.

Data association is arguably one of the most difficult problems in SLAM or localisation applications. The main reason is the poor environment representation commonly used by SLAM algorithms. Adding other source of information than landmark position estimates can drastically improve the matching process. For example, vision information used in conjunction with scanning laser data, can be used to a great advantage [60]. Most of the algorithms however, are based only in laser information and they rely on simple geometric features to represent the environment. This simple representation makes the landmarks able to be distinguished only by their positions, restricting the association algorithm to the use of only metric information [1, 58]. The next section presents a review of the most common probabilistic data association methods used in SLAM.

2.4.1 Probabilistic Data Association

There are different algorithms that can be used for probabilistic data association [5]. Most existing techniques are based on the innovation sequence and its predicted covariance. The innovation sequence ν_i relates the observations $\mathbf{z}_i$ to the underlying estimated states. In particular, it is defined as the sequence of differences between the observation $\mathbf{z}_i$ and the predicted observation based on the observation model and the predicted states:

$$\nu_{ij} = \mathbf{z}_i - \hat{\mathbf{z}}_j \tag{2.57}$$

where $\hat{\mathbf{z}}_j = \mathbf{h}(\hat{\mathbf{x}}_j)$, which represents the predicted observation for the j-th landmark.

The normalised innovation square (NIS) also known as the Mahalanobis distance is defined as:

$$d_{ij}^2 = \nu_{ij}^T \mathbf{S}_{ij}^{-1} \nu_{ij} \tag{2.58}$$

where $\mathbf{S}_i$ is the innovation covariance matrix. If ν_{ij} has a Gaussian distribution, then d_{ij}^2 will have a χ^2 distribution. The next paragraphs present a brief overview of the three most common data association techniques: gate validation, nearest neighbour and multi hypothesis tracking.

The NIS gives the basis for the *gate validation* technique, which accepts the observations that are inside a fixed region of a χ^2 distribution, and rejects the observations that make the innovation fall outside these bounds. This determination is achieved by comparing the scalar value obtained in Equation 2.58 with a threshold value that is determined by fixing the region of acceptance of the χ^2 distribution (e.g. 95%). The gate validation is defined in the observation space and is centered in the observation prediction $\mathbf{h}(\hat{\mathbf{x}}_j)$. Figure 2.3 shows an example of the validation gate. For a fixed region of d^2 the gate looks ellipsoidal. The algorithm takes only the observations that are inside the ellipsoid. In Figure 2.3 the gate validation technique will associate the measurement $\mathbf{z}_i$ with the state $\hat{\mathbf{x}}_j$.

In many cases, the gate technique will validate the same measurement for different landmarks or more than one measurement for the same landmark. This case is said to present multi-hypothesis over the data association space. Figure 2.4 shows one example of two targets and three observations. Using the gate test, it can be reasoned that: the measurement $\mathbf{z}_3$ does not belong to any of the tracks since it falls outside the two gates, the measurement $\mathbf{z}_2$ belongs to the track $\hat{\mathbf{x}}_1$ and the measurement $\mathbf{z}_1$ could be associated with any of the two tracks. In this case, a method able to manage ambiguous association is needed.

One of the most common data association techniques used in the SLAM literature is the *nearest neighbour filter* (NNF). The NNF uses the gate validation test to initially determine which landmark states are valid candidates to update with the observation received. Among all the possible landmarks, the one that is *nearest* to the observation is selected. All the other possible hypotheses are ignored. The metric used to determine the distance between observations and predicted observations is actually slightly different from NIS. Given a set of measurements within a validation gate of a target $\mathbf{x}$, a likelihood function for each observation $\mathbf{z}_i$ can be calculated as:

$$\Lambda_i = \frac{1}{(2\pi)^{n/2}\sqrt{|\mathbf{S}_i|}} \exp\left(-\frac{1}{2}\nu_i^T \mathbf{S}_i^{-1} \nu_i\right) \tag{2.59}$$

The normalised distance is obtained by taking the logs of Equation 2.59.

$$N_i = \nu_i^T \mathbf{S}_i^{-1} \nu_i + \ln |\mathbf{S}_i| \tag{2.60}$$

It is equivalent to take the maximum of Equation 2.59 or the minimum of Equation 2.60. NNF selects the observation with minimum normalised distance. For example, for the case presented in Figure 2.4, the measurement $\mathbf{z}_1$ will be associated with the landmark $\hat{\mathbf{x}}_2$ since $N_{12} < N_{11}$.

The NNF can fail to recover the true data association when validation gates overlap, and the observations fall within this overlapped region as shown in the previously example. In such cases, the NNF can associate the observation with an incorrect state. Such false data associations are known to induce catastrophic failures in the SLAM problem.

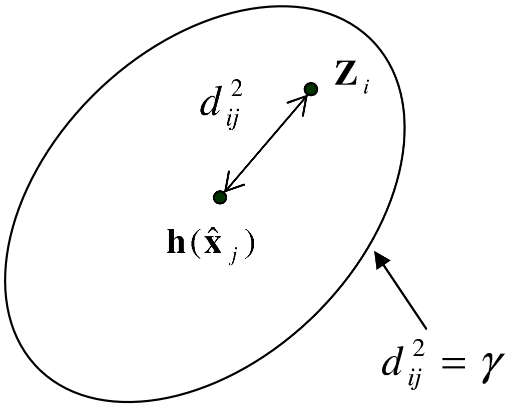


Figure 2.3: Gate validation test: the gate is evaluated in the observations space. For a fixed value, the normalised innovation square can be seen as an ellipsoid centre in the observation prediction. The measurements that fall inside the ellipsoid are associated with the respective states.

Unlike the NNF, the multi-hypothesis tracking algorithm (MHT) [5] accepts all the possible hypotheses and maintains separate tracks for each of them until the ambiguity in the association is eliminated. The algorithm runs filters in parallel for each hypothesis generated. MHT filters have been shown to be feasible for global localisation problems [37] where the state vector consists only on the robot pose. However, since the number of tracks can grow exponentially, the MHT filter would be in general very hard to apply for SLAM problems due to the computational complexity involved. That is the main reason why MHT is not usually used in SLAM.

Batch Data Association Methods

When multiple observations are jointly processed a more robust data association is obtained. It is in fact a gate validation technique applied to a multi-dimensional observation. This joint data association intrinsically considers the geometric relation and the correlation information between landmarks.

Different batch association methods can be found in the literature. The *joint compatibility branch and bound* (JCBB) [58] generates a tentative set of observations based on the gate validation test and builds an interpretation tree with all the possible combinations of solutions. Thus for a given set of associations (one possible solution) joint compatibility is determined by calculating a single joint NIS gate. Finally, the maximum set of valid joint associations is considered to be the correct hypothesis.

In [1] the *combined constraint data association* (CCDA) method is presented. The

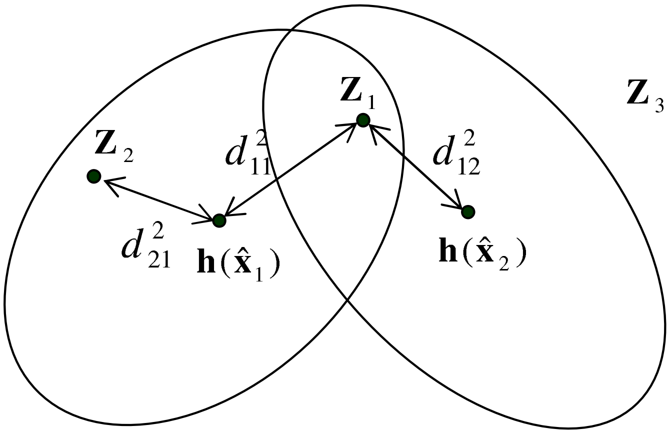


Figure 2.4: Ambiguous data association: using the gate validation test, measurement $\mathbf{z}_1$ can be associated with either state $\hat{\mathbf{x}}_1$ or $\hat{\mathbf{x}}_2$.

CCDA algorithm integrates the application of relative and absolute constraints. The fundamental data structure of the algorithm is a graph used to represent valid associations between two data sets. The edges of the graph define which pairings are mutually compatible. By performing a maximum clique search, the largest joint compatible association set may be found.

In summary, batch data association methods present a more robust solution to the data association problem since they are able to reduce the ambiguity in association based on the geometry and correlation information between landmarks.

2.5 The importance of correlations in SLAM

As the map is formed with observations taken relative to the vehicle pose, even when the sensor observations are independent, there will be a common component in the landmarks' uncertainty owing to the vehicle pose uncertainty. In other words, the uncertainty in the vehicle pose will be strongly correlated with the uncertainty in the landmarks and consequently the whole map will be correlated. Maintaining these correlations is the source of the computational complexity of SLAM problems.

In [12] it was shown that maintaining the correlations in the map is essential to ensure consistency in the estimation process. The consistency in the process is necessary for several reasons. For example, the data association process is based on the innovations. As shown in [12], in order to ensure that the covariance of the innovation is computed correctly, the covariance between the vehicle and all features estimates and also the covariance between all pairs of features estimates have to be maintained. If the landmarks are considered independent, the filter will become overconfident, and any decision process based on the filter statistics, such as data association, may be erroneous.

Another important point is the speed of convergence of the algorithm. Both overconfident or underconfident estimates will slow down the speed of convergence of the filter. When an old landmark is reobserved, the correlations between landmarks allow propagation of the corrections back through the rest of the map. If the correlations are not maintained, the rest of the map will not be corrected and the only way to update the map will be by direct observations of every single landmark in the map. Even when the old map is reobserved many times, the overconfident property of the landmarks will make the convergence much slower. An example of underconfident estimation is the Covariance Intersection filter [82]. The filter is based on the extended Kalman filter. The fusion of information is done without the correlation between state variables given an assumption of worse case conditions (all the information is considered fully correlated). As a consequence the SLAM map tends to suffer large feature uncertainty and converges very slowly.

A comparison between a full EKF-SLAM and a simplified version assuming map independence was presented in [8]. It is shown that the independence assumption yields over optimistic results in addition to a lower quality estimation (larger errors).

2.6 Computational Complexity

One of the main problems of SLAM algorithms is the computational burden required to maintain the correlations in the map. If an EKF is applied to an estimation problem that involves a state vector of n dimensions, the computational cost required for either the prediction or the update step is $O(n^3)$. This is also valid for SLAM, so the computational burden will grow cubically with the number of dimensions used to represent the vehicle pose and the map. However the sparseness property of SLAM allows the computational cost to be reduced to $O(n)$ for the prediction step and $O(n^2)$ for the update.

As explained in Section 2.3.2, using the stationary property of the map, in the prediction step the covariance matrix can be simplified as in Equation 2.36. Essentially evaluating the covariance matrix requires the calculation of two terms; the vehicle covariance $\mathbf{P}^{vv}_{k|k-1}$ and the vehicle cross-covariance with the map $\mathbf{P}^{vm}_{k|k-1}$ (since $\mathbf{P}^{mv} = (\mathbf{P}^{vm})^T$). The map covariance term $\mathbf{P}^{mm}$ remains constant since the map is assumed to be stationary. The computational cost of evaluating $\mathbf{P}^{vv}_{k|k-1}$ is $n_v^4 n_u^2$, where n_v is the dimensions of the vehicle pose and n_u is the dimensions of the control input vector. Since these two terms remain constant, the cost to update $\mathbf{P}^{vv}_{k|k-1}$ is constant time $O(1)$. The cross-correlation matrix $\mathbf{P}^{vm}_{k|k-1}$ has a computational cost of $n_v^2 n_m$. Since $n_m >> n_v$, then $n_m \approx n$ and thus the cross-covariance term is $O(n)$. Therefore the computational cost of the prediction step is proportional to the number of states.

In order to calculate the covariance matrix in the update stage, two auxiliary matrices are needed; the innovation covariance $\mathbf{S}$ (Equation 2.39) and the filter gain $\mathbf{W}$ (Equation 2.47). The computational cost to calculate $\mathbf{S}$ in a general problem will be $O(n^3)$. If the sparse nature of the observations is considered the computational cost is reduced to $O(n)$. The same situation occurs with the filter gain $\mathbf{W}$, the computational cost of the filter gain is also reduced from $O(n^3)$ to $O(n)$. Finally to obtain $\mathbf{P}_{k|k}$, $\mathbf{WSW}^T$ has to be evaluated. This multiplication does not involve any sparse matrix and the cost is $O(n^2)$. This is the most computationally expensive computation required in SLAM.

In summary, the SLAM computational burden is $O(n)$ for the prediction stage and $O(n^2)$ for the update stage .

Even when the computational cost of SLAM is reduced to $O(n^2)$ simply by exploiting the sparse nature of the observations, this still implies that the computation will grow quadratically with the number of landmarks, and for mapping large areas where hundreds of landmarks can eventually be incorporated, the problem will be intractable. Efficient solutions that reduce the computational complexity of the algorithm are reviewed next.

2.7 SLAM: Efficient Optimal Solutions

As shown previously, the computational burden involved in the SLAM update is $O(n^2)$, where n is the number of landmarks in the map. Efficient solutions which are variations of the EKF-SLAM have been proposed.

The Constraint Local Submap Filter (CLSF) was presented in [86]. By exploiting the manner in which observations are fused into the global map, the method is able to reduce the computational cost of the problem. In addition the method also improves the data association process. Rather than fusing every observation directly into the global map, the algorithm generates an independent local map of the features in the vicinity of the vehicle. This local map is then periodically fused into the global map to recover the full global map estimate. The computational savings of the CLSF algorithm arise during the update stage. Since only a local map is updated, the cost will be quadratic with the number of landmarks in the vehicle's vicinity, which will be in general much lower than the total number of landmarks. After a period of time, the local map is fused into the global map. A data association strategy is used to identify landmarks that have been duplicated in the local and global map and a constraint based projection is used to yield a single map. The other contribution of the algorithm is related to the data association process. Since the filter works in a local map, the associations are deferred until the local map is fused with the global map. A more informed choice of association is then possible, which can help to reduce the ambiguity in data association.

The Compressed EKF (CEKF) [26] is a modified version of the EKF filter, which reduces the computational cost of the filter for some estimation problems, SLAM being one of these problems. The CEKF-SLAM possesses a similar conceptual structure to the previous method reviewed [86]. The CEKF works locally, "compressing" all the sensory information into a local map formed by the *active landmarks.* The active landmarks are the ones close to the vehicle pose, and the rest of the landamrks are considered as the *passive states.* The information compressed in the active states is periodically transferred to the global map. The computational cost is reduced by working locally with a small number of landmarks. The computational burden is quadratic with the number of active states $O(n_a^2)$, which is in general smaller than the total number of states. Only when the local map is fused with the global map, does the computational cost grow quadratically with the number of landmarks in the global map.

The postponement algorithm [39] presents a similar solution to the CEKF. The method updates only the landmarks in a local region surrounding the vehicle location, and the local map is periodically fused with the global map. The computational complexity is also $O(n_a^2)$ while the vehicle remains within the local area.

It is important to remark that this section reviewed only some of the algorithms that present optimal results in respect to EKF-SLAM. There are also suboptimal algorithms that present very good solutions to the computational problem of SLAM [1, 6, 21, 66, 81]. However its analysis is beyond the scope of this thesis which is focus in representation.

2.8 Autonomous Mapping

This section presents a review of the most common mapping techniques used in autonomous mapping: occupancy grids (OGs), feature-based maps, topological maps and scan correlation techniques. In particular, their suitability to SLAM is emphasised.

2.8.1 Occupancy Grid Maps

Certainty grid mapping techniques represent the environment properties based on a grid of fixed resolution, occupancy being the most popular property represented in mapping (Occupancy grids) [18]. This is perhaps the main reason why the name occupancy grids (OGs) has been adopted as a general name for grid maps, even when other properties like traversability, colour, etc. may be represented. In this thesis OGs are also used as certainty grid maps.

The OG is a multidimensional grid that maintains stochastic estimates of the occupancy state of each cell. The technique was introduced by Elfes and Moravec [18, 56] and has been widely used due to the simplicity of its implementation. Each cell stores the probability of being occupied or free. The state variable $S(C_i)$ associated with the cell C_i is defined as a binary random variable where $P(C_i = O_{CC}) + P(C_i = Free) = 1$ since the cell states are exclusive. The evaluation of the posterior over the occupancy of each grid cell (4.6) is based on binary Bayes filters:

$$P(C_i = O_{CC}|\mathbf{x}_k^v, \mathbf{Z}^k) \propto P(\mathbf{z}_k|C_i = O_{CC})P(C_i = O_{CC}|\mathbf{Z}^{k-1}) \tag{2.61}$$

In practice the update is usually solved using Odds formulation [56].

One of the main problems with OG maps is that they do not take into account the correlation between cells that exists due to the vehicle pose uncertainty. The technique assumes that the states variables of each cell are independent, an assumption that is flawed, since the map is built up with observations taken relative to the vehicle pose, the uncertainty of which will correlate the map. The effects of this incorrect assumption will be more significant when working in large areas where the vehicle pose can potentially accumulate significant errors.

The OG has been mainly used for localisation given an a priori map and map building given the robot pose. More recently some SLAM formulations using OG have started to appear [72]. However, as stated in [8] it is of fundamental importance to have a representation of the robot pose, sensor uncertainty and their correlations to achieve convergence of the SLAM process. This is not possible using OG techniques. One obvious case where OG-based SLAM fails is when attempting to close a large loop. In this case the actual pose of the vehicle will usually have a considerable error. Even in the case where it is possible to associate the observations correctly with the global map and correct the vehicle pose, it will not be possible to propagate the corrections back through the rest of the map. This is due to the fact that OG approaches represent the uncertainty in a local frame, but do not take into account the correlation between the local and the global frames. The consequence will be an inconsistent global map. In summary, grid techniques can provide very rich environment representations, which will enable the robot to perform tasks such as global path planning but they cannot provide consistent global map estimates when there is uncertainty in the vehicle pose.

2.8.2 Feature-Based Maps

Feature map techniques [44] represent the environment with parametric features such as points, lines, cylinders, corners, etc [36]. A feature is a distinctive part of the environment that can be easily discriminated by a sensor and described with a set of parameters. This

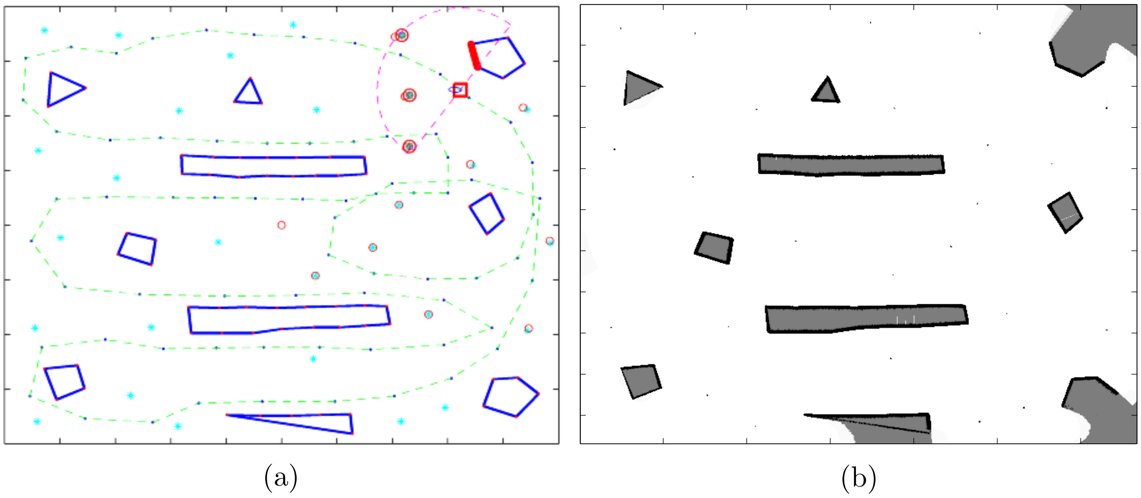
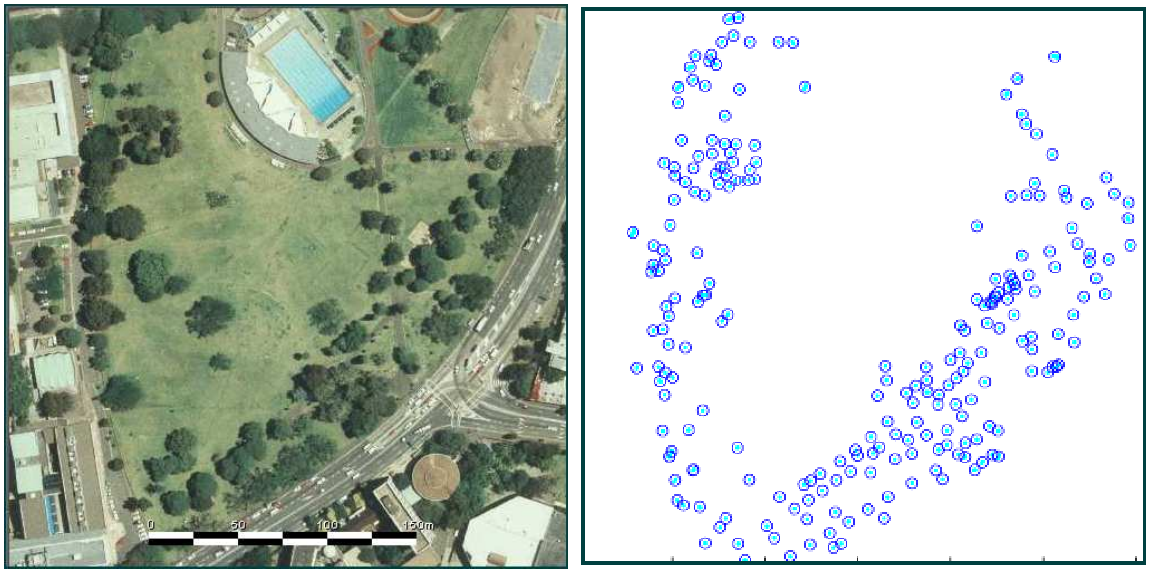

(a) (b)

Figure 2.5: (a) Navigation environment. The solid lines are objects, the '*' are landmarks and the dashed line is the control reference (b) Occupancy Grid map, the black colour denotes the highest probability of occupancy (one), the white colour the lowest (empty space) and the grey regions depict unexplored space.

method requires a model for each type of feature considered in the environment. A comprehensive description of the SLAM problem using features was presented in a seminal paper in [73]. Although only the location of the features is used to represent and maintain the map, other information such as geometry, colour, etc., could also be used in the identification stage or during the data association process. Real time implementations of SLAM have been mostly based on the use of Extended Kalman Filters (EKF) [1, 6, 9, 27, 87]. With this approach the state vector containing the robot pose is augmented when new features are observed and validated. Since the features are observed with respect to the robot position the estimated robot pose and map will be correlated due to the robot pose uncertainty. At the limit, if an infinite number of observations could be taken, the map would become fully correlated. This means that a theoretically perfect map could be achieved if the absolute location and orientation of one feature is given. Figure 2.6 shows a typical result of an estimated map and trajectory obtained with feature-based SLAM using the Victoria Park dataset [57, 26]. In this experiment the most common relevant features were trees. More detailed information on the experiment and the complete experimental dataset is available in [57].

The main advantage of feature-based SLAM is the ability to handle uncertainty and correlations. The drawback is that the map maintained is a sparse representation of the environment, which in most cases will be of limited resolution and will not be able to provide information to perform tasks such as path planning.



(a)

(b)

Figure 2.6: Figure (a) shows a satellite picture of Victoria Park and (b) the final map obtained using traditional feature-base SLAM algorithms.

2.8.3 Topological Maps

Topological approaches [41] represent the robot environment by using graphs. With metric approaches such as feature maps or occupancy grids, the objects' locations are defined in a cartesian coordinate frame. With topological maps the environment is represented by nodes and arcs that correspond to possible travel paths. Each node represents a *distinctive place* in the environment, and the edges represent the relative position between adjacent nodes. Pure topological maps represent only the connectivity between nodes and no information about the absolute position of the nodes is required. Figure 2.7 shows an example of a topological map. Topological maps are attractive for their compact representation. They are efficient representations for tasks like path planning and are appealing in indoor applications where clear distinctive places can be found between areas where navigation can be performed aided by very basic information such as wall following. One of the main problems with this representation in more complex environments is place recognition. If the robot travels between two places that look alike, the lack of metric information makes the discrimination between these two places very unreliable. In this case the logic of the topological map will be broken and the robot will not be able to evaluate its position. This is probably the main reason why most of the mapping algorithms use metric information over topological maps in large unstructured environments [1, 37, 41, 78]. Finally another limitation of this technique is that it forces the planner to follow specific trajectories to pass through or very near distinctive places.

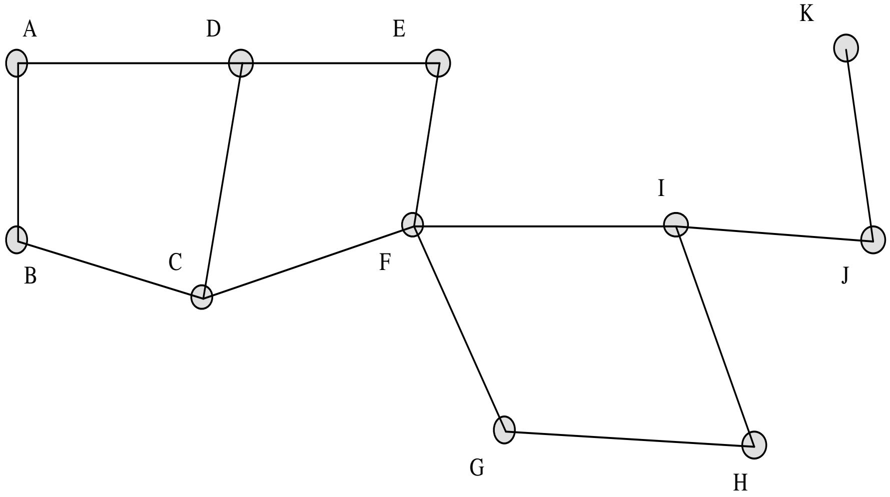


Figure 2.7: Typical topological map, the nodes are distinctive places and the arcs the path between the distinctive places.

2.8.4 Scan Correlation Methods

Scan correlation based algorithms represent the environment by unprocessed data or raw data points [40, 48]. These algorithms are particularly useful for environments where it may be difficult to apply parametric feature models. The algorithms use scan correlation or range-image registration to align a reference and an observed scan and then from the result of this alignment estimate the robot pose (localisation). The main concern with scan correlation methods is that they require to store a set of scans in order to do localisation and so they are not recursive. A review of scan correlation based algorithms will be presented in Chapter 5.

2.8.5 Hybrid Map Approaches

In the past few years various researchers have presented implementations of maps combining different approaches [77]. These studies range from integrating topological with metric maps [1, 6, 11, 78] to the use of multi-resolution OG approaches [72]. In [1] and [6], the authors introduce a hybrid approach that uses topological and feature maps to perform large-scale SLAM. This method scales very well to large environments using feature maps only. In [72] a localisation procedure using correlation techniques to register real time local OG maps with the existing global map is presented. Extensions to SLAM are also presented incorporating new features with a method based on sequential localisation and posterior map building. Unfortunately this method generates inconsistent results, especially when operating in large areas since these two tasks have to be performed simultaneously to guarantee consistency in the results.

2.8.6 Summary

In the OG map shown previously (Figure 2.5), it can be seen that every object is represented in the map, independently of their features. The map in this example is composed of point features and objects with different geometry. For the algorithm all the objects are represented in the same manner independent of their geometry. As mentioned, the main drawback of this algorithm is that it cannot handle vehicle pose uncertainty. On the other hand, in the feature-based example presented (Figure 2.6), only the objects with certain characteristics or certain features are included in the map (in the example these objects are trees). The rest of the sensory information is neglected. The main advantage of this approach is that it can cope with uncertainty in the vehicle pose, the map and the vehicle pose can be evaluated simultaneously (SLAM).

Table 2.1 presents a comparison of the algorithms analysed in this section. Topological maps were not included in this comparison because they are usually used with metric maps such as feature-based techniques. Pure topological maps are not generally used in practice. The table summarises the main characteristics of the algorithms reviewed. While occupancy grid approaches are able to obtain dense representations, the technique's main drawback is the lack of a method to manage vehicle uncertainty. On the other hand, feature-based representations are able to handle vehicle uncertainty, but they end up with sparse environment representation. Finally scan matching approaches present a promising alternative, but in the algorithms presented so far there is no measurement of uncertainty in the final map and they do not perform data fusion.

Algorithm	Occupancy Grids	Feature-based	Scan Correlation
Map Richness	Dense map	Sparse map	Dense map
Uncertainty	Posterior map	Posterior pose and map	n/a
SLAM	No	Yes	Yes

Table 2.1: Comparison between some of the most popular mapping algorithms.

2.9 Simulation

Simulation results of SLAM using the classic EKF-SLAM algorithm are presented in this section. Simulation results are necessary in order to analyse the behaviour of the algorithm since it is possible to compare the results with the ground truth.

The simulation environment is shown in Figure 2.8. The environment consists of a medium size area of approximately 100 by 80 metres. The map is formed by point feature landmarks. The control inputs are speed and steering information with added Gaussian noise with standard deviation of 0.3 m/s and 3 degrees respectively. The speed of the vehicle was 3 m/s and the time necessary to close the loop was 85 seconds, so the length of one loop was 255 metres. The simulated observations sensor returns range and bearing information with 0.15 metres and 1 degree of standard deviation. The maximum range for the sensor was set to 20 metres. Table 2.2 summarises the simulation sensors parameters.

Figure 2.8 (a) shows the ground truth and result before closing the loop. The blue stars represent the actual landmark positions and the red circles the estimates. The blue solid line depicts the actual vehicle trajectory and the red dashed line the estimated trajectory. The ellipses represent the 3σ uncertainty bound for the landmarks estimate. Figure 2.8 (b) shows the result after closing the loop.

The largest error in the trajectory can be observed on the leftmost part of the environment. This is because the vehicle starts at the centre and moves counterclockwise. Figure 2.9 (a) shows a zoom-in of the top leftmost part. Comparing the error between the actual landmarks positions and those estimated with the 3σ uncertainty bounds it can be seen that the filter is working consistently, at least with respect to landmarks error. Figure 2.9 (b) shows a zoom-in after the loop has been closed. As expected, the actual landmarks error and uncertainty have been reduced.

The actual error and standard deviations are shown in Figure 2.10. The red solid line in (a) depicts the actual error in position in the east coordinate. The red dashed line represents the 1σ estimated uncertainty. It can be seen that the estimated uncertainty is consistent with the actual error. Figure (b) shows the error and 1σ estimated uncertainty for the north coordinate and (c) for the vehicle heading. Finally, (d) shows the evolution of the uncertainty for one of the landmarks in the map, the solid line represents the east coordinate and the dashed line the north.

Figure 2.11 illustrates the evolution of the covariance matrix. The figure shows images of the normalised covariance matrix, thus each pixel in the image represents the correlation coefficient ρ_{ij}.

$$\rho_{ij} = \frac{|\mathbf{P}_{ij}|}{\sqrt{\mathbf{P}_{ii}}\sqrt{\mathbf{P}_{jj}}} \tag{2.62}$$

The grey level of the pixels indicates the level of correlation; white represents $\rho_{ij} = 0$ and black $\rho_{ij} = 1$. Figure 2.11 (a) shows an image of the normalised covariance matrix after the vehicle has completed a quarter of the loop, so not all the landmarks have been incorporated yet. Figure (b) shows the normalised covariance matrix a few seconds before the loop is closed and (c) shows the result right after the loop is closed for the first time. Comparing (b) and (c) can be seen how the map becomes more correlated after closing the loop. Finally (d) shows the result after the vehicle has gone through the whole path several times. The correlation in the map is now even stronger.

Figure 2.12 illustrates the normalised covariance matrix obtained from the EKF-SLAM as before, but now applying a few absolute vehicle pose observations. As can be seen, the correlation in the map is rapidly reduced. This example tries to illustrates the decorrelation effect when absolute observations are incorporated into the estimation problem.

Parameter	**Value**
Dead reckoning sampling time	0.025 secs
Steering sensor σ	3°
Velocity encoder σ	0.3 m/s
Observations sampling time	0.2 secs
Observations: max. range	20 metres
Observations: range σ	0.15 metres
Observations: bearing σ	1°

Table 2.2: Sensors parametres

2.10 Experimental Results

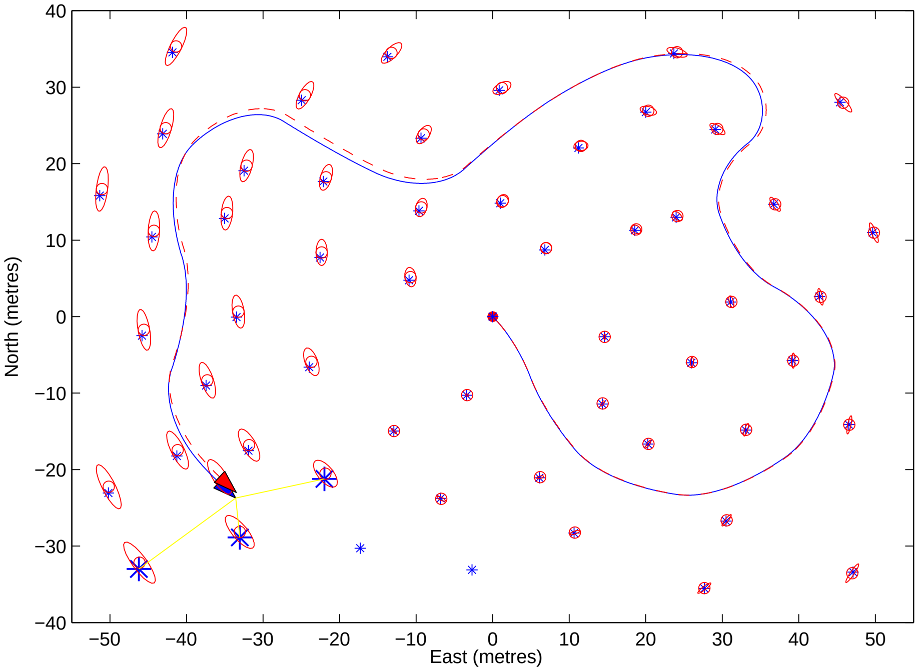
The Victoria Park dataset [57, 26] has been one of the benchmark datasets adopted by the SLAM community. The dataset was taken using the vehicle presented in Appendix A. The total vehicle trajectory was about 4000 metres, approximately 20 minutes. Laser was used in order to take observations from the environment.

Most of the relevant features useful for navigation in this environment are trees. The centres of the trees were estimated and used as landmarks. Figure 2.13 (a) shows a satellite picture of Victoria Park in Sydney.

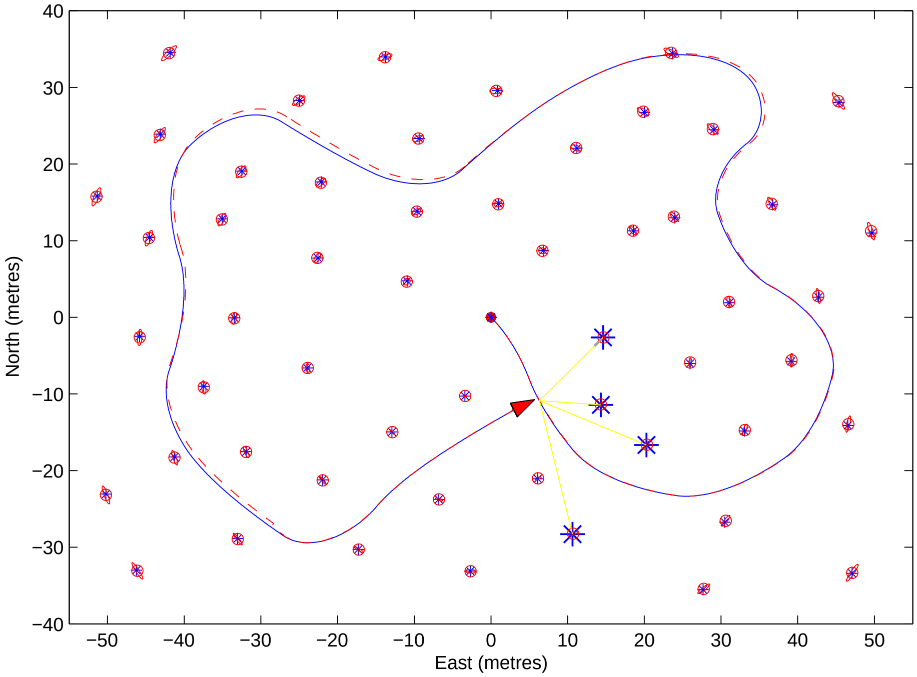
A FastSLAM algorithm was implemented and run using the Victoria Park dataset. Figure 2.13 (b) shows the result obtained with the benchmark dataset using FastSLAM [64]. The solid red line depicts the estimated vehicle trajectory which consists of the particles mean. The green stars represent the landmark's position estimates, which were obtained as the average of the Gaussian means for each landmark.

2.11 Summary

This chapter has introduced the SLAM problem and presented the approach using Bayesian and Gaussian estimation. It was noted that applying Bayesian estimation, SLAM becomes in general intractable due to the computational complexity, even for small environments. By assuming white Gaussian noise and considering the system is 'linearisable', the EKF presents a much more efficient solution to the SLAM problem.

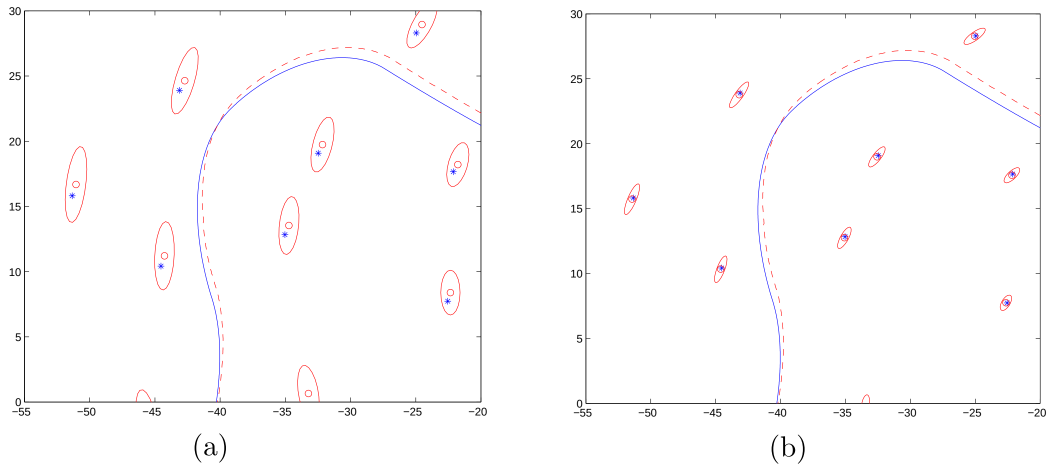


(a)



(b)

Figure 2.8: Feature-based SLAM simulation. Figure (a) shows the result before closing the loop. (b) shows the results right after the first loop is closed. The blue stars represent the actual landmarks position and the red circles the estimates. The blue solid line depicts the actual vehicle trajectory and the red dashed line the estimated trajectory. The ellipses represent the 3σ uncertainty bound for the landmarks estimate.



(a) (b)

Figure 2.9: Zoom-in of the top leftmost part of the vehicle trajectory, (a) before closing the loop and (b) after the loop is closed.

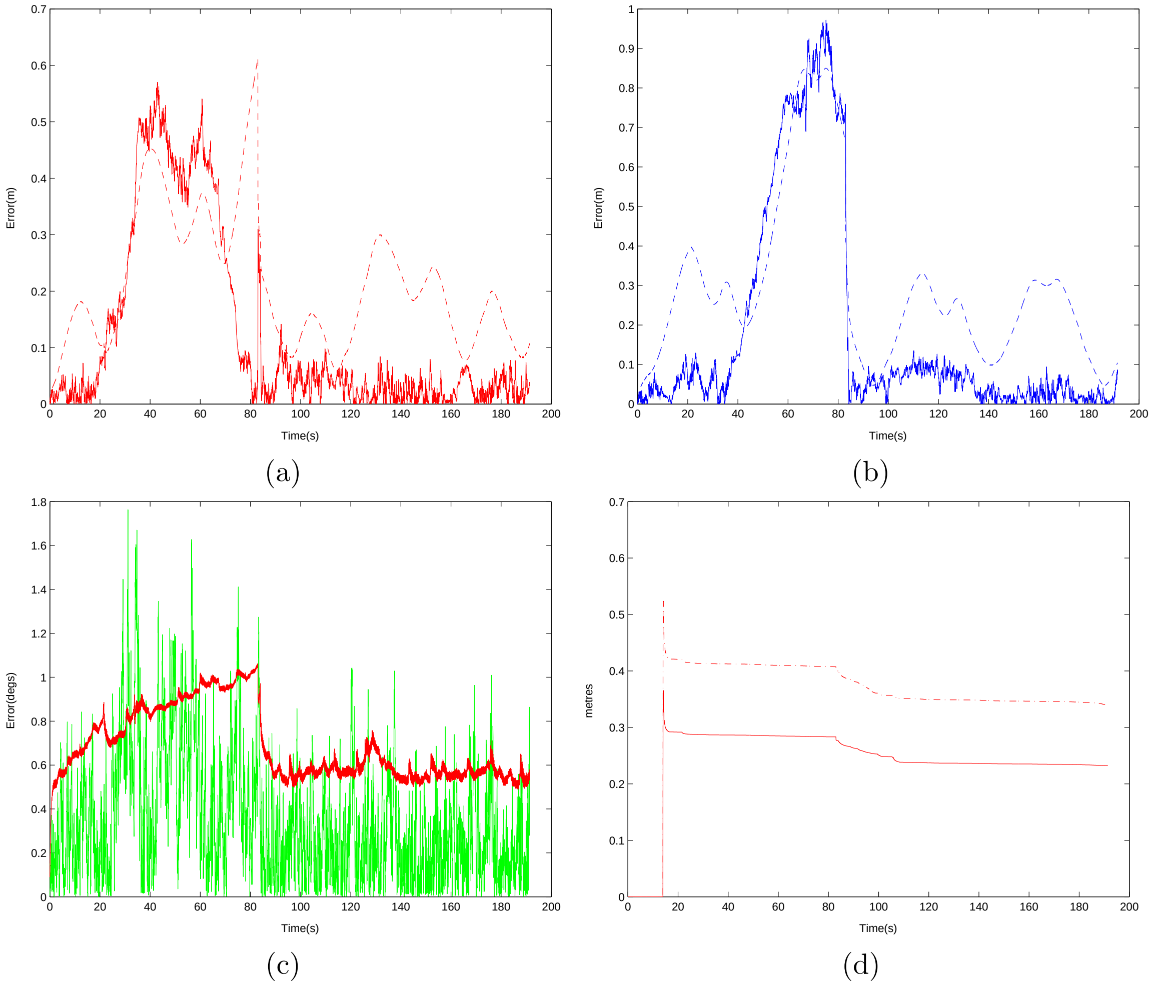
It was shown that the map becomes correlated as a consequence of the vehicle uncertainty. Maintaining these correlations is essential to ensure consistency in the final map. Assuming map independence [8] will yield over optimistic results in addition to a lower quality estimation. As a consequence of the over optimistic results, any decision based on the statistics of the filter (i.e. data association) may be incorrect and any measure of quality of the map will be incorrect.

The computational complexity of SLAM was shown to be $O(n^2)$ where n represents the number of landmarks. This computational cost can make the EKF solution intractable when applied in large areas where hundreds of landmarks can potentially be incorporated. A review of more efficient solutions that reduce the $O(n^2)$ cost of the problem was presented.

Different map representation were reviewed. In particular is was analysed how these representations can be applied to SLAM.

Finally, simulation results using the classic EKF-SLAM and experimental results with the Victoria Park dataset were presented.

Most of the solutions presented for the SLAM problem are based on a feature map, yielding sparse representations of the environment. For example, Figure 2.13 (b) shows the final map obtained in Victoria Park, which consists of point-feature landmarks representing the centre of the trees. However, the sensors render a lot more of information, which is usually rejected. The next chapter presents an algorithm that allows all the sensory information to be incorporated into the representation, obtaining dense environment representations. Furthermore, the algorithm obtains multi-layered representations, where each layer can represent different environment properties or the same property using different paradigms, e.g. grid or Gaussian maps.



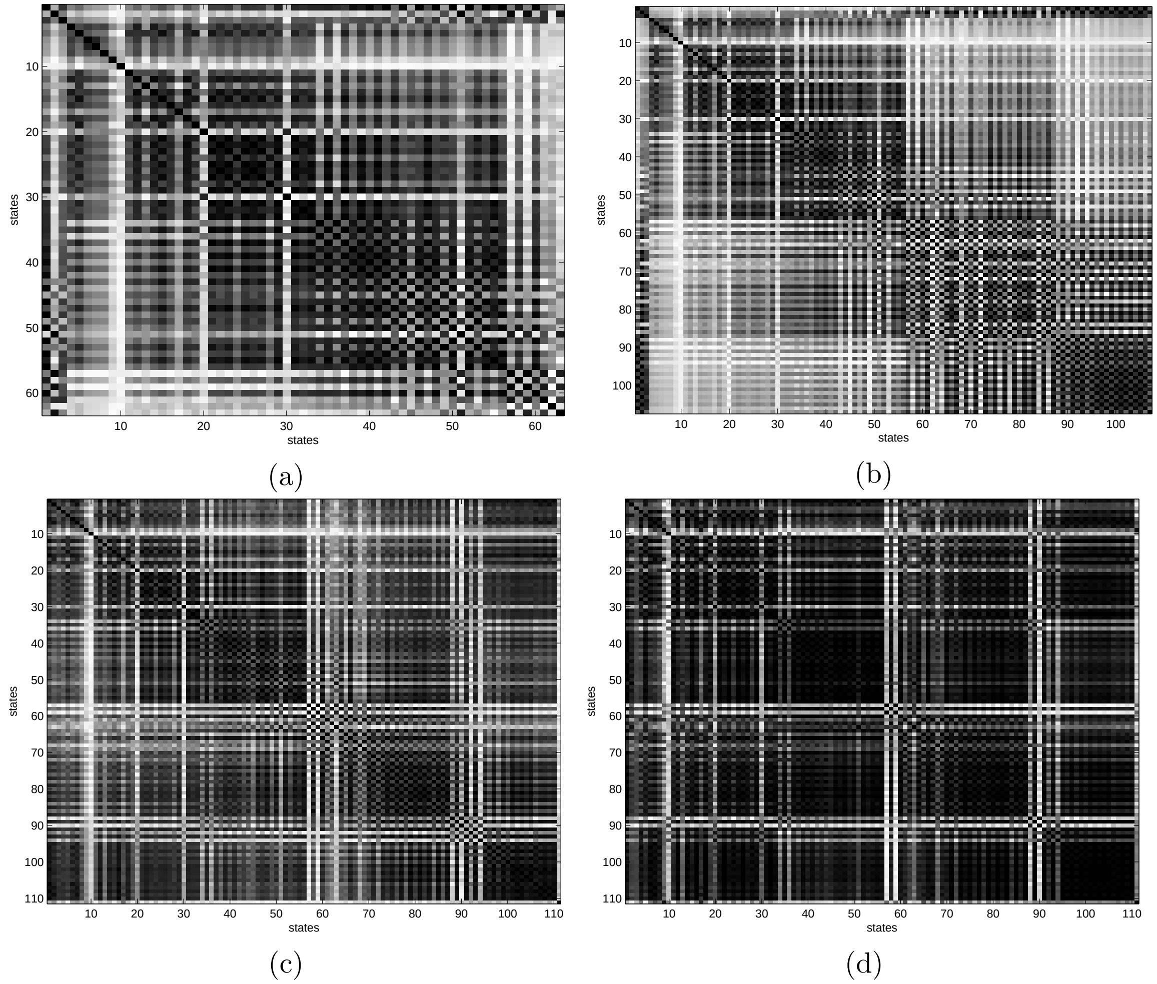
(a)

(b)

(c)

(d)

Figure 2.10: Actual errors and standard deviations. The red solid line in (a) depicts the actual error in position in the east coordinate. The red dashed line represents the 1σ estimated uncertainty. (b) shows the error and 1σ estimated uncertainty for the north coordinate and (c) for the vehicle heading. (d) shows the evolution of the uncertainty for one of the landmarks in the map, the solid line represents the east coordinate and the dashed line the north.



(a)

(b)

(c)

(d)

Figure 2.11: Normalised covariance matrix. The tone of the pixels indicates the level of correlation; white represents $\rho_{ij} = 0$ and black $\rho_{ij} = 1$. (a) shows an image of the normalised covariance matrix after the vehicle has completed a quarter of the loop, so not all the landmarks have been incorporated. (b) shows the normalised covariance matrix few seconds before the loop is closed and (c) shows the result right after the loop is closed for first time, the map becomes more correlated after closing the loop. (d) shows the result after the vehicle has gone through the loop several times, the correlation in the map is now even stronger.

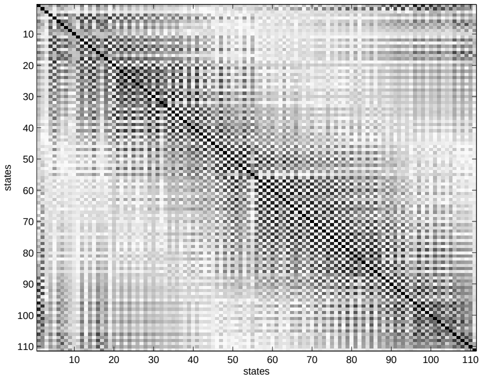
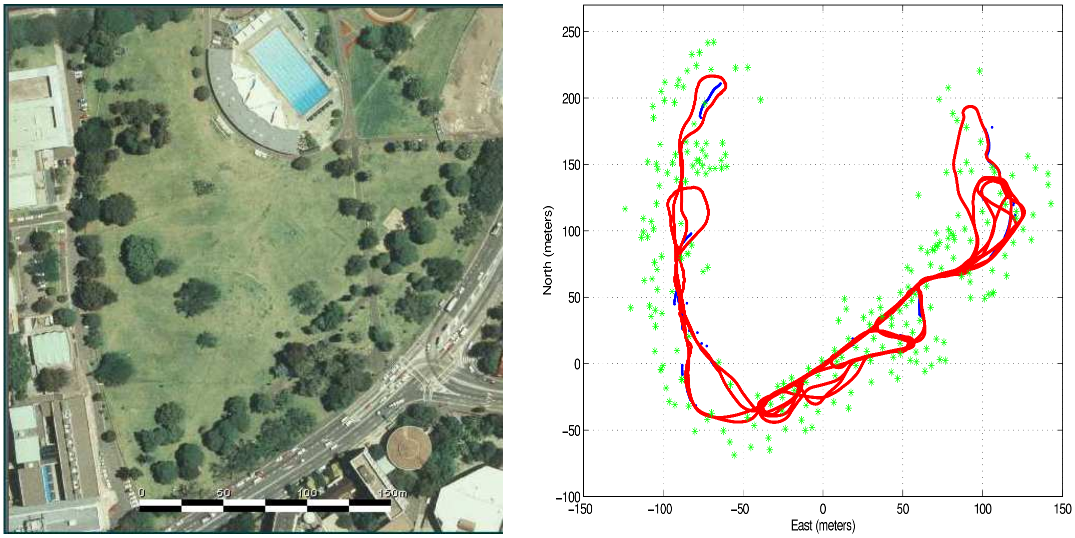


Figure 2.12: Normalised covariance matrix obtained by fusing absolute vehicle pose observations with the SLAM.



(a) (b)

Figure 2.13: SLAM result using the Victoria Park dataset. (a) shows a satellite picture of Victoria Park and (b) the result obtained using FastSLAM. The solid line depicts the estimated vehicle trajectory and the stars represent the estimated landmark position. The trunk of the trees were the main features used as landmarks in the map.

Chapter 3

DenseSLAM: The HYbrid Metric Maps (HYMMS)

This chapter introduces the concept of DenseSLAM and presents a solution for this problem named Hybrid Metric Maps (HYMMs).

