SLAM Landmark Identification Using Lidar Measurements

by

Johannes Petrus Gericke

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Engineering
at Stellenbosch University

Supervisor:

Dr C.E. van Daalen
Department Electrical and Electronic Engineering

April 2019

Plagiarism Declaration

1. I have read and understand the Stellenbosch University Policy on Plagiarism and the definitions of plagiarism and self-plagiarism contained in the Policy [Plagiarism: The use of the ideas or material of others without acknowledgement, or the re-use of one's own previously evaluated or published material without acknowledgement or indication thereof (self-plagiarism or text-recycling)].

2. I also understand that direct translations are plagiarism.

3. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

4. I declare that the work contained in this assignment is my own work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

	J.P. Gericke	***April 2019***
	Initials and Surname	Date

Abstract

This thesis investigates the use of Lidar sensors for landmark identification in simultaneous localisation and mapping (SLAM). Lidar sensors can very accurately measure the distance toward objects in the environment, but provide no other information about the environment. Modelling landmarks from this spatial representation and associating new measurements with landmarks are important problems to address in order to perform SLAM with these measurements.

A literature review shows that multiple different approaches to extracting features from Lidar measurements and modelling landmarks exist. From this study we conclude that existing methods do not reliably associate measurements to landmarks and do not have the ability to update landmark models, which could be helpful to improve the representation of the environment.

We consequently develop a probabilistic method to model landmarks from Lidar measurements. In our approach, we approximate object boundaries with continuous, piecewise linear functions. The parameters of these functions are modelled as Gaussian random variables. With this probabilistic model, a method is created to determine if measurements originate from a certain landmark. This method first aligns the measurements to the model using the iterated closest point (ICP) algorithm and then it finds the Mahalanobis distance between measurements and lines to determine if the measurements fit the model. A method is also developed to probabilistically update the model and extend the model when new segments of the landmark are observed.

In addition, a SLAM algorithm is designed to use this landmark modelling method. The extended Kalman filter (EKF) SLAM motion update is used directly with an odometry motion model. The measurement update, however, is adapted in order to update the model parameters and robot pose simultaneously. This is in contrast to most other approaches that only use the landmark model to extract a point or reference measurement.

Finally, our methods are tested in both simulated environments and with real-world datasets. The results show that the landmark models are good representations of the environment and that measurements of unique objects can be associated with their models. However, the robot tends to be overconfident about its pose and fails to close bigger loops due to faulty associations.

We conclude that this approach successfully models the environment with SLAM, but further development needs to take place to make it robust and suitable for practical applications.

Uittreksel

Hierdie tesis ondersoek die gebruik van Lidar sensors om landmerke te indentifiseer tydens gelyktydige lokalisering en kartering (SLAM). Lidar sensors meet die afstand na voorwerpe in die omgewing met hoë akkuraatheid, maar verskaf geen ander inligting oor die omgewing nie. Die modellering van landmerke met die gebruik van hierdie ruimtelike voorstelling, asook om nuwe metings met landmarke te assosieer, is belangrike probleme om aan te spreek om SLAM uit te voer.

’n Literatuurstudie toon dat verskeie benaderings tot die onttrekking van kenmerke uit Lidar metings, asook die modellering van landmerke bestaan. Uit hierdie studie kom ons tot die gevolgtrekking dat bestaande metodes nie metings betroubaar assosieer met landmerke nie en nie die vermoë het om landmerkmodelle op te dateer nie, wat kan help om die voorstelling van die omgewing te verbeter.

Ons ontwikkel dus ’n probabilistiese metode om landmerke met die gebruik van Lidar metings te modelleer. Met hieride metode benader ons objekgrense met kontinue, stuksgewys lineêre funksies. Die parameters van hierdie funksies word gemodelleer as Gaussiese ewekansige veranderlikes. Met die gebruik van hierdie probabilistiese model word ’n metode geskep om vas te stel of nuwe metings van ’n sekere landmerk afkomstig is. Hierdie metode belyn eers die metings met die model met die gebruik van die herhaalde naaste punt (ICP) algoritme en vind dan die Mahalanobis afstand tussen metings en lyne om te bepaal of die metings pas by die model. ’n Metode word ook ontwikkel om die model probabilisties op te dateer en die model uit te brei wanneer nuwe dele van die landmerk waargeneem word.

Daarbenewens word ’n SLAM-algoritme ontwerp om hierdie landmerkmodelleringsmetode te gebruik. Die bewegingsopdatering stap van uitgebreide Kalman filter (EKF) SLAM word direk gebruik met ’n odometer-bewegingsmodel. Die meetopdatering stap word egter aangepas sodat die robotposisie en landmerkmodel gelyktydig opgedateer kan word. Dit is in teenstelling met die meeste ander benaderings wat slegs die landmerkmodel gebruik om ’n punt of verwysingsmeting te onttrek.

Ten slotte word ons metodes getoets in beide gesimuleerde omgewings en met werklike datastelle. Die resultate toon dat die landmerkmodelle goeie voorstellings van die omgewing skep en dat die metings van unieke voorwerpe met die landmerkmodelle geassosieer kan word. Die robot is egter geneig om te seker die word van sy posisie en versuim om groter lusse in die omgewing te sluit weens foutiewe assosiasies.

Ons kom tot die gevolgtrekking dat hierdie metode die omgewing suksesvol modelleer tydens SLAM, maar dat verdere ontwikkeling nodig is om dit robuust en geskik te maak vir praktiese toepassings.

Contents

List of Figures

Nomenclature

Abbreviations and Acronyms

BIC	Bayes information criterion
EIF	extended information filter
EKF	extended Kalman filter
FOV	field-of-view
ICP	iterated closest point
i.i.d.	independent and identically distributed
IMU	inertial measuring unit
Lidar	light detection and ranging
SLAM	simultaneous localisation and mapping
UKF	unscented Kalman filter
UT	unscented transform
2D	two-dimensional
3D	three-dimensional

Notation

$\mathcal{C}$	Gaussian canonical form
$\boldsymbol{c}$	correspondence vector
$\mathcal{E}$	evidence
$\boldsymbol{h}$	information vector
$\boldsymbol{K}$	information matrix
k	number of vertices
$\boldsymbol{m}$	map vector
m	number of lines
$\mathcal{N}$	Gaussian normal form
$\boldsymbol{n}$	zero-mean Gaussian distributed noise
n	number of measurements or landmarks
$\boldsymbol{Q}$	measurement noise covariance matrix
$\boldsymbol{R}$	rotation matrix
$\boldsymbol{r}$	reference point of a landmark
r	range measurement

$\boldsymbol{s}$	state vector
$\boldsymbol{u}$	odometry measurements/unknown probabilities
$\boldsymbol{v}$	vertices/landmark model parameters
$\mathcal{X}$	sigma points of $\boldsymbol{x}$
$\boldsymbol{x}_{\text{pos}}$	robot position (x- and y-coordinates)
$\boldsymbol{x}_{\text{pose}}$	robot pose
$\mathcal{Y}$	sigma points of $\boldsymbol{y}$
$\mathcal{Z}$	set of measurements
$\boldsymbol{z}_c$	measurements in Cartesian coordinates
$\boldsymbol{z}_p$	measurements in polar coordinates
$\boldsymbol{\gamma}$	grouping vector
$\boldsymbol{\zeta}$	observed line encoding
θ	robot orientation
$\boldsymbol{\vartheta}$	observed vertices encoding
$\boldsymbol{\mu}$	mean vector
$\boldsymbol{\Sigma}$	covariance matrix
σ	standard deviation
ϕ	angle measurement
ψ	line angle

Subscripts

x	x-coordinate of Cartesian vector
y	y-coordinate of Cartesian vector
i and j	index of an element in a vector, matrix or list
$[N \times M]$	dimensions of matrix, $N \times M$
$a : b$	element a to b of vector or matrix

Syntax and Style

$\boldsymbol{A}$	the matrix $\boldsymbol{A}$ (usually uppercase)
$\boldsymbol{x}$	the vector $\boldsymbol{x}$ (usually lowercase)
x	scalar x
$\|\|\boldsymbol{x}\|\|$	the Euclidean length of the vector $\boldsymbol{x}$: $\|\|\boldsymbol{x}\|\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$
$\mathrm{d}x$	differential element of x
$\frac{\mathrm{d}f}{\mathrm{d}x}$	derivative of function, f, with respect to x
$\begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix}$	vector with n elements x_0 to x_{n-1}
$\{x_0, \cdots, x_{n-1}\}$	set with n elements x_0 to x_{n-1}
$(x_0, \cdots, x_{n-1})$	list with n elements x_0 to x_{n-1}