The HYMMs is a novel mapping algorithm that permits the integration of the information gathered from different sensors into one *consistent and rich* environment representation [30, 63]. The dichotomy between representational richness and mathematical complexity has been one of the main reasons why traditional SLAM maps are sparse and are built up of isolated landmarks observed in the environment. Although a dense representation may sometimes be desirable, the construction of a consistent dense map has previously not been possible for the incremental SLAM paradigm. The new framework presented here combines feature maps with other dense metric sensory information. The global feature map is partitioned into a set of connected Local Triangular Regions (LTRs), which provide a reference for a *detailed multi-layer* description of the environment. The HYMM framework permits the combination of efficient feature-based SLAM algorithms for localisation with, for example, occupancy grid (OG) maps. This fusion of feature and multi-layer dense maps has several complementary properties; for example, grid maps can assist data association and can facilitate the extraction and incorporation of new landmarks as they become identified from multiple vantage points.

The format of this chapter is as follows. Section 3.1 introduces the definition of DenseSLAM. Section 3.2 introduces the hybrid metric maps. Section 3.3 explains the fundamental basis of the algorithm. Section 3.4 describes the relative representation and in particular the one used in this thesis. Section 3.5 explains the necessary conditions to form a local region and Section 3.6 discusses error position in the dense maps. Section 3.7 talks about consistency of the representation obtained using the HYMMs. Computational complexity of the algorithm is discussed in Section 3.8 and then simulation and experimental results are presented in Sections 3.9 and 3.10.

3.1 DenseSLAM

As seen in Section 2.8, the mapping techniques that are able to handle vehicle uncertainty are not able to obtain dense representations. On the other hand, mapping algorithms that are able to obtain detailed representations, cannot cope with the problem of vehicle pose uncertainty. This section introduces the concept of DenseSLAM [63], which is defined as:

The process of simultaneous vehicle localisation and ***dense*** *map building.*

DenseSLAM is then a more ambitious problem than classic feature-based SLAM. A solution for DenseSLAM will have to deal with computational and consistency issues, arising from the problem of trying to obtain a dense representation while simultaneously doing localisation.

The next section presents a solution for the DenseSLAM problem named *Hybrid Metric Maps* (HYMMs).

3.2 The HYbrid Metric Maps (HYMMs)

The Hybrid Metric Maps (HYMMs) algorithm [30, 63] presents a novel solution to address the representation problem in mapping with unknown robot pose. The HYMM is a mapping algorithm that combines feature maps with other metric sensory information. The approach permits the localisation of the robot and at the same time produces a detailed and consistent environment representation (DenseSLAM). Rather than incorporating all the sensory information into the global map, the HYMMs augment the robot pose with features extracted from the environment (classic feature-based SLAM) and in addition, represent the rest of the information in local maps relative to the features. The state vector will be augmented with the new features positions and the rest of the information fused in the local regions using some adequate representation. The main difference between feature-based SLAM (classic SLAM) and DenseSLAM,[1] is that classic SLAM incorporates the features into the map and neglects the rest of the information, however DenseSLAM has the ability to maintain all the sensory information building a detailed environment representation. DenseSLAM is actually a natural extension to the classic SLAM algorithm that allows

[1] SLAM is actually a problem and not a solution, but the term has been related to a family of algorithms that use Kalman Filters as a solution. In similar manner, in this thesis DenseSLAM will be mentioned sometimes as a solution to a mapping problem, although the solution presented here are the HYMMs.

all the sensory information to be embodied into the environment representation without increasing the computational burden and without loss of consistency.

The algorithm uses the feature map to partition the area explored by the robot into smaller regions. Figure 3.1 illustrates this process. When the robot starts to navigate, it will extract features from the environment that will be incorporated in a feature map which will be used to partition the global map into smaller regions. The dense sensory information will be represented in these local regions. Figure 3.2 shows a hypothetical dense map. The figure shows the division of the global map into smaller regions and the multi-layer map, where each layer depicts different environment properties. The global dense map will consist of a set of local maps defined relative to the feature positions.

Figure 3.3 shows a basic diagram of the algorithm. When a sensor frame is obtained, first a feature extraction algorithm is applied and the features extracted are added to the feature-based SLAM. Then the algorithm looks for new local regions and fuses all the sensory information in the respective local frames.

The first difference between classic SLAM and DenseSLAM is the local representation used to fuse the dense information. The motivation behind the relative representation is to reduce correlations between states. Using this relative representation, the states represented in a local frame become strongly correlated and the states represented in different frames become weakly correlated. This is the key therefore, that allows the algorithm to neglect part of the correlations in the system and make the representation tractable. The algorithm chooses the local regions in a way that minimises the errors in the dense information position, that can be produced by ignoring correlations.

The second difference is the introduction of a new concept named *Unidirectional Information Flow* (UIF) which is used to ensure consistency in the system. In order to ensure consistency the algorithm needs to maintain the correlations. The consistency is maintained by augmenting the state vector only with a selected part of the sensory information (features) and by fusing the rest of the information in the local regions. By the use of the *Unidirectional Information Flow* (UIF), it will be demonstrated that the system remains consistent. The UIF considers (i) information going from the states represented in the state vector to the dense maps: when the estimated features position change, the global position of the dense maps will change as well (due to the relative representation used) and both maps will be corrected. The feature map will propagate the corrections to all the local dense maps; and (ii) the UIF considers that there is no direct flow of information from the dense maps to the landmark map and vehicle pose.

It will be shown that the estimation of the features map and the localisation process obtained with DenseSLAM are exactly the same as the estimates using classic SLAM. Moreover, the localisation process could be improved if the dense information is used to

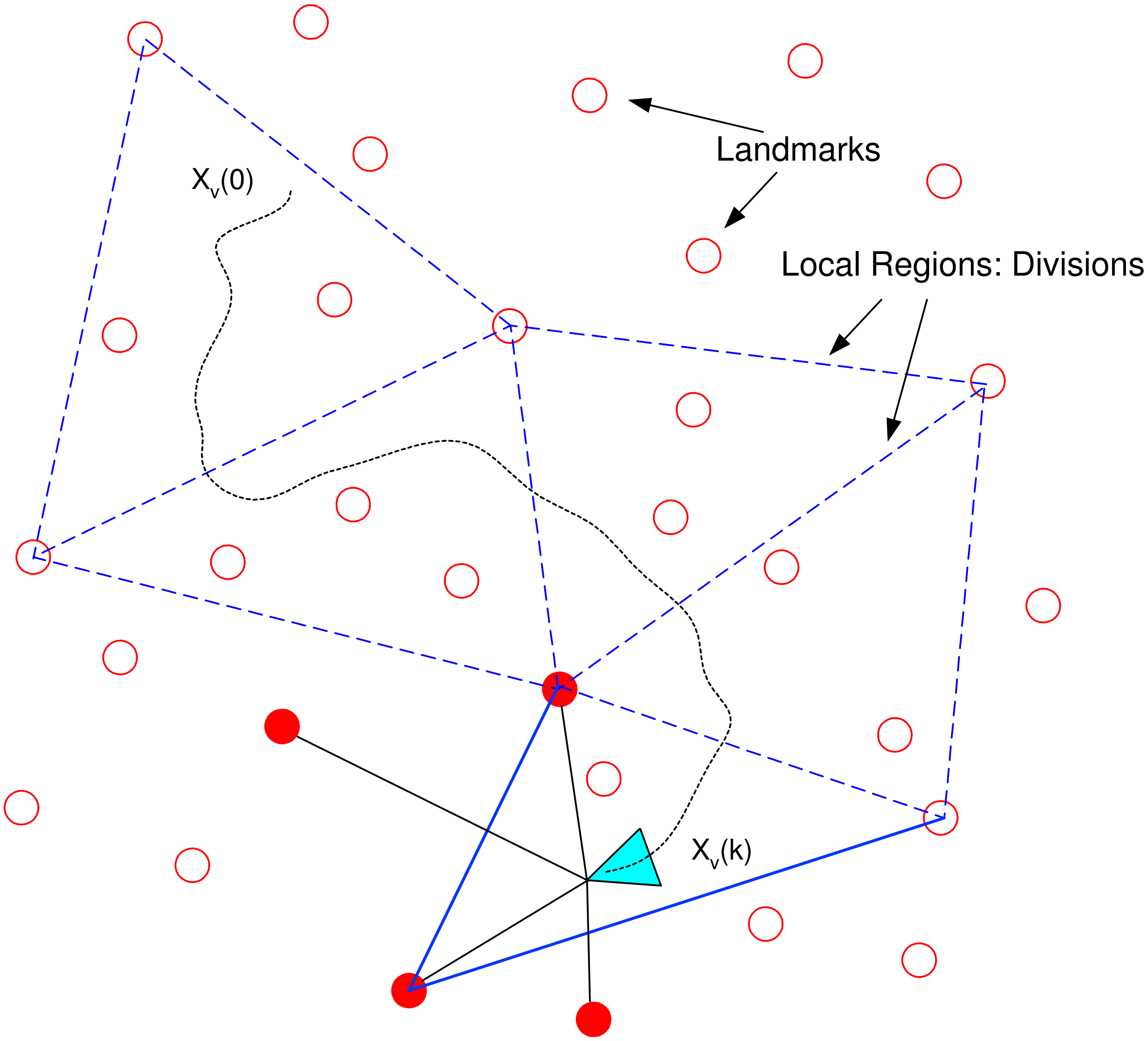


Figure 3.1: Landmarks map ('o') and a particular partition of local triangular regions (LTRs). As shown, not all the landmarks are needed as vertex points in the LTRs definition.

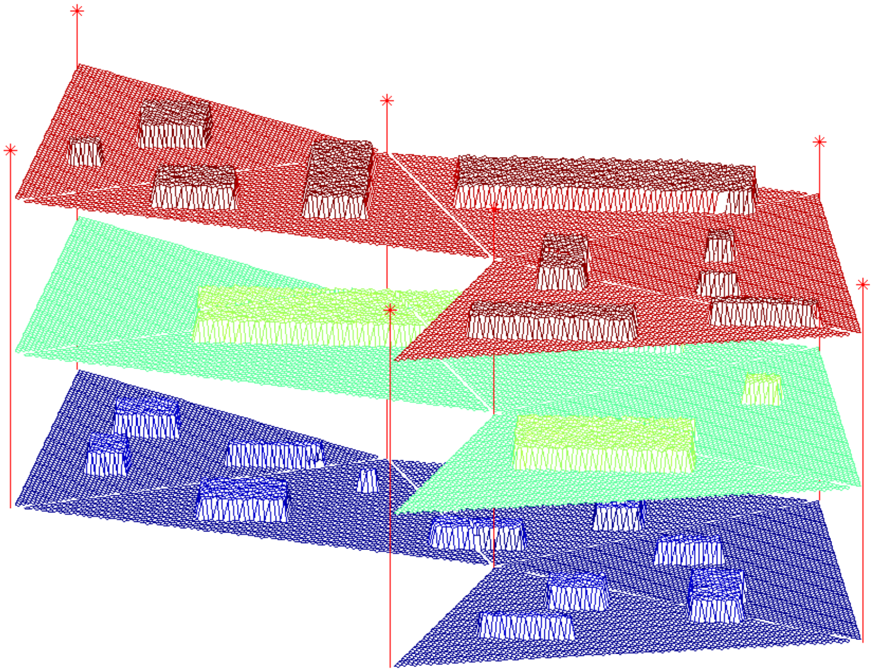


Figure 3.2: Hypothetical multi-layer dense map. The '*' represent the landmarks position and the map layers depict different environment properties captured by the sensors.

assist the SLAM process (e.g. during the data association process).

These are the basic reasons why DenseSLAM can incorporate more information than feature-based SLAM without making the computational burden intractable and without losing consistency.

The next sections explain the fundamental aspects of the algorithm.

- Fundamental Principle of the HYMMs: Incorporating all the sensory information into the state vector and maintaining all the correlations in a dense map representation is infeasible due to the computational cost. For that reason simplifications are needed to make the problem tractable. Section 3.3 explains the simplifications made by the HYMMs.

- Relative representation: The relative representation is used to fuse the dense information. Different relative representations could be used. Section 3.4 discusses the different possibilities and shows the one that will be used in this thesis.

- Base landmarks selection: The framework selects the base landmarks in order to reduce the errors that could be introduced by ignoring correlations between the dense maps and the landmarks. Section 3.5 explains the properties that have to be verified before initialising a new local region. It will also be shown that in cases where there

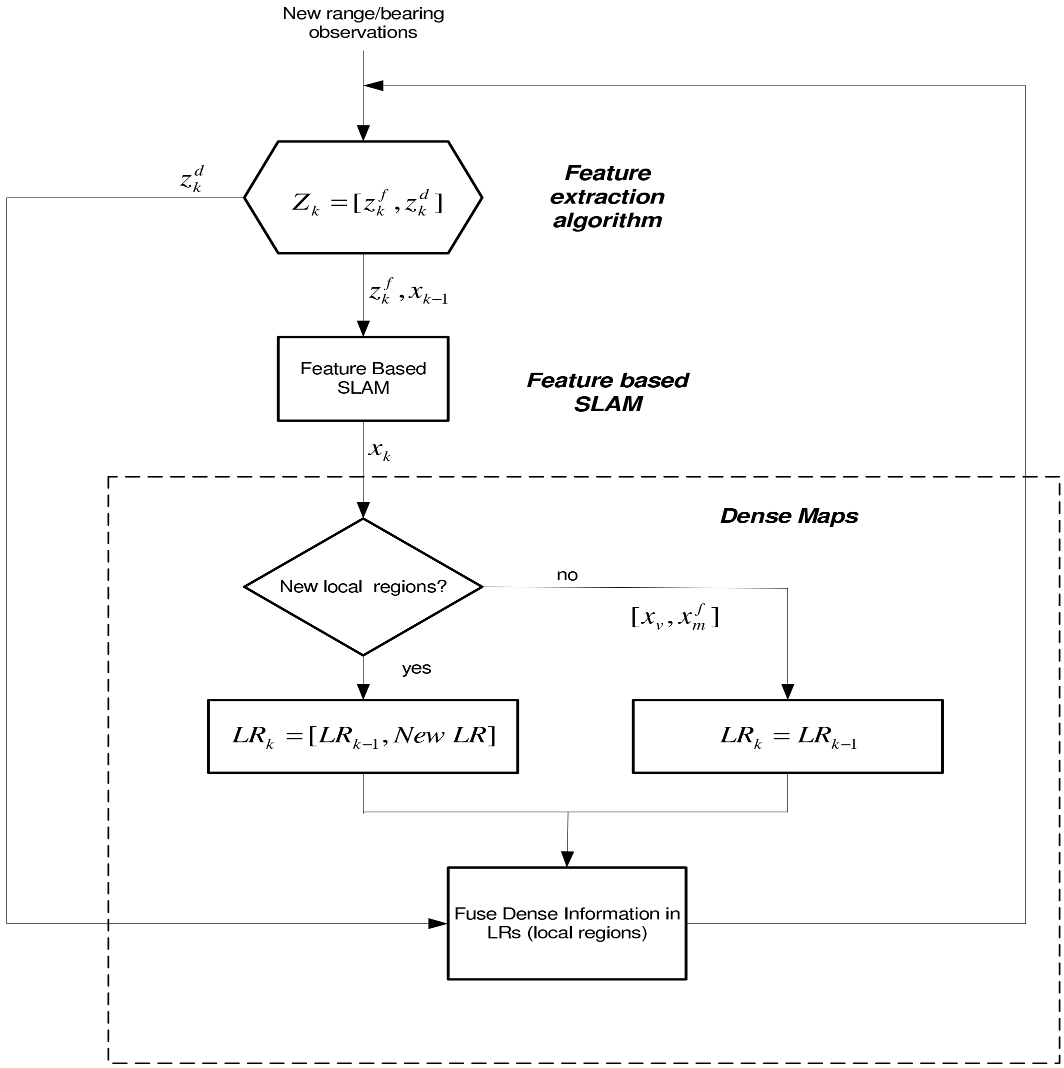


Figure 3.3: HYMM algorithm flow diagram.

are not enough landmarks to form a base, virtual observations can be used to define the local regions. These virtual observations will create virtual landmarks.

- Estimation error in the position of the dense maps: Section 3.6 shows the error in the dense maps position produced by neglecting the correlations with the landmarks map.

- Consistency: The reason why SLAM algorithms are computationally expensive is because of the correlations. Maintaining these correlations is essential to ensure consistency. Section 3.7 discusses the consistency of DenseSLAM and demonstrates that the vehicle pose and feature positions obtained with DenseSLAM are in the worst case, the same as those estimated using the classic EKF-SLAM.

3.3 Fundamental Principle of the HYMMs

Since all the observations are taken relative to the vehicle pose, any representation created will be correlated. Augmenting the state vector with all the information rendered by the sensors and maintaining the correlations is infeasible due to the computational burden involved. Therefore, DenseSLAM maintains only the necessary correlations to ensure consistency and synthesizes the rest of the sensory information into the local maps.

The approximation made by the algorithm consists in representing the dense information in the local regions without including the correlations between the locally represented information and the rest of the system. These correlations will be zero only when there is full correlation between the local property (expressed in global coordinates) and the features that define the respective local frame (assuming the same uncertainty magnitude), so their relative positions are perfectly known. Although it can be proved that in a SLAM process the map becomes fully correlated in the limit [23], in practice only high correlation is achieved. However, it can be demonstrated that the assumptions made by the HYMM framework are, in practice, very good approximations for SLAM problems. The next paragraphs explain two well known properties of SLAM that justify the approximations made in the HYMMs:

1. *Geographically close objects have high correlation:* If a set of observed objects is geographically close from the vehicle viewpoint, then the error due to the vehicle pose uncertainty will be a common component of these estimated objects' positions. This is a typical situation in SLAM where the vehicle accumulates uncertainty in its estimated position and incorporates observations that are used to synthesize a map. Due to this fact the estimates of objects that are geographically close will present similar uncertainties (high cross-correlations). Any update of a particular object will imply

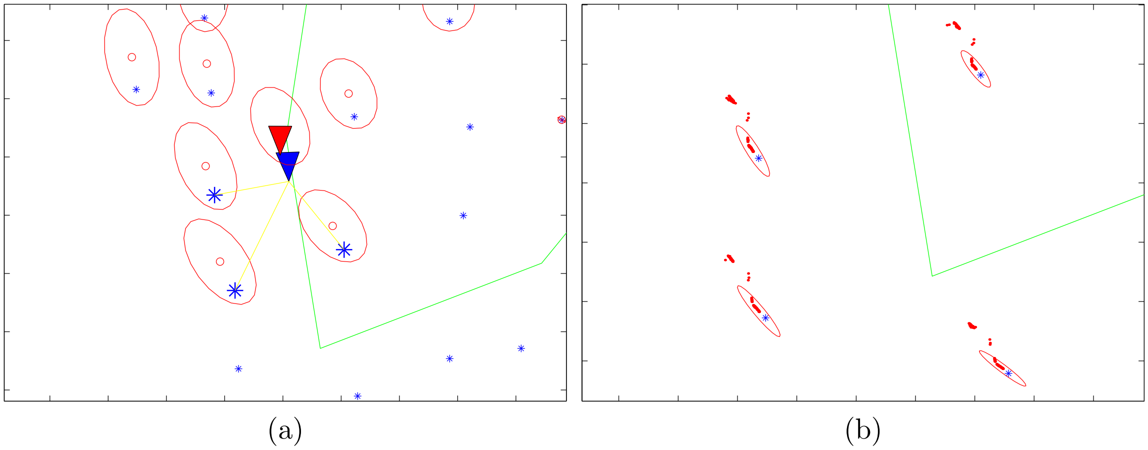


Figure 3.4: Map Correlation: The figures show that geographically close objects posses similar uncertainty. Figure (a) shows how the landmarks that are being observed have similar uncertainty to the robot pose. (b) shows how the estimated landmarks means are updated after the vehicle closes the loop. The red dots represent landmark position estimates over time. High correlation in geographically close objects is one of the SLAM characteristics; because the vehicle will observe close objects at similar instants it will propagate similar uncertainty to the objects.

a similar update of any objects sufficiently close to the first one. Figure 3.4 shows an example of a typical SLAM map. The figure shows a landmarks map with its uncertainty bounds. It can be seen that the uncertainty is very similar in landmarks that are geographically close.

2. *The relative representation stores close objects in local coordinate frames and then permits the reduction of correlation to the rest of the map [29]:* Assume a landmark can be represented in a local frame in the following way:

$$\mathbf{x}^L = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{x}_b) \tag{3.1}$$

where $\mathbf{x}^L$ represents the relative landmark position, $\mathbf{x}_v$ the vehicle position, $\mathbf{z}$ the observations and $\mathbf{x}_b$ the position of the landmarks that define the local frame (base landmarks).

Taking into account that the observation errors are independent of the vehicle position (as it is always assumed in SLAM), the cross-correlation between the vehicle and the landmark in the local frame will be:

$$\mathbf{P}_{vL} = \mathbf{P}_{vv} \nabla \mathbf{h}_{\mathbf{x}_v}^T + \mathbf{P}_{vb} \nabla \mathbf{h}_{\mathbf{x}_b}^T \tag{3.2}$$

where $\mathbf{P}_{vv}$ represents the vehicle states covariance, $\mathbf{P}_{vb}$ the cross-correlation between the vehicle states and the base landmarks position estimated and $\nabla \mathbf{h}_{\mathbf{x}_i} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_i}$ is the jacobian matrix of $\mathbf{h}$ with respect to the state $\mathbf{x}_i$.

For the one dimensional case for example, Equation 3.1 becomes:

$$x^L = h(x_v, z, x_b) = x_v + z - x_b \tag{3.3}$$

Applying Equation 3.3 to 3.2:

$$\mathbf{P}_{vL} = \mathbf{P}_{vv}(1) + \mathbf{P}_{vb}(-1) = \mathbf{P}_{vv} - \mathbf{P}_{vb} \tag{3.4}$$

Equation 3.4 shows that if the magnitudes of $\mathbf{P}_{vv}$ and the covariance of the base landmarks $\mathbf{P}_{bb}$ are similar , when the robot is highly correlated with the base landmarks there will be almost no correlation between the robot position and the local landmark ($\mathbf{P}_{vL}$) and then no correlation between the relative landmark and the rest of the map. Because the relative and the base landmarks are geographically close, whenever the robot observes the local landmark it will be highly correlated with the base landmarks. This fact will reduce the correlation between the local landmark and the robot and therefore the correlation between the local landmark and the rest of the map will be reduced as well.

A more direct way of observing the decorrelation effect will be by evaluating the cross-correlation between a landmark in the global frame with a landmark represented in a local frame and comparing this with the cross-correlation of the same two landmarks, both represented in the global frame. In a similar manner to Equation 3.2, the cross-covariance matrix between the j-th landmark and a locally represented landmark can be evaluated in the following way:

$$\mathbf{P}_{jL} = \mathbf{P}_{jb} \nabla \mathbf{h}_{\mathbf{x}_b}^T + \mathbf{P}_{jG} \nabla \mathbf{h}_{\mathbf{x}G}^T \tag{3.5}$$

where the prefix j means the j-th landmark, b the bases landmarks that define the local frame, L the locally represented landmark and G the position of the local landmark in the global frame. Then given $\mathbf{P}_{jb}$, $\mathbf{P}_{jG}$ and the transforming function from the global to the local frame $\mathbf{h}$, it is possible to evaluate the cross-correlation between the local landmark and the j landmark in the map. Although the effect of decorrelation happens regardless of the particular local representation used, finding an expression to demonstrate that $\mathbf{P}_{jL} << \mathbf{P}_{jG}$ will be dependent on the local representation used. Equation 3.5 shows that for a particular local representation $\mathbf{h}$, the decorrelation effect between the local object and the rest of the map will depend on the cross correlation between the rest of the map and the base landmarks and the cross correlation between the rest of the map and the local represented object in the global frame.

A numerical example is shown next to illustrate the decorrelation effect using the local representation utilised in this thesis. Section 3.4 gives details about the representation and Appendix C presents the equations for the derivatives using local triangular regions.

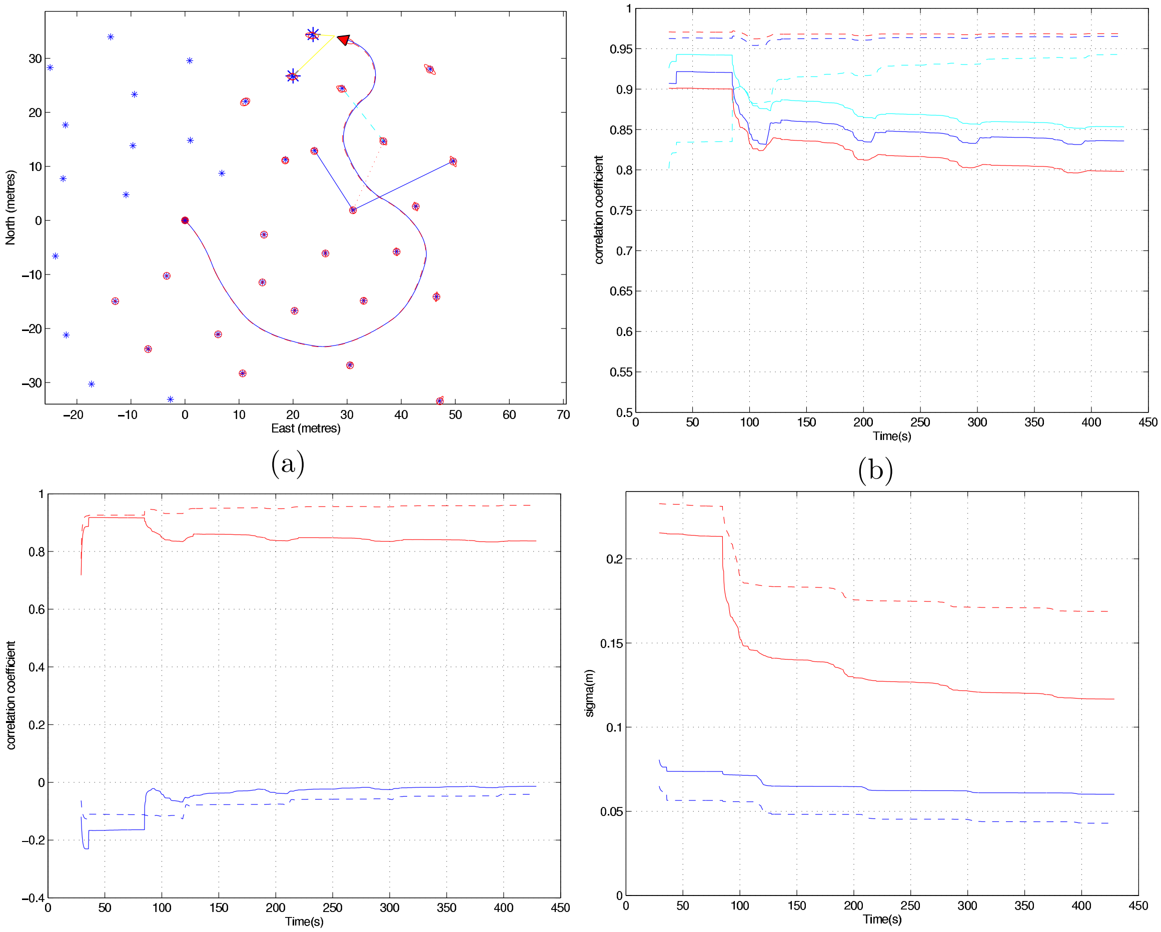
Example 3.1

The simulation environment presented in Section 2.9 is used here to illustrate the decorrelation effect. Figure 3.5 (a) shows a zoom-in of the rightmost part of the environment presented in Figure 2.8. In order to show the decorrelation effect, a local region was defined using three landmarks and one landmark was represented in this local frame. The i-th landmark is then represented in the local frame after a few observations which ensures high correlation with the base landmarks. Then, the landmark is not observed again, as it actually happens with the dense sensory information (it is assumed that is not possible to observe exactly the same point of the environment more than once). The blue solid line in Figure 3.5 (a) shows the local frame axis and the red dotted line the local landmark position $\mathbf{x}_{Li}$*. The cyan dashed line joins the local represented landmark* $\mathbf{x}_{Li}^{L}$ *with a global landmark* $\mathbf{x}_{Lj}$*. The cross-correlation between* $\mathbf{x}_{Lj}$ *and the local i-th landmark will be evaluated when the i landmark is represented in the local frame* $\mathbf{x}_{Li}^{L}$ *and when it is represented in the global frame* $\mathbf{x}_{Li}^{G}$*.*

Figure (b) shows the evolution in time of the correlation coefficient of the i landmark represented in global coordinates $\mathbf{x}_{Li}^{G}$*, with the landmarks used to define the local frame. The solid line depicts the cross-correlation in the east and the dashed line in the north. The different colours represent the cross-correlation with the different base landmarks. As can be seen, the landmark* $\mathbf{x}_{Li}^{G}$ *possesses high correlation with the base landmarks, this is due to the geographical proximity between the landmarks. Figure (c) shows the correlations between* $\mathbf{x}_{Li}^{G}$ *and* $\mathbf{x}_{Lj}$ *in red, and the correlations between the j landmark and the landmark i represented in the local frame* $\mathbf{x}_{Li}^{L}$ *(Equation 3.5) in blue. The correlation was reduced from almost one when the landmark is represented in global coordinates, to almost zero when it is represented in the local frame.*

Finally (d) shows the the variance of the landmark i, the blue line depicts the variance when the landmark is in the local frame and the red line when it is in global. Because of the high correlation between $\mathbf{x}_{Li}^{G}$ *and the base landmarks, the uncertainty in their relative position is very low, and so is the variance of* $\mathbf{x}_{Li}^{L}$*.*

In summary the relative representation used by DenseSLAM permits the locally repre-



(a)

(b)

(c)

(d)

Figure 3.5: Decorrelation effect: (a) shows a zoom of the navigation environment where three landmarks were used to define a local region and one landmark $\mathbf{x}_{Li}$ (red dashed line) was represented in this local frame. (b) shows the correlation coefficient over time between the landmark i represented in the global frame and the base landmarks. (c) shows the decorrelation effect; when the landmark i is represented in local coordinates (blue line) the cross-correlation with other landmarks is considerably reduced in respect to the correlation between other landmarks and the landmark i-th in global coordinates (red line). (d) shows the landmark position standard deviation when it is represented in local (blue line) and global (red line) coordinates.

sented information to be decorrelated with the rest of the system, allowing the incorporation of more information without increasing the computational cost.

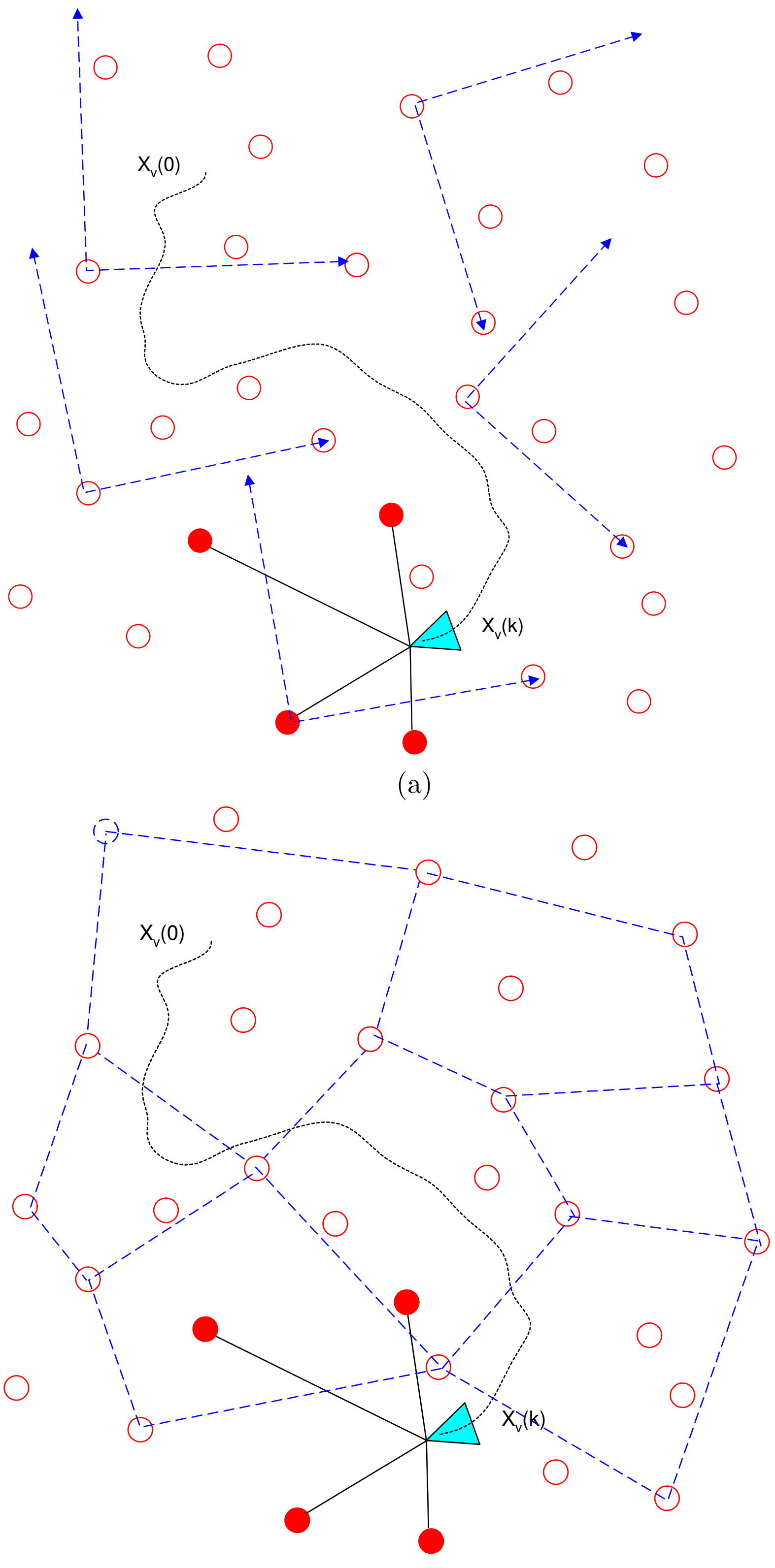
3.4 Relative representation

This section discusses different ways to define the local frames and explains the one used from now on in this thesis.

The most common relative representation employs two landmarks to represent the local frames [28] and defines rectangular local regions. The rectangular representation has an important shortcoming when used as the local regions in DenseSLAM; it presents problems in terms of map coverage. Figure 3.6 (a) shows a hypothetical map division using rectangular local regions. As seen in the figure, some local regions overlap. In addition, some areas of the map are not covered by any local region. A solution to cover the whole area would be to overlap all the regions. However this will be inefficient in terms of computation and memory, and also will add difficulty to the global map recovery task. A better solution to cover the whole area in an efficient manner will be to define all the local regions in a way that they share a common axis, thus avoiding overlapping or gaps between different local maps. Figure 3.1 shows a possible map division using three landmarks for the bases. Unlike the rectangular representation, the angle between the axes is not restricted to a particular value, the angle is defined by the landmarks position, and can be different for different regions. In this case the global map is partitioned into *local triangular regions* (LTRs)[2]. Because the local regions always have common vertices, the whole area can be covered without overlapping regions. However, this is not a particular property of the triangular division, it is actually a consequence of not confining the angle between the axes to a fixed value, so avoiding the restriction of the local maps to homogeneous regions.

More than three landmarks could be used to define local regions, presenting similar characteristics to the LTRs in terms of map coverage. Figure 3.6 (b) shows an example of a map division using more than three landmarks; four and five landmarks were used to delimit the regions. Using more than three landmarks will imply defining more than one frame per local region, which will give more robustness to the representation but will add complexity to the implementation. The complexity arises due to the larger number of landmarks needed to form a region and the repetition of information in different frames. The representation used in this thesis employs three landmarks to delimit the local regions. This is the most efficient representation for our purpose, since it involves the smallest number of landmarks while assuring full coverage of the area. For this reason all the theory and examples presented in this thesis use *Local Triangular Regions* (LTRs).

[2] Thanks to Jose Guivant for his suggestion about the use of triangular regions



(a)

(b)

Figure 3.6: Figure (a) shows an example of a map division using rectangular regions, (b) shows a map division using four and five landmarks to delimit the regions.

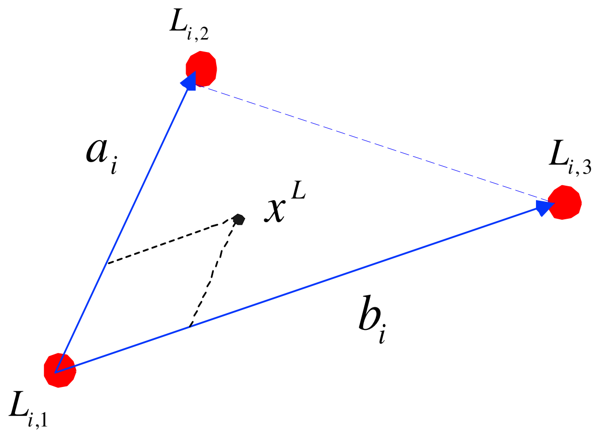


Figure 3.7: Detail of an individual LTR defined by three vertex points, $\{\mathbf{L}_{i,1}, \mathbf{L}_{i,2}, \mathbf{L}_{i,3}\}$ and the direction vectors, $\{\mathbf{a}_i, \mathbf{b}_i\}$.

3.4.1 Local Triangular Regions (LTRs)

Figure 3.7 shows an example of a LTR. Any point that belongs to a LTR can be characterised by a convex linear combination of the three vertex points (landmark's position) associated with this subregion. In Figure 3.7 a LTR Ω_i is defined by the vertex points $\{\mathbf{L}_{i,1}, \mathbf{L}_{i,2}, \mathbf{L}_{i,3}\}$. A local coordinate frame is defined based on the three vertex points according to:

$$\begin{aligned} \mathbf{a}_i &= \mathbf{L}_{i,2} - \mathbf{L}_{i,1} = [a_x, a_y]^T \\ \mathbf{b}_i &= \mathbf{L}_{i,3} - \mathbf{L}_{i,1} = [b_x, b_y]^T \end{aligned} \tag{3.6}$$

Any point that belongs to Ω_i can be expressed in the global frame as:

$$\begin{aligned} \mathbf{x} &= \mathbf{L}_{i,1} + \alpha \cdot \mathbf{a}_i + \beta \cdot \mathbf{b}_i \\ &= (1 - \alpha - \beta) \cdot \mathbf{L}_1 + \alpha \cdot \mathbf{L}_{i,2} + \beta \cdot \mathbf{L}_{i,3} \end{aligned} \tag{3.7}$$

$$\begin{aligned} \alpha > 0, \;\; \beta > 0 \\ \alpha + \;\; \beta \;\; \leq 1 \\ \forall \;\; \mathbf{x} \;\; \backslash \;\; \mathbf{x} \in \Omega_i \end{aligned}$$

Furthermore any function of the global position $\mathbf{x}$ can also be locally defined as a function of the local coordinates of $\mathbf{x}$ and the vertices points:

$$z = f(\mathbf{x}) = f(\mathbf{L}_{i,1} + \alpha \cdot \mathbf{a}_i + \beta \cdot \mathbf{b}_i) \tag{3.8}$$

Using (3.6) and (3.7) an expression for the local coordinates (α, β) can be derived.

$$\begin{aligned} \alpha &= \frac{(x_y - L_{1y})d_{13x} - (x_x - L_{1x})d_{13y}}{d_{13x}d_{12y} - d_{13y}d_{12x}} \\ \beta &= \frac{(x_x - L_{1x})d_{12y} - (x_y - L_{1y})d_{12x}}{d_{13x}d_{12y} - d_{13y}d_{12x}} \end{aligned} \tag{3.9}$$

where $\mathbf{x} = [x_x, x_y]$ are the global coordinates of the local point and $\mathbf{d}_{ij} = \mathbf{L}_j - \mathbf{L}_i = [\mathbf{d}_{ijx}, \mathbf{d}_{ijy}]^T$. In general, any local point can be expressed as:

$$\mathbf{x}^L = [\alpha, \beta]^T = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{L}) \tag{3.10}$$

where $\mathbf{x}^L$ represents the local state, $\mathbf{x}_v$ the robot pose, $\mathbf{z}$ the observations and $\mathbf{L}$ the set of landmarks that define the local region.

Assume a vehicle exploring the environment and measuring different properties such as soil salinity, humidity, terrain occupancy, etc. Not necessarily all the sensory information has to be used for the robot localisation process. Nevertheless, it may be desired to maintain all the information for additional tasks, such as path planning, or for example the detection of a particular property in the environment. The classic SLAM algorithm uses features extracted from sensed data for the localisation process and neglects the rest of the information. DenseSLAM also uses a features map, but in addition the algorithm integrates all the sensory information using local representation. The dense maps obtained consist of map layers, where each layer represents the information coming from different sensors, or the same sensor information represented in different ways.

It is worth making clear that the framework is not restricted to any particular representation for the local maps. The only restriction is that the representation used needs to be defined relative to the features positions. Figure 3.8 illustrates an example of different internal properties in a given LTR. Grid maps were used in this case to represent the local maps.

3.5 Base landmarks selection

To minimise the error in the position of the dense information, the framework establishes some conditions that have to be previously attained by the potential landmarks to form the

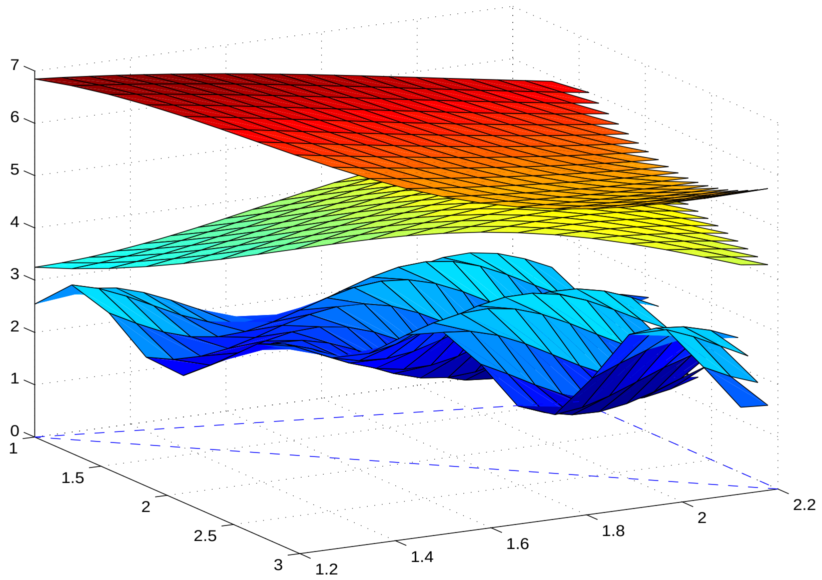


Figure 3.8: A set of properties can be defined as a function of the local coordinate variables.

vertices of a LTR. This section presents these conditions.

- *The landmarks that define a LTR must have high correlation*: As explained in Section 3.3, DenseSLAM uses the fact that in SLAM geographically close objects possess high correlation and so their relative position is almost independent of the vehicle pose uncertainty. This makes the locally represented objects have very weak correlation with the rest of the system and then allows the algorithm to neglect these correlations. In particular, because the locally represented entities are not used directly for the localisation process, the framework does not introduce any inconsistency at all in the system (Section 3.7.1 will explain this aspect in more detail).

  The ideal situation for the algorithm will be when all landmarks and objects in a common local region are fully correlated and present the same uncertainty values. In this case their relative position is perfectly known (zero uncertainty) and the correlation between the locally represented objects and the rest of the system will be zero. The higher the cross-correlation among objects in the global frame, the lower the uncertainty of their relative position.

  As a conclusion of the above discussion, it can be deduced that the higher the correlation between the base landmarks and the locally represented objects (local map), the better the quality of the local map. In particular, since the axes of the local frame are defined based on the relative position between the base landmarks (Figure 3.7) it is also important to have high correlation among the landmarks that form a base.

High correlation will prevent deformations in the frame after the landmarks positions are updated. Figure 3.9 clarifies this with an example. In (b) the region was defined with three landmarks that did not possess high correlation, and an object was defined relative to this base. After the landmarks are updated, the local region is misshapen and so is the object. Figure 3.9 (a) shows the same example, but now the region is formed after the landmarks achieved high correlation. Then, after the update, the local region has moved without major changes in the shape. This is the consequence of having low/high correlation among the base landmarks.

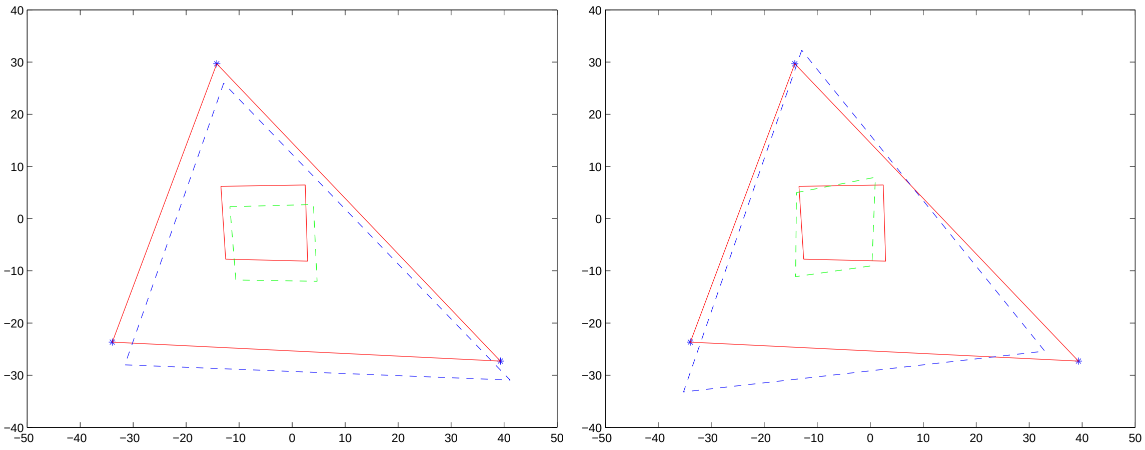
- *The landmarks must also be geometrically well conditioned (to avoid degeneration)*: This condition is related to the geometry of the local region. Before defining a local region, the angle between the local axes has to be checked. Figure 3.10 (a) shows an ill conditioned LTR. We refer to this case as *ill conditioned* because two axes are 'too close' each other, a situation that could make the triangle flip over. Deciding when the axes are too close or when a region is ill conditioned is a process that involves not only the cross correlation between the landmarks but also the individual uncertainty. With the individual uncertainty and the correlations, it can be predicted how much a landmark can change its position after an update, and then predict if any potential region could flip over. An easy and conservative test can be done considering the landmarks independent (no correlation) and then comparing the uncertainty bounds of the landmarks against the opposite axis. For example, in Figure 3.10 (a), if the uncertainty bound (ellipse) of L_2 has contact with its opposite axis (formed by L_1 and L_3), then the region could flip over. In the example presented, the uncertainty bounds of L_2 have contact with the opposite axis, and so the region is said to be *ill conditioned*. This is a very conservative test, since the fact that there is correlation between landmarks will restrict the relative movements.

- *New regions should not overlap with old regions*: The last condition to check is that the regions should not overlap to avoid repetition of the information. Figure 3.10 (b) shows an example of two overlapped regions.

Having verified all these conditions, a new local region can be created and a new local dense map will be built using some adequate representation.

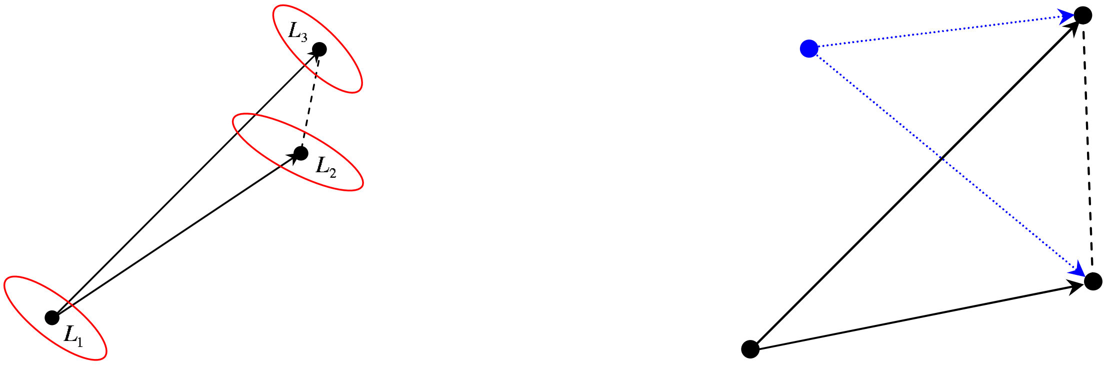
3.5.1 Virtual landmarks

There will be situations where part of the map cannot be included inside any local region. For example Figure 3.11 illustrates a case where the environment consists of objects and buildings. If the sensors cannot see through the building's walls and the robot cannot go



(a) (b)

Figure 3.9: Figure (a) shows a LTR and an object defined locally for the case where the landmarks possess high correlation. The solid line denotes the local axes and object position before the update and the dashed line after the landmarks are updated. (b) shows the case with low correlation in the base landmarks. The solid red line denotes the actual axes and object position. The green dashed line denotes the object after the region is defined and the landmarks are updated. In (b) the object is misshapen due to the low correlation in the base landmarks.



(a) (b)

Figure 3.10: Figure (a) shows a LTR which is not geographically well conditioned: the region could flip over. The dots represent the base landmark positions and the ellipses the uncertainty bounds. Figure (b) illustrates a case where two LTRs overlapped.

inside the building, using the method presented, no regions could include the walls, since all the landmarks will be outside. In order to include the walls and keep the same type of regions throughout the map, the local regions could be formed with two real landmarks, and one *virtual landmark*, which could be created inside the building (see Figure 3.11). The virtual landmark will have to be created at a moment where the vehicle is highly correlated with the other two landmarks. The difference between using two (the third landmark is artificial so does not have information) or three landmarks will be reflected in the accuracy of the dense map. The more landmarks used to define the axes of the local region, the less correlated will be the local map with the rest of the system.

The only purpose of the virtual landmark is to have only one type of local region throughout the whole map, which then makes the implementation of the algorithm easier. However, a different strategy could be adopted, and in the circumstances where a part of the environment is not enclosed by any LTR, different type of region could be used. For example, three landmarks could be used to delimit the axis, and one axis could be extended to go inside the building. This strategy is more robust since three landmarks are used to create the local region but as mentioned before, this requires the algorithm to have different types of local regions and to have a mechanism to distinguish between the different scenarios to apply each particular configuration.

3.6 Estimation error in the position of the dense maps

It was mentioned in Section 3.3 that the correlations among the dense maps and between the dense maps and the landmark map are not maintained. This was the main motivation behind the use of the local representation for the dense maps; reduce these correlations and then even when they are not maintained, make the approach a close approximation to full DenseSLAM. Neglecting these correlations does not introduce any inconsistency into the system, since only the landmark map is used for localisation. The effect of the decorrelation will be an error in the position of the dense maps. This section derives the error introduced in the dense map positions by neglecting the correlation with the landmark map.

Once the location of some environment property is fused inside the local regions, the algorithm never changes its local coordinates (α, β). A change on these coordinates can be produced by two means: direct observation of the property or through the cross-correlation with the rest of the system. In this section we investigate the changes in the dense property position produced by the cross-correlation with the rest of the system. Using the relative representation to define the dense maps results in a significant reduction in the correlation between the dense map locations and the rest of the system. Nevertheless, there will be some correlation due to the local vehicle uncertainty. The evaluation of this error is derived

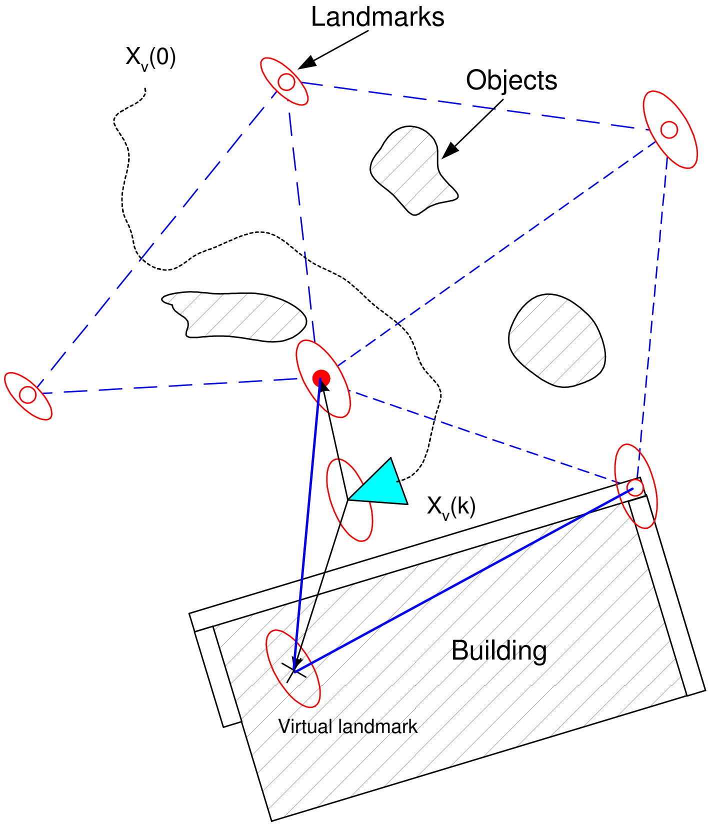


Figure 3.11: *Virtual landmark*: in order to include the walls in the representation, the vehicle creates a virtual landmark inside the building and then includes the wall in a new LTR formed by two real landmarks and one virtual.

next.

As seen in Chapter 2, after the vehicle takes new observations, the update produced to the first moment of the state vector is given by:

$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}}^{-} + \Delta\mathbf{x} \tag{3.11}$$

where

$$\Delta\mathbf{x} = \mathbf{P}^{-}\nabla\mathbf{h_x}\mathbf{S}^{-1}\nu \tag{3.12}$$

Assume a range and bearing observation of the i-th landmark is taken

$$\mathbf{z} = \begin{bmatrix} h_r(\hat{\mathbf{x}}_{\mathbf{v}}^{-}, \hat{\mathbf{x}}_{\mathbf{Li}}^{-}) \\ h_\theta(\hat{\mathbf{x}}_{\mathbf{v}}^{-}, \hat{\mathbf{x}}_{\mathbf{Li}}^{-}) \end{bmatrix} \tag{3.13}$$

And the innovations vector ν and innovations covariance $\mathbf{S}$ are calculated.

$$\mathbf{S}^{-1} = \begin{bmatrix} S_1 & S_2 \\ S_3 & S_4 \end{bmatrix} \tag{3.14}$$

$$\nu = \begin{bmatrix} \nu_r \\ \nu_\theta \end{bmatrix} \tag{3.15}$$

Using Equation 3.12 it can be shown that the update produced in the mean of a non-observed landmark j due to the correlation with the vehicle and landmark i will be:

$$\Delta\mathbf{x}_j = (\mathbf{P}_{jv}\nabla h_{r_{\mathbf{v}}}^T + \mathbf{P}_{ji}\nabla h_{r_{\mathbf{i}}}^T)(S_1\nu_r + S_2\nu_\theta) + (\mathbf{P}_{jv}\nabla h_{\theta_{\mathbf{v}}}^T + \mathbf{P}_{ji}\nabla h_{\theta_{\mathbf{i}}}^T)(S_3\nu_r + S_4\nu_\theta) \tag{3.16}$$

where $\mathbf{P}_{jv}$ is the cross-covariance matrix between the landmark j and the vehicle pose. $\mathbf{P}_{ji}$ is the cross-covariance between the landmarks j and i, and:

$$\nabla h_{r_{\mathbf{v}}} = \frac{\partial h_r}{\partial \mathbf{x}_v} \tag{3.17}$$

$$\nabla h_{r_{\mathbf{i}}} = \frac{\partial h_r}{\partial \mathbf{x}_{Li}} \tag{3.18}$$

$$\nabla h_{\theta_{\mathbf{v}}} = \frac{\partial h_\theta}{\partial \mathbf{x}_v} \tag{3.19}$$

$$\nabla h_{\theta_{\mathbf{i}}} = \frac{\partial h_\theta}{\partial \mathbf{x}_{Li}} \tag{3.20}$$

Equation 3.16 represents the error in the location of the dense maps that the algorithm makes, by not considering the correlation between the dense information and the rest of the system. An example where Equation 3.16 is calculated for a local and for a global landmark is presented below.

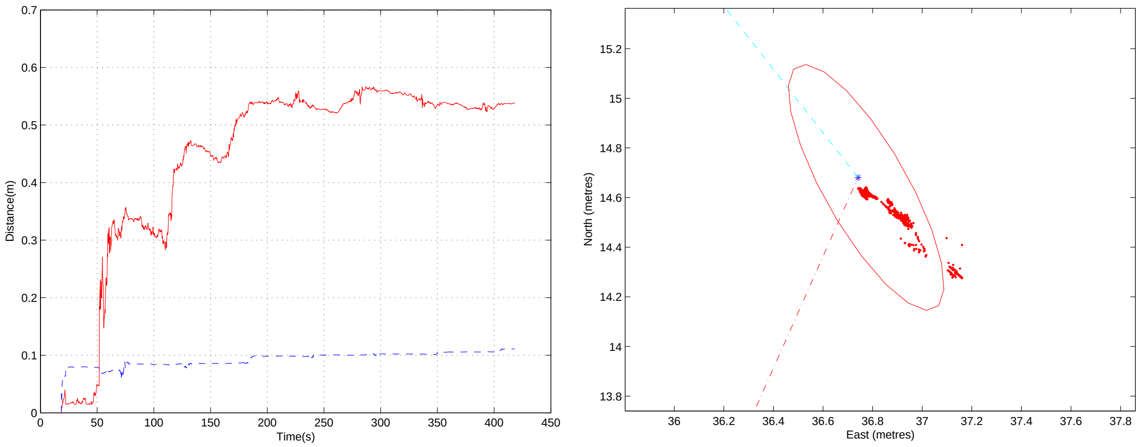
Example 3.2

The changes in position of a local represented landmark due to the correlation with the rest of the map are evaluated here, using the environment presented in Example 3.1. In Figure 3.12 (a) the dashed blue line shows the evolution in time of the landmark i represented in local coordinates $\mathbf{x}_{Li}^{L}$*, and the red solid line illustrates the landmark position evolution when the landmark is expressed in global coordinates* $\mathbf{x}_{Li}^{G}$*. The lines depict the distance in metres of the landmark position with respect to the initial position estimated. The landmark is observed several times and once it is highly correlated with the base landmarks, it is not observed again. This is to imitate what actually happens with the dense information, where there is no data association process. This means that once an observation is fused, it is assumed that the same point is never observed again. Therefore the landmark in this example, is not directly observed, the changes in position are only due to the correlations with the rest of the map, thus basically the lines in Figure 3.12 (a) are a representation of the error introduced by ignoring the update term shown in Equation 3.16, when the landmark is represented in local and in global coordinates. It can be observed that even when the position of the landmark in global coordinates changes more than 50 centimetres with respect to the initial position, its local coordinates only change approximately 10 centimetres when the loop is closed (largest update) and is rapidly stabilised to an almost constant value, while the global position continues changing. This 10 centimetres change in the local landmark coordinates is the error that the algorithm will make by not considering the correlations between the dense maps and the rest of the system, which as shown, is reduced drastically using the relative representation with respect to the error produced using the global frame to represent the dense maps.*

The red points in Figure 3.12 (b) depict the changes in position of the landmark i in global coordinates. The initial position is the bottommost point and then the estimate starts to move to where the actual landmark position is located.

3.7 Consistency in DenseSLAM

This section discusses consistency in DenseSLAM. The next subsection will introduce a new concept named UIF, which is used to demonstrate that the landmark map and the



(a) (b)

Figure 3.12: Figure (a) shows the changes in the landmark position, the red solid line depicts the changes when the landmark is represented in global coordinates and the dashed blue line when it is represented in a local frame. The red points in (b) depict the changes in position of the landmark i in global coordinates. The initial position is the bottommost point and then the estimate starts to move to the actual landmark position.

localisation process obtained with DenseSLAM are optimal in comparison with feature-based SLAM.

3.7.1 The Unidirectional Information Flow (UIF)

DenseSLAM is built upon two key variants of the basic SLAM algorithm:

- *The relative representation*: The representation stores the dense information in local coordinate frames reducing the cross-correlation between the dense representation and the landmark map.

- *The Unidirectional Information Flow*: The representation considers information going from the feature-based SLAM to the dense maps. There is no direct information flow from the dense maps to the feature-based SLAM. We call this effect Unidirectional Information Flow [62]. Due to the UIF property, it is possible to augment the feature-based SLAM algorithm to obtain a more detailed representation. The introduction of the UIF also guarantees the system remains consistent. This characteristic ensures that DenseSLAM will never perform worse than a standard feature-based SLAM.

The concept of *Unidirectional Information Flow* (UIF) for state-space estimation is now introduced, and the manner in which the HYMM representation uses this concept to ensure consistency in DenseSLAM is explained.

The notion of UIF is relevant to systems where the state-vector is divided into two groups: (i) states that both give to and receive information from the rest of the system and (ii) states that only receive information. These latter states are said to possess UIF. They are a set of states that receive indirectly the information obtained by the other states (through correlation), but they do not give information back to the rest of the system. Figure 3.13 illustrates this concept for a DenseSLAM application. In this case there is a unidirectional information flow from the feature-based SLAM to the dense map.

Example 3.3 clarifies the idea, applying the UIF concept to the feature-based SLAM problem[3].

Example 3.3

Consider a landmark map where a subset of landmarks N_1 is estimated by the SLAM algorithm and a second set of landmarks N_2 that are observed only once. The latter set will not provide information to the rest of the system (because they are observed only once). However since they are correlated to the set N_1 due to robot pose uncertainty they will receive information from N_1. As a consequence it is not necessary to maintain the cross-correlation among the landmarks set N_2, only the cross correlation between N_1 and N_2 will be necessary. A full filter implementation without considering UIF will require the update of a covariance matrix of $(N_1 + N_2)(N_1 + N_2)$. By considering the UIF property only $N_1(N_1 + N_2) + N_2$ elements of the covariance matrix have to be maintained. Figure 3.14 illustrates the example. If $N_1 << N_2$, then the computational burden will be dramatically reduced using the UIF concept.

The UIF is a general concept that can be used in different applications. Take for example an autonomous robot performing SLAM, and at the same time tracking people. Assume the motion model for the people is a random walk model. The state vector will consist of the robot pose, the static landmarks position and the states used to model the motion of the people, for example, speed and position. There will be not information flow from the people's track to the robot pose, because the robot motion model will be more accurate than the random walk model uses for the people tracks. However the static landmarks have a noiseless model, so the robot will use the information coming from the landmarks to localise itself and its position for the tracking. This is a clear example where without adding restrictions to the system about how the information flows, the system naturally behaves with UIF. There is information flowing from the feature-based SLAM to the people tracks, but not information from the people tracks to the robot pose and the landmark

[3] The example does not try to present a solution to the traditional feature-based SLAM problem, it is actually trying to illustrate the UIF concept in a very well known problem.

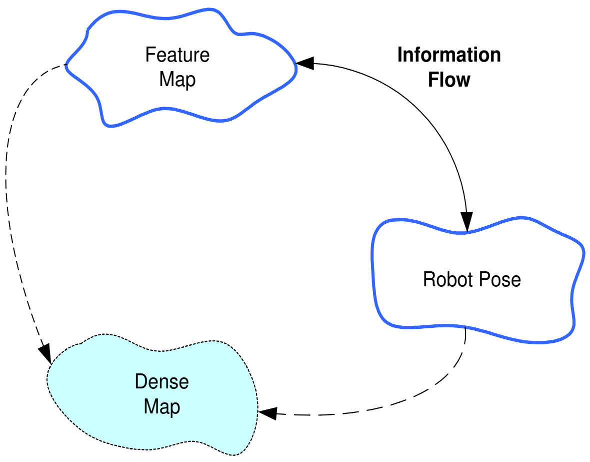


Figure 3.13: *Unidirectional Information Flow* effect in DenseSLAM: The information between feature map and robot is bidirectional while the information between robot and dense map is unidirectional

position estimates. An application similar to this example was introduced in [83], where a vehicle does SLAM in the streets of a city, and at the same time the observations are used to track the cars moving around the vehicle. The implementation presented in [83] uses the information from the SLAM in the tracking algorithm, but it does not use the information from the tracking in the SLAM. The system behaves with UIF.

In this thesis, the UIF will be used to incorporate more information into the map representation. Most stochastic mapping techniques reject a significant part of the sensory information because of the excessive computation cost involved to fuse it in a consistent manner. With the introduction of the UIF a more detailed representation can be obtained in a consistent and tractable manner. In particular the UIF together with the relative representation provide the *key* to a solution for DenseSLAM.

3.7.2 UIF in DenseSLAM

This section demonstrates that by using the UIF concept, DenseSLAM can be decoupled into two problems: The classic feature-based SLAM and the estimation of the dense map. Solving the DenseSLAM problem using a Bayesian formulation requires the evaluation of the following probability distribution:

$$\begin{aligned} & P(\mathbf{x}_k, \mathbf{f}, \mathbf{d} | \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \\ \propto \quad & P(\mathbf{z}_k | \mathbf{x}_k, \mathbf{f}, \mathbf{d}) P(\mathbf{x}_k, \mathbf{d}, \mathbf{f} | \mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \end{aligned} \tag{3.21}$$

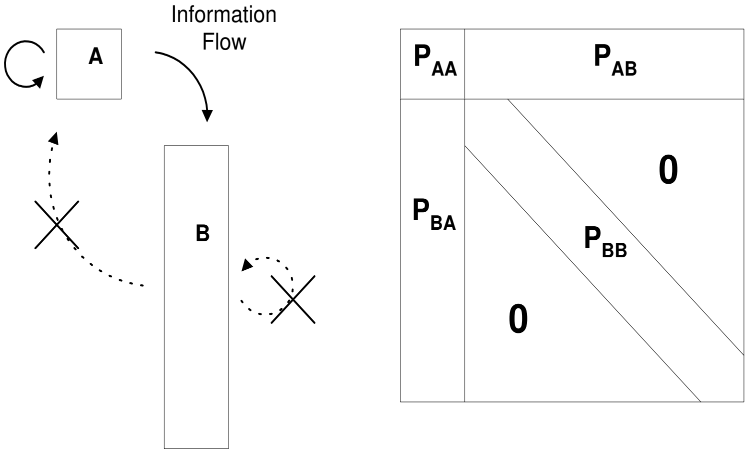


Figure 3.14: *Unidirectional Information Flow. There is information flow from states A to B but there is not information flow among the states B and from B to A.*

where $\mathbf{x}_k$ is the robot pose vector at time k, $\mathbf{f}$ is the features map, $\mathbf{d}$ is the dense map, $\mathbf{Z}^k$ is the set of observations received until time k, $\mathbf{z}_k$ are the observations at time k, $\mathbf{U}^k$ are the control inputs and x_0 the initial conditions.

Once the features are extracted, the observations can be divided into two groups; the observations belonging to the feature map and the ones belonging to the dense map.

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{f}, \mathbf{d}) = P(\mathbf{z}_k^f|\mathbf{x}_k, \mathbf{f})P(\mathbf{z}_k^d|\mathbf{x}_k, \mathbf{d}) \tag{3.22}$$

Now assume that the dense map does not provide any information to the robot pose and the feature map, then:

$$P(\mathbf{x}_k, \mathbf{f}|\mathbf{Z}^k, \mathbf{d}, \mathbf{U}^k) = P(\mathbf{x}_k, \mathbf{f}|\mathbf{Z}_f^k, \mathbf{U}^k) \tag{3.23}$$

This concept is illustrated in Figure 3.13. There is information going in both directions between the robot and the feature map, and information going from the robot and features map to the dense map, but not information going directly from the dense map to the rest of the system. This is in accordance with what happens in practice during SLAM where geographically close objects present high correlation having high mutual information. If they share the same information, then only a selected part of the map needs to be used for the SLAM process (e.g. feature map). The rest of the information, can be maintained and used for other tasks. For example, it can be used to maximise (or accumulate) all the information gathered from the sensors, (e.g. to get an occupancy grid map) [63] and then be used for the path planning algorithm.

Applying Equation 3.22 and 3.23 to 3.21, the following result can be derived:

$$\begin{aligned} & P(\mathbf{x}_k, \mathbf{f}, \mathbf{d} | \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \\ \propto \quad & P(\mathbf{z}_k^f | \mathbf{x}_k, \mathbf{f}) P(\mathbf{x}_k, \mathbf{f} | \mathbf{Z}_f^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \\ & P(\mathbf{z}_k^d | \mathbf{x}_k, \mathbf{d}) P(\mathbf{d} | \mathbf{Z}^{k-1}) \end{aligned} \tag{3.24}$$

The first factor on the right term of Equation 3.24 (second line) is a Bayesian feature SLAM, and the second factor (third line) corresponds to the dense Bayesian map estimation. This equation demonstrates that by virtue of the *Unidirectional Information* property the DenseSLAM problem can be decoupled into two problems: the estimation of the feature-based SLAM and the estimation of the dense map. This decoupling effect makes possible the implementation of an efficient algorithm able to obtain dense representations.

In summary, the UIF allows the DenseSLAM problem to be decoupled into a feature-based SLAM problem and the estimation of a dense map. The UIF guarantees consistency. And the relative representation permits the computational cost of building a dense map to be reduced. Without the relative representation, the map could not be decorrelated and building a dense representation will be intractable. And without the UIF the system will become inconsistent since the dense maps are considered independent.

3.7.3 Delay dense map initialisation using the UIF

As explained in Section 3.3 the local regions are initialised once the base landmarks and the vehicle have high correlation. In this case, the local vehicle uncertainty will be small and consequently the uncertainty of the dense local maps. In general it may take a few frames to achieve high correlation between landmarks and vehicle and then all the information observed before the local regions are initialised is not included. In most situations, this will not be a problem since the vehicle will revisit all the areas and it will eventually incorporate all the environment information into the dense representation. However there could be some applications where sensor information cannot be discarded and then all the observations must be stored until it is possible to fuse them in the local regions. Storing all the observations with the respective vehicle pose in the state vector is a possible solution [45]. But even if this process is done for a few seconds, it could turn out to be too expensive since the computational cost is quadratic with the number of states. The UIF concept can be used to reduce the computational cost of this process and to accumulate all the observations until it can be fused in a local region. Instead of maintaining a full covariance matrix between the observations and the rest of the system, the UIF is used to maintain the correlations between the vehicle pose and the observations. By virtue of the UIF, the process of updating the stored observations will be simplified to the update of a diagonal matrix. The following paragraphs explain the use of the UIF to store the observations.

When the vehicle incorporates observations from an area that is not yet covered by any local region, it will add these observations to the state vector. The observations will be represented locally, relative to a potential LTR. This potential LTR will be a local region that meets the last two conditions from Section 3.5 but does not yet attain the high correlation condition. Assume here the states are represented in a local frame. Then the augmented state vector will be:

$$\hat{\mathbf{x}} = [\hat{\mathbf{x}}_v, \hat{\mathbf{x}}_f, \hat{\mathbf{x}}_v^j, \hat{\mathbf{x}}_d^j]^T \tag{3.25}$$

where $\hat{\mathbf{x}}_v$ represents the vehicle states, $\hat{\mathbf{x}}_f$ the features map, $\hat{\mathbf{x}}_v^j$ the vehicle pose expressed in the potential local region j and $\hat{\mathbf{x}}_d^j$ the observations waiting to be fused into the local region j.

$$\hat{\mathbf{x}}_d^j = [\hat{x}_{d_1}^j, \hat{y}_{d_1}^j, \ldots, \hat{x}_{d_i}^j, \hat{y}_{d_i}^j, \ldots, \hat{x}_{d_n}^j, \hat{y}_{d_n}^j]^T \tag{3.26}$$

The local vehicle pose $\hat{\mathbf{x}}_v^j$ is also maintained to calculate the uncertainty of the dense states in the local frame. The UIF is used to maintain the correlations between the observations and the robot pose. These correlations will enable the update of the local map when the vehicle uncertainty in the local frame is reduced. This process is illustrated in Figure 3.15. At time k the vehicle obtains two observations that are represented in the potential LTR that covers that region of the map. Later, at $k+n$ the vehicle local uncertainty is reduced (observe in the figure that the global uncertainty can still be the same), and the local coordinates and uncertainty of the local map are corrected due to the correlations.

While the vehicle is navigating in the same local area, more relative states are initialised with the observations obtained from the dense map, and the correlation with the vehicle pose is calculated. Once the local vehicle uncertainty is reduced, a new LTR is initialised, the observations stored in the state vector are fused into the local region, and eliminated from the state vector. The same process is repeated for the new local areas close to the vehicle pose. Figure 3.16 shows a flow diagram of the algorithm to model the local uncertainty. When an observation from the dense map is obtained, a new state is generated. The new states' covariance and cross-correlation with the vehicle pose are calculated. To calculate the uncertainty and correlation of these states, the vehicle uncertainty in the local frame has to be evaluated. Similar to the local maps, the vehicle pose can also be represented in a local frame. For the robot position, it is possible to use α and β in the same way as in local representation of an observation (See Appendix C). The heading can be represented in the local frame relative to one of the local axes. Then the relative vehicle pose will be:

$$\mathbf{x}_v^L = [\alpha\ \beta\ \gamma]^T \tag{3.27}$$

where

$$\gamma = \phi_v - \tan^{-1}\left(\frac{L_{2y} - L_{1y}}{L_{2x} - L_{1x}}\right) \tag{3.28}$$

α and β are the vehicle position in the local frame, γ is the vehicle heading in the local frame, ϕ_v is the vehicle heading in the global frame, and L_{ij} is the component j of the landmark i of the respective local frame.

Then, the local robot pose can be represented as a function of the landmark positions that define the local region and the global robot pose. The dense map states can be expressed as a function of the local vehicle pose and the observations.

$$\mathbf{x}_d = \mathbf{h}(\mathbf{x}_v^L, \mathbf{z}) \tag{3.29}$$

The dense map covariance and cross correlations with the vehicle pose can be evaluated as:

$$\begin{aligned}
\mathbf{P}_{dd} &= \nabla\mathbf{h}_{\mathbf{x}_v^L}\mathbf{P}_{vv}^L\nabla\mathbf{h}_{\mathbf{x}_v^L}^T + \nabla\mathbf{h}_{\mathbf{z}}\mathbf{R}\nabla\mathbf{h}_{\mathbf{z}} \\
\mathbf{P}_{vd} &= \mathbf{P}_{vv}^L\nabla\mathbf{h}_{\mathbf{x}_v^L}^T
\end{aligned} \tag{3.30}$$

where $\mathbf{P}_{dd}$ is the covariance matrix of the dense states. By using the UIF concept this matrix becomes diagonal and can be updated very efficiently. $\mathbf{P}_{vv}^L$ is the vehicle covariance in the local frame and $\mathbf{P}_{vd}^L$ is the cross correlation between the local robot pose and the dense states. After the new state is initialised, these two matrices and the update of the dense states' position are evaluated using the standard EKF-SLAM.

3.8 Computational Complexity of DenseSLAM

Even when a dense representation was sometimes necessary, it has not been previously possible to obtain such a representation with SLAM algorithms. One of the main problems being the computational burden involved in updating a covariance matrix where thousand of states are included. The computational complexity of DenseSLAM is the computational cost of the feature-based SLAM, which will be $O(n^2)$ in the worst case if none of the efficient algorithms are used, plus the computational complexity of building the dense local maps. Because the local maps are independent, building the local maps can be considered

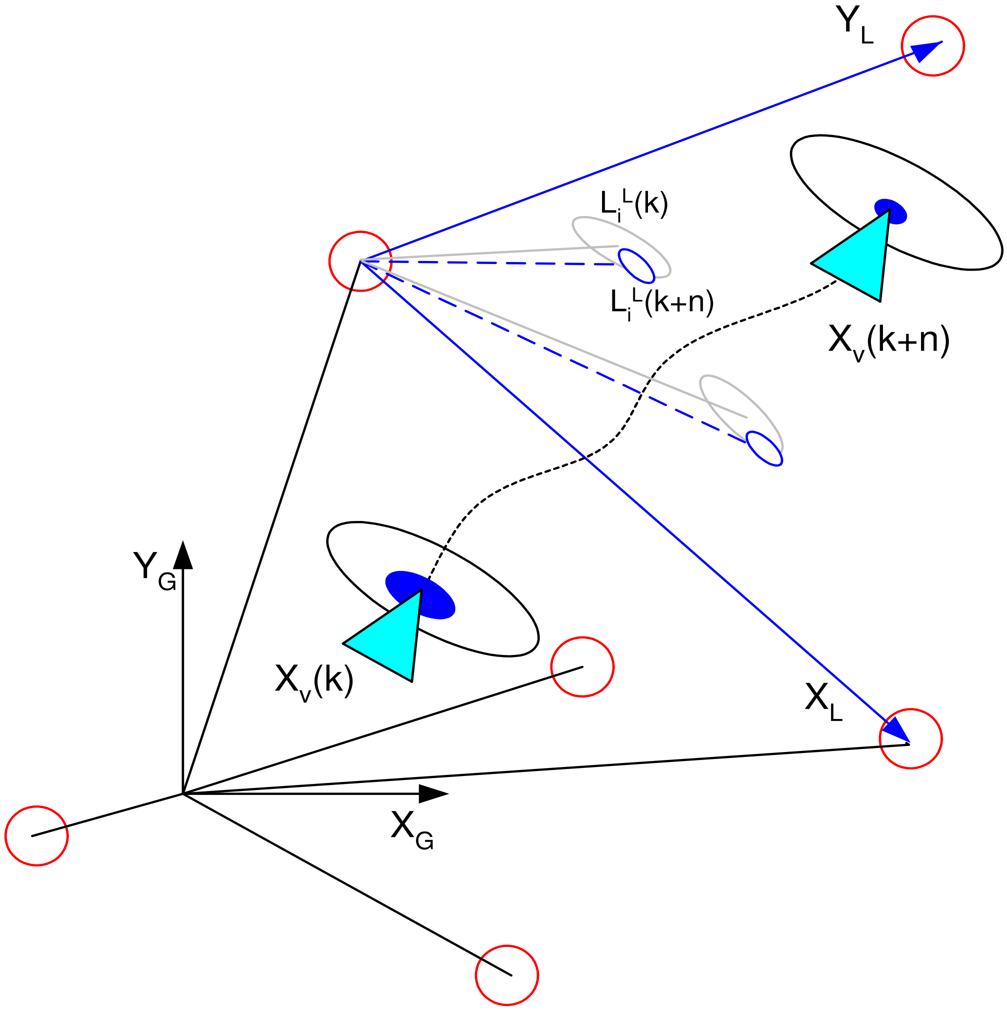


Figure 3.15: Local Error: Initially the vehicle has a large local uncertainty in the local and global frame. After some observations the uncertainty in the local frame is reduced. Maintaining the correlations between the dense map and the vehicle allows the dense map to be corrected.

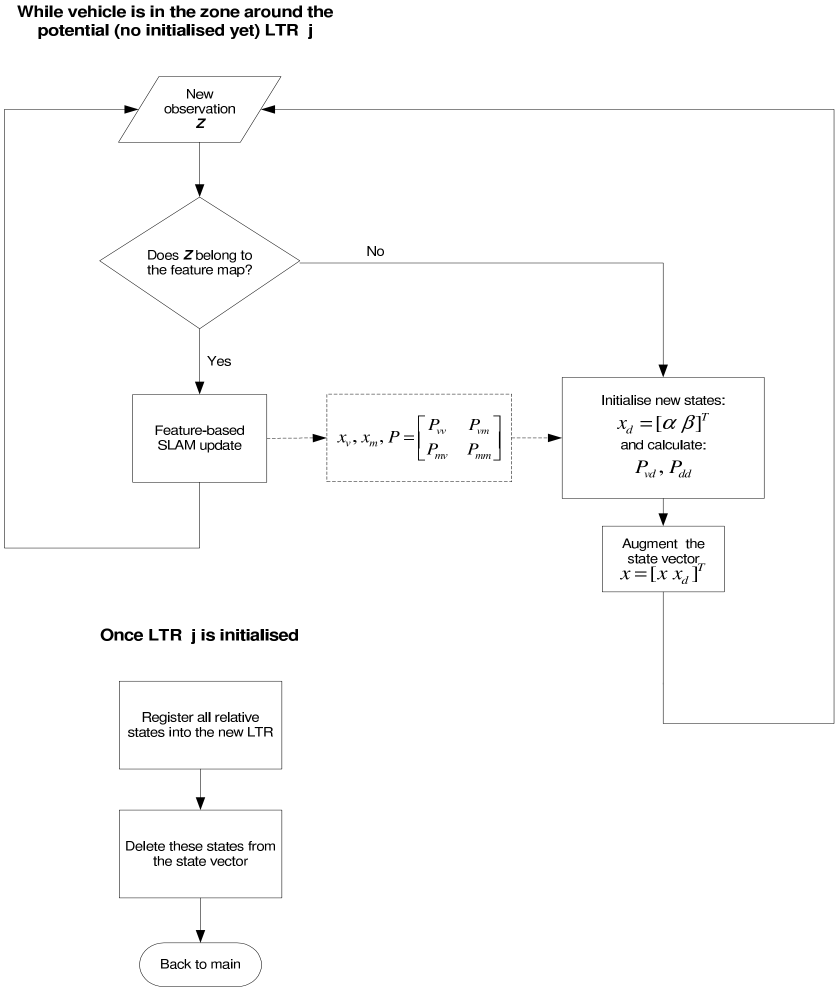

While vehicle is in the zone around the potential (no initialised yet) LTR j
New observation z
Does z belong to the feature map?
No
Yes
Feature-based SLAM update
$x_v, x_m, P = \begin{bmatrix} P_{vv} & P_{vm} \\ P_{mv} & P_{mm} \end{bmatrix}$
Initialise new states:
$x_d = [\alpha\ \beta]^T$
and calculate:
P_{vd}, P_{dd}
Augment the state vector
$x = [x\ x_d]^T$
Once LTR j is initialised
Register all relative states into the new LTR
Delete these states from the state vector
Back to main

Figure 3.16: Flow diagram of the algorithm to propagate the local error.

a constant time task $O(1)$, hence it does not add extra burden to the computation required by the feature-based SLAM algorithm.

In summary, the computational complexity of DenseSLAM is practically the same as the computational complexity required to maintain a feature-based SLAM algorithm.

3.9 Simulation

This section presents simulation results of DenseSLAM. The importance of the simulation results in this case lies in the possibility of comparing the actual objects' positions with the estimates by DenseSLAM. In the implementation presented here the observations are fused into the local regions without any processing. The dense representation is then formed by points that are the raw observations represented in cartesian coordinates.

Figure 3.17 shows the simulation environment. The simulation environment consists of an area of about 100 by 80 metres. The navigation map is formed by point feature landmarks and objects with different geometric shapes. The motion model used is the same as the one presented in Appendix A for the experimental vehicle. The control inputs are speed and steering angle corrupted with Gaussian noise with standard deviation of $0.5m/s$ and 3 degrees respectively. The vehicle travels at a constant speed of $3m/s$. The observation information consists of range and bearing with 0.15 metres and 1.5 degrees of standard deviation. The maximum range for the observations was set to 25 metres. Table 3.1 summarises the sensor settings for the experiment.

The enclosed contours in Figure 3.17 represent the actual objects' positions. The stars represent the actual landmark positions and the solid line the actual vehicle trajectory. The dark points represent the dense maps, that in this case are simply the raw observations taken by the sensor and fused into the local regions. The triangles represent the bounds of the local regions. Finally, the ellipses represent the 3σ uncertainty bound for each landmark, and the estimated landmark positions are denoted by circles.

The top figure in 3.17 shows the result before closing the loop. Since the vehicle starts to navigate at the origin and moves counterclockwise, the largest error can be observed at the leftmost side of the map. Figure 3.18 shows a zoom-in of the left-top part of the map. It it clear that the map contains a rotation error, and the whole map (landmarks and objects) seem to have the same error (i.e. high correlation). The bottom figure in 3.17 shows the result after closing the loop. At that point, DenseSLAM updates the whole map. Comparing the actual objects positions with the estimates, a large reduction in the map error can be observed.

Note: the implementation presented here does not include the delayed initialisation described in Section 3.7.3. This means that the dense observations are fused into the local regions just after the local regions are initialised.

Parameter	**Value**
Dead reckoning sampling time	0.025 secs
Steering sensor σ	3°
Velocity encoder σ	0.5 m/s
Observations sampling time	0.35 secs
Observations: max. range	25 metres
Observations: range σ	0.15 metres
Observations: bearing σ	1.5°

Table 3.1: Sensors parametres

3.10 Experimental Results

This section presents experimental results designed to validate the behaviour of the algorithm in an outdoor environment. The DenseSLAM implementation presented with simulation in the previous section was tested with real data, so the dense map is also represented here by the raw observations.

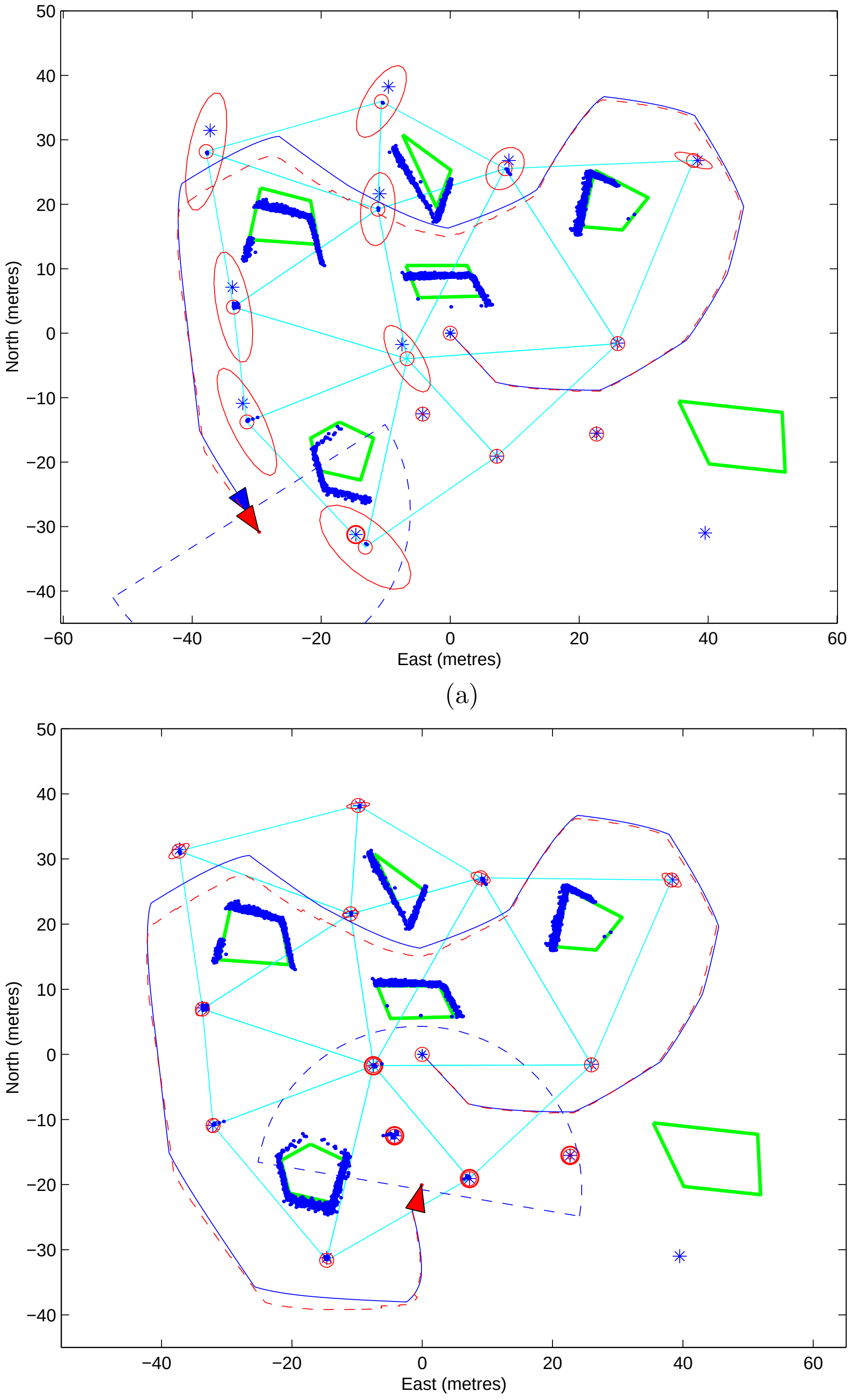
Figure 3.19 shows a satellite picture of the environment used for the experimental test. The environment is a large area of 120 by 200 metres and the run presented in this section is approximately 1 km long.

The vehicle and sensors presented in Appendix A were used to test the algorithm. An inertial unit and wheel encoder were used for the vehicle dead-reckoning process. A Sick laser was used to take external observations. Appendix A presents the typical noise variance of the sensors fitted in the vehicle.

As seen in Figure 3.19 the environment is composed mainly of buildings, trees and cars. The landmark map used by the algorithm is formed by poles and trees.

Figure 3.20 (a) presents the results obtained with the dead reckoning sensors. The figure shows the path obtained with the motion model and the GPS information. Due to the nature of the environment (buildings and trees), GPS was not always available. However there were still some sections of the path where the GPS was available with good quality and can be used as a reference. Figure 3.20 (b) shows the result obtained with the classic EKF-SLAM. The landmark mean and the 3σ ellipses covariance bounds are also shown. The vehicle starts to move at $(0, 0)$ coordinates. The vehicle initially completed a couple of loops in the car park near area the ACFR building and then it moved to the top part of the run. This is the part that presents the largest uncertainty. A zoom-in of the top part is shown in Figure 3.21.

In order to test the DenseSLAM algorithm, the GPS information was fused with the feature-based SLAM to obtain a laser image of the environment that is used as a reference to



(a)

(b)

Figure 3.17: Simulation environment. The solid line represents the actual vehicle trajectory, and the dashed line the estimated. The '*' are the actual landmarks positions and 'o' are the estimated landmarks positions. The enclosed contours are the actual objects positions and the dense points represent the dense map obtained with the raw observations. The ellipses are the 3σ contour of the landmarks covariance. (a) Shows the result before closing the loop and (b) after closing the loop and correcting the map.

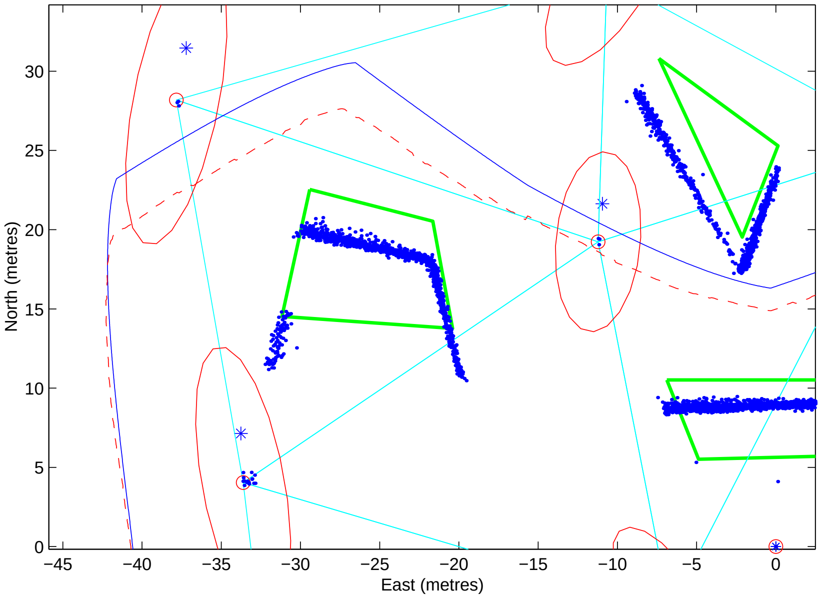


Figure 3.18: Zoom-in of the simulation result. The figure clearly shows the correlation in the errors of the objects and landmark map estimates. The figure shows the result before the first loop is closed.

compare with the estimates by DenseSLAM. Figure 3.22 shows the laser image obtained with the GPS information and the final map obtained with DenseSLAM. The figure also shows the landmarks positions. Due to the large number of buildings present in the environment, virtual landmarks (see Section 3.5.1) were needed to incorporate the walls into the map. This is why some extra landmarks can be observed in Figure 3.22 with respect to the feature-based SLAM result.

Figure 3.23 shows a zoom of the top part of the run. Figure 3.23 (a) shows the result obtained by DenseSLAM before closing the loop. The figure also shows the laser image used as a reference. The error in the estimated map before closing the loop can be easily observed. Figure 3.23 (b) shows the result after closing the first loop. Although there is still some residual error, it is clear how the estimated map has been corrected. Looking at Figure 3.20 (b) it can be seen that there is some remaining uncertainty in these landmarks even after the loop is closed. This is because the vehicle does not return to the top part of the run after closing the first loop. As a result, the error in that region is not reduced as much as in the bottom part of the run. Nevertheless, an important correction in all the regions has still been made.

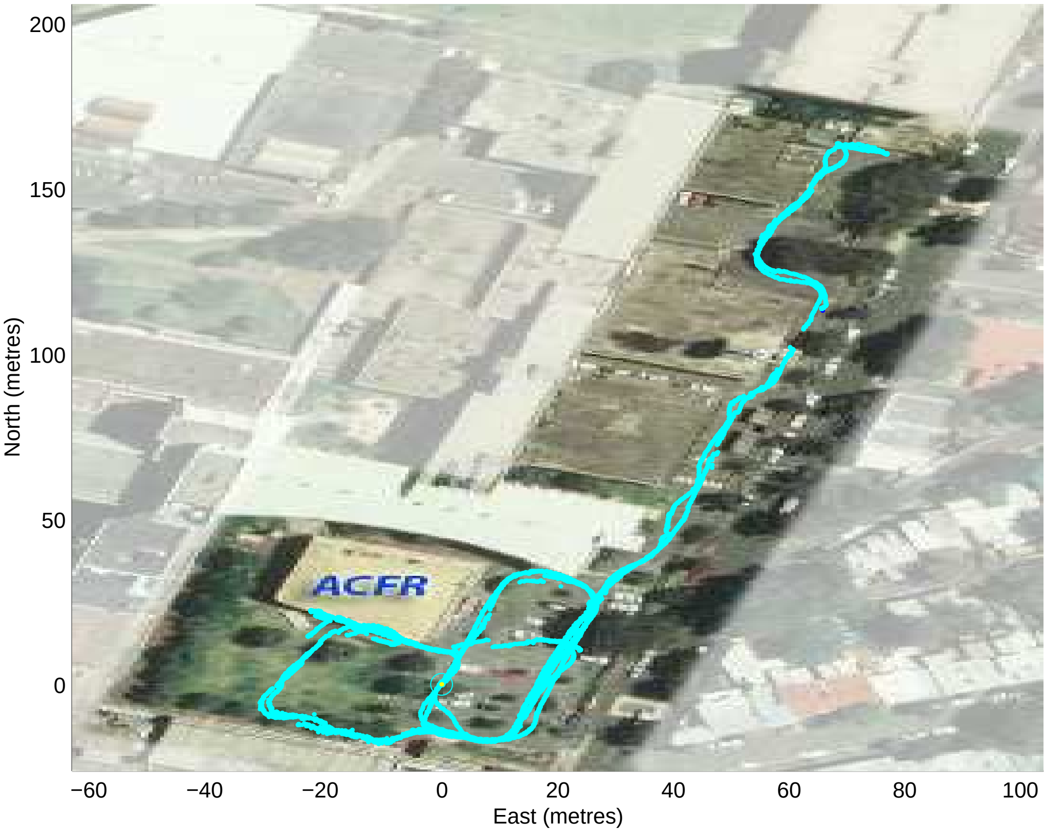
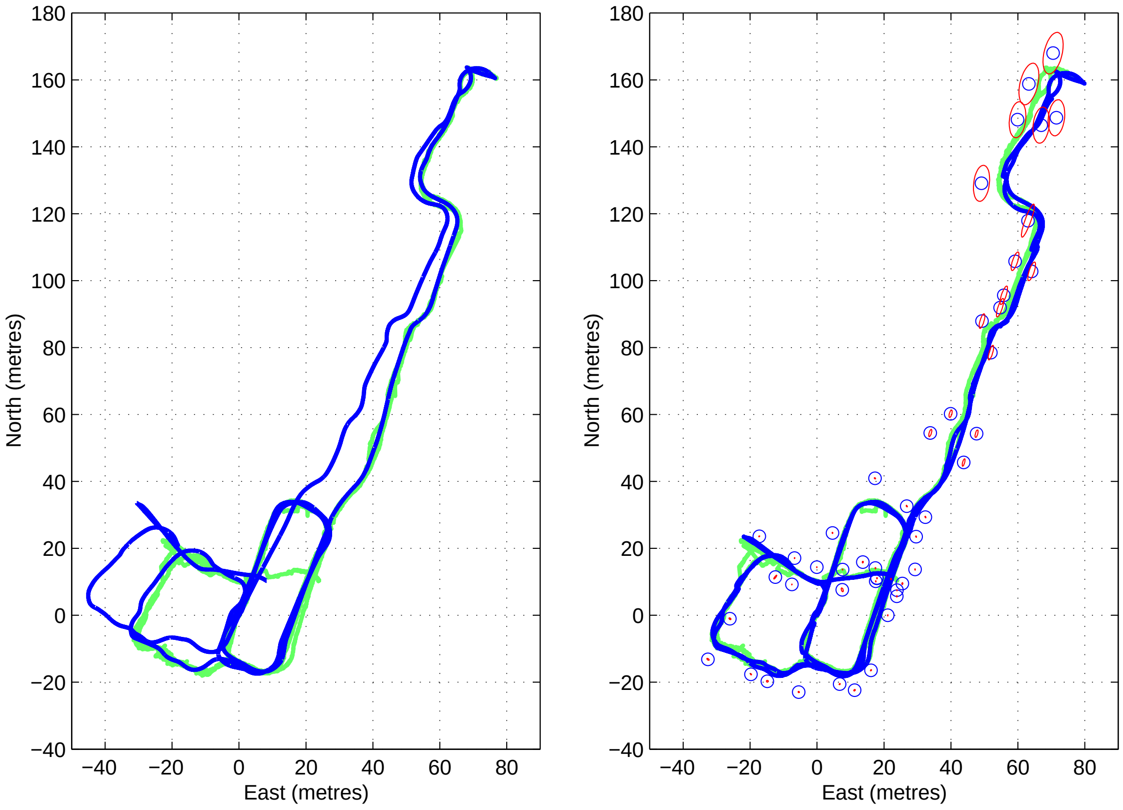


Figure 3.19: Satellite picture of the environment with the trajectory reported from the GPS.



(a) (b)

Figure 3.20: The dark line in (a) shows the path obtained with the dead-reckoning sensors and the position reported by the GPS is depicted by the light line. (b) shows the result obtained with the SLAM algorithm. The ellipses represent the 3σ bound for the landmarks uncertainty.

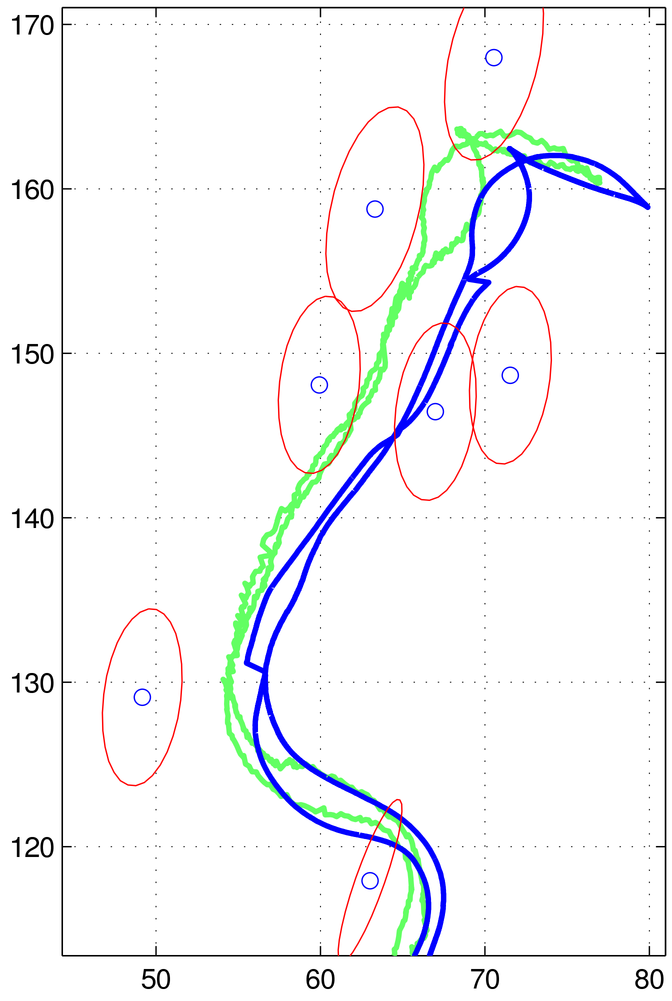


Figure 3.21: Zoom in of the top part of the environment. The figure presents the result obtained with the feature-based SLAM.

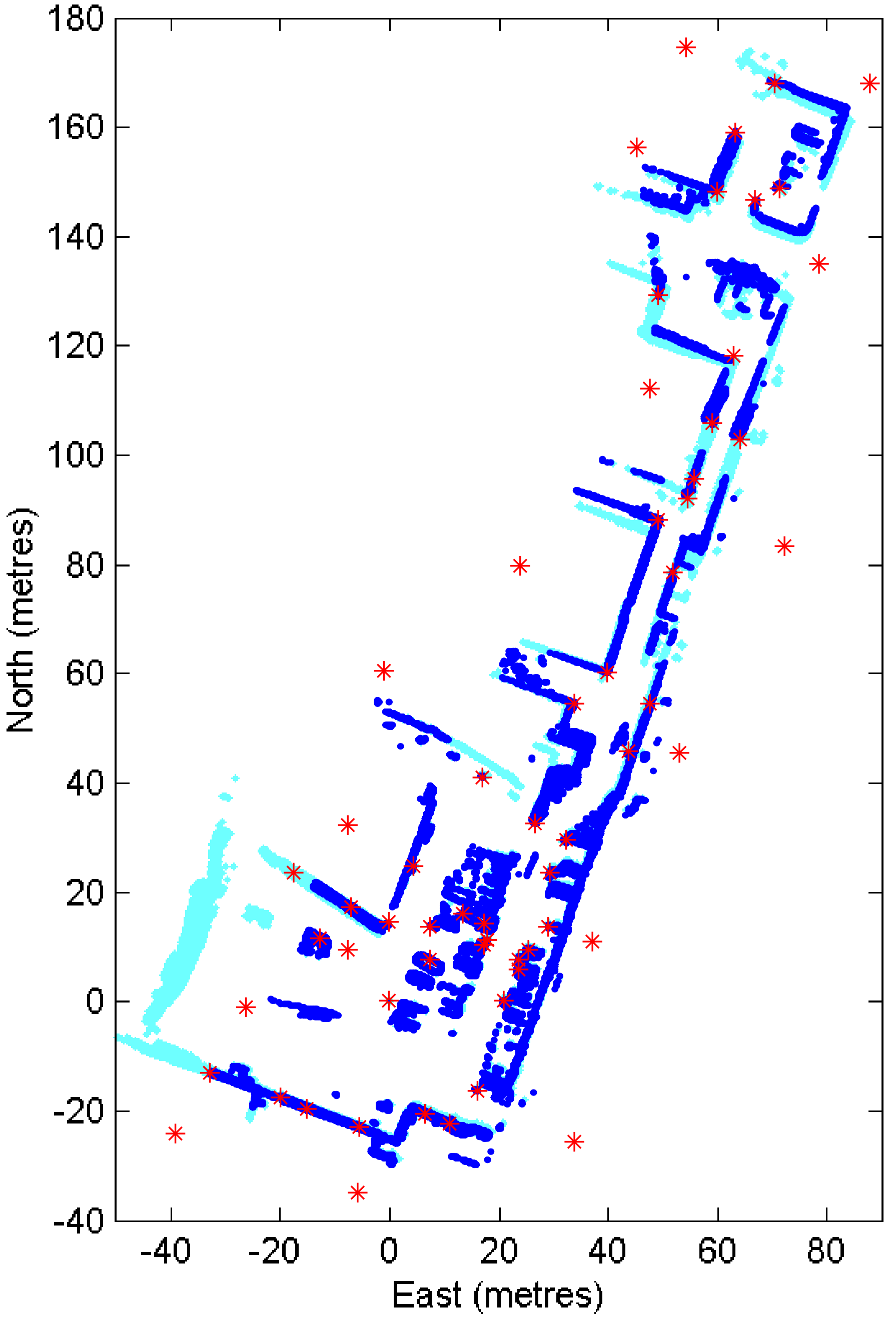
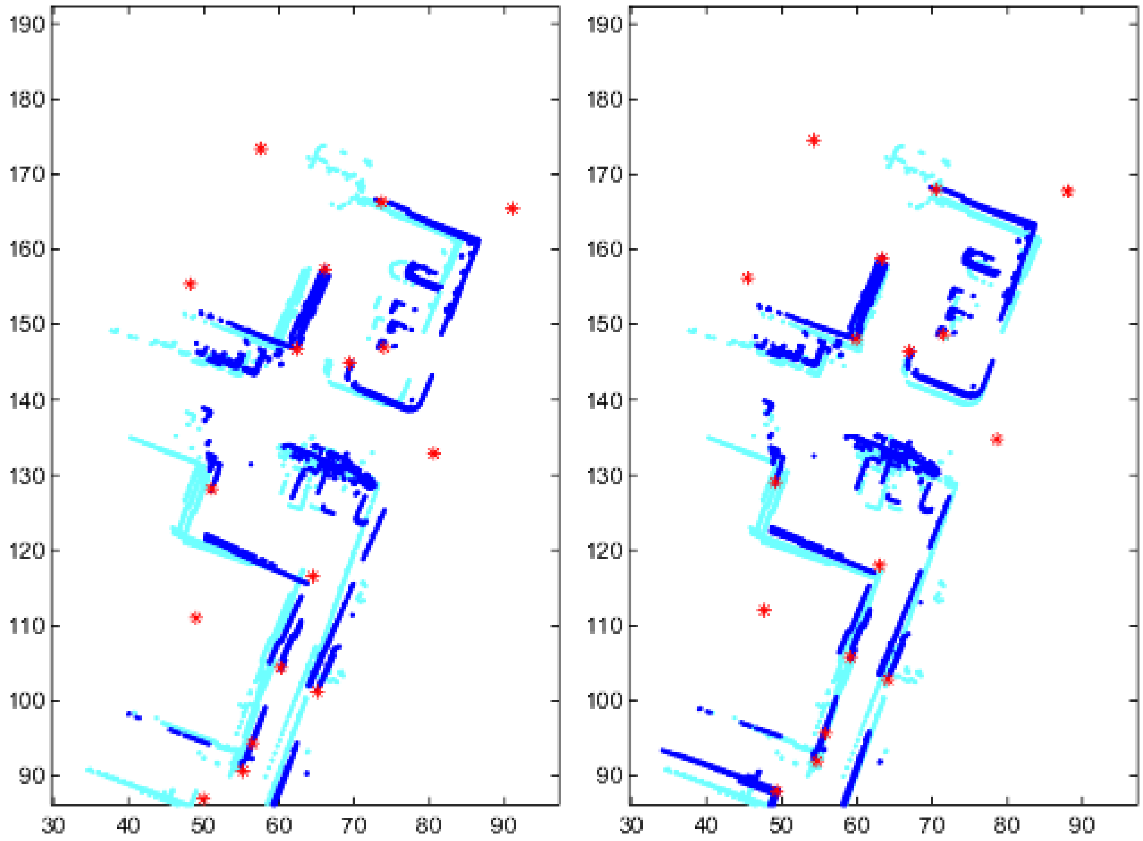


Figure 3.22: Final map obtained with DenseSLAM. The light points represent the laser image obtained using GPS and SLAM. The dark points depict the dense map estimated by DenseSLAM.