Acknowledgements

I would like to thank the following people for making my time as a masters student easier and more enjoyable:

- Dr Van Daalen for his guidance and feedback.
- Everyone at the ESL who makes it a great environment to work.
- The BTK, where I could take a break from studies.
- My parents for their love and support.

Chapter 1

Introduction

1.1 Background

Autonomous robots are used in various applications to perform tasks with higher accuracy or more efficiently than humanly possible, or tasks that are unsafe for humans. These robots are, however, often used in controlled environments or with supervision of humans.

For a mobile robot to be truly autonomous, it should to be able to model its environment and localise itself within its environment. The robot should then be able to plan accordingly in this environment to perform its task with safety and precision. To be able to achieve this, the robot needs sensing capabilities with sufficient accuracy for the specified task.

Research in the field of autonomous robotics has investigated the use of a number of different sensors, such as sonar sensors, Lidar sensors and cameras. Lidar sensors are often used for its high accuracy in spatial measurements, whereas cameras are better for seeing and tracking specific objects.

1.2 Project Aims

In this project we aim to address the problem of simultaneous localisation and mapping (SLAM) using only 2D Lidar sensors to observe the environment. The focus is on identifying and modelling landmarks from Lidar measurements, which are used in SLAM. Landmarks in the SLAM context are typically static point in the environment. The purpose of a SLAM algorithm is to estimate both the robot's trajectory through an unknown environment and the landmark locations in this environment.

Lidar sensors usually give very accurate measurements of the distance toward objects, but contain no other information about objects. Therefore, spatial information about the object, such as its shape, has to be used to associate Lidar measurements with landmarks.

The main goal of this project is to develop a method to model landmarks from Lidar measurements. These models should be developed in a manner that the landmarks are uniquely identifiable and new measurements generated from the landmarks could be associated with it. Measurement from other landmarks should, however, be rejected by the landmark. The method should also be able to update a landmark model when new information about the landmark is obtained.

The purpose of the developed modelling method is to identify landmarks for SLAM, therefore another aim of this project is to implement and adapt an existing SLAM algorithm. This SLAM algorithm should be compatible with the design choices made during the development of the modelling method.

1.3 System Overview

The approach used for modelling landmarks from Lidar measurements is discussed briefly here. The specifics of the development and motivations for design choices are discussed in later chapters.

In our approach, we attempt to model landmarks by using the shape of objects. A shape is modelled by a set of straight lines, parameterised by continuous, probabilistic variables. Using a probabilistic approach allow measurements to be associated in a probabilistic way and landmarks to be updated probabilistically.

To model multiple objects observed in a single Lidar scan, a method is used to separate measurements that come from different objects. Once these measurements are separated, a set of measurements from the same object are either associated with a landmark or used to create a new landmark.

A SLAM algorithm is implemented to test the landmark modelling method. The standard extended Kalman filter (EKF) motion update is used, since this is a commonly-used method and the motion update is not the focus of this project. The measurement update is, however, adapted to incorporate the updating of landmark models and to use the measured landmark parameters in the full state space update.

Finally, these methods are tested in multiple simulated environments as well as in real-world environments by making use of publicly available datasets.

1.4 Document Outline

We first provide an overview of existing mapping methods and algorithms to perform SLAM with in Chapter 2. In this chapter we also discuss existing approaches to SLAM with Lidar sensors.

In Chapter 3 we discuss the development of the landmark modelling technique as well as methods to detect and update these landmarks. The development of the SLAM algorithm is discussed in Chapter 4, working through the motion and measurement updates.

The implementation of the full system is described and results from simulated and real datasets are shown and discussed in Chapter 5. Finally, we make conclusions about the project and suggest possible improvements to be made in Chapter 6.

Chapter 2

Literature Review

In this chapter a study is done of existing approaches to the general simultaneous localisation and mapping (SLAM) problem, as well as approaches to solving this problem using only Lidar measurements.