(a)

(b)

Figure 3.23: The lighter points represent the laser image obtained using GPS/SLAM. The darker points represent the final map obtained with DenseSLAM. Figure (a) shows the result before closing the loop, and (b) after the loop is closed.

3.11 Summary

Feature-based SLAM algorithms are able to obtain consistent environment representations, but the final map is built up of isolated landmarks. DenseSLAM has been defined as the process of simultaneous localisation and dense mapping. This chapter has presented the HYMMs framework which offers a solution for the DenseSLAM problem. The HYMMs use the landmark map to partition the global map into a set of local regions which provide a reference for a detailed multi-dimensional description of the environment. The new algorithm is able to obtain richer representations for SLAM (DenseSLAM) without losing the major advantages of feature-based SLAM and without increasing the computational cost. The HYMM is actually a natural extension to the classic feature-based SLAM that allows the fusion of more information into the representation.

The approach was tested with simulation and experimental results. Simulation results allowed the performance of the filter to be checked because the actual map could be compared with the estimated. The algorithm was also tested with real data in an outdoor environment and the final result was compared with the map obtained using GPS. The experimental results validated the performance of the algorithm in real environments.

In all the results presented, the dense map was obtained directly by fusing the raw observations into the local regions. The next chapter presents different representations that can be used inside the local regions in order to fuse the dense sensory information, showing the flexibility of the framework with respect to the representation used for the sensed information.

Chapter 4

Dense Maps Representation

One of the main strengths of DenseSLAM is the fact that it is not restricted to any particular local representation. The only condition required by the algorithm is to fuse the sensory information into a local representation which frames are determined by the position of a set of landmarks.

This chapter presents implementations of DenseSLAM using different representations for the dense local maps. Three representations with different characteristics were chosen in order to show the flexibility of the algorithm in representing the sensory information. Implementations of DenseSLAM using Occupancy Grids, particles and Sum of Gaussians are presented.

The chapter is organised as follows. Section 4.1 presents DenseSLAM using OGs representation for the dense information. Section 4.2 presents DenseSLAM using particles. Section 4.3 introduces DenseSLAM using Sum of Gaussians (SOGs). Finally Section 4.4 presents experimental results of DenseSLAM using OGs.

4.1 DenseSLAM using Occupancy Grids

The occupancy grid (OG) mapping technique [18] represents the environment with a grid of fixed resolution. The OG is a multidimensional grid that maintains stochastic estimates of the occupancy state of each cell. As mentioned before, one of the main problems with OG maps is the lack of a method to include the correlations between cells. The technique assumes that the state variables of each cell are independent, an assumption that yields inconsistent results. This section addresses the problem of obtaining consistent OG maps. It will be shown that by using DenseSLAM it is possible to obtain grid maps, even in cases where the robot pose uncertainty is large.

DenseSLAM uses the landmark map to define the boundaries of a local grid map (see

Figure 3.2). These local regions are then used as frames to fuse the dense information. In traditional OG maps, a grid of fixed resolution covering the region of interest is defined prior to starting the mapping process. DenseSLAM does not predefine the region of interest. The algorithm defines local OG maps at the time the robot explores new areas and incorporates more landmarks. In addition, different grid granularity can be assigned to each individual local OG map.

OG-DenseSLAM presents several advantages with respect to classic OGs. First, OG-DenseSLAM overcomes the main problem of the classic OG framework: (i) The local representation used by DenseSLAM allows the corrections to be propagated to the grid maps, which makes the final map consistent with the landmark map and the robot localisation process. In addition DenseSLAM naturally overcomes two other problems of traditional OGs: (ii) DenseSLAM does not limit the grid to a previously defined area , instead local regions are created when new areas are explored and (iii) the algorithm permits regions to be defined with different resolution, a property that allows maps of more resolution to be defined in areas of more interest. This last problem is also solved using a Quadtree representation [88] which adapts the grid resolution to the sensory information's density. A Quadtree representation also presents advantages in terms of search. In comparison with traditional grids, Quadtrees provides faster access to stored information. Accessing information stored in the grid maps is also very efficient using DenseSLAM since the algorithm divides the global map into smaller local maps. Then to access information in a particular cell, first the local region is localised using the base landmark positions. This will give the global reference of the local map and then a search in the local map is needed. Quadtrees might also be implemented in the local regions.

Since the main objective of this chapter is to solve the problem (i) explained above rather than to investigate efficient grid techniques, traditional uniform grids are implemented inside the local regions. The next subsections review the observation model and the update process used by the OG algorithm.

4.1.1 Observation Model

To obtain a sensor occupancy model, it is necessary to define a model for the sensor readings. If a stochastic model is used then $P(\mathbf{r}|\mathbf{z})$ has to be evaluated, where r represents the sensor readings and z the actual observations. Assuming the sensor possesses Gaussian noise.

$$P(\mathbf{r}|\mathbf{z}) = \frac{1}{\sqrt{(2\pi)\,|\mathbf{R}|}} \exp\left(-\frac{1}{2}(\mathbf{r}-\mathbf{z})^T\mathbf{R}^{-1}(\mathbf{r}-\mathbf{z})\right) \tag{4.1}$$

where $\mathbf{R}$ represents the sensor noise covariance matrix.

For a range and bearing sensor with independent Gaussian noise in the range and bearing readings, Equation 4.1 becomes:

$$P(\mathbf{r}|\mathbf{z}) = \frac{1}{\sqrt{(2\pi)}\sigma_r\sigma_\theta} \exp\left[-\frac{1}{2}\left(\frac{(r_r - z_r)^2}{\sigma_r} + \frac{(r_\theta - z_\theta)^2}{\sigma_\theta}\right)\right] \tag{4.2}$$

where $\mathbf{z} = [z_r \; z_\theta]^T$ and $\mathbf{R}$ is as shown in Equation 2.41.

A stochastic sensor occupancy model is defined as:

$$P(\mathbf{r}|s(C_i) = Occ) \tag{4.3}$$

where $s(C_i)$ represents the state of the cell C_i which has two possible values; occupied and free.

The sensor occupancy model is obtained from the sensor model (Equation 4.1) applying Kolmogorov's theorem [17]. Closed form solutions for Equation 4.3 can be derived for certain sensor models and numerical solutions are used in other cases. Different closed forms approximations for Equation 4.3 can be found in the literature [70, 65]. The next example presents occupancy models for the one and two-dimensional cases.

Example 4.1

Equation 4.4 illustrates an approximated closed form solution for the one-dimensional case [43].

$$P(\mathbf{r}|s(C_i) = Occ) = \begin{cases} (1 - p_{max}) + (p_{max} - 0.5)(1 - \lambda) & -1 \leq \lambda \leq 0 \\ 0.5 + (p_{min} - 0.5)\lambda & 0 < \lambda \leq 1 \wedge r \leq r_u \\ 0.5 & r > r_u \end{cases} \tag{4.4}$$

where $\lambda = (1 - 2\lambda_r)$ and λ_r is the normalised sensor model. p_{min} and p_{max} correspond to the maximum and minimum of the occupancy model. And r_u corresponds to the closest range where:

$$0.5 + (p_{min} - 0.5)\lambda = 0.5 \tag{4.5}$$

See Appendix A in [43] for more details about this closed-form solution.

Figure 4.1 (a) shows a plot of the closed form occupancy profile for a one-dimensional range sensor with Gaussian sensor noise. A measurement $r = 20$ was represented with a 0.1 grid resolution. The figure also shows with a dashed line the sensor model distribution which is a Gaussian model with a variance of 0.8.

Similar to the one-dimensional case, a closed-form solution of the occupancy model can be derived for a range and bearing sensor. Figure (b) illustrates an occupancy distribution profile in cartesian space for a range and bearing measurement. A measurement of 6 metres in range and 10 degrees in angle was represented.

4.1.2 Update

OG approaches use Bayes formulation to fuse the sensor information collected.

$$P(s(C_i) = Occ|\mathbf{r}^{k+1}) = \frac{P(r_{k+1}|s(C_i) = Occ)P(s(C_i) = Occ|\mathbf{r}^k)}{\sum_{s(C_i)} P(r_{k+1}|s(C_i) = Occ)P(s(C_i) = Occ|\mathbf{r}^k)} \tag{4.6}$$

where $\mathbf{r}^{k+1}$ represents the set of observation vectors received until time k and $P(s(C_i) = Occ|\mathbf{r}^k)$ is the previous estimate of the cell state. It is important to note that the vehicle pose is assumed to be known in OG approaches, so the left term in Equation 4.6 should actually be written as $P(s(C_i) = Occ|\mathbf{r}^{k+1}, \mathbf{x}_v)$, where $\mathbf{x}_v$ represents the vehicle pose. To avoid repetition in the notation, $\mathbf{x}_v$ is not written here.

In practice the update is often solved using Odds formulation [56, 77]. The odds of a variable x with probability $P(x)$ is defined as $\frac{P(x)}{1-P(x)}$. The logarithmic form is used because it is computationally advantageous. The update using log Odds formulations is expressed as (for the sake of brevity, $s(C_i) = Occ$ is noted as C_i):

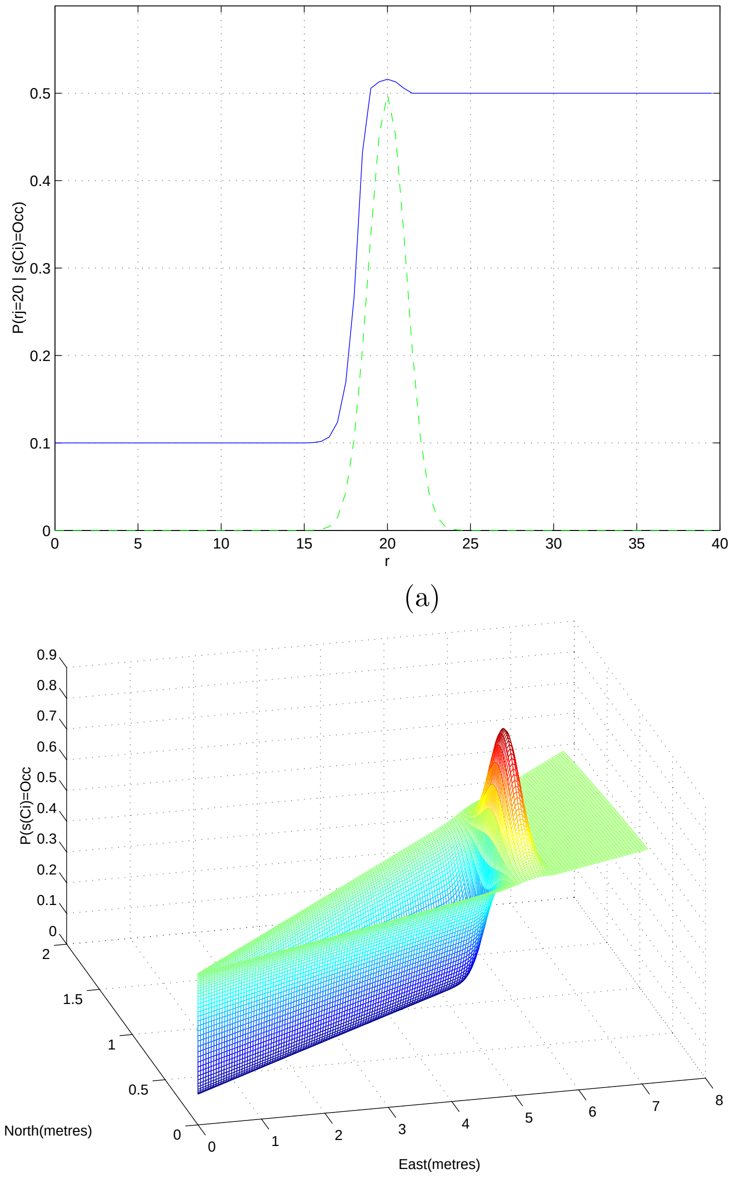
$$\log \frac{P(C_i|\mathbf{r}^{k+1})}{1 - P(C_i|\mathbf{r}^{k+1})} = \log \frac{P(C_i|\mathbf{r}_k)}{1 - P(C_i|\mathbf{r}_k)} + \log \frac{P(C_i|\mathbf{r}^k)}{1 - P(C_i|\mathbf{r}^k)} + \log \frac{1 - P(C_i)}{P(C_i)} \tag{4.7}$$

where $P(C_i)$ is the prior which is usually set to 0.5 and so the last term goes to zero. Note that Equation 4.7 does not need a normalisation factor as in Equation 4.6.

4.1.3 Simulation

This section presents simulation results of OG-DenseSLAM. Classic OG maps are also presented in order to compare them with the results obtained using DenseSLAM.

Figure 4.2 shows the simulation environment. The environment is similar to the one introduced in Section 3.9, it is formed by objects and point feature landmarks. The standard deviation for the noise added to the control inputs speed and steering were 0.5 m/s and 5 degrees respectively. These values are actually higher than the typical values in the UTE sensors, but they were set high to accumulate large errors and to highlight the effects of



(a)

(b)

Figure 4.1: Figure (a) represents an occupancy distribution for a range measurement $r = 20$. The sensor model is represented with the dashed line. (b) shows an occupancy distribution profile for a range and bearing measurement of 6 metres and 10 degrees respectively.

having large vehicle pose uncertainty. The vehicle travelled at $3m/s$. The sampling time for the control sensors was 0.025 secs. and for the observations 0.35 secs. The standard deviations for the observations were 0.10 metres for range and 1 degree for bearing. The maximum sensor range was set to 25 metres. Table 4.1 summarises the settings used for the simulation.

Figure 4.3 (a) shows the result obtained with standard OG techniques after the vehicle has closed the first loop. The dark points represent the cells C_i with $P(C_i = Occ) > 0.8$. The largest error is on the leftmost part of the run. It can be observed that the object located on the bottom left part is totally misshapen. This is because one part of the object is observed at the beginning of the run when the vehicle has very low uncertainty, and the other part (the left part of the object) at the end of the run, before closing the loop where the vehicle possesses large uncertainty. The inconsistencies are due to the fact that the vehicle uncertainty is not included in the OG map. The estimates for the three objects located on the centre-top part also present large errors which are clearly inconsistencies when compared with the estimated position of the landmark map around them, which present a very small error. Figure 4.5 presents a sequence of images from the result obtained with the standard OG approach for different times during the run. Figures (a)-(e) show the result after the first, second, third, fourth and fifth vehicle loop respectively. Figure (f) shows the result obtained after the tenth loop. In Figure 4.5 (a), the OG map is created when the vehicle possesses a large error, which is mainly when travelling over the left part. Then, the first loop is closed and the vehicle uncertainty is severely reduced, but the OG map does not include the correlations so the corrections are not propagated. After the first loop, the OG map is rebuilt with a small vehicle uncertainty and it can be observed after the second and third loop (Figure (b) and (c)) how the objects are incorporated in different positions. The observations are now taken from a vehicle with small uncertainty, so the same object is represented in two different places of the map. The map presents a 'shadow effect' since each object is observed several times with different vehicle error and the vehicle pose uncertainty is never incorporated into the observation model.

Figure 4.3 (b) presents the OG map obtained with DenseSLAM. The figure shows the result after the vehicle has closed the loop and corrected the position. The cells with $P(C_i = Occ) > 0.8$ are shown with dark points. The OG map is consistent with the landmark map as the objects' positions are corrected along with the vehicle pose after closing the loop. The figure also shows the LTRs used to partition the global map into a set of local regions. Figure 4.4 (b) displays a 3D plot of the final map obtained with OG-DenseSLAM. Figure 4.4 (a) shows the 3D grid map result obtained with the classic OG approach. There are three main differences that can be observed from the two figures:

- Map accuracy: In (b) the estimated map presents objects with similar shape to the actual objects, while in (a) the objects are misshapen. In (a) the wall widths of the objects is larger that in (b). In general, the map obtained by OG-DenseSLAM is much more accurate than the one obtained with classic OGs.

- Map coverage: In (a) a rectangular area (uniform 2D grid) where the vehicle is believed to explore was predefined. In (b) only the area actually explored by the robot is included in the map, there is no need to predefine the map area. The map coverage is automatically defined on-line.

- Grid resolution: The classic OG map is built using a uniform rectangular grid, while DenseSLAM builds many smaller grid maps. The uniform grid used by classic OG techniques forces the algorithm to have a uniform resolution in all the areas, unless quadtree maps are implemented. DenseSLAM obtains the global map joining many smaller maps and the algorithm presents the possibility of using different resolutions for different areas. Figure 4.6 shows the result using OG-DenseSLAM with different resolutions defined for the grids in different regions. A function running in the background was implemented which searches for regions that are 'almost empty'. In this case the function looks for regions where the percentage of cells with $P(Occ) < minProb$ is bigger than $cellsPercent$. For the experiment $cellsPercent = 0.99$ and $minProb = 0.2$. When a local region with these characteristics is found, it is considered an empty region and the grid resolution of the region is reduced. For the simulation, the cell resolution was reduced from 0.2 to 0.5 metres. The result illustrated in Figure 4.6 shows four regions with a lower resolution than the rest. It can be seen that these regions are almost empty, except for the corners where the landmarks which delimit the regions are located. Even when the example presented uses only two different grid map resolutions, many grid resolutions could be set for different regions and assigned according to some criteria. Furthermore, some regions can be built with different resolutions and some can be built using other representations such as Quadtrees.

Parameter	**Value**
Dead reckoning sampling time	0.025 secs
Steering sensor σ	5°
Velocity encoder σ	0.5 m/s
Observations sampling time	0.35 secs
Observations: max. range	25 metres
Observations: range σ	0.1 metres
Observations: bearing σ	1°

Table 4.1: Sensors parametres set for the simulation.

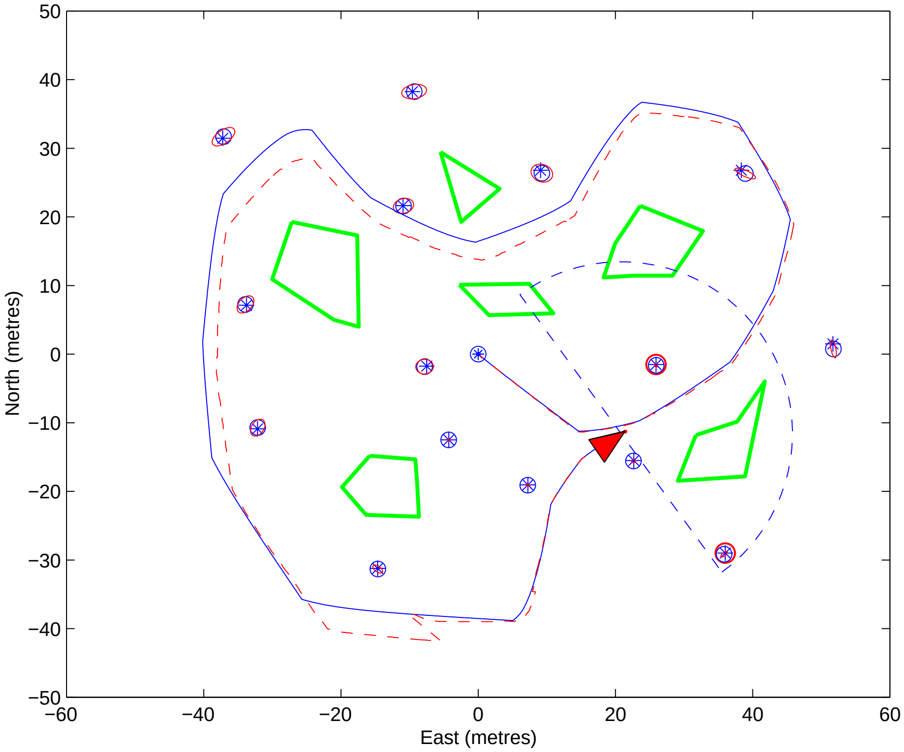
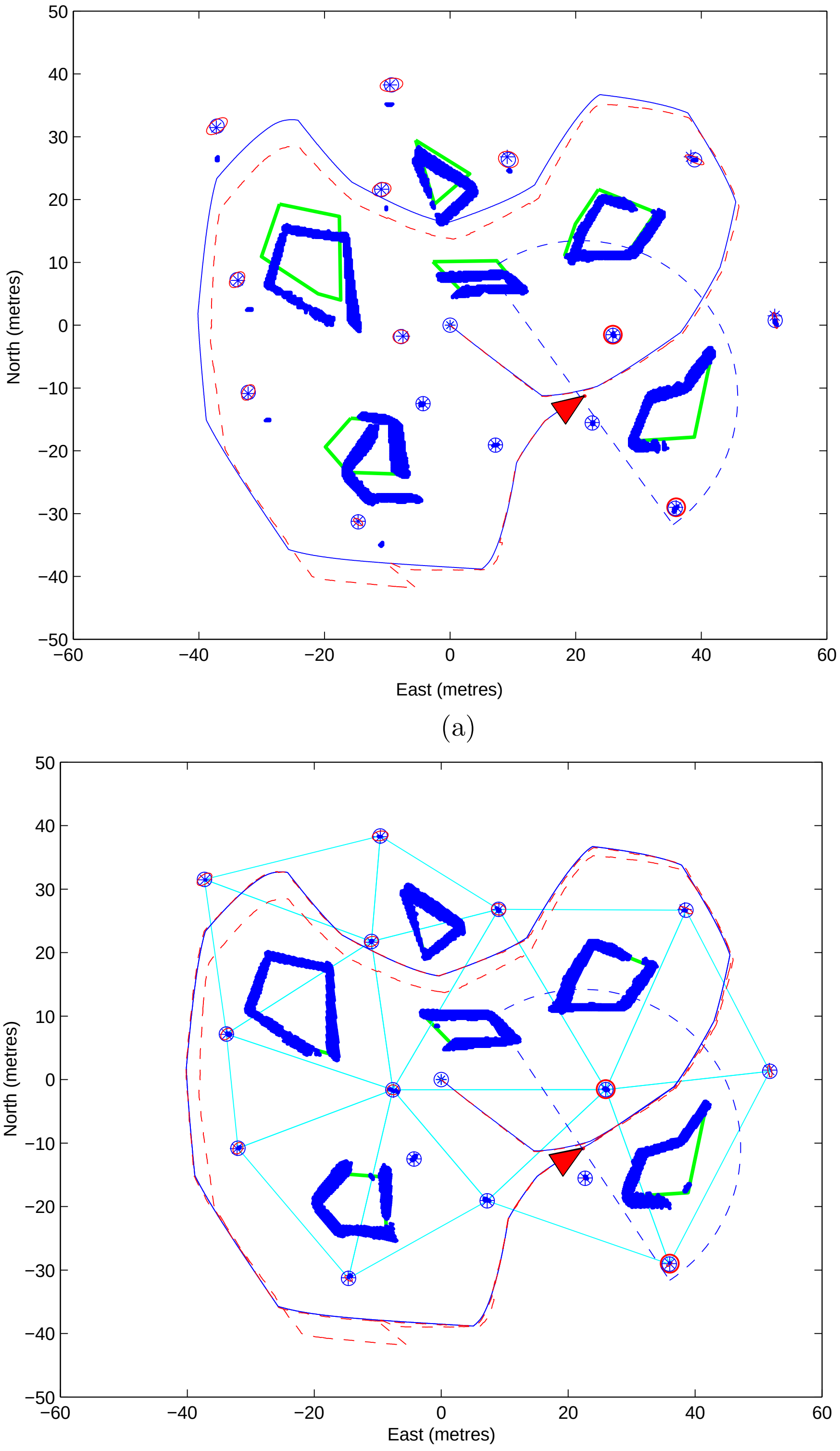


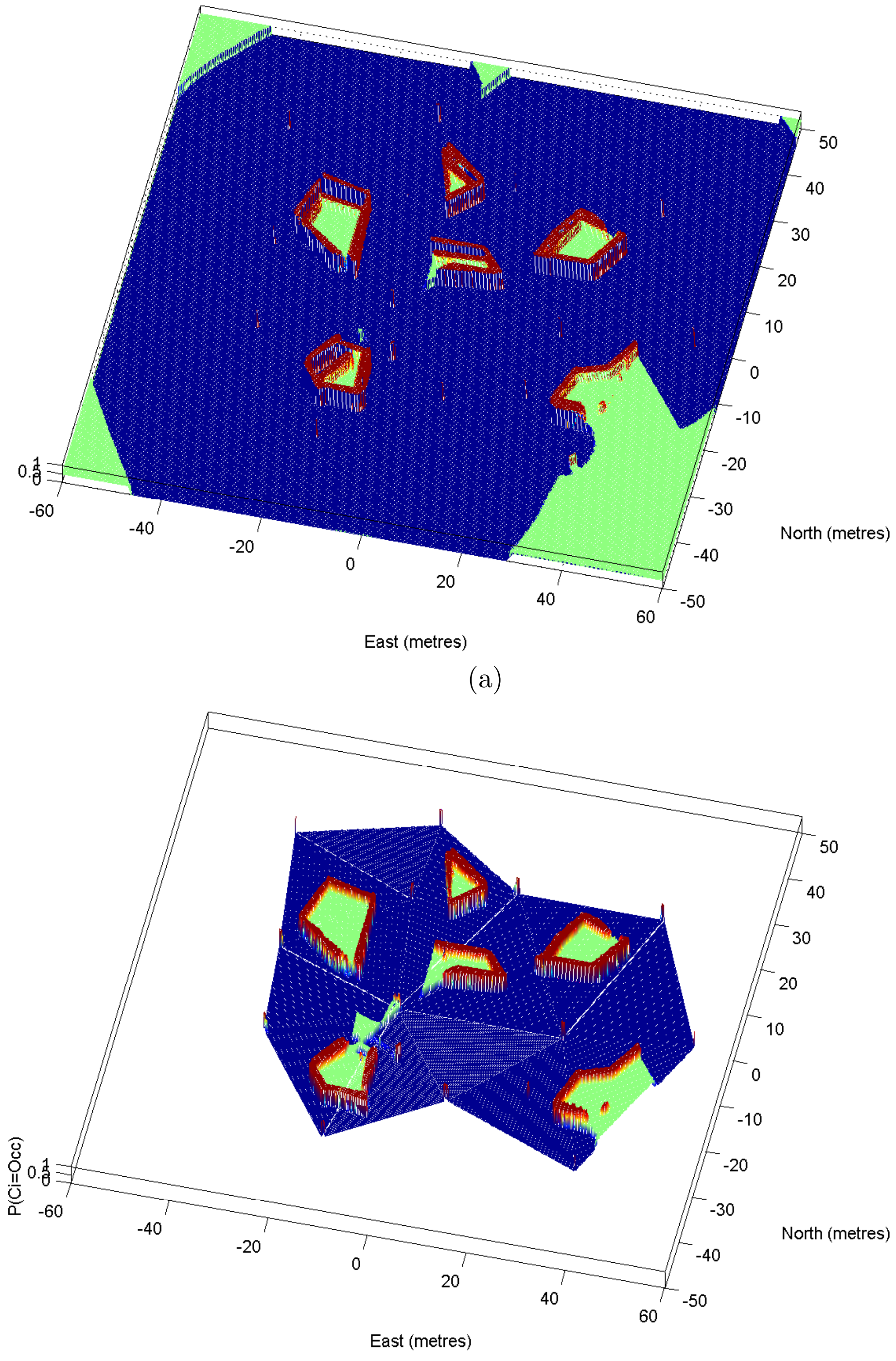
Figure 4.2: Simulation environment. The figure shows the result obtained with the feature based SLAM algorithm. The '*' are the actual landmarks positions and 'o' are the estimated landmarks positions. The closed contours are the actual objects' positions. The ellipses are the 3σ contour of the landmarks covariance. The solid line represents the actual vehicle pose, and the dashed line represents the vehicle pose estimated by the SLAM.



(a)

(b)

Figure 4.3: Figure (a) shows the result obtained using classic OG approaches to represent the objects. The figure shows the result after closing the first loop. The points represent the cells with $P(s(Ci) = Occ > 0.8)$. It can be seen that even when the vehicle has corrected its position after closing the loop, the OG map is not corrected due to the independent assumption. The result obtained using OG-DenseSLAM can be seen in (b). In here, the OG map is corrected along with the vehicle pose when the loop is closed.



(a)

(b)

Figure 4.4: Surface plot of the occupancy representation obtained (a) using OG-SLAM, and (b) using OG-DenseSLAM.

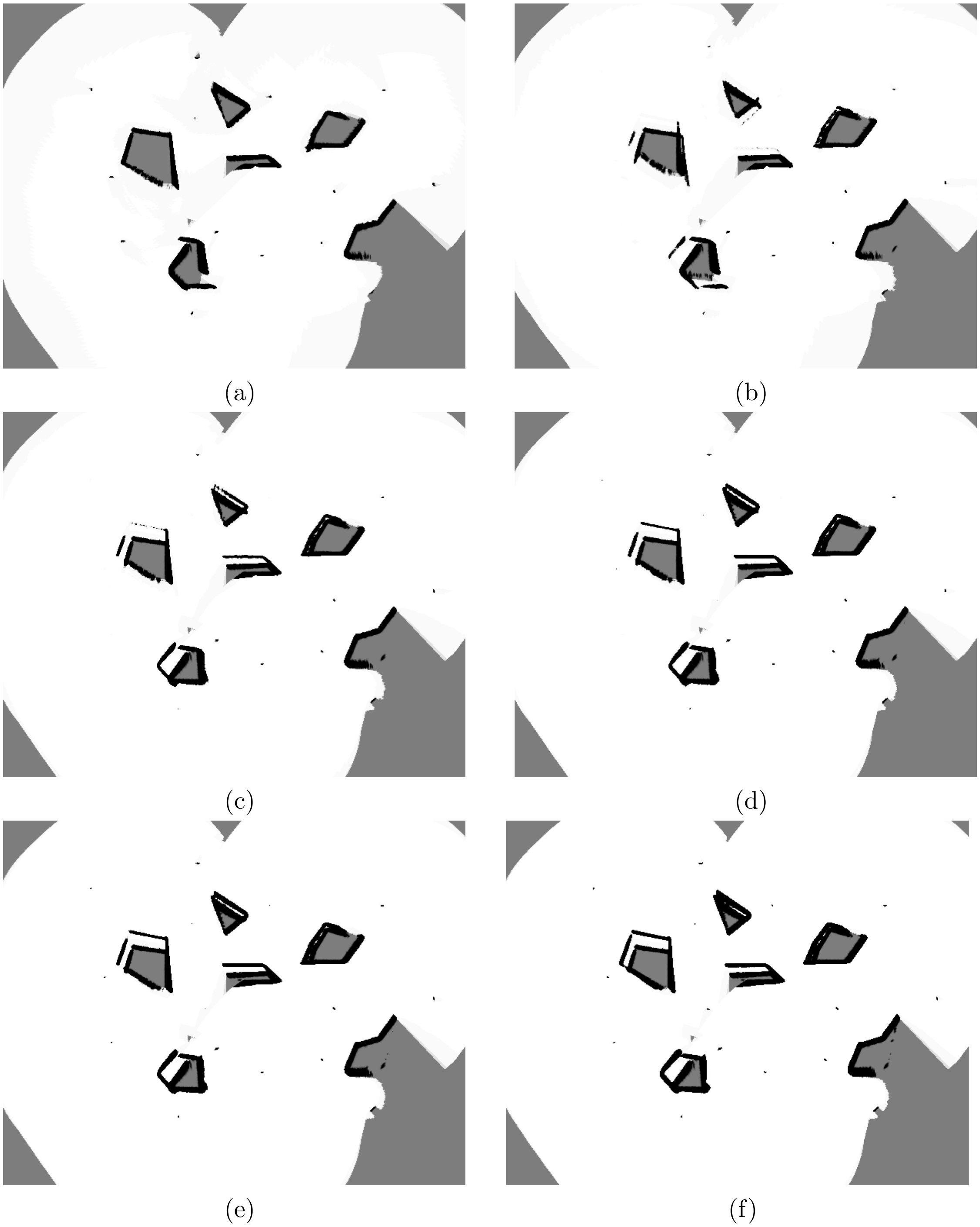

(a) (b) (c) (d) (e) (f)

Figure 4.5: Sequence of images of the result obtained with OG-SLAM for different times during the run. Figures (a)-(e) show the result after the first, second, third, fourth and fifth vehicle loop respectively. (f) shows the result obtained after the tenth loop. As a consequence of the independent assumptions the map is never corrected and is misshapen due to the inconsistency with the vehicle pose uncertainty.

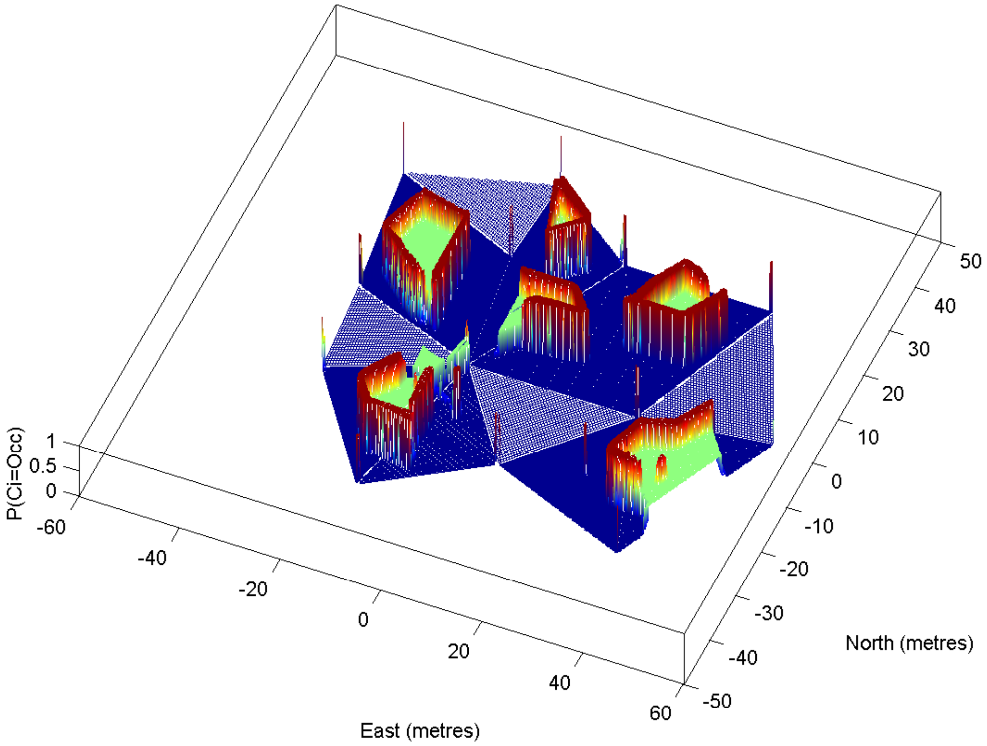


Figure 4.6: The figure shows a surface representation of the result obtained with OG-DenseSLAM using different cell resolutions. In the implementation, two different cell resolutions were used: 0.2 metres by default for all the regions, and then for the regions in which most of the cells were empty the resolution was reduced to 0.5. For the example it can be seen that four regions have lower resolution than the rest (lighter regions).

4.2 DenseSLAM using Particles

Particle filters have become very popular in the last few years [15]. In the robotics community, particle filters have been used to solve localisation problems [13, 36]. Hybrid approaches that use particle filters have also appeared for mapping problems. An algorithm that uses particle filters to represent the vehicle pose distribution and Kalman filters to represent the map probabilities was introduced in [54]. In [20], particle filters are used to estimate the vehicle pose and OG techniques are used to represent the environment. In all these approaches, the particles are used to represent the vehicle pose (usually a low dimensional space), while the map is maintained using more efficient representations.

An approach which uses samples to represent the environment was presented in [43]. The algorithm called *Sample Environment Map* (SEM) maintains spatial probabilistic information over discrete environment locations (samples). The approach incrementally constructs an environment representation on discrete points based on the sensor model. An occupancy representation of the environment using SEM is presented, where the particle's weight represents the occupancy probability of the particle's location. It is argued that the advantage of performing such a sample-based inference on the target distribution (occupancy for example), lies mostly in the compactness and adaptability of the sample set. Samples can be simply spawned into specific regions of the environment, thus obtaining more detail in the environment distribution. The algorithm spawned the particles according to the sensor uncertainty. However, the issue of sensor pose uncertainty is not approached by the author, and the results presented are just the representation of sensor frames obtained from a stationary sensor. It is mentioned that including the vehicle uncertainty in the representation can be done by inflating the noise of the sensors, however this is only partially correct. In the case of a mobile vehicle, the vehicle pose uncertainty will be propagated to the map, an effect that could be approximated by inflating the sensor noise. But the other consequence of having vehicle pose uncertainty is the correlation in the map. Thus the consequences of vehicle uncertainty are not just a *blurring effect* as mentioned in [43], the correlations in the map representation have to be maintained.

This section presents particle-DenseSLAM. By representing the environment samples in the local regions, the algorithm permits the vehicle corrections to be propagated to the samples making the map representation consistent with the estimates by SLAM.

4.2.1 Observation Model

The observation model will vary according to the property to be represented. This section presents an observation model for occupancy in order to compare it with the representation obtained with grid maps in Section 4.1.

The same Gaussian range and bearing sensor model as the one presented in Equation 4.2 is used here. Since the environment property to be modelled is occupancy, the closed form solution for occupancy presented in Example 4.1 will be used in order to obtain samples representing the occupancy distribution. Equation 4.3 can be rewritten to give a sensor occupancy model for samples:

$$P(\mathbf{r}|s(\mathbf{x}_i) = Occ) \tag{4.8}$$

where $\mathbf{x}_i$ denotes the position of an n-dimensional particle. Since a range and bearing sensor model is used, the samples represent a position in the plane, $\mathbf{x}_i = [x_i, y_i]^T$. By evaluating the samples $\mathbf{x}_i$ in the closed-form for the occupancy distribution, weights $\mathbf{w}_i$ for each sample can be obtained. The weights will represent the probability of the sample location being occupied:

$$w_i = P(\mathbf{x}_i|s(\mathbf{x}_i) = Occ) \tag{4.9}$$

Different sampling methods can be implemented which will yield different results. For example, the occupancy distribution could be sampled using a uniform sampling rule, in which case the result will be equivalent to the one obtained using grid approaches. The sampling method implemented in this thesis is the same as in [43]. The samples are obtained from the sensor model and the weights are then evaluated by passing each sample through the occupancy model. Figure 4.7 shows an example of the samples obtained using a Gaussian sensor model. The occupancy and sensor distribution are also shown in the figure. The red dots denote the samples obtained from the sensor model and the result when these samples are propagated through the occupancy model. The solid blue line represents the occupancy model and the dashed green line the sensor model. It can be seen from the figure that most of the particles are in the area where the sensor model has the highest probability. The occupancy distribution can be divided basically into three main regions: the free space; from the sensor position to the observation, the observation sensor model at the observation position and the unknown space behind the observation. The problem of sampling from the sensor model is that most of the samples will be concentrated near the peak of the sensor distribution, leaving the free space and unknown regions of the occupancy model without samples. Figure 4.8 shows an example of the sampled occupancy distribution obtained after sampling the observation model of a range and bearing sensor with 1000 samples. The same effect seen in the one dimensional example is observed here, most of the particles are in the area with the highest probability of occupancy and the free space and unknown area are not represented by the samples.

In summary, sampling from a uniform distribution will give the same result as grid

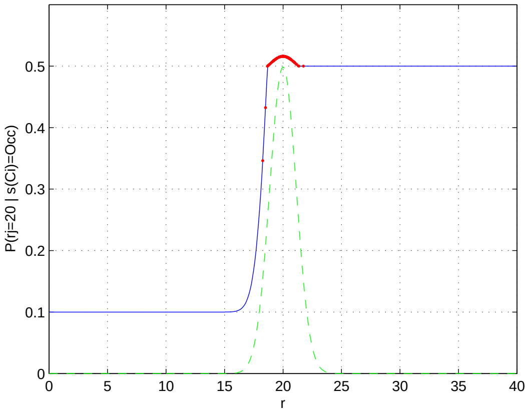


Figure 4.7: The solid line represents the occupancy distribution for a range sensor measurement, the dashed line the sensor model, and the dots the samples from the sensor model propagated through the occupancy distribution. The Gaussian sensor model was sample using 1000 particles. It can be seen that most of the samples are in the region with high probability of being occupied and there are almost no samples in the free space and unknown regions.

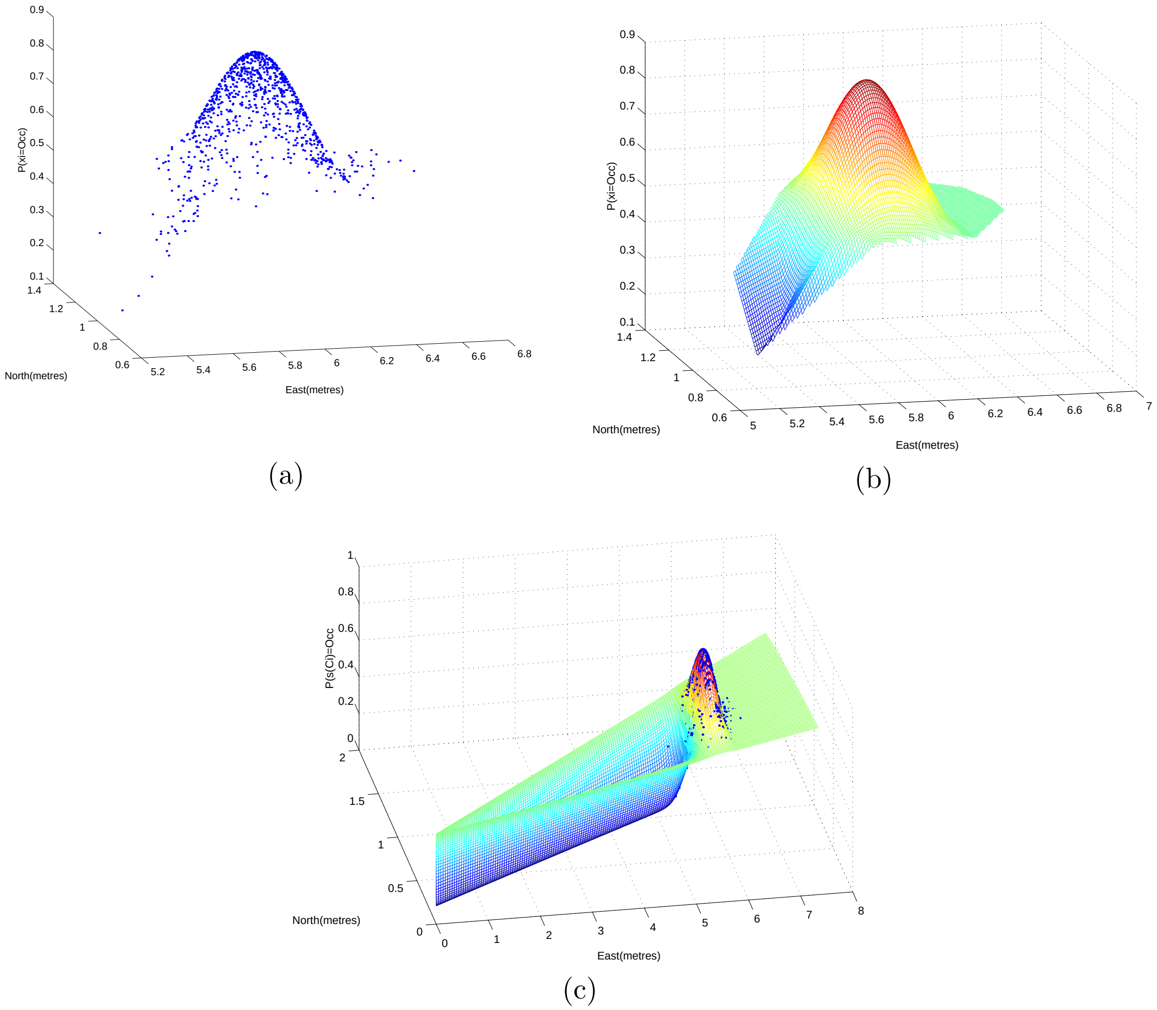
methods and the advantage of using particles that can be spawned anywhere is lost. On the other hand, sampling from the sensor model will include only the regions with high probability of being occupied and it will miss information regarding the free space. This suggests that a hybrid approach between grids and particles might be the ideal. Grids could be used to cover big regions as they are usually the free spaces, and particles could be used to obtain details about the occupied regions.

4.2.2 Update

Similar to grid approaches, Bayes rule can also be used to update the particle weights. A local map is built with the new observations and is then composed with the global map using Equation 4.6.

The local map in particle-DenseSLAM is obtained by sampling from the sensor model and evaluating the samples in the occupancy distribution. Once the local grid is done, the particles in the local map have to be associated with the corresponding particles in the global map. Unlike cells, particles do not have volume, and no particles will ever be spawned in exactly the same position. In practice, this is solved by considering particles which are closer than a minimum distance to be samples from the same state and then applying Bayes rules to update their state as in OG. Assume the following situation:

$$distance(\mathbf{x}_i, \mathbf{x}_j) < d_{min} \tag{4.10}$$



(a)

(b)

(c)

Figure 4.8: Samples obtained from the sensor occupancy distribution. (a) samples, (b) surface reconstructed interpolating the samples and (c) occupancy probability distribution obtained with a grid with the samples superimposed.

where $\mathbf{x}_i$ is a sample from the sensor local map and $\mathbf{x}_j$ is a sample from the a priori map. The occupancy state of the j-th particle is updated using the i-th particle in the following form:

$$P(s(\mathbf{x}_j) = Occ|\mathbf{r}^{k+1}) = \frac{P(r_{k+1}|s(\mathbf{x}_i) = Occ)P(s(\mathbf{x}_j) = Occ|\mathbf{r}^k)}{\sum_{s(\mathbf{x}_i)} P(r_{k+1}|s(\mathbf{x}_i) = Occ)P(s(\mathbf{x}_i) = Occ|\mathbf{r}^k)} \tag{4.11}$$

Figure 4.9 shows an example of the nearest neighbour algorithm for a $d_{min} = 0.2$. The green points denote the particles from the a priori map and the blue points represent the particles obtained with the new observations. The red lines illustrate the associations. It is seen that more than one sample from the a priori map can be associated with the same new sample, which means that one sample from the observations will update more than one sample from the a priori map.

When a sample is not associated with any sample of the a priori map, the sample is updated with an a priori probability of 0.5. Equation 4.11 becomes:

$$P(s(\mathbf{x}_j) = Occ|\mathbf{r}^{k+1}) = P(r_{k+1}|s(\mathbf{x}_j)) \tag{4.12}$$

One disadvantage of sampling from the sensor model is that the particles are concentrated only around the occupied regions. If the vehicle is mapping an environment where there are dynamic objects, once an object is observed and its corresponding region is defined as occupied, this area will remain occupied even if the object is later removed. Consider an object that is moved from its original position after it has been included it the map. If the vehicle only spawns particles in the regions with high probability of occupancy, even when the vehicle revisits the place where the object was and this place is now empty, it will never update the occupancy state of the particles in that region, because now these particles are under the free space region of the occupancy distribution and as shown before, the observation model does not sample this region. However, even when the particles only represent the regions with high probability of occupancy, the free space information obtained with the new observations could be used to update the a priori map, setting the weights of corresponding particles as free space. The first step is to obtain from the a priori map, the particles that are under the free space region of the occupancy model for the new observations. Then new particle weights are calculated using the occupancy model. These new weights are now used to update the weights of the old particles using Equation 4.11. This simple extra step in the update stage ensures that a region that changes from being occupied to free is not wrongly preserved as occupied.

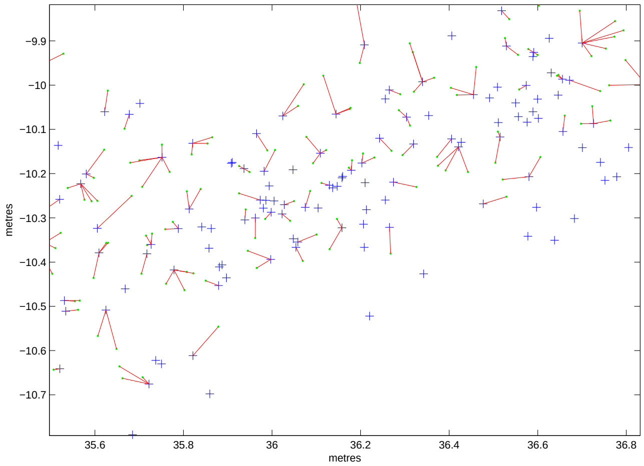


Figure 4.9: Result of the nearest neighbour algorithm for a $d_{min} = 0.2$. The green points denote the particles from the a priori map, the blue '+' represent the particles obtained with the new observations. The red lines illustrate the associations.

4.2.3 Simulation

This section presents simulation results using particle-DenseSLAM. The environment presented in Section 4.1.3 is also used here to demonstrate the algorithm.

Figure 4.10 shows the result obtained with particle-DenseSLAM. The blue dots in (a) denote the particles' position and the red lines the actual objects' positions. For the nearest neighbour algorithm, d_{min} (see Section 4.2.2) was set to 0.2. The result obtained is very similar to the one presented in Section 3.9 which fuses just the raw observations. The similarity in the result obtained from the two implementations was expected since particle-DenseSLAM spread most of the particles near the peak of the sensor model (observation position). In the particle-DenseSLAM implementation presented here, in addition to the weights, the position of the samples is also updated with the mean position between the position of the sample from the a priori map and the sample from the new observation. Because the resolution was set to 0.2, not much difference was seen between the implementation without updating the particles' positions and the one presented here. A higher value of d_{min} may make the difference more noticeable.

Figure 4.10 (b) shows the occupancy plot. It can be observed that most of the particles possess high probability of occupancy. The whole map has around 50,000 particles. The number of samples used to represent the map can be reduced using subsampling techniques, but in the results presented here the map is not subsampled. Only 2% of the samples have weights below 0.5, 11% have weights between 0.5 and 0.8, and the other 87% possess weights above 0.8. These numbers show that the largest concentration of particles is near occupied regions.

4.3 DenseSLAM using mixture of Gaussians

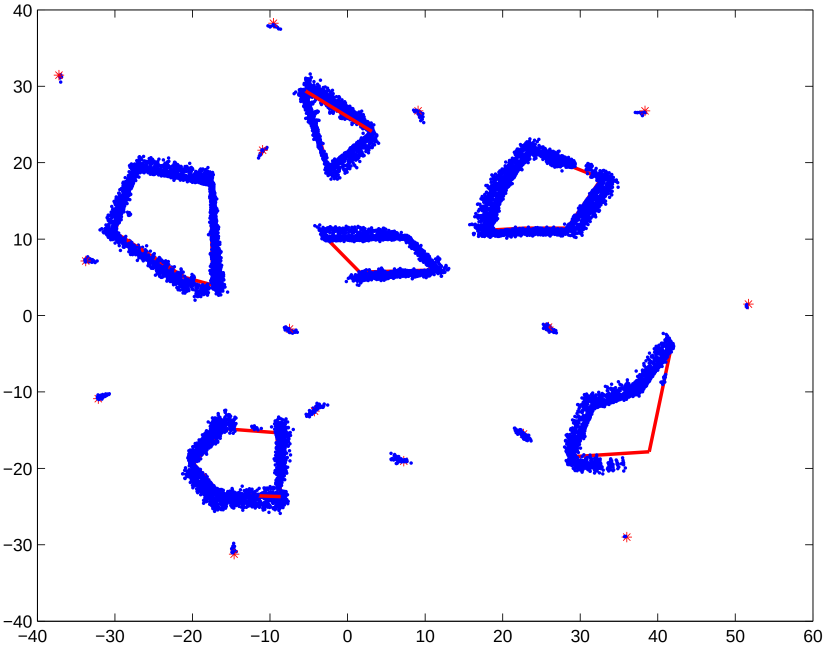
A Bayesian based SLAM algorithm using Sum of Gaussians (SOG) to represent the sensor model was presented in [49]. A SOG distribution is used to build a feature map describing the environment. The algorithm is applied in a subsea environment using sonar sensors. Each sonar return is modelled as a SOG in range and a Gaussian in bearing. The likelihood functions, and the motion model are defined as a SOG. The algorithm first computes the time update (convolution) for each Gaussian using the motion model and the prior distributions, and the resulting SOG is then multiplied by the SOG observation distribution to yield the posterior SOG. The convolution and multiplication steps are equivalent to the prediction and update step of the EKF but applied now to a SOG. After each multiplication the number of Gaussians will multiply as well so the resulting posterior must in general be resampled to produce a lower complexity mixture. The algorithm is applied to map building through multiplication of scans, localisation using scan correlation, and SLAM. However, as explained in [1] the use of SOG for recursive estimation as presented in [49] is not correct. The SOG used to approximate the sonar returns is valid, but its fusion via multiplication with other observations is incorrect. Basically each sensor return does not observe the same point target but different points of a no-point surface, thus the recursive estimation via multiplication will give inconsistent results (a clear explanation can be found in Chapter 4 of [1]).

This section presents a SOG-DenseSLAM algorithm. A SOG distribution is used to represent the sensory information. This SOG is then fused into the map using local representation in the same way as the particles or the OGs shown in previous sections. It is important to clarify that the final SOG map does not represent a distribution. Furthermore, for the reasons explained before, the resulting map cannot be used for recursive mapping. However the 2D PDF representation can be used for example to assist the data association process via scan correlation or for localisation after a final map has been obtained with SOG-DenseSLAM. The next subsections explain the observation model used for the SOG representation and the update step.

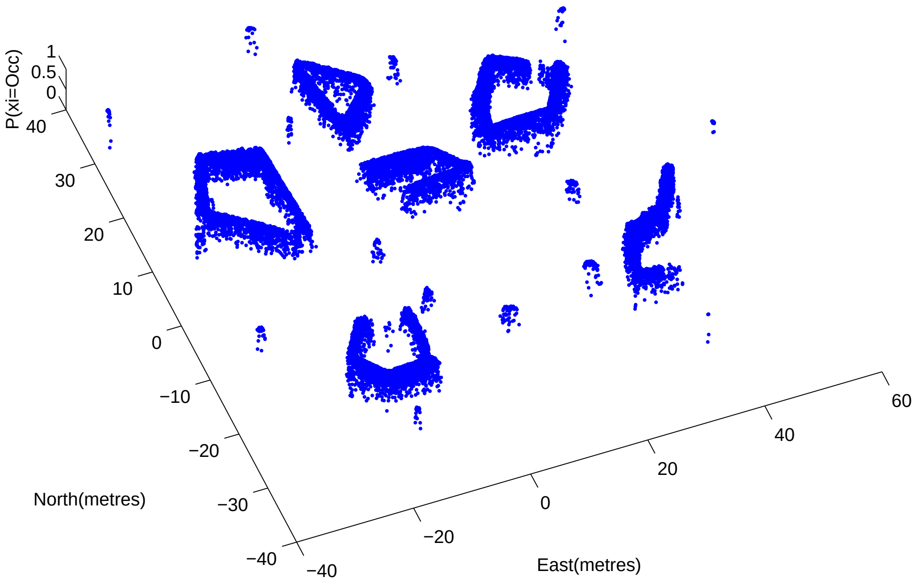
4.3.1 Observation Model

An n-dimensional Gaussian distribution is defined as

$$g(\mathbf{x};\bar{\mathbf{x}},\mathbf{P}) \triangleq \frac{1}{\sqrt{(2\pi)^n\,|\mathbf{P}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^T\mathbf{P}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\right) \tag{4.13}$$

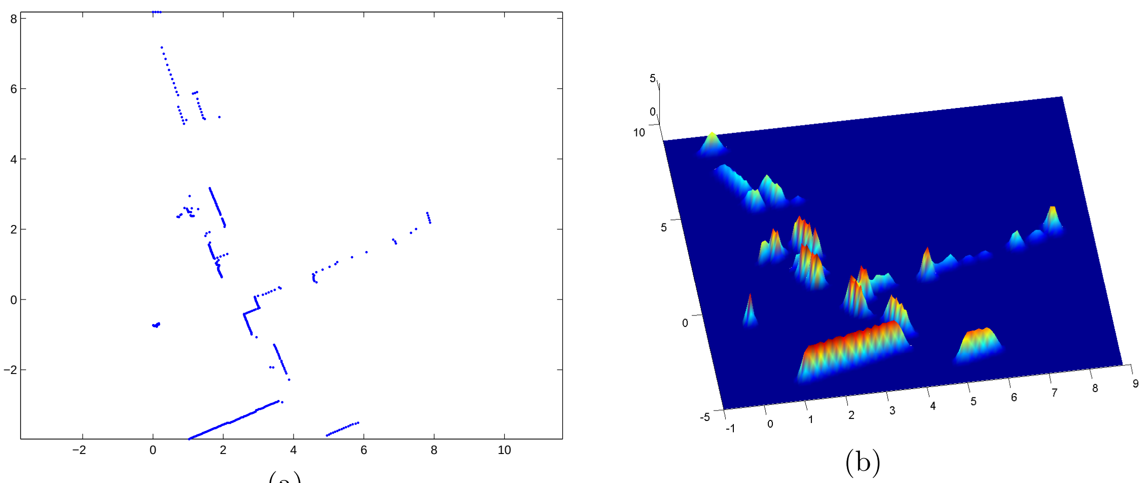


(a)



(b)

Figure 4.10: Environment representation obtained using particle-DenseSLAM. (a) shows with blue dots the particle map and with red lines the actual objects position. (b) shows the particles, where the z axis represents the $P(x_i = Occ)$



(a)

(b)

Figure 4.11: The left-hand figure shows the set of raw range-laser data points transformed to a sensor-centric coordinate frame. The right-hand figure shows the SOG representation of this scan.

where $\bar{\mathbf{x}}$ and $\mathbf{P}$ are the mean and covariance, respectively. An n-dimensional sum of Gaussians (SOG) is defined as the sum of k scaled Gaussians.

$$G(x) \triangleq \sum_{i=1}^{k} \alpha_i g(\mathbf{x}; \bar{\mathbf{x}}_i, \mathbf{P}_i) \tag{4.14}$$

where, for a *normalised* SOG, the sum of the scaling factors α_i is one.

Any set of point measurements with Gaussian noise can be represented as a SOG. In particular, a range and bearing measurement in polar coordinates $\mathbf{z}_i = (r_i, \theta_i)$ with covariance matrix R, can be represented in cartesian coordinates as

$$\mathbf{x}_i = \mathbf{f}\left(\mathbf{z}_i\right) = \begin{bmatrix} r_i \cos\theta_i \\ r_i \sin\theta_i \end{bmatrix} \tag{4.15}$$

with covariance matrix

$$\mathbf{P}_i = \nabla \mathbf{f}_{\mathbf{z}_i} \mathbf{R}_i \nabla \mathbf{f}_{\mathbf{z}_i}^T \tag{4.16}$$

where $\nabla \mathbf{f}_{\mathbf{z}_i} = \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i}$.

Figure 4.11 shows an example of a laser scan represented as a SOG. In this example, the SOG representation was built with Gaussians of equal height, which means that the scaling factors in 4.14 are equal to $\sqrt{(2\pi)^n \left|\mathbf{P}\right|}$, where $\mathbf{P}$ is the Gaussian covariance matrix.

4.3.2 Update

To understand the update it is necessary, first, to clarify what the Gaussians are actually representing. The Gaussians represent the probability of the area under the Gaussian shape of being occupied. But they do not represent an actual distribution and so the representation cannot be used for recursive data fusion. However the Gaussians give a good representation of the shape of the environment. The strategy for the update will be to cover the areas where observations are received with the smallest possible number of Gaussians. This could be done for example by fusing two Gaussians that are close in only one Gaussian with mean equal to the average of the means and variance equal to the sum of the variances.

The algorithm implemented here works as follows:

1. When an observation frame is received, it is represented as a SOG as shown in Section 4.3.1.

2. To fuse the new SOG, a nearest neighbour algorithm associates each Gaussian from the a priori map that are inside the sensor view, a Gaussian from the observation SOG.

3. Similar to the particle-DenseSLAM, a d_{min} value is also defined, so if two Gaussians from the a priori map and the observation scan are associated and the distance between the Gaussians mean is smaller than d_{min} then the two Gaussians will be fused in one new Gaussian.

4. The Gaussian's fusion consists of obtaining a new Gaussian from the two associated Gaussians. This is done as follows. The mean of the new Gaussian $\mathbf{x}_{new}$ is equal to the average of the means (as in particle-DenseSLAM).

   $$\mathbf{x}_{new} = \frac{\mathbf{x}_{priori} + \mathbf{x}_{obs}}{2} \tag{4.17}$$

   To obtain the new Gaussian variance σ_{new}, the variances of the two associated Gaussians are first analysed.

   If the sum of the Gaussian variance from the a priori map and the Gaussian variance from the observation is smaller than $k \cdot d_{min}$

   $$\sigma_{priori} + \sigma_{obs} \leq k \cdot d_{min} \tag{4.18}$$

   Then the covariance matrix of the new Gaussian is equal to the sum of the covariances.

$$\mathbf{P}_{new} = \mathbf{P}_{priori} + \mathbf{P}_{obs} \tag{4.19}$$

If the sum of the Gaussian variance from the a priori map and the Gaussian variance from the observation is larger than $k \cdot d_{min}$ then the covariance matrix of the new Gaussian is equal to the covariance matrix of the a priori Gaussian.

$$\mathbf{P}_{new} = \mathbf{P}_{priori} \tag{4.20}$$

This last point is to ensure that the Gaussians variance will not grow all the time when more observations are received, so if the sum of the Gaussians is large in comparison with the distance between them, then the covariance of the new Gaussian stays as the one from the a priori Gaussian.

4.3.3 Simulation

Results of SOG-DenseSLAM are presented here, using the simulation environment presented previously.

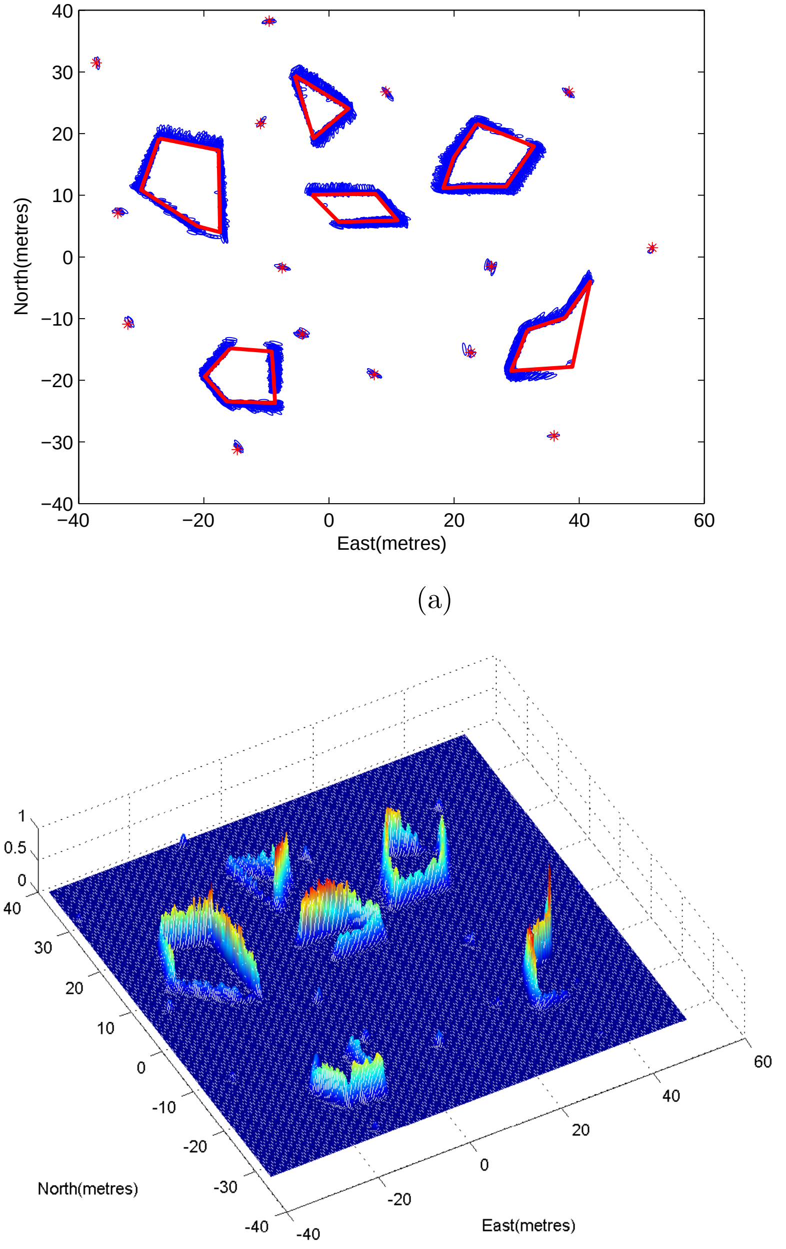
Figure 4.12 illustrates the result obtained using SOG-DenseSLAM. The blue lines in (a) depict the actual objects' positions, and the blue star the actual landmarks' positions. The red lines represent the 1σ uncertainty bound for the Gaussians. For the experiment d_{min} was set to 0.2 and the constant k (Equation 4.18) was set to 0.8. Figure 4.13 shows the Gaussians 1σ ellipses for one of the objects in the map. It is clear from the figure that the SOG could be subsampled and still maintain a representation of similar quality. The implementation presented here does not include subsampling (see [49] for possible subsampling techniques to apply with SOG representations).

Finally Figure 4.12 (b) shows a 3D plot of the final SOG map obtained. The Gaussians were drawn using non-normalised Gaussians (peak equal to one) with a resolution of 0.2 metres.

4.4 Experimental Results

This section presents experimental results of OG-DenseSLAM. The algorithm was implemented with the car park dataset introduced in Section 3.10.

Figure 4.14 displays a zoom-in of the OG-DenseSLAM result. The lighter points represent the laser image obtained using GPS/SLAM. The darker points represent the OG's cells with $P(C_i = Occ) > 0.8$. Figure (a) shows the result obtained before closing the loop, and (b) after the loop is closed. The accuracy of the map is similar to the one obtained in



(a)

(b)

Figure 4.12: Environment representation obtained using SOG-DenseSLAM. The red lines in (a) denote the actual object positions and the blue ellipses the 1σ Gaussians bounds. (b) shows a $3D$ map of the SOG map obtained. The graph was done using non-normalised Gaussians.

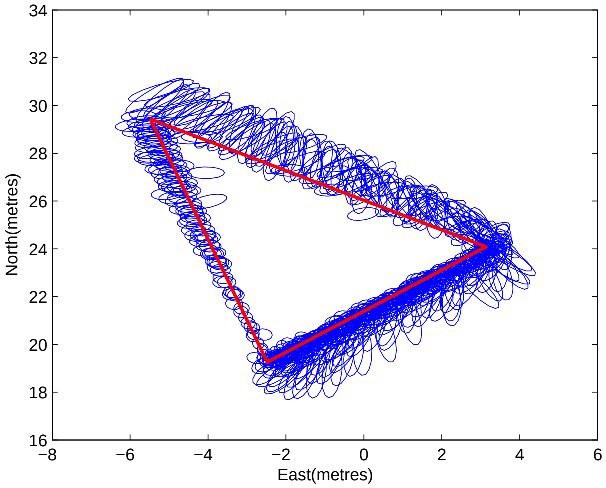


Figure 4.13: The figure shows the Gaussians 1σ ellipses for one of the objects in the map.

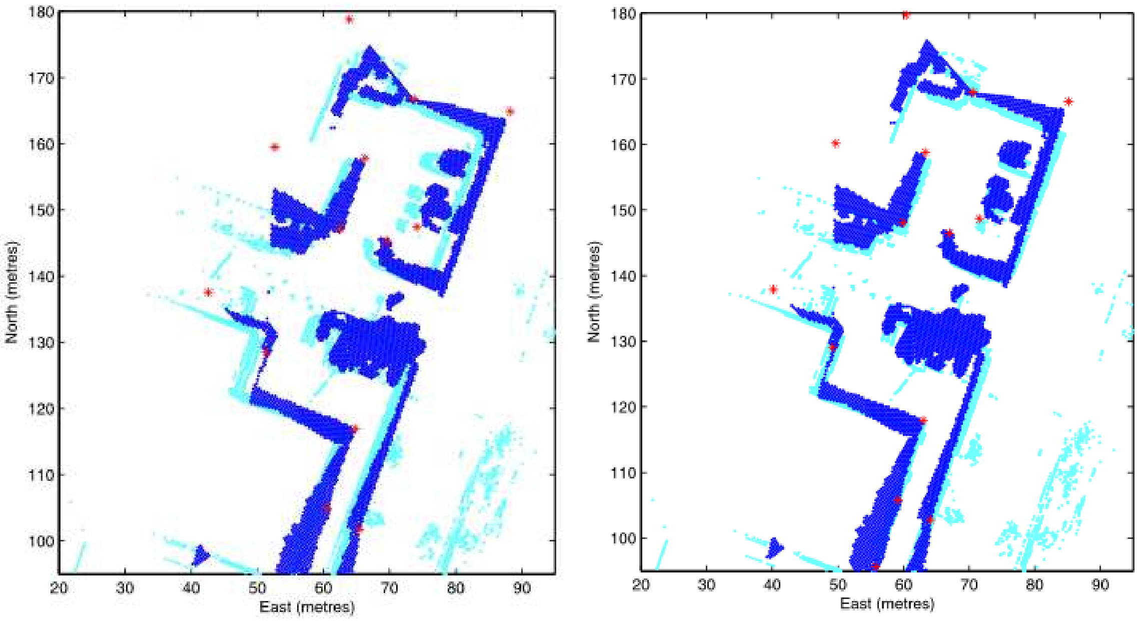
Section 3.10 using the raw observations. This could be expected, since both implementations are using the same landmarks for the localisation process so the quality of the vehicle localisation is identical in both cases. They are actually different map layers of the same SLAM implementation. If the estimated map is compared with the laser-image obtained using GPS/SLAM, an error of approximately 3 metres can be observed in (a). And after the loop is closed, the error is substantially reduced as seen in Figure 4.14 (b).

Figure 4.15 shows the final occupancy grid map obtained with OG-DenseSLAM. The grid cell resolution was set to 0.4 metres.

4.5 Summary

This chapter presented implementations of DenseSLAM using different representations to describe the dense maps. Three different methods to represent the sensed information were shown. Figure 4.16 illustrates the three map layers obtained with OG-DenseSLAM, particle-DenseSLAM and SOG-DenseSLAM. The results presented showed the flexibility of DenseSLAM to work with different representations inside the local regions. In addition since the computational cost of obtaining the dense maps is the computational cost of building a local map, the algorithm allows different map layers of the same sensed data to be obtained at the same time. Each layer will represent the information in a different manner. This fact allows a more robust navigation system, since different tasks will benefit from having access to different environment representations.

DenseSLAM relies on a landmark map which is used to partition the global map into a set



(a)

(b)

Figure 4.14: The lighter points represent the laser image obtained using GPS/SLAM. The darker points represent the final map obtained with DenseSLAM. Figure (a) shows the result before closing the loop, and (b) after the loop is closed.

of local regions. Traditional feature-based SLAM depends on geometric models to define the landmarks. This property restricts feature-based SLAM algorithms to environments suited for the *predefined geometric models*. The next chapter presents a mapping algorithm which, rather than relying on geometric models to define the landmarks, uses raw sensor information to define landmarks' templates avoiding geometric models, thus broadening the suitable environments for SLAM algorithms.

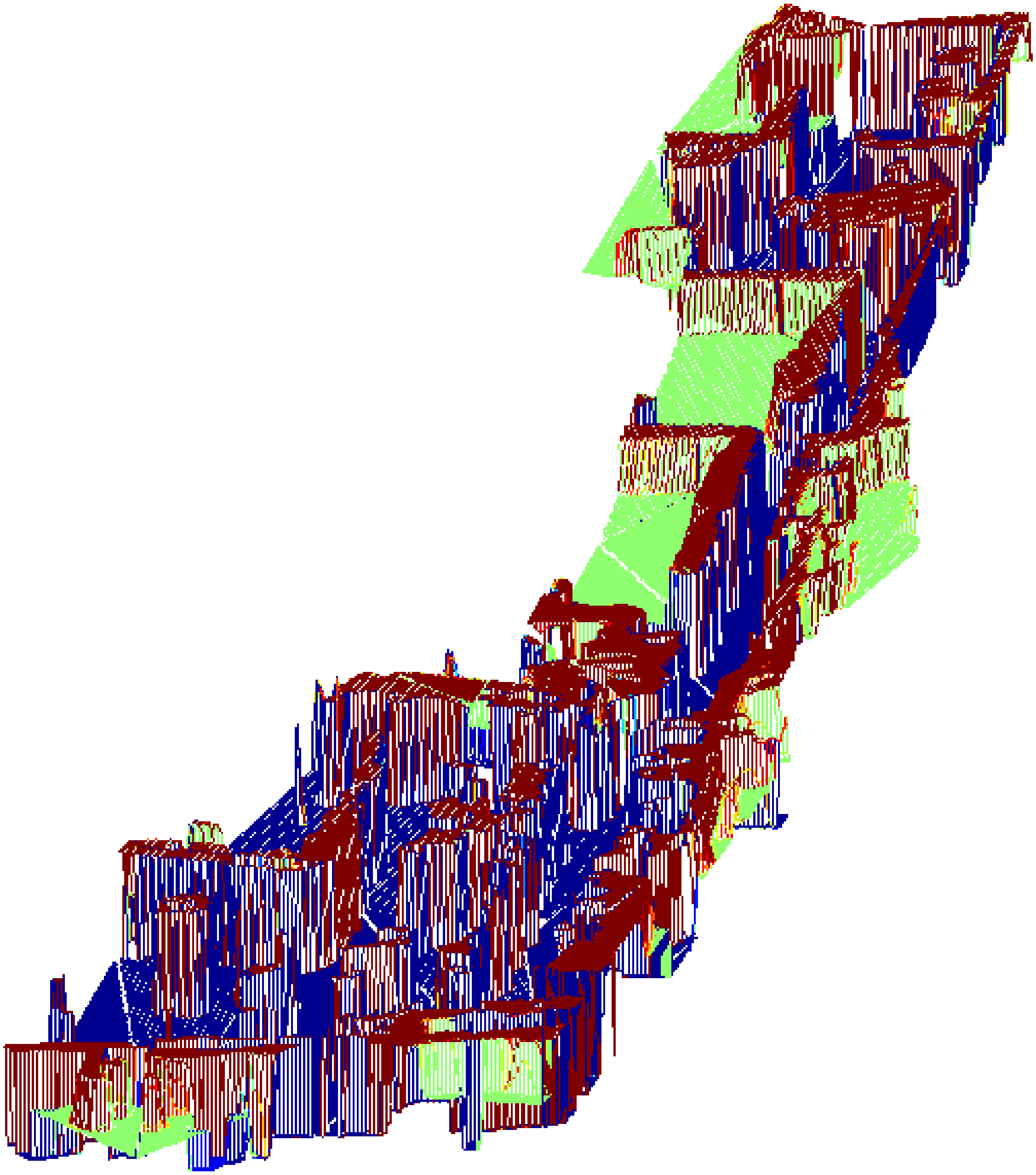


Figure 4.15: Map obtained with OG-DenseSLAM using the car park dataset.

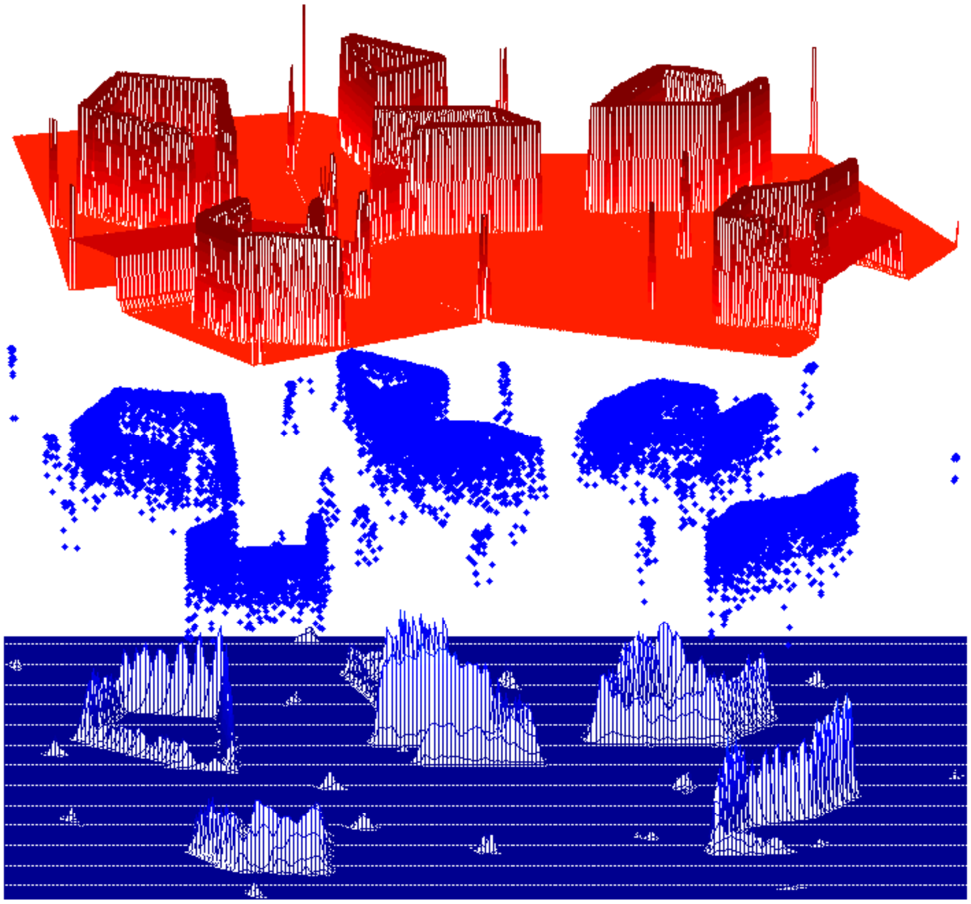


Figure 4.16: The three map layers obtained during the experiments, from top to bottom: the Occupancy grid, the particles and the sum of Gaussians layers.

Chapter 5

SLAM with non-geometric features

This chapter presents a new generalisation of simultaneous localisation and mapping (SLAM). SLAM implementations based on *extended Kalman filter* (EKF) data fusion have traditionally relied on simple geometric models for defining landmarks [46, 68]. This limits EKF-SLAM to environments suited to such models and tends to discard much potentially useful data. An alternative to geometric feature models is a procedure called *scan correlation*, which computes a maximum likelihood alignment between two sets of raw sensor data. Thus, given a set of observation data and a reference map composed similarly of unprocessed data points, a robot can locate itself without converting the measurements to any sort of geometric primitive. The observations are simply aligned with the map data so as to maximise a correlation measure. Scan correlation has primarily been used as a localisation mechanism from an a priori map [85, 32, 7, 40], with the *iterated closest point* (ICP) algorithm [3, 48] and occupancy grid correlation [19] being the most popular correlation methods. However, two significant methods have recently been presented that perform scan correlation based SLAM. The first [79] uses *expectation maximisation* (EM) to maximise the correlation between scans, which results in a set of robot pose estimates that give an "optimal" alignment between all scans. The second method [31] accumulates a selected history of scans, and aligns them as a network. The main concern with both of these methods is that they do not perform data fusion, instead requiring a (selected) history of raw scans to be stored, and they are not compatible with the traditional EKF-SLAM formulation.

The approach presented in this chapter is a marriage of EKF-SLAM with scan correlation [61]. Instead of geometric models, landmarks are defined by templates composed of raw sensed data, and scan correlation is shown to produce landmark observations compatible with the standard EKF-SLAM framework. The resulting *Scan-SLAM* combines the general applicability of scan correlation with the established advantages of an EKF implementation: recursive data fusion that produces a convergent map of landmarks and maintains an estimate of uncertainties and correlations.

The format of this chapter is as follows. The next section presents a review of the most common scan correlation methods. Section 5.2 presents the Scan-SLAM algorithm. Section 5.3 explains the scan matching method used in this thesis. The scan correlation variance estimation is explained in Section 5.4. Finally Section 5.6 and 5.7 show simulation and experimental results of the algorithm.

5.1 Scan-Correlation based SLAM

This section reviews two seminal works for scan-correlation based SLAM.

5.1.1 Consistent pose estimation (CPE)

An algorithm for registration of multiple range scans was presented in [47]. The algorithm maintains all the local frames of data as well as a network of spatial relations among the local frames. The robot pose is used to define the local frames of the scans. Then, spatial relations between local frames are derived by both odometry measurements and matching pairs of scans. ICP [3] is used for the scan matching. The history of robot poses (origin of local frames for the scans) are stored and treated as variables. The goal is to estimate all these pose variables using a network of constraints based on odometry and scan matching, and then register the scans based on the updated poses. All the spatial relations are used simultaneously.

The algorithm can be summarised as follows. Assume a network of uncertain measurements taken about $n+1$ nodes $X_0, X_1, ..., X_n$. A link D_{ij} between two nodes i and j is defined which represents the difference between the positions of the nodes. An observation of D_{ij} is defined as $\overline{D}_{ij} = D_{ij} + \Delta D_{ij}$, where ΔD_{ij} is a random Gaussian variable with zero mean and known covariance matrix C_{ij}. As mentioned, the measurements are based on the constraints obtained from the matching between pairs of scans as well as the odometry measurements. Then, given a set of measurements $\overline{D}_{ij}$ between a pair of nodes, and the covariance C_{ij} the aim is to obtain the optimal estimate of the position $X_i's$ based on the covariance matrices and measurements. The criterion of optimal estimation is based on the maximum likelihood concept and the objective is to minimise the following Mahalanobis distance:

$$W = \sum_{0 \leq i < j \leq n} (X_i - X_j - \overline{D}_{ij})^T C_{ij} (X_i - X_j - \overline{D}_{ij}) \tag{5.1}$$

This optimal estimator for a network of relations is then applied to the robot pose estimation and scan data registration problem. The computation needed for the algorithm is $O(n^3)$, where n represents the number of poses stored.

5.1.2 Local Registration and Global Correlation (LRGC)

Although an incremental CPE is presented in [47], it basically runs the same algorithm with all the poses accumulated to the current point without computational savings. An incremental mapping approach based on CPE was presented in [31]. The method called *Local Registration and Global Correlation* (LRGC) integrates the scans into the map as they are taken by the sensor. However, when closing loops, additional computational power is needed.

One of the problems of the basic CPE algorithm is that it requires a good initial estimate of the scan poses in order to generate good results. These initial estimates are obtained from the odometry. However, in large environments the robot could accumulate large uncertainty and the method could converge to a local minimum that is incorrect. To avoid this problem, LRGC uses CPE to create local patches (local maps of the last few scans), since in this case very little odometric error has been accumulated. The algorithm also uses scan matching to build the local patches. When a new scan is added to the map, it is first registered with the last K scans (local neighborhood) for proper alignment and for improving the position estimated from odometry. The new scan pose together with its links are then added to the current map.

For closing a loop, first topological relationships have to be obtained. In order to do that, the algorithm runs a map correlation algorithm in the background which checks for matches against the old map. A recent portion of the map around the pose and the older portions of the map are compared using map correlation. Where there is a good match according to the map correlation algorithm, it is likely that the new pose is topologically connected to one of the older poses. This loop detection algorithm is activated every few seconds. After the map correlation algorithm has confirmed the topological relations between the new map and the old map, CPE is first run with the new links added to the map and then, after closing the loop leading to a topologically correct map, CPE is run again with the new scan matches for fine-tuning the map. An analysis of uncertainty and consistency of the estimation was not presented.

In summary, the advantage of LRGC when compared with classic CPE is that it can integrate the new information into the map at the time the observations are taken. However, the algorithm presented in [31] requires ad-hoc solutions when part of the map is revisited.

5.2 Scan-SLAM

A significant issue with EKF-SLAM is the design of the observation model. Current implementations require landmark observations to be modelled as geometric shapes, such as lines or circles. Measurements must fit into one of the available geometric categories in order to be classified as a feature, and non-conforming data is ignored. The chief problem with geometric observation models is that they tend to be environment specific, so that a model suited to one type of environment might not work well in another and, in any case, a lot of useful data is thrown away.

This section presents a new approach to scan correlation that is integrated with the EKF-SLAM framework. The map is constructed as an on-line data fusion problem and maintains an estimate of uncertainties in the robot pose and landmark locations. There is no requirement to accumulate a scan history. Unlike previous EKF-SLAM implementations, landmarks are not represented by simplistic geometric models, but rather are defined by a template of raw sensor data. This way the feature models are not environment specific and good data is not thrown away. The result is *Scan-SLAM* that uses raw data to represent landmarks and scan matching to produce landmark observations. In essence, this approach presents a new way to define generic observation models, and in all other respects Scan-SLAM behaves in the manner of conventional EKF-SLAM.

5.2.1 Scan-SLAM Overview

When a scan segment is accepted as a landmark, a landmark definition template is created by extracting the selected set of measurements from the current scan . These measurements form a Sum of Gaussians (SOG), which is transformed to a coordinate frame local to the landmark. While there is no inherent restriction as to *where* this local axis is defined, it is more intuitive to locate it somewhere close to the landmark data-points and, in this thesis, the local coordinate frame is defined as the centroid of the template SOG. A new landmark is added to the SLAM map by adding the global pose of its coordinate frame to the SLAM state vector. Note that the landmark description template is not added to the SLAM state and is stored in a separate data structure.

As new scans become available, the SLAM estimate of existing map landmarks can be updated by the following process. First, the location of a map landmark relative to the vehicle is predicted to determine whether the landmark template SOG $G_L(\mathbf{x})$ is in the vicinity of the current scan SOG $G_o(\mathbf{x})$. This vehicle-relative landmark pose is the predicted observation $\hat{\mathbf{z}}$. If the predicted location is sufficiently close to the current scan, the landmark template is aligned with the scan, using $\hat{\mathbf{z}}$ as an initial guess (see Figure 5.1).

$$[\mathbf{z},\ \mathbf{R}] = \texttt{scan\_align}(G_L(\mathbf{x}),\ G_o(\mathbf{x}),\ \hat{\mathbf{z}})$$

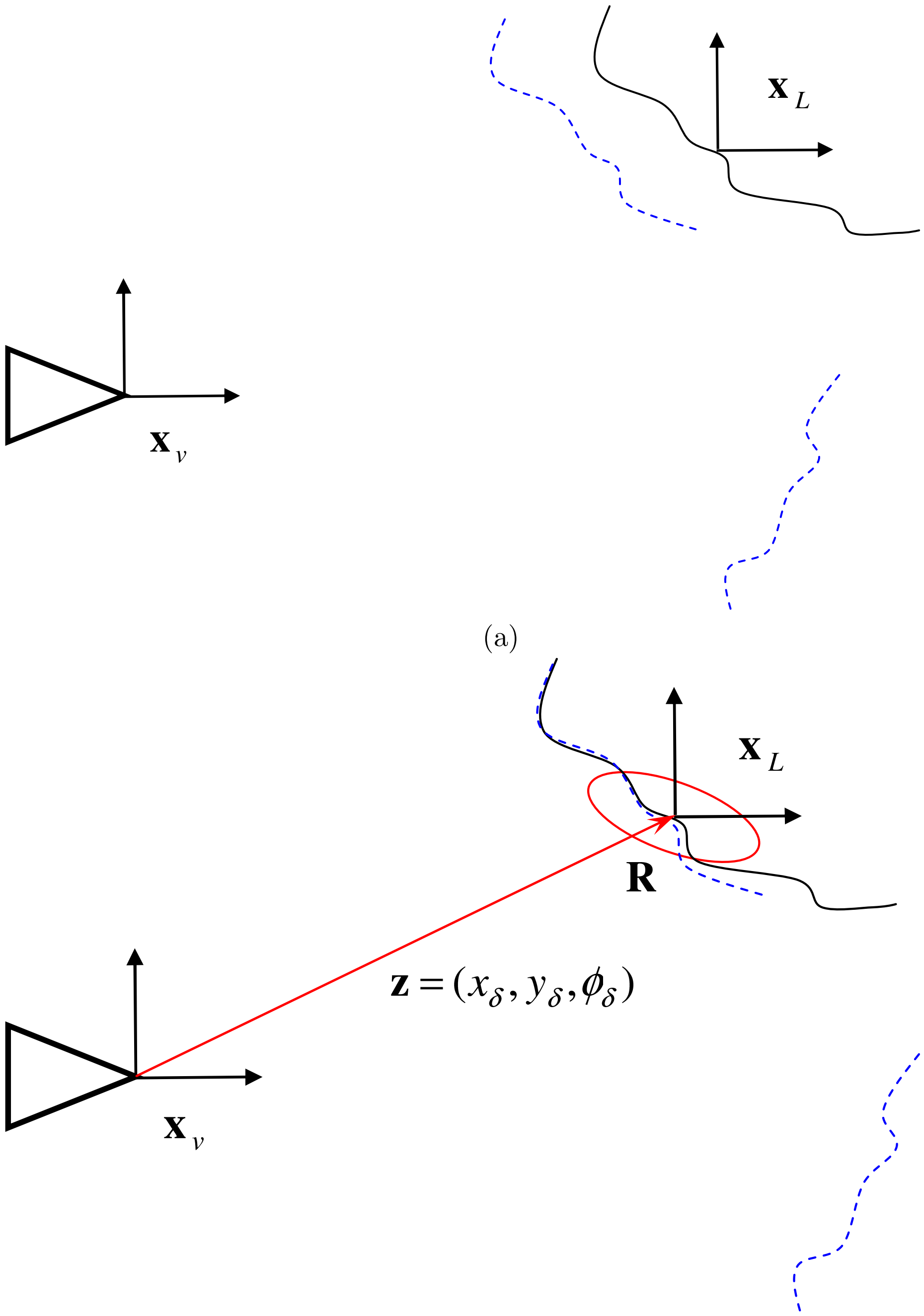
The result of scan alignment gives the pose of the landmark template frame with respect to the current scan coordinate frame, which is defined by the current vehicle pose. This is the new landmark observation $\mathbf{z}$ with uncertainty $\mathbf{R}$. Having obtained the observation $\mathbf{z}$ and $\mathbf{R}$, the SLAM state is updated in the usual manner of EKF-SLAM.

5.2.2 Scan Segmentation

Segmentation is the first process after a new scan is received. The segmentation criterion used here is based on distance. It selects sensor segments that contain a minimum number of neighbour points. This is justified since big objects will in general have a more stable position than small objects. Figure 5.2 shows an example of segments extracted from a laser frame. In the example three segments were selected which are depicted by different colours.

Since the segmentation is only based on distance, it is necessary to have a procedure that gives an estimate of the pose error one might expect when registering the particular segment. A parameter called *typical boundary error* is introduced in [67]. The objective is to find a parameter that can give a simple idea of the level of accuracy of the alignment. The covariance of the transformation could be used to estimate the quality of the alignment, but is too rich to give an idea of the alignment to an end user. Instead it is proposed to use the RMS error expectation of the points aligned. This concept is applied to images, where the averaged RMS for the vertices of the 3D image is called the *typical boundary error*. Similarly, in [74] another parameter is defined called *registration index*. The registration index gives an indication of how well two surfaces may be registered. In [83] a parameter called *object saliency score* is defined. This parameter is defined as the inverse of the covariance matrix trace, obtained after aligning two shapes. The larger the object saliency score, the more certain the pose estimate from the registration process.

These parameters can then be used to predict the accuracy of the registration process with a particular segment and then, to assist in deciding whether the segment will be incorporated as a new landmark or not. The procedure is as follows: when a new frame is received it is first segmented using the distance criterion. A copy of each segment is made and a random translation and rotation are applied to each copy. Then the copies are registered with their respective original segments and for each segment a measure of the quality of the matching is obtained using, for example, the object saliency score.



(a)

(b)

Figure 5.1: The top figure shows a stored scan landmark template (solid line) and a new observed scan (dashed line). The bottom figure shows the scan alignment evaluated with the scan correlation algorithm from which the observation vector z is obtained.

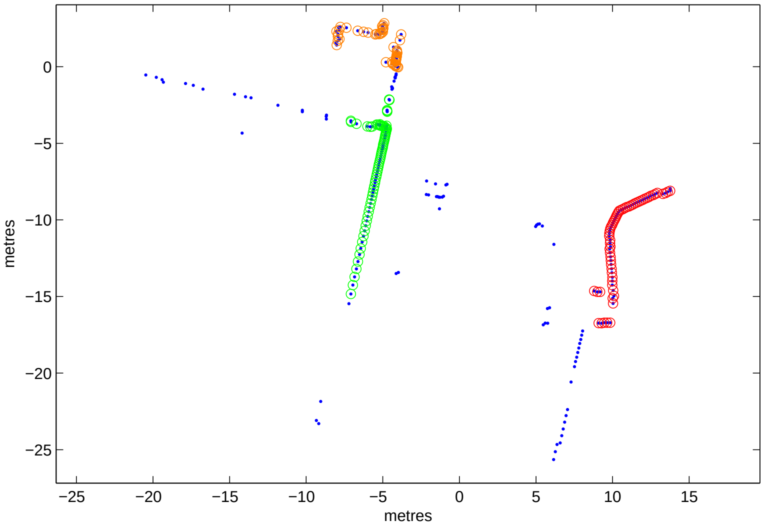


Figure 5.2: Scan segmentation based on distance. The blue points represent the laser scan and the ‘o’ denote the segments. Colours are used to differentiate different segments.

5.2.3 Local representation

Once the scan has been segmented and the new segment has been accepted as a new landmark, a local coordinate system is defined. In the case in which the robot is moving in a plane, the local axis will be defined as

$$\mathbf{x}_L = [x_L, y_L, \phi_L]^T \tag{5.2}$$

where (x_L, y_L) is the position of the axis in the plane and ϕ_L the orientation.

There is no intrinsic restriction as to where the local axis pose is defined, however it will be more intuitive and easier to visualise if the local axis pose is located somewhere near the new scan landmark points. Figure 5.3 illustrates the concept. A new sensor frame is received and a local axis is defined with position equal to the mean of the scan and orientation equal to the vehicle heading.

As mentioned before, the new landmark is added to the SLAM map by adding the global pose of its coordinate frame to the state vector. The augmented state vector will be:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{x}}_L \end{bmatrix} \tag{5.3}$$

where $\hat{\mathbf{x}}_v$ represents the vehicle states and $\hat{\mathbf{x}}_L$ the landmarks map.

$$\hat{\mathbf{x}}_v = [\hat{x}_v, \hat{y}_v, \hat{\phi}_v]^T \tag{5.4}$$

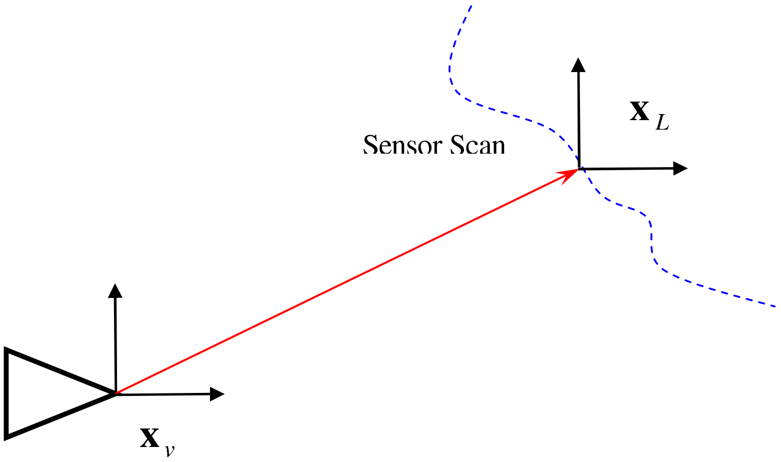


Figure 5.3: The local axis position is defined as the centroid of the template points and orientation equal to the vehicle heading.

$$\hat{\mathbf{x}}_L = [\hat{x}_{L_1}, \hat{y}_{L_1}, \hat{\phi}_{L_1}, \ldots, \hat{x}_{L_i}, \hat{y}_{L_i}, \hat{\phi}_{L_i}, \ldots, \hat{x}_{L_n}, \hat{y}_{L_n}, \hat{\phi}_{L_n}]^T \tag{5.5}$$

The points representing the scan landmark are stored in a separate data structure, thus what actually represents the landmark in the SLAM is the global pose of the coordinate system. The rest of the information is used for the scan correlation algorithm to obtain the observation vector as explained in Section 5.3.1.

Note: the objective of using a local coordinate system in this chapter is totally different to the objective pursued in DenseSLAM. DenseSLAM defines a local coordinate system formed by the position of three landmarks in order to decorrelate the internal map with the rest of the system. In this chapter, the local coordinate system is used to describe a sensor scan as a point and then make it compatible with the state-space representation.

Fusing scan-landmarks

It could be the case that two scan landmarks incorporated already in the map are very close and then both could be represented using only one local axis, forming one landmark with more points in the scan-template. This section presents the equations necessary to represent a Gaussian ($\bar{\mu}_2$,$\mathbf{\Sigma}_2$) that is originally represented locally to the landmark axis L_2, in the landmark axis L_1. Figure 5.4 illustrates the case. There are two landmarks L_1 and L_2 and there is a Gaussian (depicted with an ellipse) represented in L_2. This Gaussian $\bar{\mu}_2$ needs to be represented now locally in L_1. The new Gaussian is denoted as $(\bar{\mu}_2^1, \mathbf{\Sigma}_2^1)$.

The new Gaussian mean can be obtained applying a rotation $\mathbf{R}$ and a translation $\mathbf{T}$ of the original Gaussian position $\bar{\mu}_2$.

$$\bar{\mu}_2^1 = \mathbf{f}(\mathbf{x}_{L1}, \mathbf{x}_{L2}, \bar{\mu}_2) = \mathbf{R}\bar{\mu}_2 + \mathbf{T} \tag{5.6}$$

where

$$\mathbf{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{5.7}$$

$$\mathbf{T} = \mathbf{x}_{L2} - \mathbf{x}_{L1} \tag{5.8}$$

being

$$\psi = \phi_{L2} - \phi_{L1} \tag{5.9}$$

$$\mathbf{x}_{L1} = [x_{L1}, y_{L1}]^T \tag{5.10}$$

$$\mathbf{x}_{L2} = [x_{L2}, y_{L2}]^T \tag{5.11}$$

The covariance of the new Gaussian represented in the new local frame can be calculated as:

$$\mathbf{\Sigma}_2^1 = \nabla\mathbf{f}_{\bar{\mu}_2}\mathbf{\Sigma}_2\nabla\mathbf{f}_{\bar{\mu}_2}^T + \nabla\mathbf{f}_{L_1}\mathbf{P}_{11}\nabla\mathbf{f}_{L_1}^T + \nabla\mathbf{f}_{L_2}\mathbf{P}_{22}\nabla\mathbf{f}_{L_2}^T - (\nabla\mathbf{f}_{L_1}\mathbf{P}_{12}\nabla\mathbf{f}_{L_2}^T + \nabla\mathbf{f}_{L_2}\mathbf{P}_{21}\nabla\mathbf{f}_{L_1}^T) \tag{5.12}$$

where $\mathbf{\Sigma}_2$ is the covariance of the Gaussian in the local frame number two, $\mathbf{P}_{ij}$ is the cross-covariance between the landmark i and j-th, $\nabla\mathbf{f}_{\bar{\mu}_2}$ is the derivative of f with respect to the Guassian position $\bar{\mu}_2$ and $\nabla\mathbf{f}_{L_i}$ is the derivative of f with respect to the landmark L_i. Equations 5.6 and 5.12 give the necessary equations to represent a Gaussian in the local axis L_1 originally represented with respect to the local axis L_2.

5.3 Scan Matching

Scan matching algorithms present an alternative solution to geometric feature models. Given a set of observation data and a reference map, scan matching computes a maximum alignment between the two sets of raw sensor data. Thus, through the result from this alignment the robot can locate itself without converting the measurements to any geometric template. *Iterative closest point* (ICP) and occupancy grid correlation have been

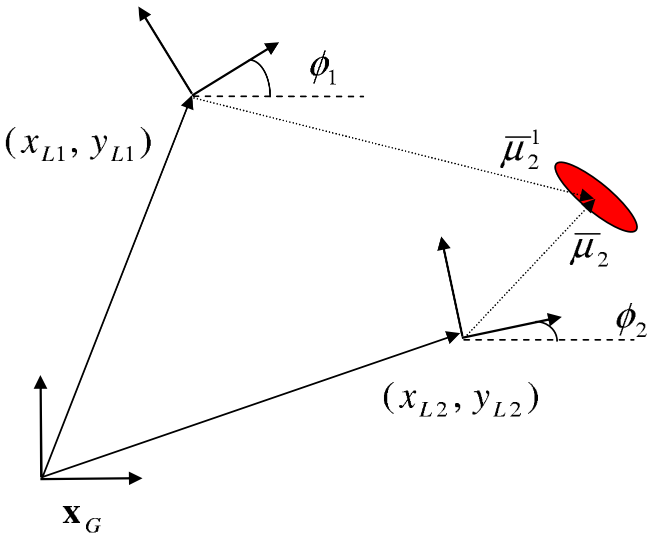


Figure 5.4: A Gaussian $\bar{\mu}_2$ originally represented in the local frame number two is now represented in the local frame number one $\bar{\mu}_2^1$.

the two most popular algorithms for scan matching.

ICP [3, 48] is the most popular range-image registration technique mainly due to its simplicity. Given a set of observed points P_o and a set of reference points P_r, the algorithm iteratively calculates the pose $\mathbf{x}_o$ from where P_o was taken by a process of point to point data association, searching for the pose that minimises the least-squared error between the associated points. The original ICP method does not estimate pose uncertainty in its solution, but a number of approaches which include pose uncertainty estimation appeared some years later [67, 74, 83]. In [83] for example, instead of using only one initial pose for the alignment, the registration process is run N times with randomly generated initial relative transformations. The variance in the final pose obtained for the different samples is used as the sensor pose uncertainty.