2.1 SLAM

SLAM is a key problem in robotics [1], addressing the situation of a mobile robot with an unknown pose in an unknown environment. The robot has to localise itself to build an accurate map of the environment, but it needs an accurate map to localise itself. In SLAM, the robot uses the measurements it obtains from the environment to build a relative map around itself and localise itself in this map. In this section we discuss different mapping techniques as well as existing SLAM algorithms found in literature.

2.1.1 Mapping

In SLAM, the robot needs to represent the environment using the measurements it obtains. In this subsection we look at two methods used to represent the environment. Although these are not the only representations, these methods are commonly used in SLAM applications.

Occupancy Grids

A possible representation for mapping an environment is to divide it into a grid of cells, where each cell has a probability of it being occupied or empty. The map created is called an occupancy grid and was developed by Elfes [2].

The occupancy grid is typically initialised with probability values and an inverse model of the sensor that the measurements come from is created to update these probabilities when new measurements are received. For range-angle measurements, the cells in the measurement beam have low occupancy probabilities, the measurement cells at the measurements have high occupancy probabilities and the rest of the environment is not updated. For known poses, the map is updated using this sensor model each time new measurements are received. Occupancy grids create a dense representation of the environment, which is good for applications such as planning where an accurate map of the entire environment is needed, but could make it computationally expensive in localisation applications.

In his article, Elfes [2] also addresses the problem of uncertain robot poses, as well as estimating the robot pose using the measurements and the map. Handling this pose uncertainty and localisation is critical in solving the SLAM problem. The cells in the occupancy grid are, however, assumed to be independent from each other. This means that once new information of a cell's occupancy is obtained, it has no effect on other cells. When used in SLAM, this independence results in the robot being unable to correct other parts of the environment once it revisits a previously observed part of the map and improves its pose estimate.

Landmark Maps

Another mapping representation often used in SLAM applications is landmark maps. In these maps, recognisable features in the static environment are modelled and stored as landmarks in the map [1]. Each of these landmarks is typically represented by its location in the global environment with a signature or model describing the landmark associated with it.

Low-dimensional features are typically extracted from high-dimensional measurements, creating a sparse representation of the environment. The challenge with these maps are, however, to extract reliable features to create a model describing the landmark and to associate new measurements to existing landmark models.

Most classical SLAM algorithms that use landmark maps only use point landmark measurements and assume that the modelling and association are done separately, which simplifies the problem significantly. A number of different methods have been proposed to approach the modelling and association problem with specific sensor measurements. Some of these methods are discussed later.

Landmark maps do not map the entire environment, but rather extract features from the environment and some measurements are not used to model a landmark and has no effect on the map. Although this is computationally less expensive when mapping and localising a robot, a dense representation of the environment is required in addition to the landmark map for certain applications.

2.1.2 SLAM algorithms

Once an environment representation has been chosen, an algorithm to perform SLAM with this representation can be designed. We now discuss some of the existing classical SLAM algorithms.

EKF SLAM

The EKF SLAM algorithm is the earliest of the SLAM algorithms, first presented by Smith and Cheeseman [3] and Smith *et al.* [4]. This algorithm applies the EKF to SLAM with a map of point landmarks. With EKF SLAM, all state variables are represented by Gaussian random variables. A consequence of this assumption is that negative information, or the absence of landmarks, cannot be processed by the algorithm [1].

EKF SLAM is a Bayes filter approach, where only the current belief over the states is calculated, which is also known as online SLAM. The algorithm maintains a joint distribution over the robot states and all the landmark locations, introducing dependencies between all elements. These dependencies allow landmarks not observed to also be updated when other states are updated.

The algorithm consists of two steps: the motion update and the measurement update. In the motion update, the controls given to the robot's actuators or odometry measurements of the robot's motion are used to update the belief over the robot's pose. This step typically increases the uncertainty over the robot's pose due to the noise of the controls.

The measurement update involves using the measurements of landmarks to update the belief over the robot pose and landmarks simultaneously. With this update, a predicted measurement of the landmark is evaluated against the actual measurement and all states are updated accordingly. This update usually decreases the uncertainty over the robot pose and the landmarks due to the added information of new measurements. If a new landmark is measured, it is added to the map. The motion update and measurement update are performed at each timestep and EKF linearisation is applied to the nonlinear motion and measurement models.