Another popular method to perform scan correlation is occupancy grid correlation [19, 76, 71]. Occupancy grid correlation is applied in a similar way to template matching methods in image processing. The vehicle pose space is sampled by shifting the vehicle pose. The correlation value is calculated by multiplying the overlapping grid cells and summing the result for each grid. This procedure is done for each cell in the observation space. The final correlation distribution is then a function of the robot pose.

The scan matching algorithm used in this chapter is the one presented in [1], which uses Gaussian Sum to represent the scans. However, the Scan-SLAM framework is independent of the scan matching technique used and other methods like ICP could be used, just by replacing the module for the scan matching algorithm. The next subsection reviews the Gaussian sum scan matching approach introduced in [1].

5.3.1 Scan Matching using Gaussian Sum representation

The SOG representation permits efficient correlation of two scans of data, and has a Bayesian justification which ensures that, under certain conditions, the scan alignment estimate is consistent (see [1]). SOG correlation also avoids limitations inherent to occupancy grid and ICP correlation methods, these being fixed-scale granularity and point-to-point data associations, respectively. Given two scans of Cartesian data points, where each point has a mean and variance, the respective scans may be represented by two SOGs.

$$\begin{aligned} G_1(\mathbf{x}) &= \sum_{i=1}^{k_1} \alpha_i g(\mathbf{x}; \bar{\mathbf{p}}_i, \mathbf{P}_i) \\ G_2(\mathbf{x}) &= \sum_{i=1}^{k_2} \beta_i g(\mathbf{x}; \bar{\mathbf{q}}_i, \mathbf{Q}_i) \end{aligned}$$

A likelihood function for the correlation of these two SOGs is given by their cross-correlation.

$$\begin{aligned} \Lambda(\mathbf{x}) &= G_1(\mathbf{x}) \star G_2(\mathbf{x}) \\ &= \int \sum_{i=1}^{k_1} \alpha_i g(\mathbf{u} - \mathbf{x}; \bar{\mathbf{p}}_i, \mathbf{P}_i) \sum_{j=1}^{k_2} \beta_j g(\mathbf{u}; \bar{\mathbf{q}}_j, \mathbf{Q}_j) d\mathbf{u} \\ &= \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} \alpha_i \beta_j \gamma_{ij}\left(\mathbf{x}\right) \end{aligned} \tag{5.13}$$

where $\gamma_{ij}\left(\mathbf{x}\right)$ is the cross-correlation of two Gaussians $g(\mathbf{x}; \bar{\mathbf{p}}_i, \mathbf{P}_i)$ and $g(\mathbf{x}; \bar{\mathbf{q}}_j, \mathbf{Q}_j)$.

$$\gamma_{ij}\left(\mathbf{x}\right) = \frac{1}{\sqrt{(2\pi)^n \left|\mathbf{\Sigma}\right|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mu})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \bar{\mu})\right)$$

$$\begin{aligned} \bar{\mu} &= \bar{\mathbf{p}}_i - \bar{\mathbf{q}}_j \\ \mathbf{\Sigma} &= \mathbf{P}_i + \mathbf{Q}_j \end{aligned}$$

The result is a likelihood function that provides a measure of scan alignment, and a maximum-likelihood alignment can be obtained as

$$\mathbf{x}_M = \arg\max_{\mathbf{x}} \Lambda(\mathbf{x}) \tag{5.14}$$

where pose $\mathbf{x}_M$ is the maximum-likelihood location of scan 1 with respect to scan 2. More precisely, $\mathbf{x}_M$ is the location of the coordinate frame of scan 1 with respect to the coordinate frame of scan 2.

Full details of Gaussian sum correlation can be found in [1, Section 4.4]. In particular, it describes SOG scaling factors, SOG correlation in a plane (with alignment over position *and* orientation), and various alternatives for efficient implementation.

Figure 5.5 shows the alignment result using Gaussian sum correlation, for two scans taken with a SICK laser in an office environment. Figure (a) shows the two original scans and (b) shows the alignment made by the algorithm. Figure 5.6 illustrates the result using ICP, a fixed number of iterations $N = 20$ was used. For the example presented, both algorithms appeared to obtain good alignments.

5.3.2 Generic Observation Model

The algorithm defines a landmark by a SOG in a local landmark coordinate frame, and the Scan-SLAM map stores a global pose estimate of this coordinate frame in its state vector. Thus, all observations of landmarks obtained by scan matching can be modelled as the measurement of a global landmark frame $\mathbf{x}_L$ as seen from the global vehicle pose $\mathbf{x}_v$.

The generic observation model for the pose of a landmark coordinate frame with respect to the vehicle is as follows.

$$\begin{aligned} \mathbf{z} = [x_\delta, y_\delta, \phi_\delta]^T &= \mathbf{h}\left(\mathbf{x}_L, \mathbf{x}_v\right) \\ &= \begin{bmatrix} (x_L - x_v)\cos\phi_v + (y_L - y_v)\sin\phi_v \\ -(x_L - x_v)\sin\phi_v + (y_L - y_v)\cos\phi_v \\ \phi_L - \phi_v \end{bmatrix} \end{aligned} \tag{5.15}$$

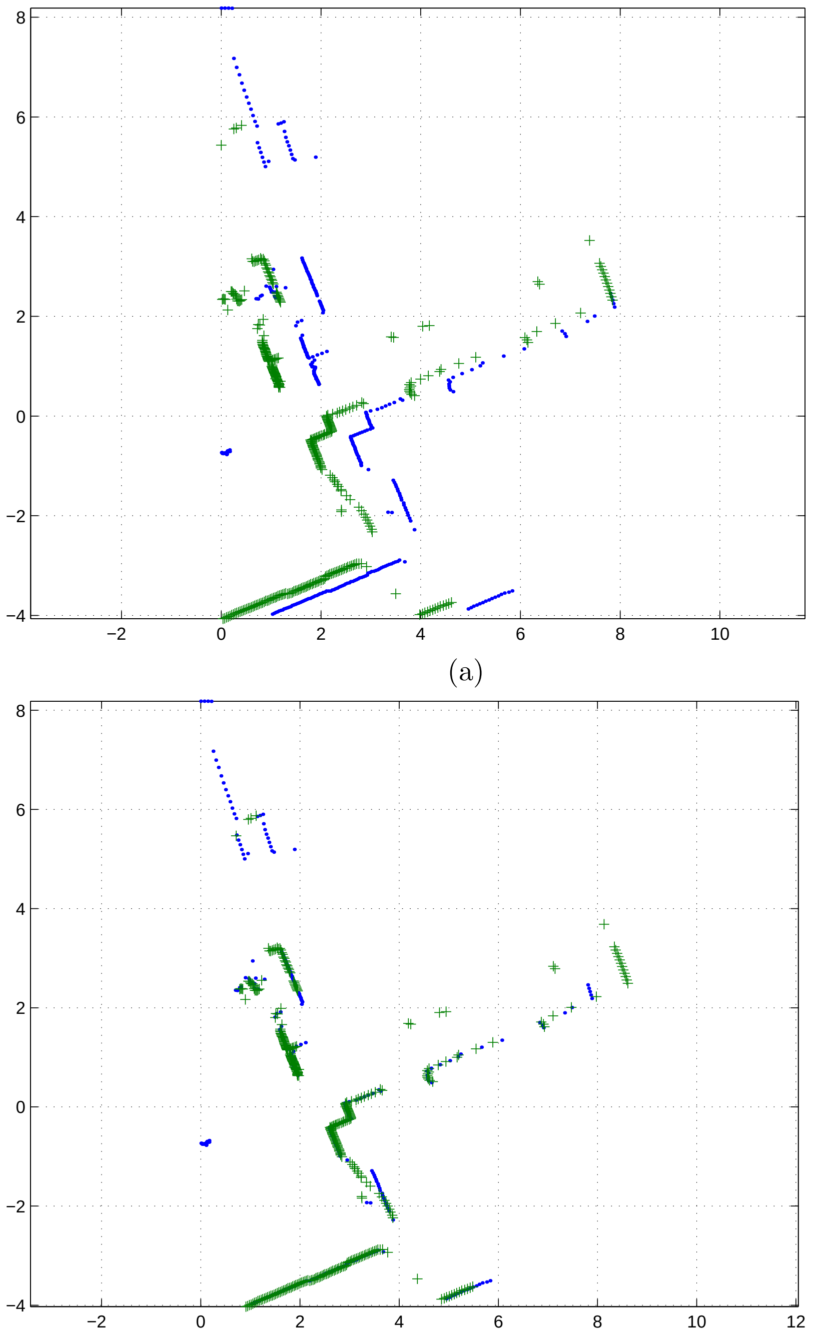
Figure 5.7 illustrates the observation model. In the figure, the vehicle pose is represented by $\mathbf{X}_v$ and the landmark pose by $\mathbf{X}_L$

5.4 Scan Correlation Variance

For scan correlation to be compatible with EKF-SLAM, it is necessary to approximate the correlation likelihood function in Equation 5.13 by a Gaussian. This section presents a method to compute a mean and variance for scan correlation based on the shape of the likelihood function in the vicinity of the point of maximum-likelihood[1]. The resulting approximation is reasonable because the likelihood function tends to be Gaussian in shape in the region close to the maximum-likelihood.

The first step in deriving this approximation is to compute the variance of a Gaussian

[1] Thanks to Tim Bailey for this derivation



(a)

(b)

Figure 5.5: Scan alignment using Gaussian sum scan matching: the reference scan is denoted by '.' and the observation scan by '+'. Figure (a) shows the original scans position and (b) the result after the alignment.

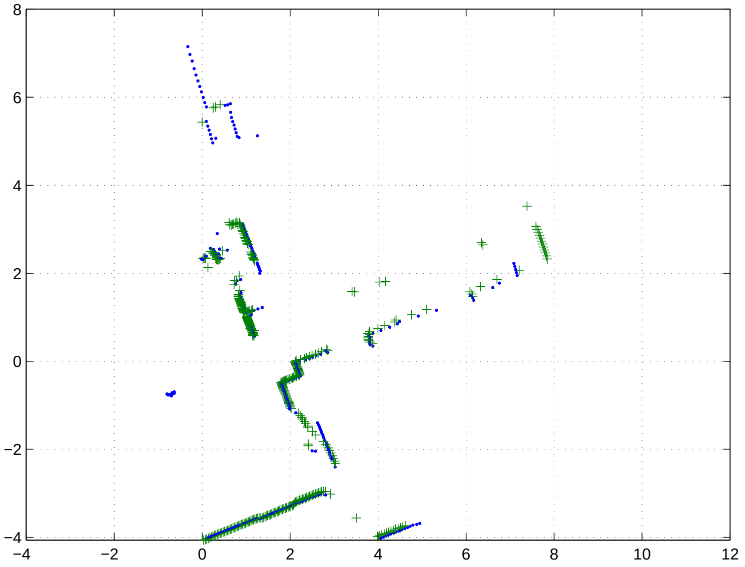


Figure 5.6: Scan alignment result using traditional ICP: the reference scan is denoted by '.' and the observation scan by '+'.

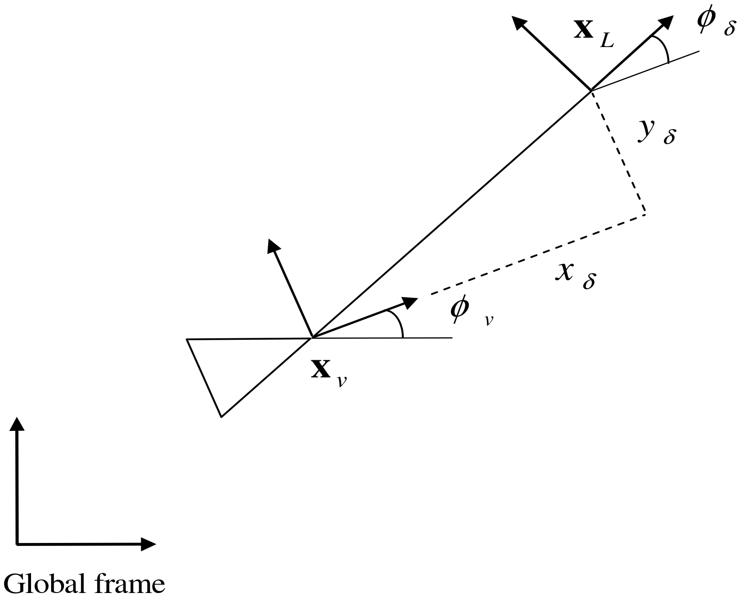


Figure 5.7: All SOG features are represented in the SLAM map as a global pose identifying the location of the landmark coordinate frame. The generic observation model for these features is a measurement of the global landmark pose $\mathbf{x}_L$ with respect to the global vehicle pose $\mathbf{x}_v$. The vehicle-relative observation is $\mathbf{z} = [x_\delta, y_\delta, \phi_\delta]^T$.

function given only a set of point evaluations of the function. Given a Gaussian PDF

$$g(\mathbf{x}; \bar{\mathbf{p}}, \mathbf{P}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{P}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{p}})^T \mathbf{P}^{-1} (\mathbf{x} - \bar{\mathbf{p}})\right)$$

the maximum-likelihood value is found at its mean

$$g(\bar{\mathbf{p}}; \bar{\mathbf{p}}, \mathbf{P}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{P}|}} = C_M$$

Any other sample $\mathbf{x}_i$ from this distribution will have the value

$$\begin{aligned} g(\mathbf{x}_i; \bar{\mathbf{p}}, \mathbf{P}) &= C_M \exp\left(-\frac{1}{2}(\mathbf{x}_i - \bar{\mathbf{p}})^T \mathbf{P}^{-1} (\mathbf{x}_i - \bar{\mathbf{p}})\right) \\ &= C_i \end{aligned}$$

By taking logs and rearranging terms, we get

$$(\mathbf{x}_i - \bar{\mathbf{p}})^T \mathbf{P}^{-1} (\mathbf{x}_i - \bar{\mathbf{p}}) = -2(\ln C_i - \ln C_M) \tag{5.16}$$

Thus, given a set of samples $\{\mathbf{x}_i\}$ and their associated function evaluations $\{C_i\}$, along with the maximum-likelihood parameters $\bar{\mathbf{p}}$ and C_M, then the inverse covariance matrix $\mathbf{P}^{-1}$ (and hence $\mathbf{P}$) can be evaluated. The only requirement is that the number of samples equals the number of unknown elements in $\mathbf{P}^{-1}$.

In this thesis, we are concerned with 3-dimensional Gaussians, (i.e., to represent the distribution of a landmark pose $[x_L, y_L, \phi_L]^T$), and so we present the full derivation of variance evaluation for this case. We define the following variables

$$\begin{aligned} C_i^{'} &= -2(\ln C_i - \ln C_M) \\ \mathbf{x}_i - \bar{\mathbf{p}} &= [x_i, y_i, z_i]^T \\ \mathbf{P}^{-1} &= \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \end{aligned}$$

Substituting these into Equation 5.16 and expanding terms gives

$$x_i^2 a + 2x_i y_i b + 2x_i z_i c + y_i^2 d + 2y_i z_i e + z_i^2 f = C_i^{'} \tag{5.17}$$

The result is an equation with six unknowns $(a, \ldots, f)$, and so a solution can be found given six samples from the Gaussian. This is posed as a matrix equation of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where the i-th row of $\mathbf{A}$ is $[x_i^2, 2x_iy_i, 2x_iz_i, y_i^2, 2y_iz_i, z_i^2]$, $\mathbf{x}$ is the unknowns $[a, b, c, d, e, f]^T$, and $\mathbf{b}$ is the set of solutions $\{C_i^{'}\}$. For a Gaussian function, the solution of this system of equations gives the exact covariance matrix of the function.

Since the scan correlation likelihood function is not exactly Gaussian (although it is presumed it has approximately Gaussian shape near the maximum likelihood location), different sets of samples will produce different values for $\mathbf{P}$. To reduce this variation, more than the minimum number of samples are evaluated and compute a least-squares solution using *singular value decomposition* (SVD), which results in a much more stable covariance estimate.

In summary, two SOGs are aligned according to a maximum likelihood correlation, to give the pose $\mathbf{x}_M$ between scan coordinate frames. A number of samples, $N > 6$, from the region near $\mathbf{x}_M$ are evaluated and the alignment variance $\mathbf{P}_M$ is computed by SVD. At a higher level of abstraction, the result of this algorithm can be described by the following pseudocode function interface

```
[$\mathbf{x}_M$, $\mathbf{P}_M$] = scan_align($G_1(\mathbf{x})$,$G_2(\mathbf{x})$, $\mathbf{x}_0$)
```

where $\mathbf{x}_0$ is an initial guess of the pose of $G_1(\mathbf{x})$ relative to $G_2(\mathbf{x})$.

5.5 Data Association

Since the approach is based on scan alignment methods, there is no need for data association. However, because the observations obtained are compatible with EKF-SLAM, any of the data association techniques developed for SLAM algorithms [1, 5, 58] can be used after the alignment to verify the consistency of the observation.

The procedure would be as follows: all the segments that are inside the sensor view are matched against the new observed scans and a new observation vector is obtained. Then the new observations are verified with, for example, a gate validation test (see Section 2.4) that will decide whether the individual observations are accepted or not.

5.6 Simulation

This section presents simulation results for the Scan-SLAM algorithm. The simulation environment is shown in Figure 5.8. The experiment was done in a large area of 180 by 160 metres with a sensor field of view of 30 metres. The vehicle travels at a constant speed of $3m/s$. The sensor observations are corrupted with Gaussian noise with standard deviations

0.1 metres in range and 1.5 degrees in bearing. The simulation map consists of objects with different geometry and size. In order to select the segments to be added in the navigation map, a basic segmentation algorithm was implemented that selects sensor segments that contain a minimum number of neighbour points.

The results for the Scan-SLAM algorithm are shown in Figure 5.8. The solid line depicts the ground truth for the robot pose and the dashed line the estimated vehicle path. The actual object's position is represented by the light solid line and the segment's position by the dark points. The local axis pose for each scan landmark is also shown and the ellipses indicate the 3σ uncertainty bound of each scan landmark. The local axis position was defined equal to the average position of the raw points included in the segment and the orientation equal to the vehicle orientation. Figure 5.9 shows a zoom of the left top area of the simulation which depicts in more detail the sensory information that is added to the representation obtained by the algorithm. Figure 5.8 (b) shows the result after the vehicle closes the loop and the EKF-SLAM updates the map. A bigger error reduction between the actual object's position and the estimates by the algorithm after closing the loop can be observed.

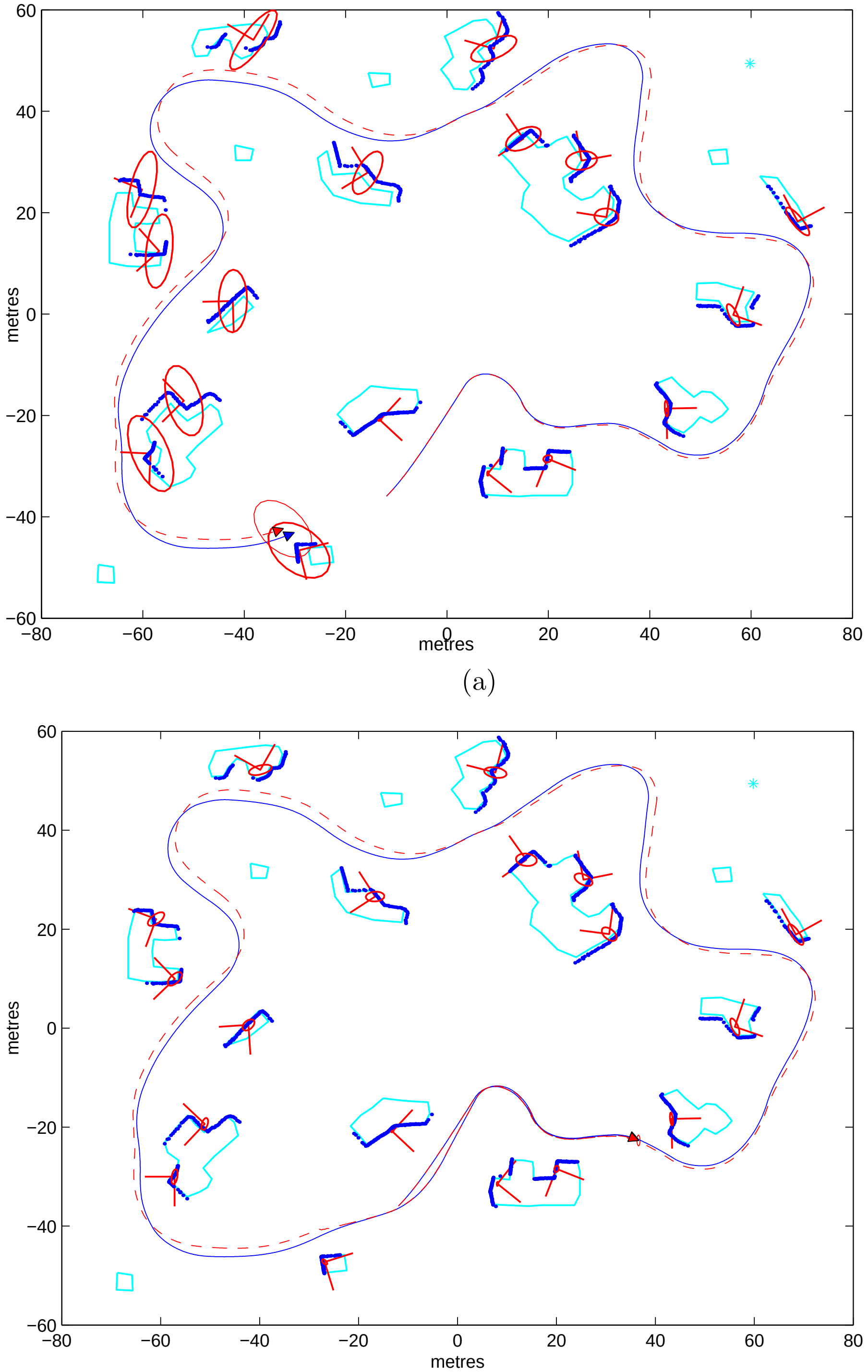
Parameter	**Value**
Dead reckoning sampling time	0.025 secs
Steering sensor σ	5°
Velocity encoder σ	0.5 m/s
Observations sampling time	0.2 secs
Observations: max. range	35 metres
Observations: range σ	0.1 metres
Observations: bearing σ	1°

Table 5.1: Sensors parameters set for the simulation.

5.7 Experimental Results

Figure 5.10 shows the experimental environment used to test the algorithm. The green line denotes the trajectory reported by the GPS, and the light points depict a laser image of the environment.

The segmentation criterion used is based on distance. When a new frame is received, the segmentation algorithm searches for segments of the scan that have a minimum number of neighbour points. The minimum number of points for a segment was set to 100 points, and for two consecutive points to be considered neighbours they need to have less than 5 metres difference in range and 3 degrees in angle. Once a segment has been detected, a procedure is used to predict the level of accuracy in the alignment with that particular segment and decide whether the segment is accepted as a new landmark (see Section 5.2.2).



(a)

(b)

Figure 5.8: The top figure shows the simulation environment. The solid line depicts the ground truth for the robot pose and the dashed line the estimated vehicle path. The actual object positions are represented by the light solid line and the segment positions by the dark points. The ellipses indicate the 3σ uncertainty bound of each scan landmark. The bottom figure shows the simulation result after closing the loop.

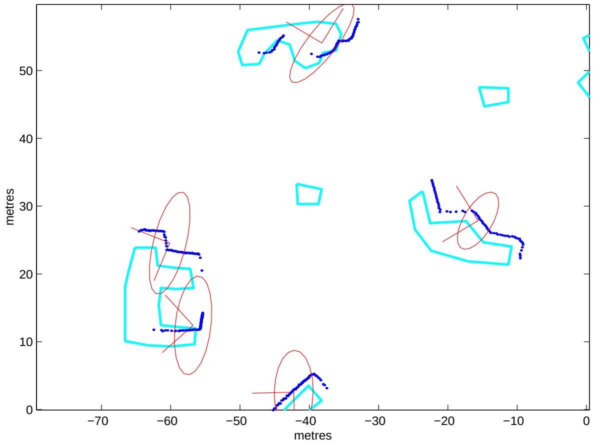


Figure 5.9: Zoom of the left top area of the simulation environment. The figure shows the actual object poses and the 3σ ellipses for the local axis representing the scan landmarks.

The procedure consists in adding noise to the observations, adding a random translation and rotation to the segment and registering the new segment with the original. Segments in which alignment error is less than 10% of the original displacement are accepted as new landmarks.

Figure 5.10 shows the segments accepted with dark points. The dark solid line denotes the estimated vehicle trajectory. The red ellipses represent the 1σ covariance bounds. Six landmarks were incorporated by the algorithm. Figure 5.11 shows the landmark map, the local axes for each landmark are also shown in the figure.

Figure 5.12 illustrates an alignment yielded by the scan matching algorithm. The circles denote the landmark template, and the sensor scan that is being aligned with the landmark is represented by points. Landmarks that are in the sensor view are aligned against the whole new scan. A landmark is considered to be in the sensor view if more than 90% of the points that define the template are inside the sensor view. After the observations are obtained with the scan matching algorithm, a gate validation test is applied. This test decides whether the observation obtained with the alignment is accepted or not.

As can be seen, the results are not as accurate as those obtained using feature-based SLAM. The main reason is that sometimes the segments do not have enough information in order to make a good match; the segments obtained here do not possess as much shape as the segments from the simulation. However, as can be observed from the results, the landmark covariances are consistent with the errors, indicating that there is still remaining uncertainty in the map. The results can be improved if more information is added into the landmarks template, facilitating a better scan alignment.

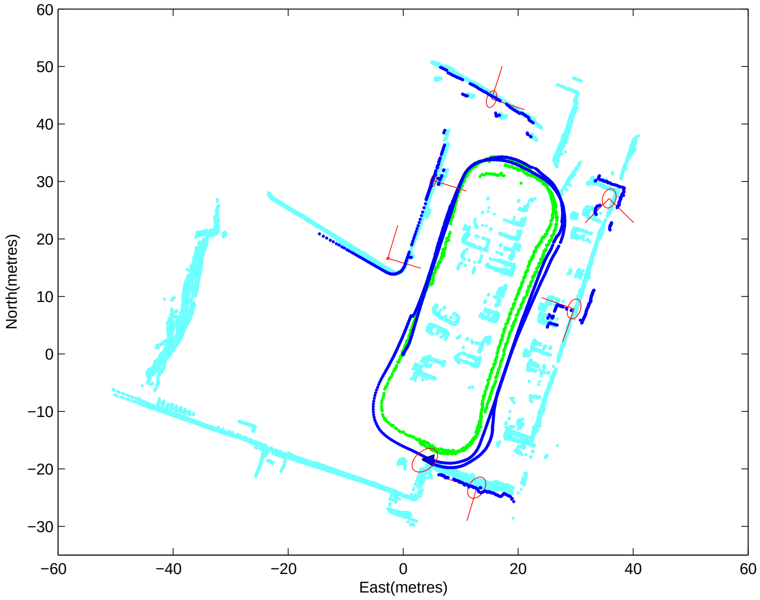


Figure 5.10: Scan-SLAM result obtained in the car park area. The solid line denotes the trajectory estimated. The light points represent a laser-image obtained using feature-based SLAM and GPS. The dark points represent the template scans and the ellipses the 1σ covariance bound.

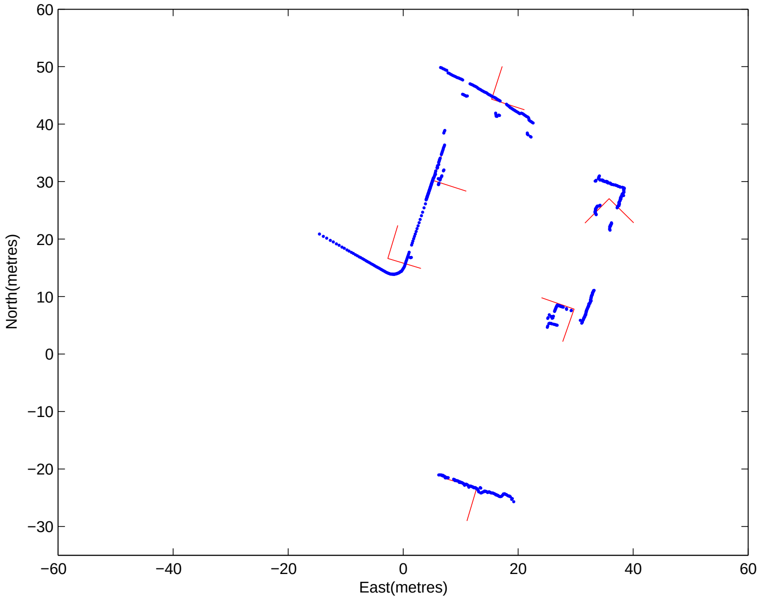


Figure 5.11: Scan-landmarks map. The points indicate the landmarks template. The local axes for each landmark are shown with red lines.

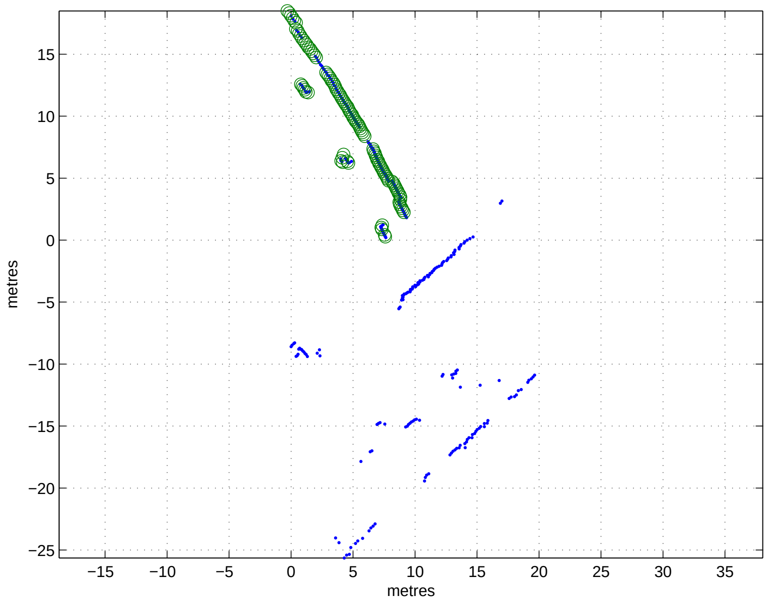


Figure 5.12: Scan alignment example: the green ‘o’ depict the landmark template and the blue ‘.’ the observations.

5.8 Summary

EKF-SLAM is currently the most popular filter used to solve stochastic SLAM. An important issue with EKF-SLAM is that it requires sensory information to be modelled as geometric shapes and the information that does not fit in any of the geometric models is usually rejected. On the other hand, scan correlation methods use raw data and are not restricted to geometric models. Scan correlation methods have mainly been used for localisation given an a priori map. Some algorithms that perform scan correlation based SLAM have appeared [31], but they do not perform data fusion and they require storage of a history of raw scans.

The Scan-SLAM algorithm presented in this chapter combines scan correlation with EKF-SLAM. The hybrid approach uses the best of both methods; it incorporates raw data into the map representation and so does not require geometric models. In addition the algorithm is able to estimate the map in a recursive manner without the need to store the scan history. The main advantage of the algorithm presented, in comparison with other scan correlation based SLAM methods, is that it works as an EKF-SLAM. It is actually an EKF-SLAM that uses raw data as landmarks and utilises scan correlation algorithms to produce landmark observations.

Simulation results were presented which allowed the filter performance to be analysed by comparing the result with the ground truth. Finally experimental results were presented.

Chapter 3 presented an algorithm that allows the integration of dense mapping techniques with EKF-SLAM. This chapter presented a method that uses scan correlation techniques to obtain, from templates of raw scan data, observations that fit into the state-vector representation and which are then suitable for EKF-SLAM. The scan matching algorithm could converge to a local minimum when the vehicle uncertainty is large. In particular, when the observed scan contains more information than the scan-landmark, the template could match the wrong part of the observations. A solution to this problem will be to include more environment information into the scan-landmarks, thus reducing the possibilities of converging to a local minimum. The next chapter explains how to integrate Scan-SLAM with DenseSLAM. This combination will yield a more robust SLAM, not restricted to landmarks that fit into a predetermined geometric model. Having the dense information obtained by DenseSLAM available, will then allow a more robust observation model.

Chapter 6

SLAM in outdoor environments

As shown previously, DenseSLAM is an algorithm that permits the fusion of all the sensory data into the environment representation, thus obtaining a detailed description of the robot's surroundings.

This chapter shows how this detailed multi-dimensional description of the environment can be used to improve the vehicle navigation process. For example, the information can be used to assist the data association process. The vehicle localisation could also be improved by the extraction and incorporation of complex landmarks as they become identified using the dense representation.

The chapter is divided essentially into three parts. Section 6.1 explains how the detailed representation can be used to extract features that cannot be identified from one vantage point.

Scan-SLAM can converge to a local minimum if the vehicle pose uncertainty is large (i.e. the initial estimate of the scans possesses large errors). The probabilities of converging into a local minimum will be reduced if more information is added to the template scans. Section 6.2 shows how the dense maps can assist the scan matching process adding more information to the landmarks template.

Section 6.3 investigates the possibility of using the dense maps to estimate variations in time of the areas explored by the robot and then being able to discriminate whether a region has potential dynamic objects (e.g. a car park). This information will help to prevent the vehicle from using dynamic objects for the localisation process. In addition, this information will allow DenseSLAM to create a map layer of *dynamic areas*, which can be used for different tasks. For example, the map could be used in rescue to detect areas with motion.

Section 6.4 discusses about data association and finally, Section 6.5 presents experimental results.

6.1 High Level Landmarks (HLLs)

One of the main problems in SLAM algorithms is the error accumulation due to non-linearities in the system. This error accumulation can be reduced if more information is added into the localisation map, since the vehicle error will remain smaller. Among the reasons to avoid including more landmarks is the computational burden required to maintain the map. However, in many situations, even if the computational cost would not be a problem, the difficulties of finding *stable and easily detectable* features cause the algorithm to use a small number of landmarks for the localisation process, which results in a major accumulation of errors due to non-linearities.

DenseSLAM yields a rich environment representation, which gives the possibility of adding landmarks extracted from the dense maps into the landmarks map. In many situations an object cannot be detected using the measurements taken from only one vantage point. This can be due to a variety of reasons: occlusion between objects, the size of the object in relation to the sensor field of view, an inappropriate feature model, or just because the nature of the sensor makes the estimation of the landmark location impossible from only one vantage point (e.g. wide-beam sonar; [52, 45]). Estimating partially observable features has been an important research topic in computer vision using stereo vision and bearing only information, where the initialisation of the feature position is a significant problem. The problem of partially observable features has also been studied for localisation and SLAM applications. In [45] an approach is presented that delays the decision to incorporate the observations as map landmarks. Consistent estimation is achieved by adding the past vehicle positions to the state vector and combining the observations from multiple points of view until there is enough information to validate a feature. In [52] intersection of range of constant depth of ultrasonic sensors is used to determine the location of features from multiple vantage points.

Having a comprehensive representation of the environment will enable a delayed processing to determine whether part of the map can qualify as a landmark. The rich representation obtained by DenseSLAM, will enable postprocessing capabilities to continuously detect high-level landmarks using the dense map layers. The newly detected landmarks can then be added to the feature map. This approach has the potential of incorporating a large number of landmark models, some of them to be applied online at the time the observations are taken and the rest to run in the background when computer resources become available. The landmarks can then be incorporated in a consistent manner into the features map.

6.1.1 Initialisation: incorporation of dense information into the features map

This section explains how the HLLs extracted from the dense map can be added to the landmark map.

An algorithm searching for high level landmarks (HLL) can be run in the background. As explained before, the HLLs will be objects with particular features that cannot be extracted from only one vantage point, requiring some data processing before they are identified.

Consider the environment presented in Chapter 3 (see Section 3.10). Since the environment is mainly dominated by buildings, a good feature to use as a landmark for the robot localisation, will be corners for example. However, it will not always be possible to identify the corners from only one vehicle's viewpoint. Hence, a corner detector algorithm could be running in the background of the main mapping algorithm, and the HLLs (in this case corners) could be identified and extracted from the dense map. Figure 6.1 shows the result after running a Harris corner detector in the OG map. The red crosses represent the position of the corners identified by the corners detector algorithm. Figure (b) shows a zoom of the upper part of the environment. This example illustrates how a dense representation facilitates the extraction of particular features which, in many cases, cannot be identified straight away.

When a HLL is found and accepted, it will be incorporated in the landmark map. Hence, the HLL has to be represented in a state-vector form. The next subsection explains possible representations for the HLLs. Once the HLL is represented it will remain as a passive landmark until the vehicle actually observes it again. At that moment, the HLL will be initiliased as shown in Section 2.3.4, using the new observations.

6.1.2 HLL representation

The only condition for the incorporation of a HLL is to represent the information in EKF form. An option will be to use a representation similiar to the one described in Section 5.2.3, where the landmarks are represented as a local coordinate system and a template defined relative to the local axis. Another form of representing the HLLs could be by using geometric parameters. Experimental results of SLAM using trees as landmarks are presented in [26]. An EKF is run which estimates the trees' parameters, which consist of the centre and the diameter of the trees' trunk. In [80] an algorithm is presented that employs the expectation maximization algorithm to fit a low-complexity planar model to 3D data collected by range finders and a panoramic camera. After this model is obtained, its parameters could be added to the state vector to represent a HLL.

Since the HLLs are extracted from the dense map, the representation used for the

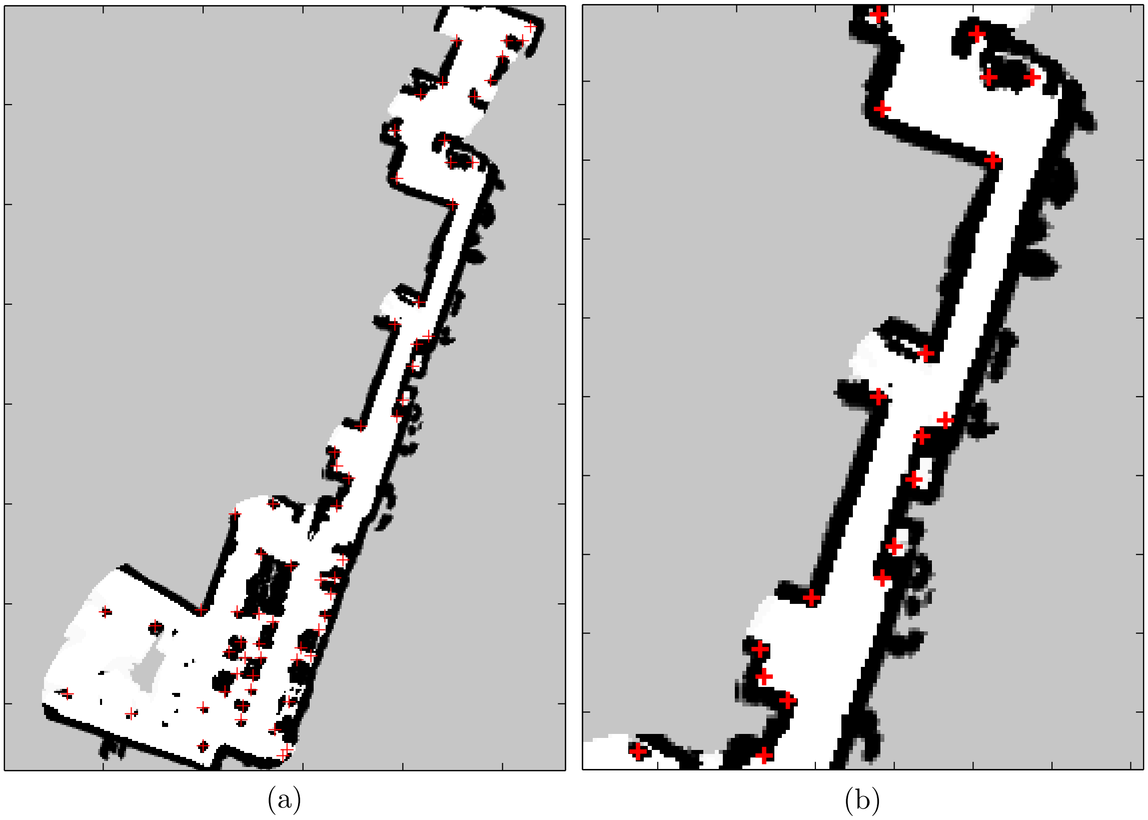
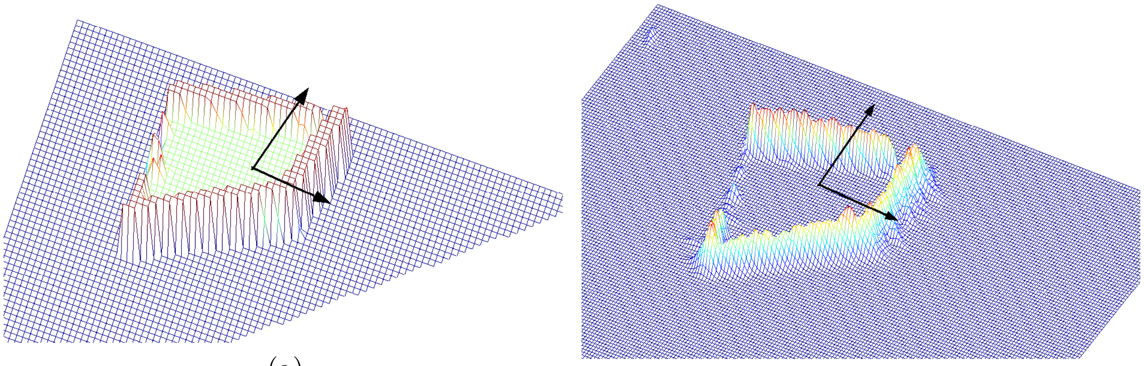


Figure 6.1: The figure shows the corners detected by a Harris corner detector algorithm in the OG map. (a) shows the whole OG map obtained. The red '+' depict the corner positions. (b) shows a zoom of the upper part of the run.



(a)

(b)

Figure 6.2: The figure shows a HLL extracted from the dense maps. (a) shows the HLL represented using the OG map and (b) the SOG map.

template will depend on the representation used for the dense map. As shown in Chapter 4, DenseSLAM is not restricted to any particular representation, furthermore, different layers using different representations can be obtained allowing different representations to be used for the landmark template. Figure 6.2 shows a HLL extracted from the dense map, (a) shows the OG-HLL and (b) the SOG-HLL. The figure also shows the local axes whose pose will be included in the state vector. The next section presents an implementation of DenseSLAM including HLLs.

6.1.3 Simulation

The environment presented in Chapter 4 is used in this section to show results of DenseSLAM using high level landmarks. Figure 4.2 shows the environment which consists of point feature landmarks and objects of diverse geometric shapes.

In this experiment, DenseSLAM was run yielding two dense maps, an occupancy grid and a sum of Gaussians map. The HLLs are represented as templates of SOGs relative to a local axis. Then the state vector will be formed by the vehicle pose, the point feature landmarks and the HLLs which are represented as the scan landmarks shown in Chapter 5.

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{x}}_f \\ \hat{\mathbf{x}}_{hl} \end{bmatrix} \tag{6.1}$$

where $\hat{\mathbf{x}}_v$ represents the vehicle states, $\hat{\mathbf{x}}_f$ the features map and $\hat{\mathbf{x}}_{hl}$ the HLLs map.

$$\hat{\mathbf{x}}_v = [\hat{x}_v, \hat{y}_v, \hat{\phi}_v]^T \tag{6.2}$$

$$\hat{\mathbf{x}}_f = [\hat{x}_{L_1}, \hat{y}_{L_1}, \ldots, \hat{x}_{L_i}, \hat{y}_{L_i}, \ldots, \hat{x}_{L_n}, \hat{y}_{L_n}]^T \tag{6.3}$$

$$\hat{\mathbf{x}}_{hl} = [\hat{x}_{hl_1}, \hat{y}_{hl_1}, \hat{\phi}_{hl_1}, \ldots, \hat{x}_{hl_i}, \hat{y}_{hl_i}, \hat{\phi}_{hl_i}, \ldots, \hat{x}_{hl_n}, \hat{y}_{hl_n}, \hat{\phi}_{hl_n}]^T \tag{6.4}$$

The OG layer was used to detect the high level landmarks and the SOG map, to obtain SOG templates to be used for the HLLs representation.

In the implementation presented, for a part of the map to qualify as a HLL, it needs to have two corners and the corners need to be closer than a predefined distance. A template area is delimited around the two corners, and if the dense map under the template area possesses more than a minimum number of Gaussians, the template is accepted as a HLL. Figure 6.4 presents one example. Assume the points represent the SOG map obtained with DenseSLAM. As explained, the first stage in finding HLLs is to detect corners. Four corners are detected in the example shown, and they are pairwise associated by using a distance criterion. After a pair of corners is found, a rectangular region is drawn around the pair of corners which delimits the HLL template. Figure 6.4 shows two potential HLLs. In order to qualify as a HLL, the template needs to have a minimum number of points under the template area (rectangle). In the example shown, only the first template qualified as a HLL since the second template does not have enough points.

The detection of HLLs is done by a function running in the background. In the implementation presented here, this function is running every one minute. The function searches first for corners in the environment using the OG map obtained with DenseSLAM. Once a HLL is detected, it will be initialised with the next observation as in standard feature-based SLAM (see Section 2.3.4). The algorithm only searches for HLL in the local areas around the vehicle position, which are the areas the vehicle could visit next. This will make the computational cost of the HLLs search independent of the global map size. Once all the corners are detected, they are associated pairwise with the nearest corner. In the implementation, the minimum distance required to associate two corners was set to 12 metres. A rectangular area is defined around the corners to delimit the template area. The bounds were set to 5 metres from the corners' position. Next, the template bounds information is used together with the SOG map to analyse which template qualified as a HLL. A minimum of 200 points (Gaussians) are required by the implementation to accept a HLL. This is to make the HLLs richer and facilitate the observation of the landmark templates. The SOG template is represented in a local coordinates system as explained in Section 5.2.3 and SOG correlation is used to generate the HLL observations.

Figure 6.5 (a) illustrates the result after the function that searches for HLLs is run for the first time. The circles denote the corners detected by the Harris algorithm using the OG layer. The rectangles depict the bounds of the potential template landmarks. There are three possible HLLs detected. After the number of points under the template area are examined only one is accepted which is depicted with a solid line. The other two are rejected. Figure (b) shows the result after the vehicle explored the whole area and closed the

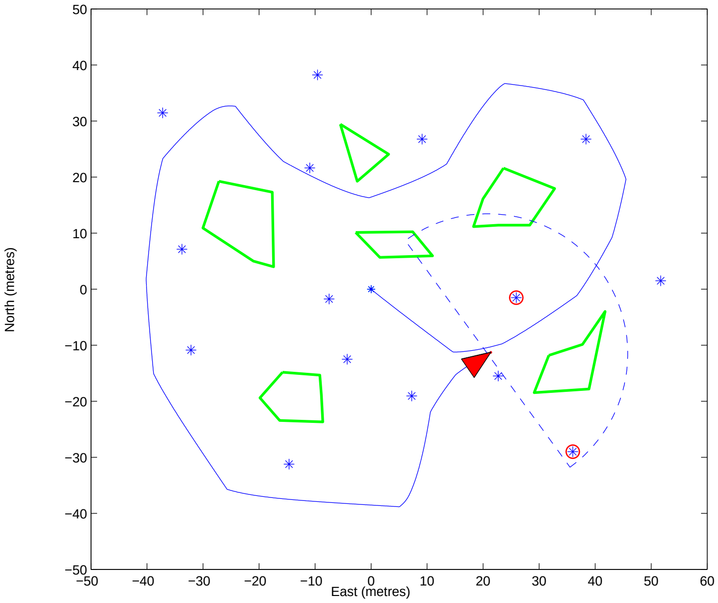


Figure 6.3: Simulation environment.The '*' are the actual landmarks. The solid line represents the vehicle trajectory, and the enclosed contours depict the actual objects' positions.

loop for the first time. Four more HLLs have been detected and incorporated. These HLLs are then incorporated into the state vector and are used for the vehicle localisation process. Scan matching was used to generate observations and update the HLLs position. The axes inside the rectangles denote the local coordinate axis position. Each axis is represented by its $[x, y]$ coordinates and orientation ϕ. The ellipses represent the HLL 1σ uncertainty bounds.

Figure 6.6 illustrates the final HLLs map. In addition to the point feature landmarks, the dense information selected in the templates is also used for the vehicle navigation process, making the vehicle localisation much more robust.

6.2 Combining Scan-SLAM with DenseSLAM

As mentioned in Chapter 5, the main objective of Scan-SLAM is to avoid the dependency on geometric features for the landmark representation. Thus, the algorithm uses templates composed of raw sensed data as landmarks. When a new sensor frame is received, a segmentation procedure searches for possible segments to be used as landmarks. The new segments are then represented in a local coordinate system and the local coordinate pose is incorporated into the state vector.

Using segments of points extracted from only one sensor frame can make the scan matching algorithm fragile, especially if the vehicle pose uncertainty is large (i.e. the initial

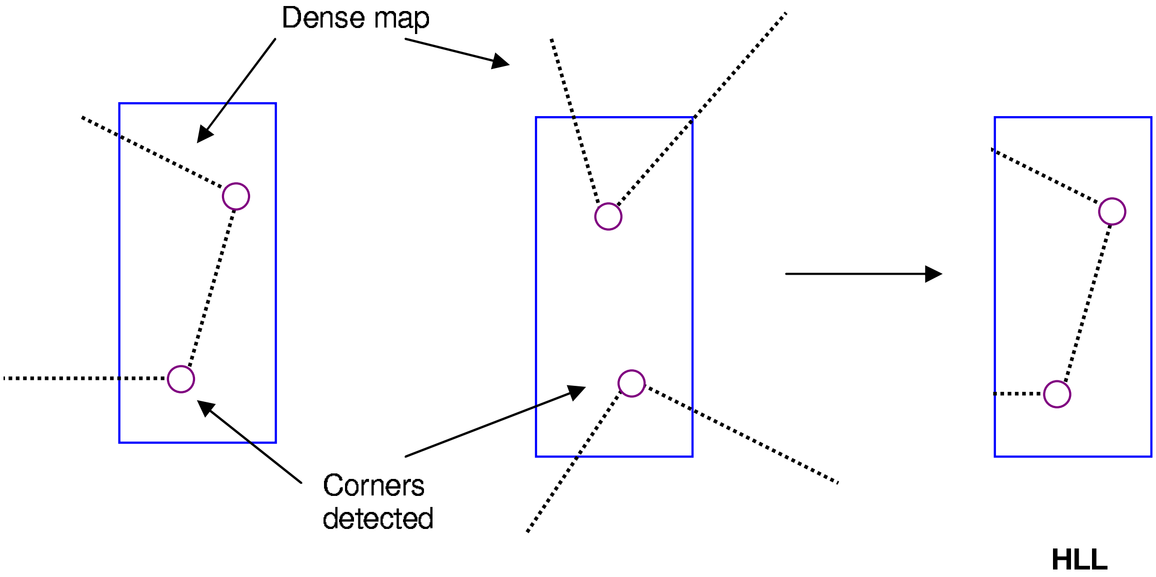


Figure 6.4: The HLLs consist of rectangular areas of the map which possess two corners and have a minimum number of Gaussians. This last property is checked using the SOG map. The figure shows two examples of HLLs detected. The 'o' depict the corners detected and the '.' the Gaussians position. From the example, the second HLL is not accepted since it does not have enough Gaussians.

estimate of the scan possesses large error). Any scan matching algorithm could converge to a local minimum if a good initial pose is not provided and the templates' landmarks do not have enough information to promote reliable convergence. The probability of converging to a local minimum can be reduced if more information is used in the landmark templates.

Scan-SLAM can benefit from the complementary characteristics of DenseSLAM. On the one hand DenseSLAM is able to obtain a dense representation, but it is based on a landmark map which is used to define the local regions. On the other hand, Scan-SLAM can provide non-geometric based landmarks, but in order to make the scan matching process robust, it needs to add more information to the template landmarks when the vehicle pose uncertainty becomes large.

The combination of the two algorithms can be done as follows. When the vehicle starts to navigate, templates of raw sensed data can be used to define landmarks as explained in Section 5.2.3. These landmarks are then used by DenseSLAM to define local regions and fuse all the sensory information. Thus, the combination will make possible to obtain a dense representation without relying on geometric features for the landmark definitions. The scan matching process using scan segments will be reliable while the local vehicle uncertainty remains small. Therefore, the segments can be used to build a local region. However, if the vehicle pose uncertainty is not highly correlated with the scan landmarks that are being

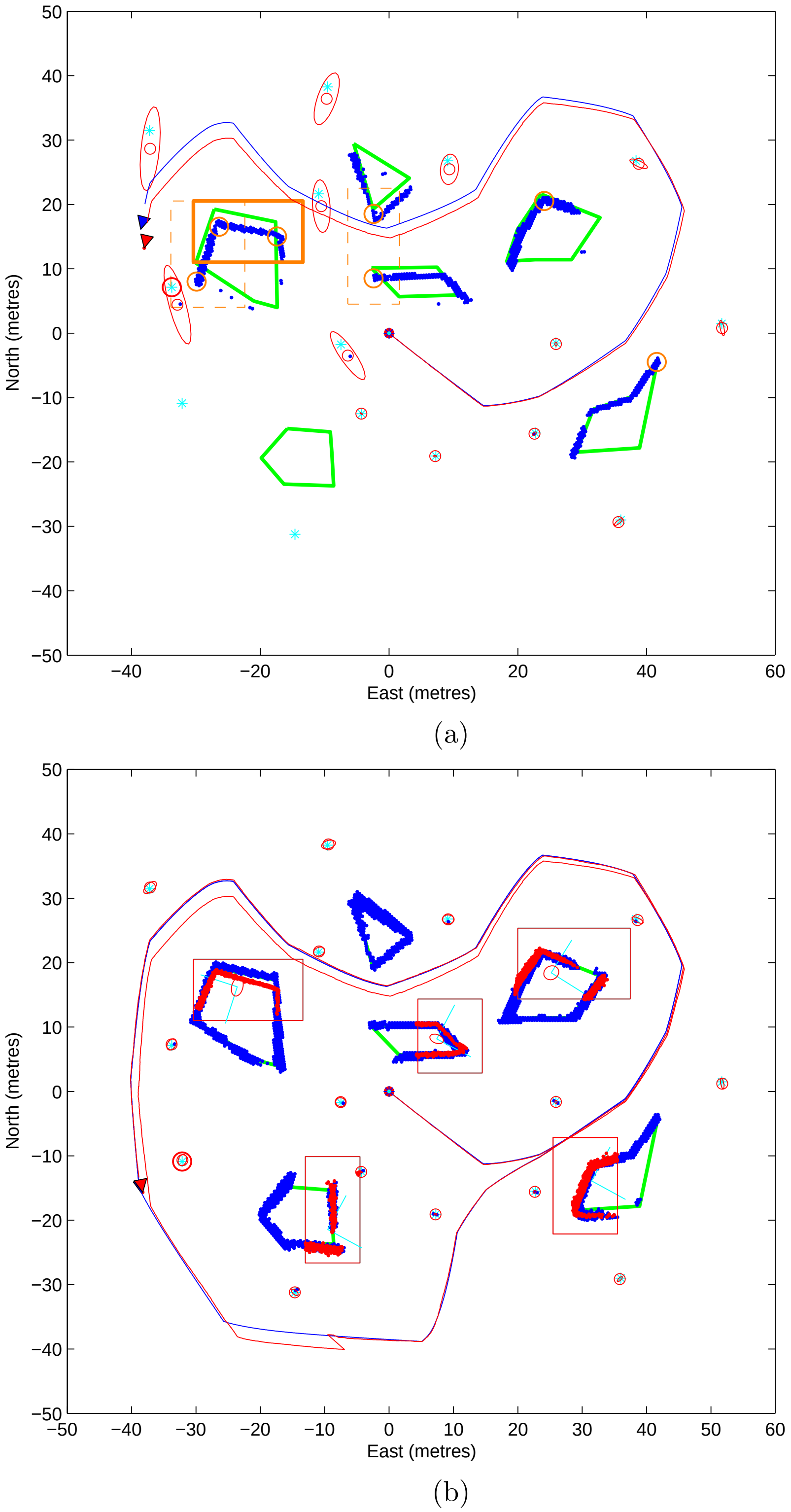


Figure 6.5: Result obtained with DenseSLAM using HLLs. Figure (a) shows three HLLs detected by a HLL algorithm running in the background. The rectangles depict the area to be taken as a HLL. Only the HLL represented with solid line rectangle is accepted. (b) shows the final result where five HLLs are incorporated into the landmarks map.

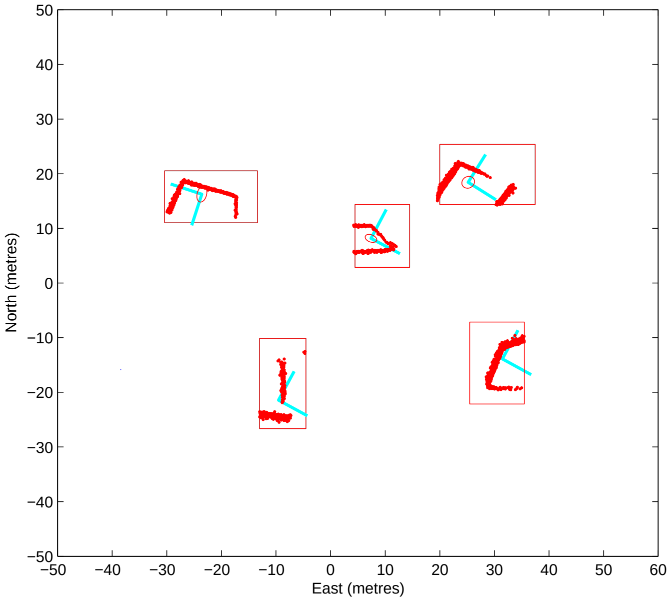


Figure 6.6: The HLLs map obtained. The '.' denote the SOG templates. The local axes are also shown in the figure.

observed, it means that the initial pose used for the scan matching process may not be close enough to ensure global convergence of the algorithm. At that point the information stored in the dense maps can be used to assist the scan matching process by defining more robust landmarks. For example if grid correlation is being used, the information in the occupancy grid map around the template landmarks can be used to define a more informative landmark and so reduce the probabilities of converging to a local minimum.

6.2.1 Simulation

This section shows simulation results of Scan-SLAM combined with DenseSLAM. Figure 6.7 (a) illustrates the environment used for the experiment, which consists of an area of 120 by 100 metres. The solid blue line represents the actual vehicle trajectory and the dashed red line the estimated. The environment is formed by objects with different geometries. There are no point feature landmarks in the environment.

The algorithm implemented uses the scan landmarks defined by Scan-SLAM to make a partition of the environment into local triangular regions, in the same way as was done before using the point feature landmark map. Then, DenseSLAM is run using the LTRs defined by the scan landmarks. Figure 6.7 (b) shows the scan landmarks incorporated by the algorithm. The figure shows with dots the Gaussians mean of the template landmarks and the local axis positions are shown with cyan lines. The ellipses represent the 3σ uncertainty bounds. Figure (c) and (d) show the two dense maps obtained. Figure (c) illustrates the

OG layer, where the points represent the cells with $P(Occ) > 0.8$, and (d) shows the SOG map, where the red ellipses represent the Gaussians uncertainty bounds.

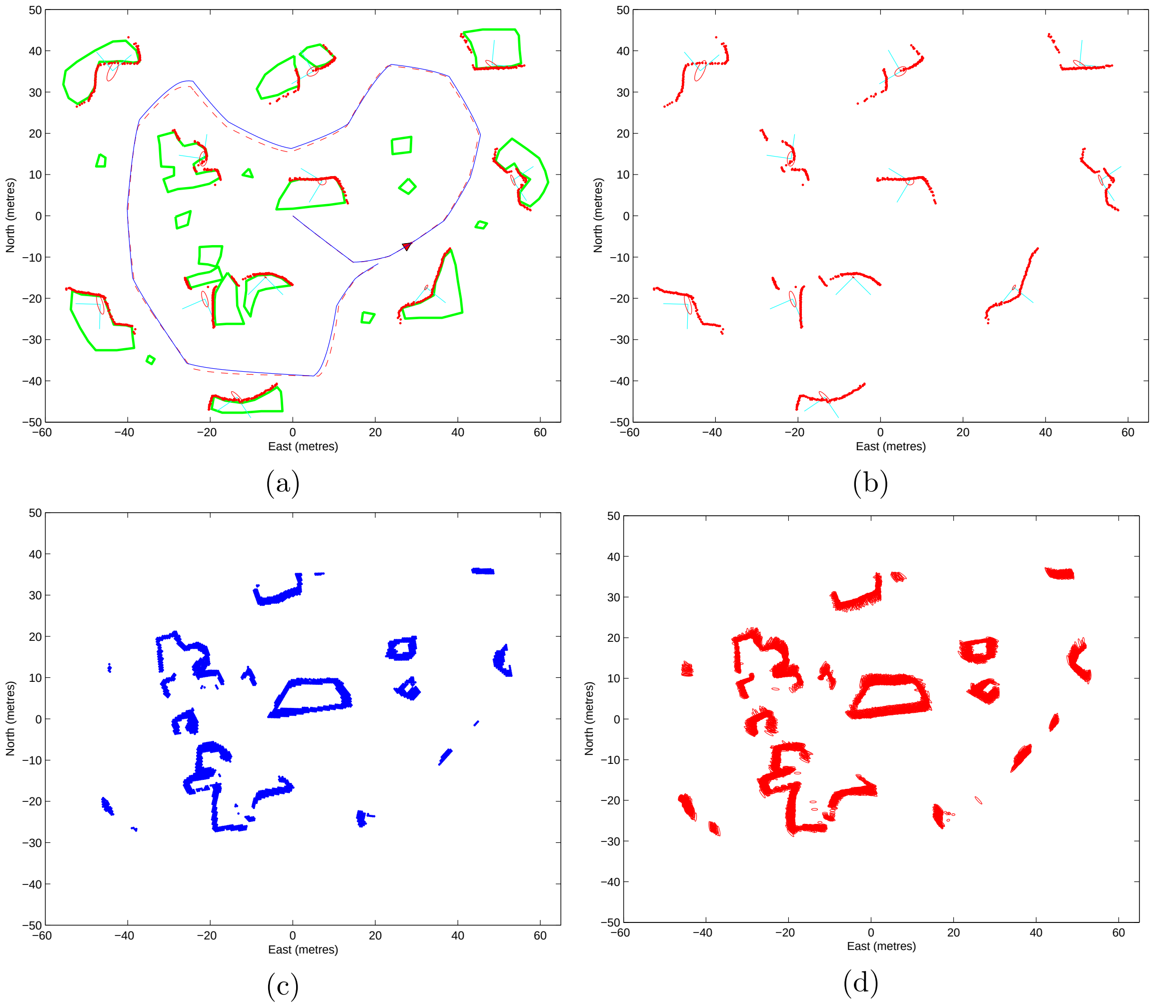
It can be seen in Figures (c) and (d) that the dense maps have much more information than simply the landmarks' templates by themselves. This information can be used by the algorithm to assist the scan matching process. Once the dense maps have been obtained, when a scan landmark is associated with a new sensor scan, the dense map surrounding the scan landmark is recovered and is added to the landmark template. The richer landmark is then used by the scan matching algorithm. Figure 6.8 shows a zoom of one scan landmark and the SOG map obtained by DenseSLAM. Figure 6.8 (a) shows the scan landmark template and the local axis. The actual object positions are represented by the green solid line. Figure 6.8 (b) shows the scan landmark, denoted by red points and the SOG map used to augment the landmark template, which is represented by blue points. The rectangle represents the bounds of the dense map area used to augment the landmark template.

6.3 Dynamic Environments

Most of the mapping algorithms assume the world is static [77]. Dynamic environments require an extension of the typical representation used for static environments, which should allow for modelling of temporal evolution of the environment. Dynamic objects can induce serious errors in the robot localisation process. Only a few approaches that include moving objects have been presented so far. The next paragraphs review some of them.

A SLAM algorithm with generic objects (static and dynamic) was presented in [83]. Similar to classic SLAM, the approach calculates the joint posterior over robot and object's pose, but unlike traditional SLAM it includes also the objects' motion model. However the problem is shown to be computationally intractable and so a simplified version called *SLAM with Detection and Tracking of Moving Objects* (SLAM with DATMO) [84] is presented. The latest algorithm decomposes the estimation process into two separate problems: (i) the SLAM problem, using static landmarks as the classic approach, and (ii) the detection and tracking of moving objects, using the robot pose estimated by the SLAM algorithm. This simplification makes updating both the SLAM and the tracking algorithm possible in real-time since they are now considered two independent filters.

In [34] an algorithm for mapping in dynamic environments is presented. The aim of the approach is to determine which measurements correspond to dynamic objects and then filter them out for the mapping process. The approach uses the EM algorithm; the expectation step computes an estimate about which measurements might correspond to static objects. These estimates are then used in the maximization step to determine the position of the robot and the map.



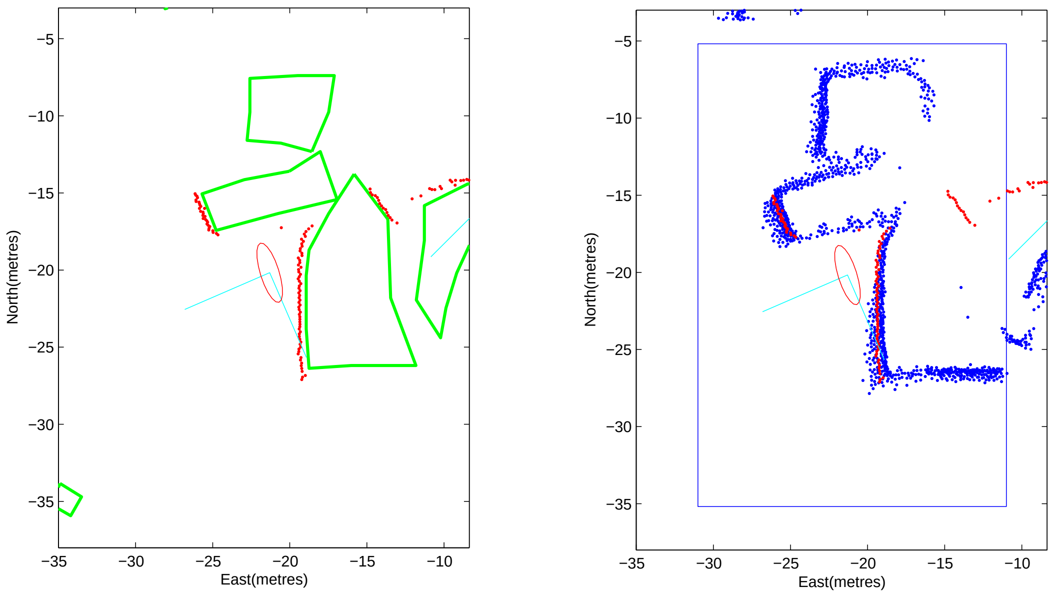
(a)

(b)

(c)

(d)

Figure 6.7: Result obtained using Scan-SLAM combined with DenseSLAM. (a) shows the simulation environment and the result after the vehicle has closed the loop. The local axes represent the scan landmark positions. (b) illustrates the scan landmarks map. (c) shows the OG map obtained. The dots represent the cells which $P(C_i = Occ) > 0.8$. (d) shows the SOG map. The ellipses represent the 1σ bound for the Gaussians.



(a) (b)

Figure 6.8: Zoom of one scan landmark and the SOG map obtained by DenseSLAM. Figure (a) shows the scan landmark template and the local axis. The actual object position is represented by the green solid line. Figure (b) shows the scan landmark, denotes by red points and the SOG map used to augment the landmark template, which is represented by blue points. The rectangle represents the bounds of the dense map area used to augment the landmark template.

An approach called *Robot Object Mapping Algorithm* (ROMA) was presented in [4]. The main goal is to identify non-stationary objects and model their time varying locations. The approach assumes that objects move sufficiently slowly that they can safely be assumed static for the time it takes to build an occupancy grid map of the whole area explored by the robot. Assuming the robot is able to acquire static occupancy grid maps at different times, changes in the environment are detected using a differencing technique. The algorithm learns models of the objects using EM. The expectation step calculates the correspondences between objects at different points in time and the maximisation step uses these correspondences to generate refined object models, represented by occupancy grid maps.

Finally, algorithms for SLAM and people tracking have also been presented [55].

The algorithms presented in [84] and [55] have one thing in common, they rely on predefined models of the specific objects they aim to track. ROMA, however, is able to learn about the shape of the objects, but the algorithm presents a number of limitations. Objects have to move slowly (it is not able to cope with fast-moving objects such as people), it is assumed the robot is able to obtain static maps at different times. The results presented include only four different objects in an environment where these objects can be perfectly segmented from a laser scan. The extension to a real environment with a larger number of objects may not be possible and will be computationally very expensive.

If navigation is the primary objective, the accurate shape of objects, or even their classification may not be important in general. What may be more useful is an algorithm such as the one presented in [34] that is able to identify observations that may be coming from objects that are no static and eliminate them from the list of observations to be used for the navigation process. Furthermore the algorithm could identify areas where it is more likely to find dynamic objects (e.g. a corridor where people walk) and then avoid their use or give a low priority to observations coming from objects in those areas.

The rich environment representation obtained by DenseSLAM allows a map layer identifying the most likely areas to possess dynamic objects to be built. As shown in [4], dynamic objects can be identified by differentiation of maps taken at different times. There are two different classes of object motions in a dynamic environment; slow motion, as for example the motion of a bin, which will be static during most of the day but will eventually be moved; and fast motion, such as people. The next subsections show how using DenseSLAM, and applying a straightforward differentiation, makes it possible to identify regions with dynamic objects for either fast or slow motion objects.

6.3.1 Slow Motion

One of the main problems with the differentiation is that maps obtained at different times will have different uncertainty. If only one global map is maintained and two maps acquired at different times want to be differentiated, to detect dynamic objects for example, since the uncertainty in the maps will be different, the matching process will require ad-hoc techniques. However, in DenseSLAM the global map is divided into smaller regions, so the whole map can be differentiated by applying differentiation between the corresponding local regions. As a consequence, the differentiation process will be prone only to local errors (which as shown in Section 3.6 are much smaller than the global ones) eliminating detection errors due to vehicle uncertainty.

6.3.2 Fast Motion

Fast motion can be captured in a similar way to slow motion, by differentiation. The main difference is that the differentiation is done over shorter periods of time and only in the region under the sensor view. As in the detection of objects with slow motion, using DenseSLAM the differentiation is done using local regions instead of using a global map. The motion detection will be included in a map layer as will the other properties captured by the sensors, then the global position of the dynamic map will be updated together with the other map properties (colour, occupancy, etc.).

6.3.3 Simulation

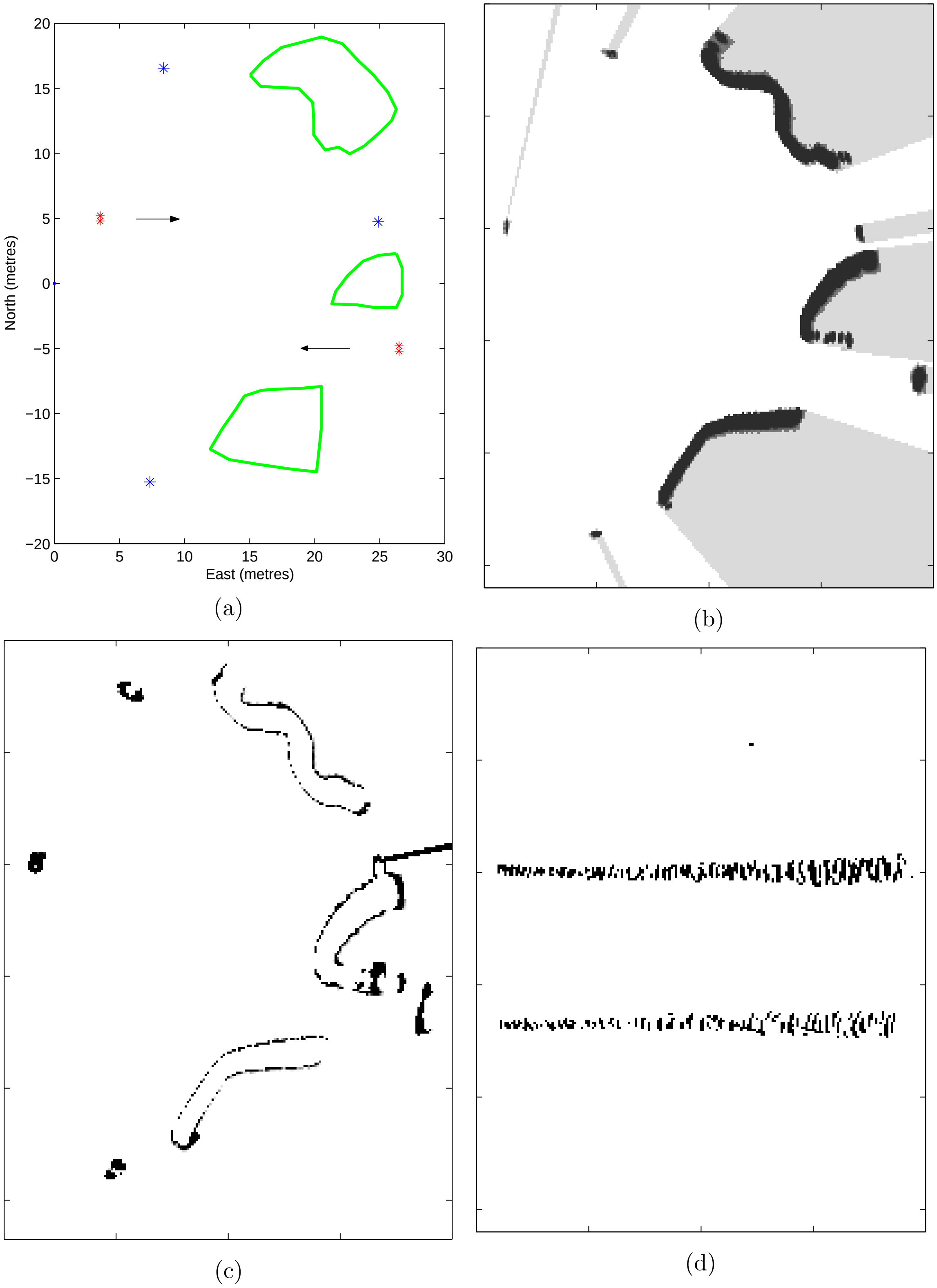
This section shows simulation results of DenseSLAM including the identification of regions with moving objects. DenseSLAM will obtain map layers with information about the regions that are more likely to have moving objects. Two different layers will be obtained, (i) the slow motion detection layer, which is obtained by differentiating the map over long periods of time, and (ii) the fast motion layer, where the differentiation is done over short periods of time.

Figure 6.9 (a) illustrates the simulation environment used to show the fast motion detection using grid maps. The environment consists of point feature landmarks, static objects and dynamic objects. The dynamic objects are denoted with red stars. Two dynamic objects were included, one moving from left to right and the second one moving from right to left (see arrows in figure). Both objects are moving parallel to the east axis. The first one has a north coordinate equal to 5 and the second one equal to -5. The observations are taken from a range and bearing sensor located in the position $(0, 0)$ with 0 degrees in orientation. The sensor takes one frame every 0.1 seconds. The observations are corrupted with Gaussian noise with 0.15 metres and 1 degree of standard deviation. Figure 6.9 (b) shows a snapshot of the occupancy grid representation obtained. The figure shows the OG map obtained at the beginning of the run. The map resolution was 0.2 metres.

As a consequence of the dynamic objects, the grid map will have temporal variations.

The detection of dynamic objects is done by a function running in the background which differentiates maps obtained at different times. For the experiment, the running time was set to 1 second, thus, every one second the OG map is differentiated with the map obtained a second before. Figure (c) shows one of the results obtained by differentiating two maps. As can be seen, not only the dynamic objects are included in the result, but also the contours of the static objects. This effect is due to the sensor noise. In order to eliminate this effect, a Gaussian lowpass filter was run after the differentiation which removes the contours of the static objects. After this, a threshold operation is done, and then cells with difference in occupancy bigger than 0.5 are set as dynamics, and cells with difference smaller than 0.5 remain as statics. After applying the thresholds the result is a binary map with 1 for the dynamic cells and 0 for the static cells. The dynamic map is then fused with the a priori dynamic map simply by applying a logical OR operation. Thus, if one cell was found as dynamic in either the a priori map OR the result from the new differentiation, the cell will be defined as dynamic, and only the cells found as statics in the a priori AND the new map remain as static. Figure (d) shows the final dynamic map result. The black colour denotes the cells where variation in the occupancy was detected and white means cells with no variation in their occupancy. The result shows two dynamic areas detected, which clearly coincide with the movement of the dynamic objects. The good results obtained provide evidence of the usefulness of having a dense representation of the environment when it is necessary to detect areas with temporal variations.

Figure 6.10 shows the environment utilised to demonstrate the slow motion detection. DenseSLAM was run in order to obtain an OG map of the environment. The algorithm started to run using the environment shown in Figure 6.10 (a). After 100 seconds the environment was modified as shown in Figure 6.10 (b). One of the objects was removed (object in the middle), and two new objects were added, one on the right and one on the left. DenseSLAM continued running and an updated OG map of the new environment is obtained. In order to detect the changes in the environment, a function running in background differentiated the maps obtained every 150 seconds. The differentiation is done in the same way as explained before for the fast motion detection algorithm; the map is differentiated, a lowpass filter is applied and then the result is converted to a binary map using thresholds. Figure 6.10 (c) illustrates the slow motion detection map obtained. As seen, three areas were detected as dynamic, which are the area where one of the objects was removed, and the two areas where new objects were added. The results show the robustness of the slow motion detection algorithm.



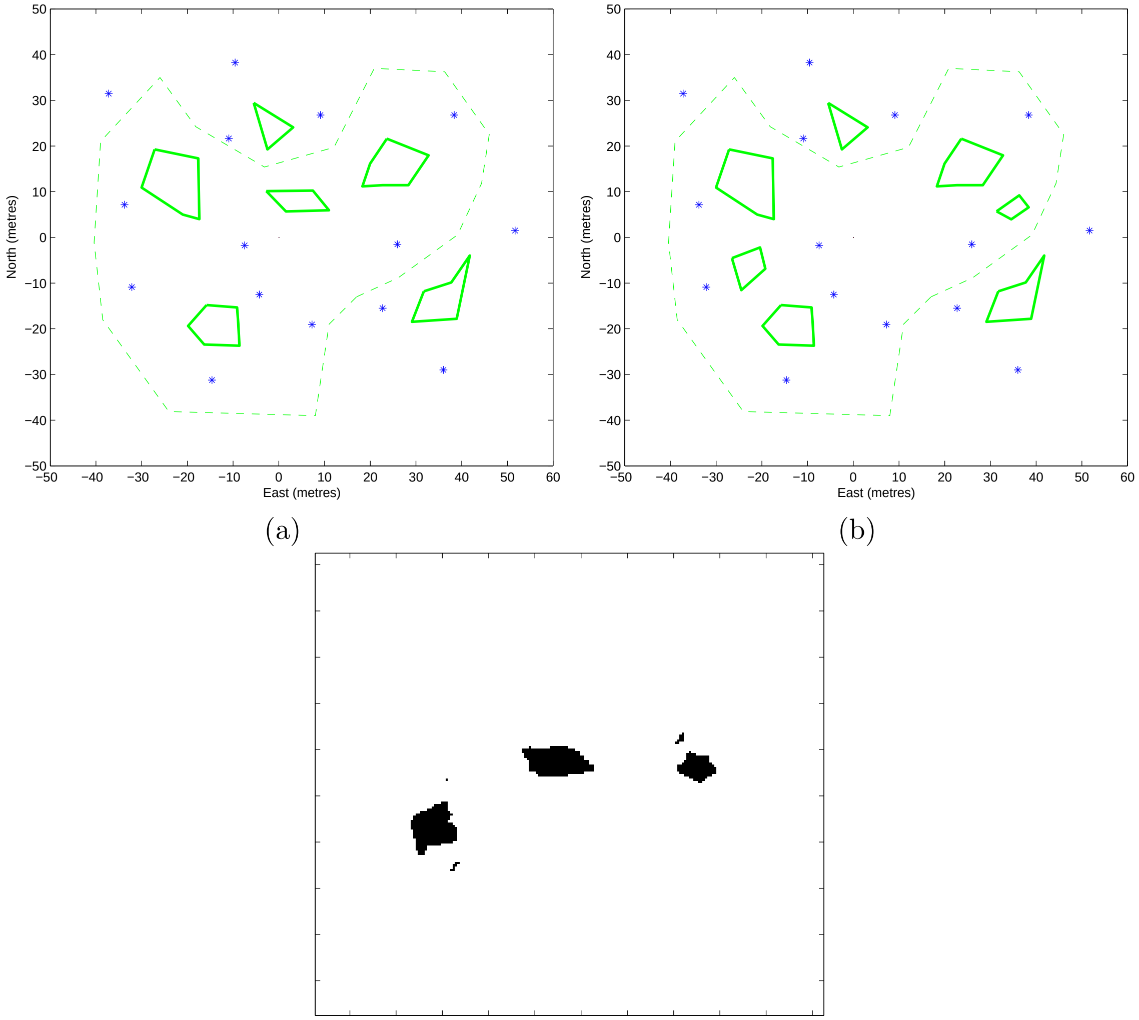
(a)

(b)

(c)

(d)

Figure 6.9: Fast motion detection algorithm. (a) illustrates the testing environment which is composed of static and dynamic objects. The dynamic objects are depicted by the red '*'. The environment has two dynamic objects, both moving parallel to the east axis but in opposite direction. (b) shows one of the OG maps obtained at the beginning of the run. (c) shows the result of the differentiation of two OG maps before the low pass filter is applied. The contours of the static objects are also detected. (d) shows the final results. The cells detected as dynamic are represented with black.



(a)

(b)

(c)

Figure 6.10: Slow motion detection algorithm. (a) and (b) show the testing environment at two different times. In (b), two objects have been added and one object has been removed. (c) shows the result from the motion detection algorithm. Three spots have been identified as dynamics.

6.4 Data Association

Data association is one of the hardest problems in SLAM. The main reason is the poor environment representation commonly used by SLAM algorithms [64]. Classic SLAM algorithms have relied on simple geometric features to represent the environment while rejecting the rest of the sensor information. This simple representation makes the landmarks able to be distinguished only by their positions, restricting the association algorithm to the use of only metric information to match the new measurements with the correct states [1, 58]. The poor information inherent in the classic feature representation makes the data association problem difficult when objects are reobserved, in particular when closing loops and big uncertainty in the robot pose is accumulated.

Data association failures can be drastically reduced if more information is added into the landmark representation. Consider a scenario where the map is formed by pole-like landmarks of different colours. Assume a vehicle takes a range-bearing observation of one of these poles and a gate validation test gives two possible landmarks to be associated with that measurement. If the association ambiguity cannot be reduced, the measurement will have to be rejected. Consider the same scenario, but the map is now formed by the landmarks position and in addition a colour map is obtained. The ambiguity in association can be eliminated using the colour information and the measurement will not have to be rejected. This is just a simple example of how adding information into the landmarks can reduce ambiguities in data association.

DenseSLAM allows different map layers with different environment properties (occupancy, colour, texture) and different representations of the same sensed data (OG, SOG) to be obtained. The use of only position information to describe the landmark makes the association very fragile, in particular in highly populated environments. If more information is added to the landmark representation, more distinctive landmarks will be obtained, and the map would not be restricted to identical landmarks which are distinguished only by their positions.

6.5 Experimental Results

This section presents experimental results in an outdoor environment. The test was done in the environment presented in Section 3.10, but the results presented here are obtained using a part of that run. The use of a smaller area will allow the concepts introduced in this chapter to be better illustrated. Figure 6.11 shows the environment. The green solid line depicts the vehicle trajectory as reported by the GPS. The blue dots represent a laser image obtained using SLAM in combination with GPS. The initial vehicle pose is in (0,0) which is indicated by a small triangle.

Different representations of the environment were obtained using DenseSLAM. Figure 6.12 shows the map obtained fusing just the raw observations in the local regions. Figure 6.13 illustrates the occupancy map layer obtained by DenseSLAM. The grid resolution is 0.5 metres. Figure 6.14 shows the SOG map layer acquired. The figure shows with red solid lines the ellipses which represent the 1σ bounds for the Gaussians covariance. The SOG map was obtained using a $d_{min} = 0.5$ metres for the nearest neighbour algorithm. Figure 6.15 illustrates the SOG map result. The SOG surface was built using a grid of 0.5 metres. Figure 6.16 illustrates the three map layers obtained.

An algorithm to detect high level landmarks (HLLs) was run using the representations obtained with DenseSLAM. Since the environment is dominated by buildings and cars (recall that it is a car park area), the same algorithm used for the simulation (see Section 6.1.3) which searches for corners is used here. Basically the algorithm first searches for corners using the OG map layer. Figure 6.17 shows the result of the corners detector algorithm. The red crosses represent the position of the identified corners. After the corners are detected, the algorithm searches for pairs of corners whose distance apart is less than 7 metres. Then a rectangle area around the two corners is delimited and the HLL is extracted from the SOG taking the Gaussians under the rectangle. A HLL is accepted only if it possesses a minimum number of Gaussians. Figure 6.18 (a) illustrates the HLLs found. The figure shows the laser image, the corners detected and the HLL bounds are represented by the rectangles. Nine HLLs were accepted by the algorithm. Figure 6.18 (b) shows the HLLs map. The points represent the means of the Gaussians that form the templates for the landmarks. After the HLLs are detected, they are incorporated into the state vector and updated using SOG scan matching as explained in Chapter 5.

The slow motion detection algorithm was also tested. Figure 6.19 (a) and (b) show images of OG map layers obtained in the car park area at different times. The map shown in (b) was obtained after three cars were removed from the car park. The grid cells that changed from occupied to free are depicted with red colour in (a). The slow motion detection was run using these two maps. 6.19 (c) illustrates the result. The black colour denotes cells that have been identified as dynamic. As can be seen, variations have been identified in two different places. The first place on the right is where two cars were parked, and the second spot is where the third car was parked.

The main objective of identifying dynamic regions is to avoid the use of possible landmarks formed by dynamic objects. Then, after the algorithm detects dynamic areas, the result is superimposed with the navigation map and the landmarks that are under one of the dynamic areas are removed from the state vector. Figure 6.20 shows the HLLs map and the dynamic areas detected which are denoted by red points. A HLL represented with a dashed line is clearly under a dynamic detected region, and is therefore removed from the

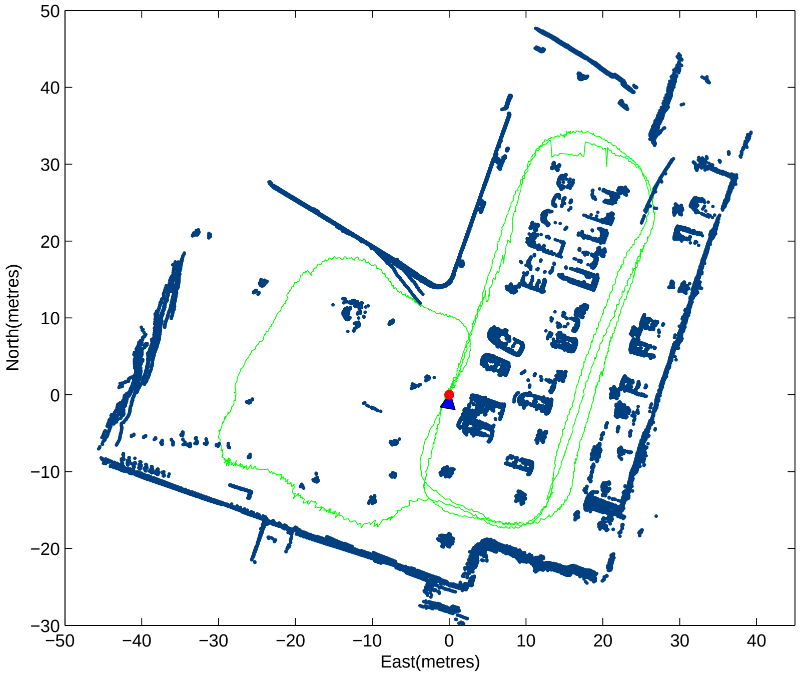


Figure 6.11: Experimental environment. The green solid line depicts the vehicle trajectory as reported by the GPS. The blue dots represent a laser image obtained using SLAM in combination with GPS.

landmark map.

6.6 Summary

This chapter has shown how the representation rendered by DenseSLAM can be used to improve the overall vehicle navigation process.

Outdoor environments require the vehicle navigation process to be able to cope with problems that could make the localisation process fail. This chapter showed how having a dense representation helps to deal with some of these problems. The first issue addressed is the possibility of using the dense maps to detect features that are not possible to detect from one vantage point, and once they are detected incorporating them into the landmark map. Increasing the information used for the vehicle localisation process will reduce the error accumulation due to non-linearities. In addition, a more comprehensive environment representation can eliminate association ambiguity.

The other problem addressed is related to the fact that in real environments there are always dynamic objects. These objects could make the vehicle localisation fail drastically if used as static navigation landmarks. This chapter showed an approach to detect areas where

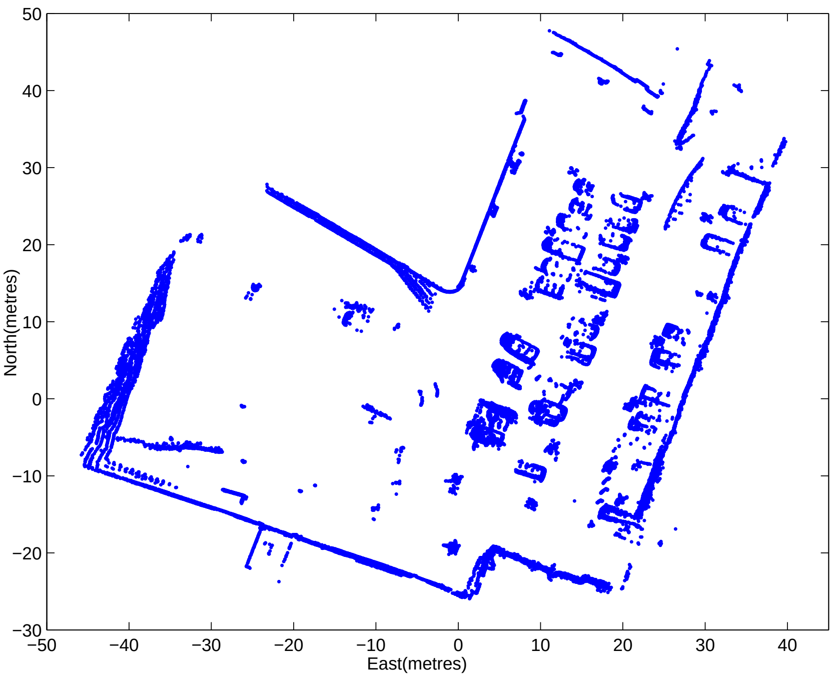


Figure 6.12: Map obtained by DenseSLAM, by fusing the raw observations into the local regions.

there are dynamic objects with either fast or slow motion. Observations that come from the detected dynamic areas can be discarded, making a more robust localisation algorithm.

Finally, simulation and experimental results of the algorithms presented were shown.

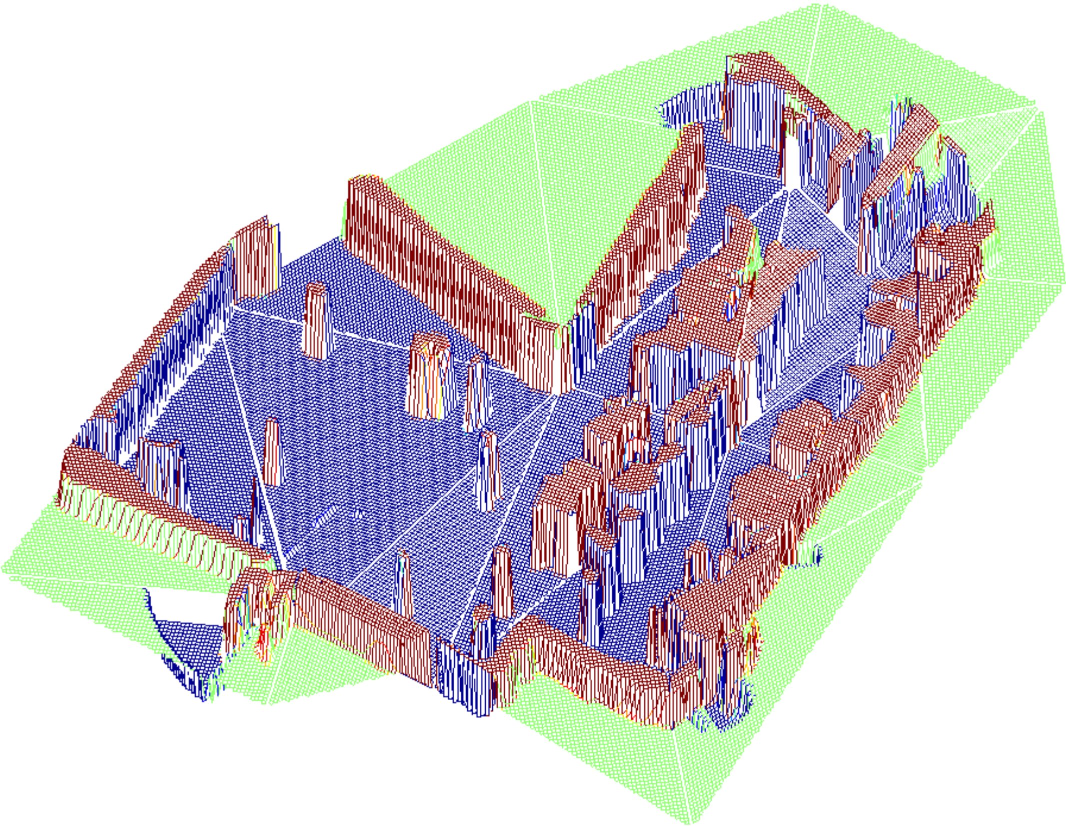


Figure 6.13: The surface represents the OG map obtained with OG-DenseSLAM.

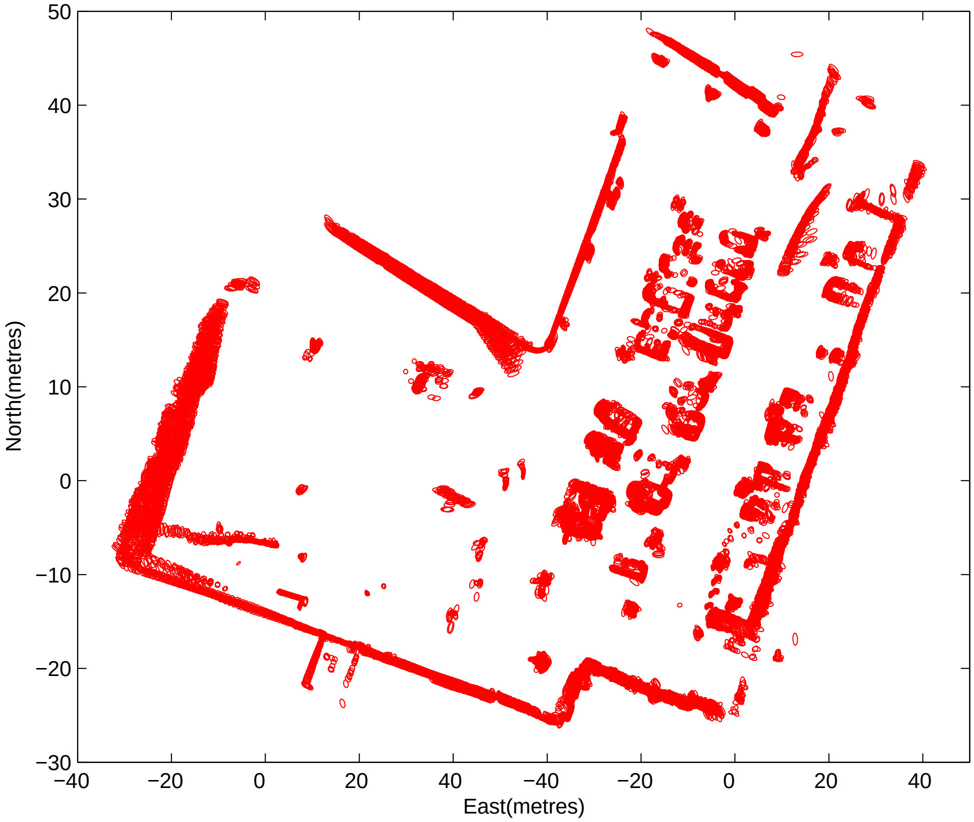


Figure 6.14: SOG map layer. The red solid lines represent the 1σ bounds ellipses for the Gaussians covariance.

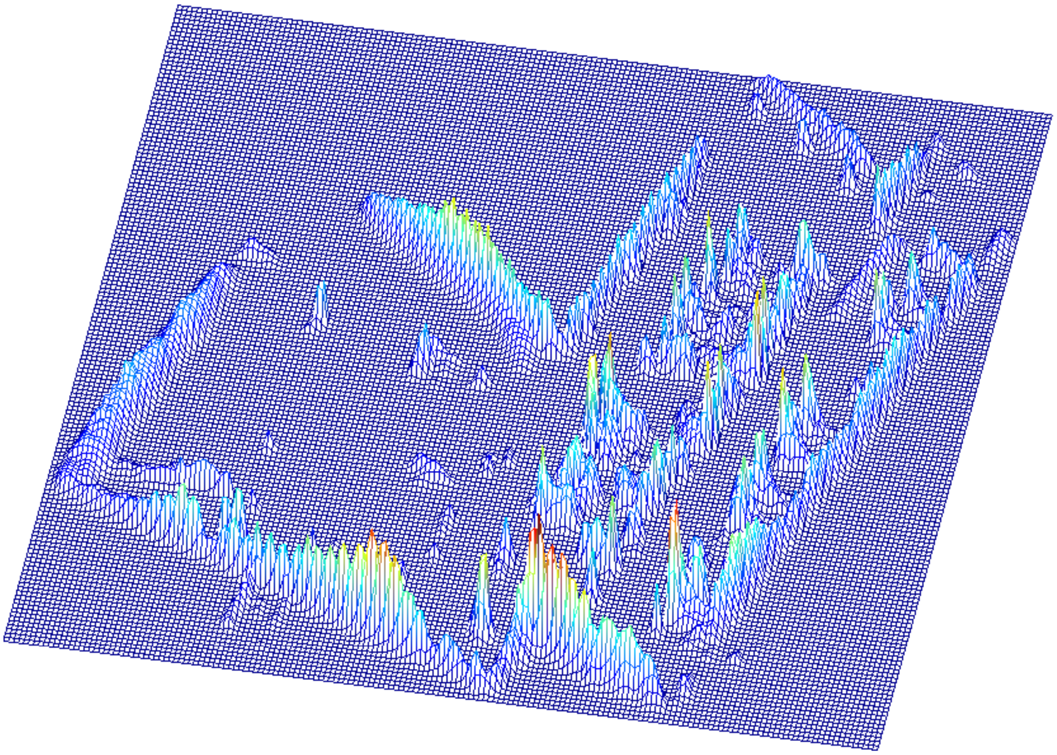


Figure 6.15: The surface represents the SOG map obtained with SOG-DenseSLAM.

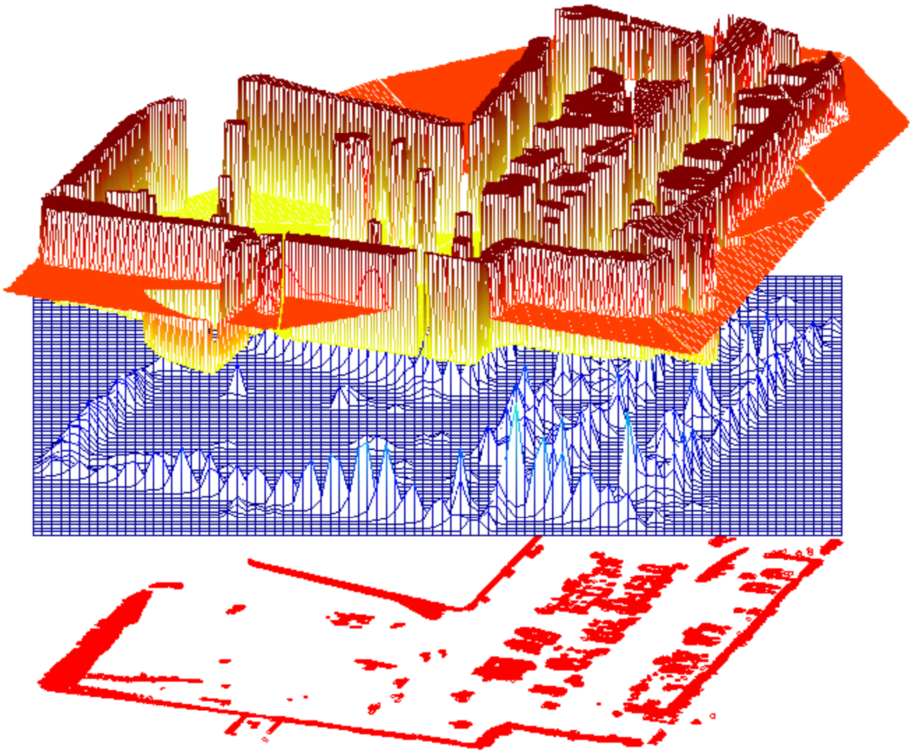


Figure 6.16: Three map layers obtained with the laser information. From bottom to top, the the raw data, the SOG and the OG layers.