The usual formulation of EKF SLAM assumes that measurements are associated with the correct landmarks. This is, however, not always the case and faulty associations could lead to irreparable errors. Therefore, the design of reliable landmark models is very important when using this approach.

Other similar Bayes filter approaches have been implemented as well using different filters such as the unscented Kalman filter (UKF) and the extended information filter (EIF) [1].

Pose-Graph SLAM

EKF SLAM discussed above only maintains the posterior distribution over the latest pose and landmark estimates. The pose-graph SLAM algorithm, however, estimates the posterior distribution over all robot poses and landmarks, resulting in the complete trajectory of the robot. This idea was originally introduced by Lu and Milios [5].

Pose-graph SLAM is known as an offline SLAM algorithm [1], since it accumulates all robot controls and measurements over all timesteps and the posterior distributions are only calculated afterwards. Lu and Milios [5] applied this method by representing the environment with the point clouds of the raw measurements and performed scan matching to get estimates of the relationship between poses. However, we discuss the method presented by Thrun *et al.* [1], which uses a landmark-based approach.

The distribution over the full state space of all poses and landmarks is typically represented as a Gaussian random variable in the canonical form. A link is created between consecutive poses due to the robot controls describing the transformation between these poses. For each landmark measurement, a link is created between the landmark and the pose when the landmark is measured. These links are typically nonlinear relationships which are linearised to create a sparse information matrix.

Once all controls and measurements are factored in, the resulting estimate of all states is obtained from the canonical form distribution.

Since the algorithm maintains the pose estimates and measurements of all timesteps, the linearisation of the nonlinear relationships can be repeated once a posterior distribution over all states is obtained, which could lead to a better estimate of the posterior distribution.

Landmark association in pose-graph SLAM is also important. More robust methods to make landmark associations can be designed since measurements do not need to be processed sequentially [1]. As with the linearisation, the association step could be repeated once a posterior distribution over all states is obtained, which can lead to fixing faulty associations.

Although pose-graph SLAM can result in a more accurate representation of the environment and pose estimates of the complete robot trajectory, it could be very computationally expensive. This is due to the fact that the state space dimensions is proportional to the number of timesteps of the robot and the number of landmarks created.

Particle Filter SLAM

Another method of approaching SLAM is with the use of particle filters. Particle filters take a number of random samples, called particles, from a distribution of a random variable, where each particle is a hypothesis of the true states of the variable [1]. Any distribution can be approximated by this nonparametric method of drawing samples and therefore it is suitable for very nonlinear models.

Thrun [6] proposed the first mapping algorithm using particle filters and Murphy [7] introduced Rao-Blackwellised particle filters to the SLAM problem applied to occupancy grids. However, we discuss the approach developed by Montemerlo *et al.* [8], named FastSLAM, which applies particle filters to a landmark-based SLAM problem.

This approach maintains a particle set for each robot pose and estimates the mean and covariance of each landmark for this specific pose. This simplifies the problem to a mapping problem for each particle, where the EKF is used to update the landmark estimates. The landmark estimates are independent since the robot pose is assumed to be known for each particle.

For each timestep, new poses are sampled from the previous pose particles and robot controls. Once new poses are sampled, the measurements are used to update the estimates of the observed landmarks for each particle. An importance weight is calculated for each particle, which is related to the likelihood that the states represented by the particle are the true states. The particles are then resampled using these importance weights, which removes unlikely particles and creates more particles with high likelihoods.

The particle filter SLAM approach solves both the online and offline SLAM problems. At each timestep, an estimate of the current pose and map is calculated, which solves online SLAM and the algorithm also maintains all previous poses for each particle, which solves offline SLAM.

No assumptions are made regarding the distribution of the pose of the robot or the motion noise, such as the Gaussian assumption made with the other SLAM algorithms. This could result in a more accurate estimate of the robot pose, especially in nonlinear systems. The accuracy of the filter is dependent on the amount of particles used.

Due to the use of particles, where each particle is assumed to be the true robot pose, this SLAM approach is able to associate measurements with different landmarks for different particles. This can lead to more robust landmark associations, since bad associations will lead to unlikely particles that will not be resampled.