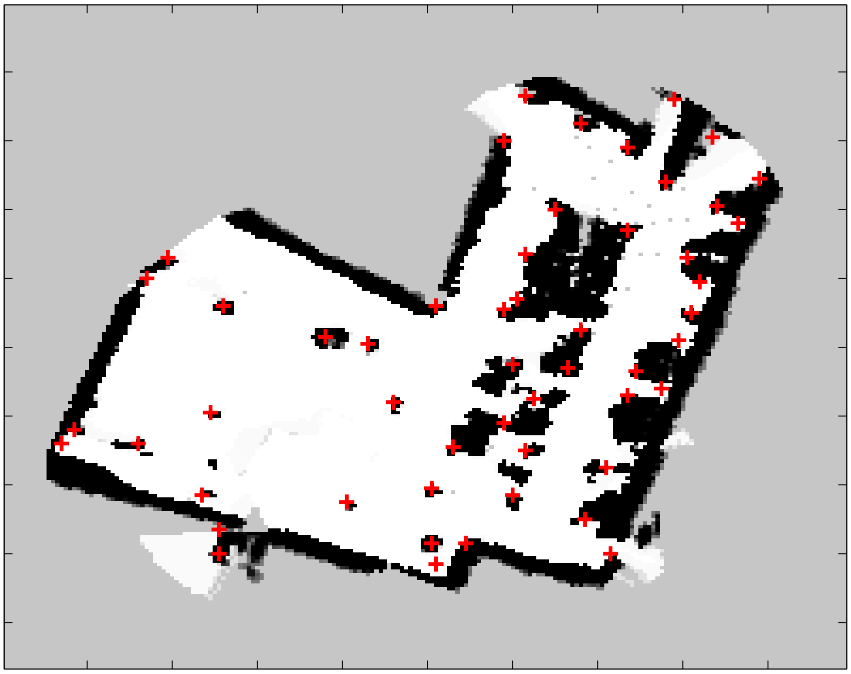
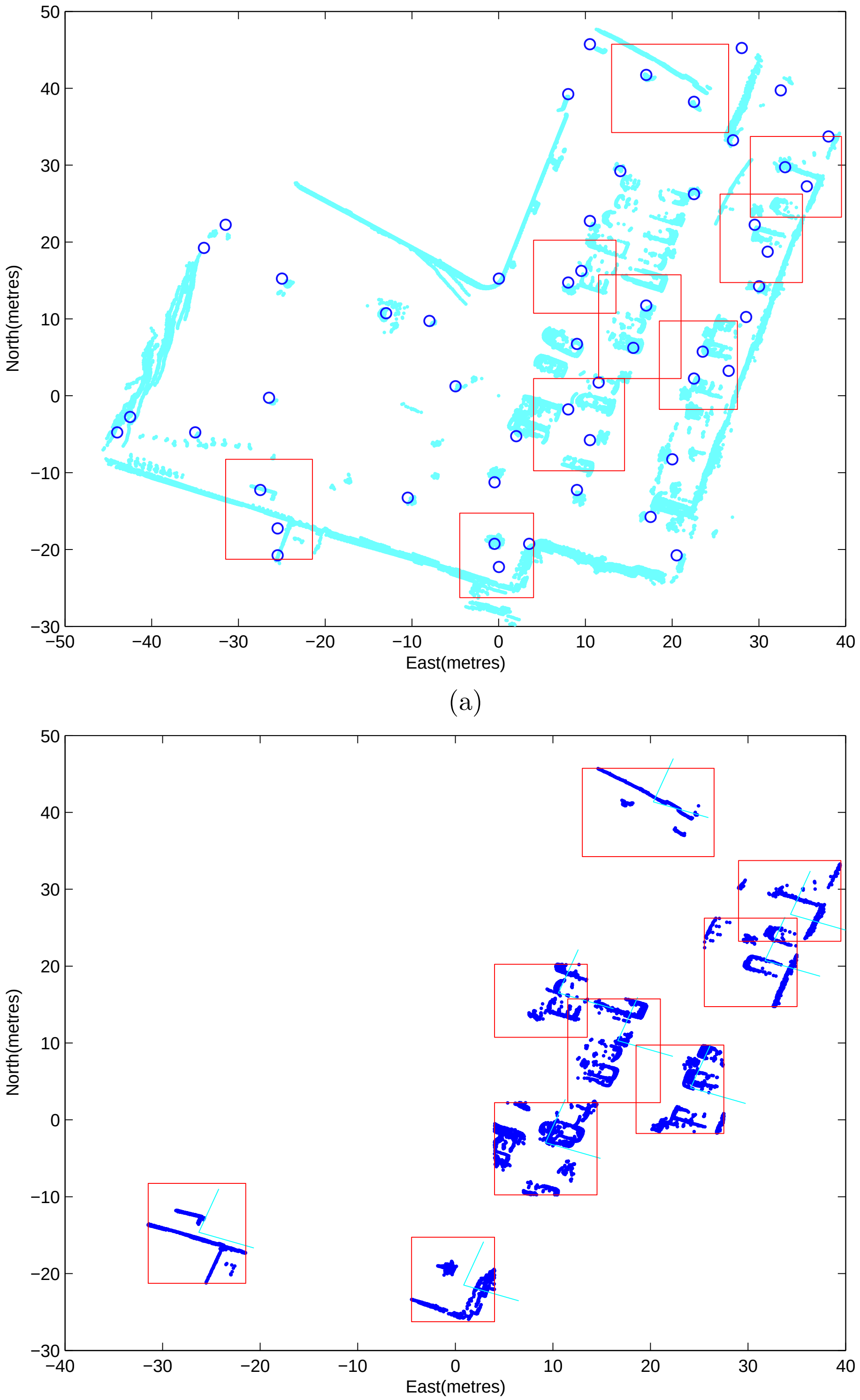


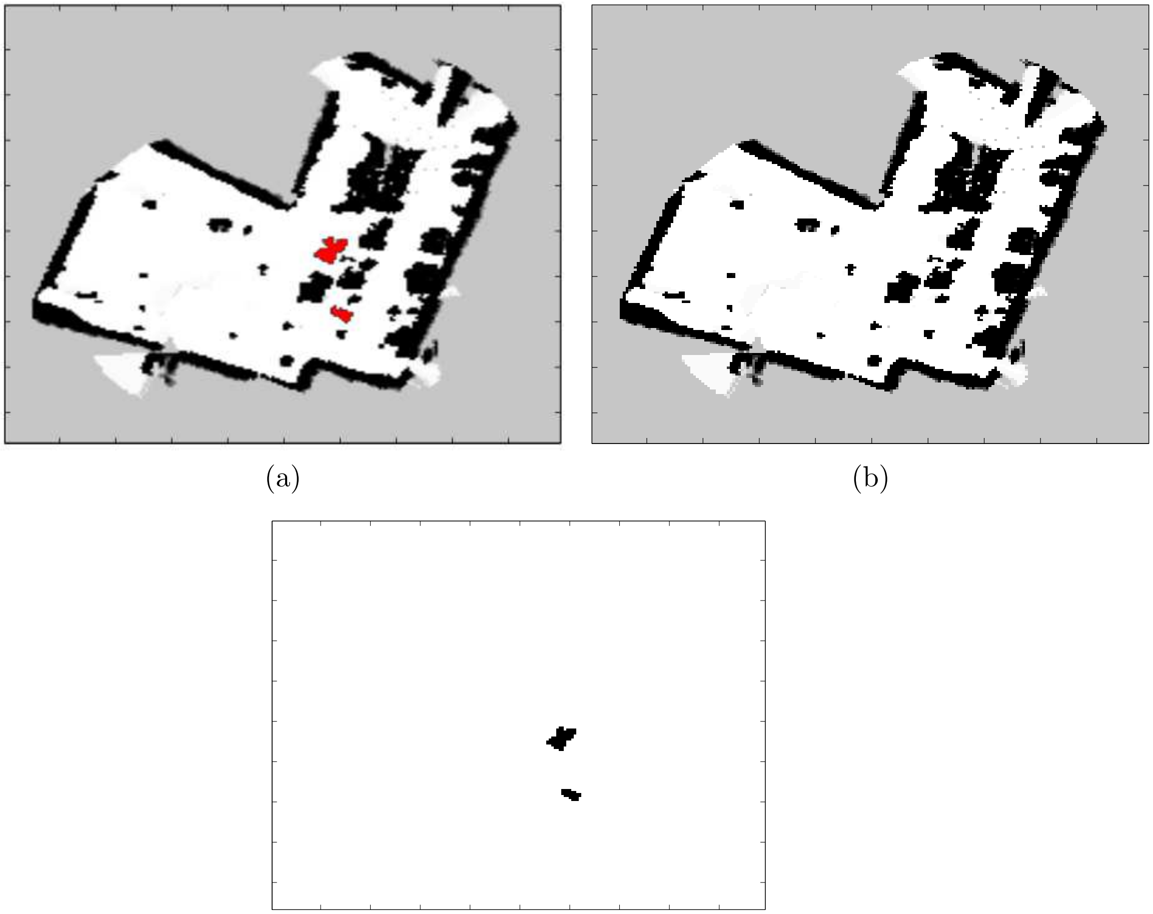
Figure 6.17: OG map layer with corners detected.



(a)

(b)

Figure 6.18: Figure (a) shows the HLLs detected. The rectangles represent the bounds used to define the landmarks template. (b) shows the HLLs map.



(a)

(b)

(c)

Figure 6.19: Slow motion detection algorithm. (a) and (b) show the OG map obtained at two different times. In (b), three cars have been removed from their parking places which are shown in (a) in red. (c) shows the result from the motion detection algorithm. Two spots have been identified as dynamics. The first spot is the detection of two cars that were near and the second spot for the third car removed.

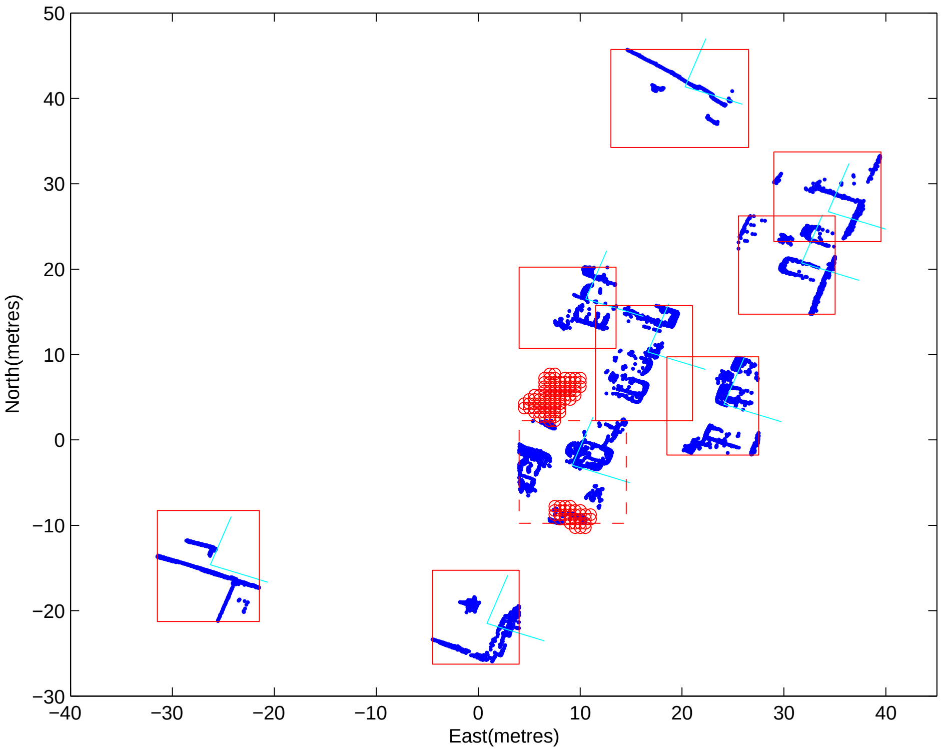


Figure 6.20: The red points denote the two dynamic areas found. The HLL represented with dashed line will be removed from the landmarks map since it is located under one of the dynamic areas detected.

Chapter 7

Conclusions

This thesis investigated environment representation for the Simultaneous Localisation and Mapping problem, in particular environment representations for the deployment of autonomous vehicles in large unstructured environments.

This chapter presents a summary of the thesis contributions and discusses areas of future research. The chapter is organised as follows. Section 7.1 presents a summary of the thesis contributions. Section 7.2 discusses areas of future research. Finally, Section 7.3 provides concluding remarks of the thesis.

7.1 Summary of contributions

Traditionally, SLAM algorithms have relied on sparse environment representations. A substantial number of works have appeared during recent years with the objective of reducing the computational complexity of SLAM algorithms. Most of these approaches are based on Kalman filters, which require the map to be modelled in the state-vector form. Consequently, the environment representation maintained by most SLAM algorithms is a landmark map where the landmarks are extracted from sensory information that possesses particular features, and the rest of the information is rejected. One important problem with traditional SLAM algorithms is that the sparse representation maintained cannot provide enough information for autonomous navigation. Despite the large amount of work focusses on issues related with the computational complexity of SLAM algorithms, very little work has been done to investigate alternatives representations rather than the classic feature-based.

This thesis focussed on the investigation of alternatives representations for SLAM algorithms and also in the investigation of how a richer representation can improve the overall vehicle navigation process. The next subsections present a summary of the thesis contributions.

7.1.1 DenseSLAM

This thesis introduced the concept of DenseSLAM. DenseSLAM was defined as the process of simultaneous localisation and dense mapping. The introduction of this concept presents a new line of research for the SLAM community to investigate the possibilities of obtaining a dense representation (e.g. an occupancy grid map) while simultaneously localising the vehicle. A solution for DenseSLAM will have to deal with computational and consistency issues. As described in this thesis, the map becomes correlated due to the robot pose uncertainty. Maintaining these correlations is necessary to guarantee consistency. At the same time, the computational cost required to update these correlations will increase quadratically with the number of objects stored in the map. A solution for DenseSLAM will have to deal with these two main problems.

7.1.2 The HYbrid Metric Maps

A solution for DenseSLAM was presented which was referred as the *Hybrid Metric Maps* (HYMMs).

The new algorithm combines feature maps with other dense metric sensory representation. The global feature map is partitioned into a set of connected Local Triangular Regions (LTRs), which provide a reference for a detailed multi-dimensional descriptions of the environment. The dense information is fused inside the local regions, relative to a local frame determined by the landmark positions. The relative representation used to represent the dense information reduces the correlations between states. Using this relative representation, the states represented in a local frame become strongly correlated and the states represented in different frames become weakly correlated. This is the key, therefore, that allows the filter to neglect part of the correlations, and minimise the errors in the dense map's position that could result from not maintaining these correlations. The consistency of the system is ensured by the use of a newly introduced concept referred to as *Unidirectional Information FLow* (UIF). The UIF is relevant to systems where there are (i) states that both give and receive information and (ii) states that only receive information. Using the UIF in DenseSLAM, where the dense maps are the states that only receive information, the consistency of the system is maintained.

The relative representation and the UIF are the basis of the HYMMs that permit the algorithm to obtain dense environment representation, without increasing the computational cost and without introducing inconsistencies.

7.1.3 SLAM with non-geometric features

EKF-SLAM is currently the most popular filter used to solve stochastic SLAM. An important issue with EKF-SLAM is that it requires sensory information to be modelled as geometric shapes.

Scan matching algorithms do not require geometric models, instead they use templates of raw scan data. The observations are aligned with an *a priori* map in order to estimate the robot pose. Scan matching algorithms have mainly been used for localisation given a priori map. Some scan correlation based SLAM algorithms have appearer in recent years. They usually require a selected history of scans to be accumulated [31] and align them as a network. One of the issues with these methods is that they do not perform data fusion and they are not compatible with the classic EKF-SLAM formulation. One of the objectives in this thesis was to find a method to make scan-matching algorithms compatible with EKF-SLAM.

This thesis presented Scan-SLAM, which is a marriage of EKF-SLAM with scan correlation. Instead of geometric models, landmarks are defined by templates composed of raw sensed data, and scan correlation is shown to produce landmark observations compatible with the standard EKF-SLAM framework. The resulting *Scan-SLAM* combines the general applicability of scan correlation with the established advantages of an EKF implementation: recursive data fusion that produces a convergent map of landmarks and maintains an estimate of uncertainties and correlations.

7.1.4 Dense mapping in outdoor environments

One of the objectives of this thesis was to investigate the improvements that a richer environment representation can introduce in the overall vehicle navigation process, in particular when the vehicle navigates in large unstructured environments. Three main issues were pointed out, and it was shown how a dense representation can help to deal with them.

The first issue is related to the detection of landmarks in unstructured environments. In many situations an object cannot be detected using the measurements taken from only one vantage point. Having a comprehensive environment representation allows post-processing of the sensory information to determine whether part of the map can qualify as a landmark. This thesis presented a method to detect and incorporate these landmarks which were referred as *High Level Landmarks* (HLLs).

The second issue is related to algorithms that are based on scan-matching to generate landmark observations. Any scan matching algorithm could converge to a local minimum if a good initial pose is not provided and the reference scan does not have enough information to promote reliable convergence. The probabilities of converging to a local minimum can be reduced if more information is used in the reference scan. If a dense environment representation is available, the template defining the reference scan can be augmented with information extracted from the dense map. The addition of information into the scans will produce a more robust scan alignment, the result will be less prone to convergence into a local minimum. It was shown how the map layers obtained with DenseSLAM can be used to assist the scan matching process.

The last issue tackled is associated with the fact that real environments are dynamic. Most of the SLAM algorithms assume the world is static. Consequently, dynamic objects can induce serious errors in the robot localisation process. If navigation is the main objective, it will be necessary to detect the dynamic objects to avoid incorporating them as navigation landmarks. This thesis presented a method that creates regions where dynamic objects are detected. The dynamic objects were classified into two main groups, objects with fast motion and objects with slow motion. The method presented can detect either objects. The dynamic regions detected are then superimosed with the landmark map to remove possible dynamic landmarks.

7.1.5 Experiments

This thesis presented simulation and experimental results in order to validate the performance of the proposed algorithms.

Simulation experiments are necessary since it is possible to compare the estimates with the actual vehicle pose and map position.

Experimental results in outdoor environments were presented. Even when the ground truth is usually not available as in simulation, experimental results are necessary to validate the algorithms in practical environments.

7.2 Future Research

Several of the algorithms presented in this thesis can be extended. This section presents areas of future research related to this thesis contributions.

7.2.1 Extensions to 3D

One of the pending areas of research is the extensions of SLAM algorithms to 3D. Except for very few cases [35, 38], SLAM has been studied for the 2D case. The extensions to $2\frac{1}{2}$D and 3D is a necessary step that will allow a more robust navigation process specially in natural environments.

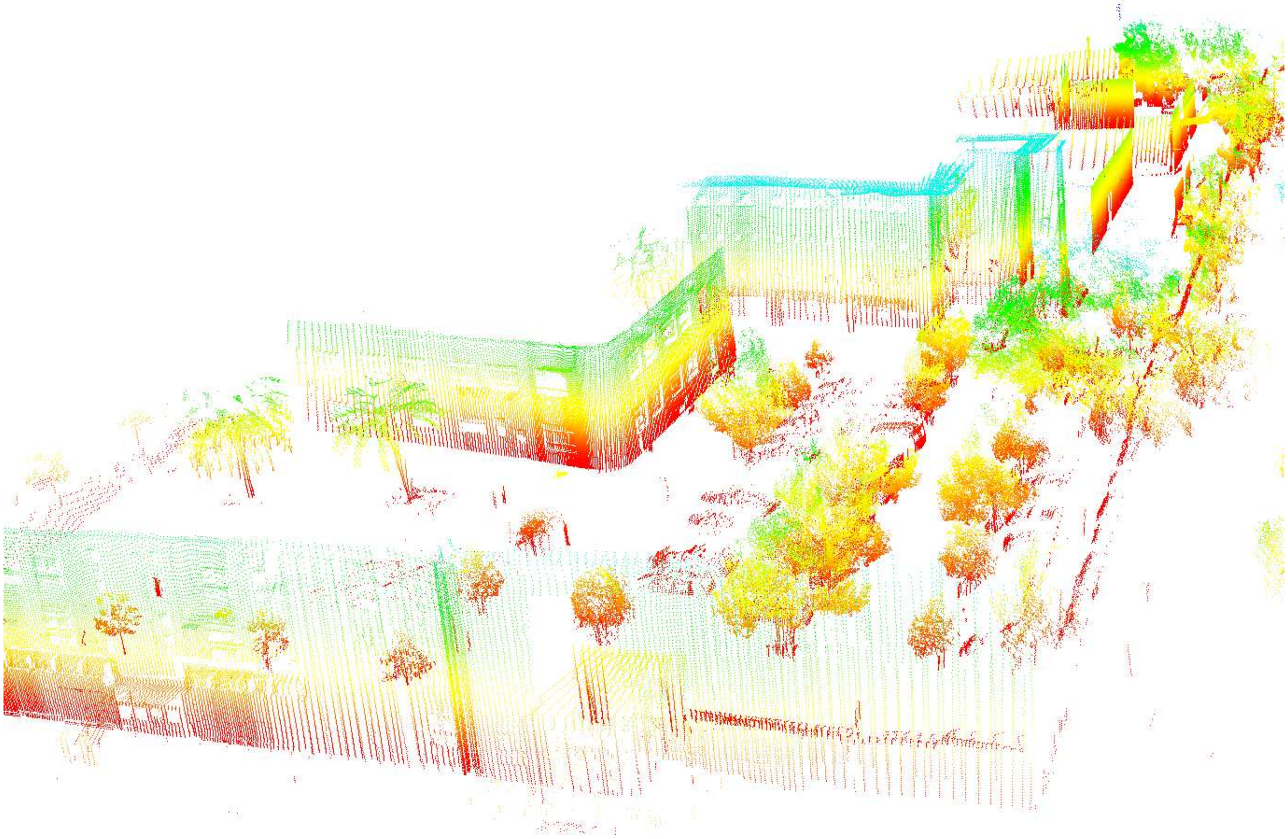


Figure 7.1: 3D laser plot of the experimental environment used along this thesis.

Figure 7.1 shows a 3D plot of the experimental environment used to present the results of this thesis. The representation was obtained using one laser perpendicular to the vehicle movement axis. It is clear from the figure that the third dimension possesses a lot of information that can be used for navigation.

The incorporation of more dimensions to the representation will make the localisation less sensitive to variations in the environment. At the same time, it will allow richer landmarks to be defined. The detection and incorporation of 3D landmarks is also an area of future research. In particular, the Scan-SLAM algorithm presented in Chapter 5 can be extended to 3D by representing the template landmarks with 3D points and applying 3D scan matching algorithms.

7.2.2 Sensor integration

DenseSLAM allows a multi-dimensional representation of the environment to be obtained. This thesis showed implementations where different map layers were obtained by representing the same sensory information using different representations such as Sum of Gaussians or Occupancy Grids. The framework also allows the information of different sensors to be fused into the local regions. One area of future work is the incorporation of the information gathered by other sensors in the dense maps, in particular vision sensors. Vision sensors could be used to obtain map layers such as colour or texture. In addition, vision sensors are better suited for 3D representations than range lasers. One of the problems with vision sensors is the detection of landmarks. Having a map with different layers obtained by different sensors will allow each sensor to benefit from the other sensors' capabilities allowing more robust long term autonomous navigation.

7.2.3 Scan-matching algorithms

This thesis proposed an algorithm that uses scan matching approaches to obtain observations from raw sensed data without the need for geometric models. The algorithm uses Sum of Gaussians correlation to align the scans. A thorough comparison between different scan matching algorithms like ICP, grid correlation and sum of Gaussians correlation would be a necessary step for the improvement of the algorithm presented here, in particular, their sensitivity to initial pose error.

7.2.4 Integrating control

SLAM has been studied as a problem independent of the vehicle control system. However, connecting the control decisions to the mapping process will be necessary for autonomous navigation. On the one hand, the control planning needs a map to plan the trajectory, for example based on shortest distance [22]. On the other hand, the mapping process can be optimised by adding goals to the control level. For example, one of the goals of the control planning could be to evaluate the path that maximises the overall information gain of the system. Another goal could be to maintain the smallest possible vehicle pose uncertainty during all the trajectory [69].

These high level tasks require the vehicle control system to be coupled with the SLAM process in order to take the decisions.

7.3 Summary

This thesis has presented algorithms that permit dense multi-layer maps to be obtained. DenseSLAM allows a detailed multi-dimensional description of the environment to be maintained. Scan-SLAM was developed to overcome the restriction of SLAM algorithms to geometric based-landmarks. In general, a more detailed environment representation improves many aspects of the vehicle navigation process. Not only localisation issues such as data association or error accumulation will benefit from a more comprehensive representation, but also high level decision tasks such as path planning.

Appendix A

Vehicle Models and Jacobians

The experimental platform used for the experiments is a conventional Holden UTE (see Figure A.1). The platform is equipped with Sick lasers, linear variable differential transformer sensor for the steering mechanism, back wheel velocity encoder, inertial unit, compass and GPS.

Table A.1 shows the typical parameters of the sensors used on the UTE.

Sensor	**Parameter**	**Unit**	**Value**
Dead reckoning	Sampling time	s	0.025
	Steering sensor σ	$\circ$	2
	Velocity encoder σ	m/s	0.2
Laser	Sampling time	s	0.2
	Max. Range	m	60
	Range σ	m	0.15
	Bearing σ	$^\circ$	1.5
IMU	Sampling time	s	0.008
	Acc. σ	m/s^2	0.01
	Gyro σ	$^\circ/s$	0.05

Table A.1: Sensors parameters

A.1 Motion Model

The motion model used for the vehicle moving in a plane is:

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v_c cos(\phi) \\ v_c sin(\phi) \\ \frac{v_c}{L} tan(\alpha) \end{bmatrix} \tag{A.1}$$

where (x_c, y_c) represent the vehicle cartesian coordinates taken a the centre of the vehicle

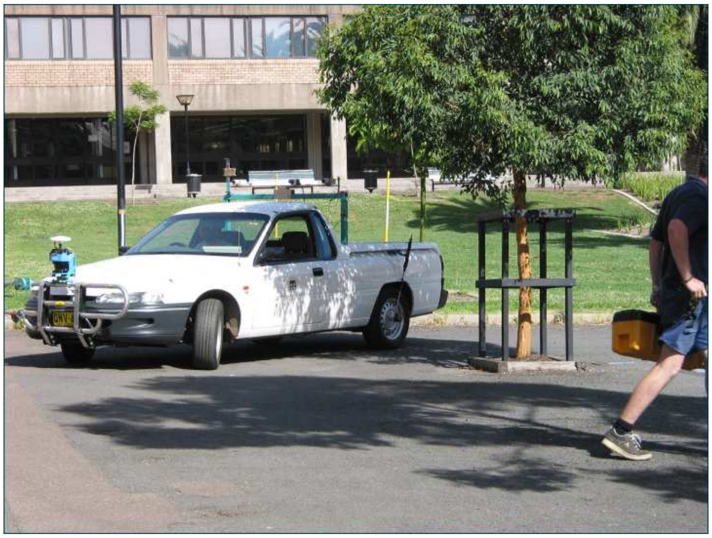


Figure A.1: The utility car used for the experiments is equipped with Sick lasers range and bearing sensor, linear variable differential transformer sensor for the steering mechanism, back wheel velocity encoder and inertial unit.

(see Figure A.2) and ϕ the vehicle orientation. v_c and α are the speed and steering control inputs respectively.

The laser sensor stands at the front of the vehicle. Applying a geometric translation to A.1, the motion model at the laser point $[x_L, y_L]$ can be obtained:

$$\begin{bmatrix} x_L \\ y_L \end{bmatrix} = \begin{bmatrix} x_c + acos(\phi) - bsin(\phi) \\ y_c + asin(\phi) + bcos(\phi) \end{bmatrix} \tag{A.2}$$

The velocity is measured with an encoder located at the back left wheel. This velocity, v_e, is translated to the center of the axle with the following equation:

$$v_c = \frac{v_e}{(1 - tan(\alpha)\frac{H}{L})} \tag{A.3}$$

For the UTE, $H = 0.76, L = 2.83, b = 0.5$ and $a = 3.78$ metres.

Finally the discrete model in global coordinates can be approximated using Euler with the following set of equations:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \phi_{k+1} \end{bmatrix} = \mathbf{f}(x_k, u_{k+1}) = \begin{bmatrix} x_k + \Delta T(v_c cos(\phi) - \frac{v_c}{L}tan(\alpha)(asin(\phi) + bcos(\phi))) \\ y_k + \Delta T(v_c sin(\phi) + \frac{v_c}{L}tan(\alpha)(acos(\phi) - bsin(\phi))) \\ \phi_k + \Delta T\frac{v_c}{L}tan(\alpha) \end{bmatrix} \tag{A.4}$$

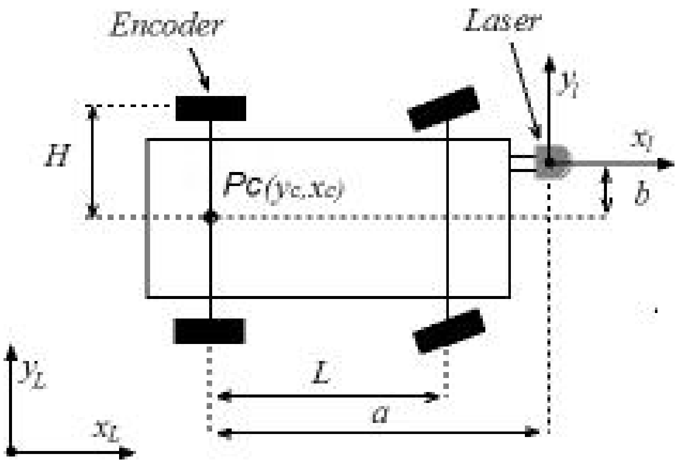


Figure A.2: Diagram of the UTE: sensor positions and geometry.

A.1.1 Motion Model Jacobians

The derivative of the motion model **f** with respect to the control inputs for steering and velocity inputs can be derived as:

$$\mathbf{B} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial \mathbf{x_v}}{\partial \mathbf{v_e}} & \frac{\partial \mathbf{x_v}}{\partial \alpha} \\ \frac{\partial \mathbf{y_v}}{\partial \mathbf{v_e}} & \frac{\partial \mathbf{y_v}}{\partial \alpha} \\ \frac{\partial \phi}{\partial \mathbf{v_e}} & \frac{\partial \phi}{\partial \alpha} \end{bmatrix} \tag{A.5}$$

$$\frac{\partial x_v}{\partial v_e} = \left(cos(\phi) - \frac{tan(\alpha)}{L} T_1 \right) \frac{\partial v_c}{\partial v_e} \Delta t \tag{A.6}$$

$$\frac{\partial x_v}{\partial \alpha} = -T_1 \frac{v_c}{L} cos(\alpha)^{-2} + \left(cos(\phi) - \frac{tan(\alpha)}{L} T_1 \right) \frac{\partial v_c}{\partial \alpha} \Delta t \tag{A.7}$$

$$\frac{\partial y_v}{\partial v_e} = \left(sin(\phi) + \frac{tan(\alpha)}{L} T_2 \right) \frac{\partial v_c}{\partial v_e} \Delta t \tag{A.8}$$

$$\frac{\partial y_v}{\partial \alpha} = -T_2 \frac{v_c}{L} cos(\alpha)^{-2} + \left(cos(\phi) - \frac{tan(\alpha)}{L} T_1 \right) \frac{\partial v_c}{\partial \alpha} \Delta t \tag{A.9}$$

$$\frac{\partial \phi}{\partial v_e} = \frac{tan(\alpha)}{L} \frac{\partial v_c}{\partial v_e} \Delta t \tag{A.10}$$

$$\frac{\partial \phi}{\partial \alpha} = \frac{v_c}{L} cos(\alpha)^{-2} + \frac{tan(\alpha)}{L} \frac{\partial v_c}{\partial \alpha} \Delta t \tag{A.11}$$

$$\tag{A.12}$$

Where

$$T_1 = asin(\phi) + bcos(\phi) \tag{A.13}$$

$$T_2 = acos(\phi) - bsin(\phi) \tag{A.14}$$

The derivative of the motion model $\mathbf{f}$ with respect to the vehicle states can be derived as:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x_v}} = \begin{bmatrix} 1 & 0 & -\Delta t \left(v_c sin(\phi) + \frac{v_c}{L} tan(\alpha) T_2 \right) \\ 0 & 1 & \Delta t \left(v_c cos(\phi) - \frac{v_c}{L} tan(\alpha) T_1 \right) \\ 0 & 0 & 1 \end{bmatrix} \tag{A.15}$$

A.2 Observation Model

The observation vector consists of range and bearing information from a SICK laser:

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = \mathbf{h}(x_k) = \begin{bmatrix} \sqrt{(x_L - x_v)^2 + (y_L - y_v)^2} \\ atan(\frac{y_L - y_v}{x_L - x_v}) - \phi \end{bmatrix} \tag{A.16}$$

where (x_L, y_L) represent the cartesian coordinates of the object being observed by the sensor.

A.2.1 Observation Model Jacobians

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial z_r}{\partial (x_v, y_v, \phi, \{x_L, y_L\})} \\ \frac{\partial z_\theta}{\partial (x_v, y_v, \phi, \{x_L, y_L\})} \end{bmatrix} \tag{A.17}$$

$$\frac{\partial z_r}{\partial (x_v, y_v, \phi, \{x_L, y_L\})} = \frac{1}{\Delta}[-\Delta_x, -\Delta_y, 0, 0, ...0, \Delta_x, \Delta_y, 0, ..., 0] \tag{A.18}$$

$$\frac{\partial z_\theta}{\partial (x_v, y_v, \phi, \{x_L, y_L\})} = \frac{1}{\Delta^2}[-\Delta_y, \Delta_x, -\Delta^2, 0, ...0, \Delta_y, -\Delta_x, 0, ..., 0]$$

Where

$$\Delta_x = x_L - x_v \tag{A.19}$$

$$\Delta_y = y_L - y_v \tag{A.20}$$

$$\Delta = \sqrt{\Delta_x^2 + \Delta_y^2}$$

Appendix B

Bayesian Estimation

This Appendix presents the equations for recursive bayesian estimation and gives the necessary assumptions to obtain the Kalman filter equations from the bayesian approach.

B.1 Predictions

The prediction step consists in finding the next probability:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \tag{B.1}$$

Assume the *a priori* probability is:

$$P(\mathbf{x}_{k-1}|\mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) \tag{B.2}$$

Then, applying the total probability theorem:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) = \int_{-\infty}^{\infty} P(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) d\mathbf{x}_{k-1} \tag{B.3}$$

$P(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0)$ can be decomposed by applying the chain rule:

$$P(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) = P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) P(\mathbf{x}_{k-1}|\mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) \tag{B.4}$$

Then, if $\mathbf{x}_{k-1}$ contains all the information obtained by the system until time $k-1$ ($\mathbf{Z}^{k-1}$) and the motion model is a Markov process:

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{Z}^{k-1}, \mathbf{U}^{k-1}, \mathbf{x}_0) = P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{B.5}$$

Combining Equation B.3, Equation B.4 and Equation B.5 the Chapman-Kolmogorov equation is obtained:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) = \int_{-\infty}^{\infty} P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)P(\mathbf{x}_{k-1}|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)d\mathbf{x}_{k-1} \tag{B.6}$$

Then given the stochastic motion model $P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)$ and the initial distribution $P(\mathbf{x}_{k-1}|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)$, and the control inputs, the prediction step can be calculated.

B.2 Update

The update step consists in calculating the next probability distribution:

$$P(\mathbf{x}_k|\mathbf{Z}^k,\mathbf{U}^k,\mathbf{x}_0) \tag{B.7}$$

The *a priori* information here is the result obtained from the prediction step Equation B.6:

$$P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) \tag{B.8}$$

Applying bayes rules:

$$P(\mathbf{x}_k|\mathbf{Z}^k,\mathbf{U}^k,\mathbf{x}_0) = K \cdot P(\mathbf{z}_k|\mathbf{x}_k)P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0) \tag{B.9}$$

where:

$$K = \int_{-\infty}^{\infty} P(\mathbf{z}_k|\mathbf{x}_k)P(\mathbf{x}_k|\mathbf{Z}^{k-1},\mathbf{U}^k,\mathbf{x}_0)d\mathbf{x}_k \tag{B.10}$$

Combining B.6 and B.9 a recursive solution for the update can be obtained:

$$P(\mathbf{x}_k|\mathbf{Z}^k,\mathbf{U}^k,\mathbf{x}_0) = K \cdot P(\mathbf{z}_k|\mathbf{x}_k)\int_{-\infty}^{\infty} P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)P(\mathbf{x}_{k-1}|\mathbf{Z}^{k-1},\mathbf{U}^{k-1},\mathbf{x}_0)d\mathbf{x}_{k-1} \tag{B.11}$$

B.3 Kalman filter assumptions

The Kalman filter solution assumes the noises are all Gaussians, temporally uncorrelated and zero mean and the models are linear.

Assuming the motion model noise and the initial conditions are also corrupted by Gaussian noise, since the convolution of two Gaussians is Gaussian [49], the Chapman-Kolmogorov equation (B.6) yields a Gaussian distribution.

Similarly, the product of two Gaussians is Gaussian, so Equation B.9 yields a Gaussian distribution.

The Kalman filter estimate is the minimum mean-square error (MMSE) conditioned on the observations sequence received which is found to be the Gassian mean of the result from Equation B.6 for the prediction step, and the Gaussian mean from Equation B.9 for the update. The estimates variance is then calculated taking the variance of the conditional mean.

The Kalman filter then works by propagating the Gaussian first and second moments using the Chapman-Kolmogorov equation for the prediction step and Bayes rule for the update. Under the assumptions noted above, these two steps are simplified to the addition and multiplication of matrices instead of convolution and products of distributions as is required for general Bayesian estimation.

For further information and the Kalman filter and the extended Kalman filter see [24, 51]

Appendix C

Relative Representation: Local Triangular Regions (LTRs)

Figure 3.7 shows an example of a LTR. Any point that belongs to a LTR can be characterised by a convex linear combination of the three vertex points (landmarks position) associated with this sub-region. In Figure 3.7 a LTR Ω_i is defined by the vertex points $\{L_{i,1}, L_{i,2}, L_{i,3}\}$. A local coordinate frame is defined based on the three vertex points according to the following equation:

$$\begin{aligned} \mathbf{a}_i &= \mathbf{L}_{i,2} - \mathbf{L}_{i,1} = [a_x, a_y] \\ \mathbf{b}_i &= \mathbf{L}_{i,3} - \mathbf{L}_{i,1} = [b_x, b_y] \end{aligned} \tag{C.1}$$

Any point that belongs to Ω_i can be expressed in the global frame as:

$$\begin{aligned} \mathbf{x}^G &= \mathbf{L}_{i,1} + \alpha \cdot \mathbf{a}_i + \beta \cdot \mathbf{b}_i \\ &= \quad (1 - \alpha - \beta) \cdot \mathbf{L}_1 + \alpha \cdot \mathbf{L}_{i,2} + \beta \cdot \mathbf{L}_{i,3} \end{aligned} \tag{C.2}$$

$$\begin{aligned} \alpha > 0, \ \ \beta > 0 \\ \alpha + \ \ \beta \ \ \leq 1 \\ \forall \ \ \mathbf{x} \ \setminus \ \mathbf{x} \in \Omega_i \end{aligned}$$

Then from (C.2) and (C.1) the local coordinates will be:

$$\alpha = \frac{(x_y - L_{1y})d_{13x} - (x_x - L_{1x})d_{13y}}{d_{13x}d_{12y} - d_{13y}d_{12x}} \tag{C.3}$$
$$\beta = \frac{(x_x - L_{1x})d_{12y} - (x_y - L_{1y})d_{12x}}{d_{13x}d_{12y} - d_{13y}d_{12x}}$$

where $\mathbf{x} = [x_x, x_y]$ are the global coordinates (C.2) of the local point and $\mathbf{d}_{ij} = \mathbf{L}_j - \mathbf{L}_i$. In general, any local point can be expressed as:

$$\mathbf{x}^L = [\alpha, \beta] = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{L}) \tag{C.4}$$

where $\mathbf{x}^L$ represents the relative state, $\mathbf{x}_v$ the robot pose, $\mathbf{z}$ the observations and $\mathbf{L}$ the set of landmarks that define the local region.

C.1 Jacobians

The derivative of a point expressed in local coordinates $\mathbf{x}^L$ with respect to its global position $\mathbf{x}^G$ is derived as:

$$\frac{\partial \mathbf{x}^L}{\partial \mathbf{x}^{\mathbf{G}}} = \frac{1}{\Delta}\begin{bmatrix} -d_{13y} & d_{13x} \\ d_{12y} & -d_{12x} \end{bmatrix} \tag{C.5}$$

Where

$$\Delta = d_{13x}d_{12y} - d_{13y}d_{12x} \tag{C.6}$$

The derivative of the point expressed in local coordinates with respect to the base landmarks is derived as:

$$\frac{\partial \alpha}{\partial \mathbf{L_1}} = \frac{\partial \alpha}{\partial d_{xL_1}}\frac{\partial d_{xL_1}}{\partial \mathbf{L}_1} + \frac{\partial \alpha}{\partial \mathbf{d}_{13}}\frac{\partial \mathbf{d}_{13}}{\partial \mathbf{L}_1} + \frac{\partial \alpha}{\partial \Delta}\frac{\partial \Delta}{\partial \mathbf{L}_1} \tag{C.7}$$
$$\frac{\partial \beta}{\partial \mathbf{L_1}} = \frac{\partial \beta}{\partial d_{xL_1}}\frac{\partial d_{xL_1}}{\partial \mathbf{L}_1} + \frac{\partial \beta}{\partial \mathbf{d}_{12}}\frac{\partial \mathbf{d}_{12}}{\partial \mathbf{L}_1} + \frac{\partial \beta}{\partial \Delta}\frac{\partial \Delta}{\partial \mathbf{L}_1}$$

$$\frac{\partial \alpha}{\partial \mathbf{L_2}} = \frac{\partial \alpha}{\partial \Delta}\frac{\partial \Delta}{\partial \mathbf{L}_2} \tag{C.8}$$
$$\frac{\partial \beta}{\partial \mathbf{L_2}} = \frac{\partial \beta}{\partial \mathbf{d}_{12}}\frac{\partial \mathbf{d}_{12}}{\partial \mathbf{L}_2} + \frac{\partial \beta}{\partial \Delta}\frac{\partial \Delta}{\partial \mathbf{L}_2}$$

$$\begin{aligned}\frac{\partial\alpha}{\partial\mathbf{L_3}} &= \frac{\partial\alpha}{\partial\mathbf{d}_{13}}\frac{\partial\mathbf{d}_{13}}{\partial\mathbf{L}_3} + \frac{\partial\alpha}{\partial\Delta}\frac{\partial\Delta}{\partial\mathbf{L}_3} \qquad (C.9)\\ \frac{\partial\beta}{\partial\mathbf{L_3}} &= \frac{\partial\beta}{\partial\Delta}\frac{\partial\Delta}{\partial\mathbf{L}_3}\end{aligned}$$

Where

$$d_{xL_1} = \mathbf{x}^G - \mathbf{L}_1 \tag{C.10}$$

$$\frac{\partial\alpha}{\partial d_{xL_1}} = \frac{1}{\Delta}\begin{bmatrix} -d_{13y} & d_{13x} \end{bmatrix} \tag{C.11}$$

$$\frac{\partial\beta}{\partial d_{xL_1}} = \frac{1}{\Delta}\begin{bmatrix} d_{12y} & -d_{12x} \end{bmatrix} \tag{C.12}$$

$$\frac{\partial d_{xL_1}}{\partial\mathbf{L}_1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{C.13}$$

$$\frac{\partial\alpha}{\partial d_{13}} = \frac{1}{\Delta}\begin{bmatrix} d_{xL_1x} & -d_{xL_1y} \end{bmatrix} \tag{C.14}$$

$$\frac{\partial\beta}{\partial d_{12}} = \frac{1}{\Delta}\begin{bmatrix} -d_{xL_1y} & d_{xL_1x} \end{bmatrix} \tag{C.15}$$

$$\frac{\partial d_{13}}{\partial\mathbf{L}_1} = \frac{\partial d_{12}}{\partial\mathbf{L}_1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{C.16}$$

$$\frac{\partial d_{12}}{\partial\mathbf{L}_2} = \frac{\partial d_{13}}{\partial\mathbf{L}_3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{C.17}$$

$$\frac{\partial\alpha}{\partial\Delta} = \frac{-\alpha}{\Delta} \tag{C.18}$$

$$\frac{\partial\beta}{\partial\Delta} = \frac{-\beta}{\Delta} \tag{C.19}$$

$$\frac{\partial\Delta}{\partial\mathbf{L}_1} = \begin{bmatrix} d_{23y} & -d_{23x} \end{bmatrix} \tag{C.20}$$

$$\frac{\partial\Delta}{\partial\mathbf{L}_2} = \begin{bmatrix} -d_{13y} & d_{13x} \end{bmatrix} \tag{C.21}$$

$$\frac{\partial\Delta}{\partial\mathbf{L}_3} = \begin{bmatrix} d_{12y} & -d_{12x} \end{bmatrix} \tag{C.22}$$

Bibliography

[1] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Phd thesis, University of Sydney, Australian Centre for Field Robotics, 2002.

[2] BBC. A history of navigation. http://www.bbc.co.uk/history/discovery/exploration/.

[3] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[4] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 1014–1019, 2003.

[5] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library, 1999.

[6] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller. An atlas framework for scalable mapping. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1899–1906, 2003.

[7] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 730–735, 1998.

[8] J.A. Castellanos, J. D. Tardos J.D, and Schmid. Building a global map of the environment of a mobile robot: the importance of correlations. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1053–1059, 1997.

[9] J.A. Castellanos, J.M. Martnez, J. Neira, and J.D. Tardós. Experiments in multisensor mobile robot localization and map building. In *Third IFAC Symposium on Intelligent Autonomous Vehicles*, pages 173–178, 1998.

[10] J.A. Castellanos, J.M.M Montiel, J. Neira, and J.D. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transaction on Robotics and Automation*, 15(5):948–952, 1999.

[11] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transaction on Robotics and Automation*, 17:125–137, April 2001.

[12] M. Csorba. *Simultaneous Localisation and Map Building.* Phd thesis, Robotics Research Group, Department of Engineering Science, University of Oxford, 1997.

[13] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1322–1328, 1999.

[14] M. Devy, R. Chatila, P. Fillatreau, S. Lacroix, and F. Nashashibi. On autonomous navigation in a natural environment. *Journal of Robotics and Autonomous Systems*, (16):5–16, 1995.

[15] A. Doucet, J.F.G. de Freitas, and N.J. Gordon. *Sequential Monte Carlo Methods In Practice.* Springer Verlag, New York, 2001.

[16] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.

[17] A. Elfes. *Occupancy Grids: A probabilistic framework for robot perception and navigation.* Phd thesis, Department of Electrical Engineering, Carnegie Mellon University, 1989.

[18] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22:46–57, June 1989.

[19] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Autonomous Mobile Robots: Perception, Mapping,and Navigation*, pages 60–71, 1991.

[20] A. Eliazar and R. Parr. Dp-slam 2.0. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1314–1320, 2004.

[21] J. Folkesson and H. Christensen. Graphical slam - a self-correcting map. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 383–390, 2004.

[22] J. Gancet and S. Lacroix. Pg2p: a perception-guided path planning approach for long range autonomous navigation in unkown natural environments. In *IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 2992–2997, 2003.

[23] P. W. Gibbens, G. M. W. M. Dissanayake, and H. F. Durrant-Whyte. A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pages 191–196, 2000.

[24] M.S. Grewal and A.P. Andrews. *Kalman Filtering: Theory and Practice.* Prentice Hall, 1993.

[25] M.S. Grewal, L.R. Weill, and A.P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration.* John Wiley and Sons, 2001.

[26] J. Guivant. *Efficient Simultaneous Localization and Mapping in Large Environments.* Phd thesis, University of Sydney, Australian Centre for Field Robotics, 2002.

[27] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transaction on Robotics and Automation*, 17(3):242–257, June 2001.

[28] J. Guivant and E. Nebot. Improving computational and memory requeriments of simultaneous localization and map building algorithms. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2731–2736, 2002.

[29] J. Guivant and E. Nebot. Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *IEEE Transaction on Robotics and Automation*, (19):749 –755, August 2003.

[30] J. Guivant, J. Nieto, F. Masson, and E. Nebot. Navigation and mapping in large unstructured environments. *The International Journal of Robotics Research*, 23(4):449–472, April 2004.

[31] J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 318–325, 1999.

[32] J.S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *1st Euromicro Workshop on Advanced Mobile Robots (Eurobot'96)*, pages 61–67, 1996.

[33] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211, 2003.

[34] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1557–1563, 2003.

[35] J. Il-Kyun and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Ninth IEEE International Conference on Computer Vision*, volume 2, pages 946–951, 2003.

[36] P. Jensfelt, D.J. Austin, O. Wijk, and M. Andersson. Feature based condensation for mobile robot localization. In *IEEE International Conference on Robotics and Automation*, pages 2531–2537, 2000.

[37] P. Jensfelt and S. Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transaction on Robotics and Automation*, 17(5):748–760, 2001.

[38] J. Kim and S. Sukkarieh. Airborne simultaneous localisation and map building. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 406–411, 2003.

[39] J. Knight, A. Davison, and I. Reid. Towards constant time SLAM using postponement. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 405–413, 2001.

[40] K. Konolige and K. Chou. Markov localization using correlation. In *International Joint Conference on Articial Intelligence*, pages 1154–1159, 1999.

[41] B. Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, (8):47–63, 1991.

[42] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation in a unknown terrains: Functions and integrations. *The International Journal of Robotics Research*, 21(10–11):917–942, 2002.

[43] J. Leal. *Stochastic Environment Representation.* Phd thesis, University of Sydney, Australian Centre for Field Robotics, 2003.

[44] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation*, 7:376–382, 1991.

[45] J. J. Leonard, R. J. Rikoski, P. M. Newman, and M. Bosse. Mapping partially observable features from multiple uncertain vantage points. *The International Journal of Robotics Research*, 21(10):943–975, October 2002.

[46] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE International Workshop on Intelligent Robots and Systems*, pages 1442–1447, 1991.

[47] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[48] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.

[49] S. Majumder. *Sensor Fusion and Feature-Based Navigation for Subsea Robots.* Phd thesis, University of Sydney, Australian Centre for Field Robotics, 2001.

[50] F. Masson, J. Guivant, and E Nebot. Hybrid architecture for simultaneous localization and map building in lare outdoor areas. In *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 570–575, 2002.

[51] P. Maybeck. *Stochastic Models Estimation and Control*, volume 1. Academic Press, 1982.

[52] P. J. McKerrow. Echolocation - from range to outline segments. In *Intelligent Autonomous Systems*, pages 238–247, 1993.

[53] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association.* PhD thesis, Carnegie Mellon University, 2003.

[54] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *American Association for Artificial Intelligence*, pages 593–598, 2002.

[55] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 695–701, 2002.

[56] M. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, (2):61–77, 1988.

[57] E. Nebot, J. Guivant, and J. Nieto. Acfr, experimental outdoor dataset. http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm, 2002.

[58] J. Neira and J. D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transaction on Robotics and Automation*, 17:890–897, 2001.

[59] P. Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem.* Phd thesis, University of Sydney, Australian Centre for Field Robotics, 1999.

[60] P. Newman and K. Ho. Slam- loop closing with visually salient features. In *IEEE International Conference on Robotics and Automation*, 2005.

[61] J. Nieto, T. Bailey, and E. Nebot. Scan-slam: Combining ekf-slam and scan correlation. In *International Conference on Field and Service Robotics*, pages 129–140, 2005.

[62] J. Nieto, J. Guivant, and E. Nebot. Denseslam: The unidirectional information flow (uif). In *5th IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[63] J. Nieto, J. Guivant, and E. Nebot. The hybrid metric maps (hymms): A novel map representation for denseslam. In *IEEE International Conference on Robotics and Automation*, pages 391–396, 2004.

[64] J. Nieto, J. Guivant, E. Nebot, and S. Thrun. Real time data association for fastslam. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 412–418, 2003.

[65] D. Pagac, E.M. Nebot, and Durrant-Whyte H. An evidential approach to map-building for autonomous vehicles. *IEEE Transaction on Robotics and Automation*, 14(4):623–629, 1998.

[66] Mark A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1157–1164, 2003.

[67] X. Pennec and J. P. Thirion. A framework for uncertainty and validation of 3d registration methods based on points and frames. *International Journal of Computer Vision*, 25(3):203–229, 1997.

[68] S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 1304–1311, 2003.

[69] S. Rezaei, J. Guivant, J. Nieto, and E. Nebot. Simultaneous information and global motion analysis (sigma) for car-like robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1339–1344, 2004.

[70] M. Ribo and A. Pinz. A comparison of three uncertainty calculi for building sonar-based occupancy grids. *Journal of Robotics and Autonomous Systems*, 35:201–209, 2001.

[71] A.C. Schultz and W. Adams. Continuous localization using evidence grids. In *IEEE International Conference on Robotics and Automation*, pages 2833–2839, 1998.

[72] A.C. Schultz, W. Adams, B. Yamauchi, and M. Jones. Unifying exploration, localization, navigation and planning through a common representation. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 2651–2658, 1999.

[73] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Fourth International Symposium of Robotics Research*, pages 467–474, 1987.

[74] A.J. Stoddart, S. Lemke, A. Hilton, and T. Renn. Estimating pose uncertainty for surface registration. *Image and Vision Computing*, 16(2):111–120, 1998.

[75] Studyworks. History of navigation at sea timeline. http://www.studyworksonline.com/cda/content/explorations/.

[76] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[77] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[78] S. Thrun and A. Bucken. Integrating grid-based and topological maps for mobile robot navigation. In *Thirteenth National Conference on Artificial Intelligence*, pages 944–950, 1996.

[79] S. Thrun, W. Bugard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *IEEE International Conference on Robotics and Automation*, pages 321–328, 2000.

[80] S. Thrun, W. Burgard, D. Chakrabarti, R. Emery, Y. Liu, and C. Martin. A real-time algorithm for acquiring multi-planar volumetric models with mobile robots. In *The 10th International Symposium of Robotics Research (ISRR'01)*, 2001.

[81] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 2004. To Appear.

[82] J.K. Uhlmann, S.J. Julier, and M. Csorba. Nondivergent simultaneous map building and localization using covariance intersection. In *SPIE Aerosense Conference on Navigation and Control Technologies for Unmanned Systems II*, volume 3087, pages 2–11, 1997.

[83] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.

[84] Chieh-Chih Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *IEEE International Conference on Robotics and Automation*, 2003.

[85] G. Weiß, C. Wetzler, and E. Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *IEEE International Conference on Intelligent Robots and Systems*, pages 595–601, 1994.

[86] S.B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. Phd thesis, University of Sydney, Australian Centre for Field Robotics, 2001.

[87] S.B Williams, P. Newman, G. Dissanayake, and H. Durrant-Whyte. Autonomous underwater simultaneous localisation and map building. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1792–1798, 2000.

[88] A. Zelinsky. A mobile robot exploration algorithm. *IEEE Transaction on Robotics and Automation*, (8):701–717, 1992.