2.2 Lidar Sensors

For SLAM, the environment needs to be observed by a sensor. In this project we use light detection and ranging (Lidar) sensors to accomplish this. Lidar sensors are very accurate sensors that take a scan of the environment and measure distances to the closest objects. These sensors are commonly used in SLAM applications due to their accuracy regarding positional information of objects in the environment. We give a brief overview of the operation of Lidar sensors and then discuss different approaches to the SLAM application using Lidar measurements.

2.2.1 Lidar Sensor Operation

Lidar sensors use laser beams to measure the distance to objects. A Lidar emits laser pulses, which is reflected back by objects and sensed by the Lidar. The measured time of flight and speed of light is used to calculate the distance to these objects. A Lidar sensor emits these pulses in multiple directions, measuring the distance in each direction, which results in a scan of range-angle measurements. This is typically made possible by having a single laser source, with its direction being altered by a rotating mirror. Due to the rotating mechanical parts in the Lidar, it cannot reach the same measurement frequency as similar solid-state electronic sensors.

Lidar sensors with 2D and 3D sensing capabilities exist and they usually have a number of characteristics that are useful for localisation and mapping applications. The Lidar's aperture angle or field-of-view indicates the angular range it can take measurements in and the operating range indicates the maximum distance it can measure. A Lidar also has an angular resolution associated with it, which refers to the angle between consecutive range measurements, and a scanning frequency, which indicates how many scans it can take per second.

The last characteristic which is important for probabilistic applications is the accuracy of the Lidar regarding the angle and range measurements, which is often given as a statistical error. Lidar sensors are typically very accurate and the beam width of a laser source is small enough to be negligible.

Environmental effects can also cause Lidar sensors to make faulty measurements. These are typically due to objects with low reflectivity or objects with high focussed reflectivity, in which case the laser is reflected in another direction. The Lidar can also measure a pulse which is reflected by multiple objects, resulting in a longer time of flight and a higher measurement than the true distance. These errors can be avoided in controlled environments or filtered out with outlier detection methods.

Lidar sensors return range-angle measurements, which can be modelled as points in the environment, where the line between a point and the Lidar contains no objects and the point lie on an object boundary. When a Lidar measures no objects in a certain direction, a maximum range is typically measured by the Lidar. These maximum range measurements are usually filtered out by the Lidar sensor or before the measurements are used further. The Lidar measurements also have errors, which can be modelled as Gaussian distributed noise. This results in each measurement in the point cloud to be modelled as a Gaussian distribution. The remaining measurements form a point cloud of objects in the environment and these measurements can be used for purposes such as landmark identification.

2.2.2 SLAM with Lidar measurements

A number of different approaches to SLAM exist using only Lidar measurements to observe the environment. These approaches differ in mapping representations as well as the SLAM algorithms implemented.

Lu and Milios [5] implemented a pose-graph SLAM algorithm for 2D Lidar data, as mentioned in Section 2.1.2. The map in their approach is represented by the raw Lidar scans. Each Lidar scan is aligned to the previous scan with a scan-matching algorithm to obtain an estimate of the relationship between the two poses. Lidar scans from poses far apart in time, but close in space are also aligned to obtain estimates of the relationships between these poses. This approach only calculates the posterior distribution over all the poses and a point cloud map can be constructed by using the measurements and posterior poses.

A similar approach is proposed by Mendes *et al.* [9]. The differences are that they use 3D Lidar data and instead of constructing a graph of all poses, they create local maps and only insert a single pose for each local map in the graph. These local maps typically consist of a number of consecutive measurements that are aligned to each other; when new measurements are out of the previous local map's scope, a new local map is created, which is also considered for loop closure with other local maps.

A real-time SLAM algorithm using 2D Lidar measurements have been implemented by Hess *et al.* [10]. In this approach, occupancy grid submaps are created and these submaps are updated by aligning new measurements to the submap. As the robot moves out of the range of a submap it creates a new submap. Each submap has a local reference frame, which can be adjusted in the global frame once a previous submap is revisited to perform loop closure.

Other approaches to SLAM with Lidar data identifies landmarks in the environment and implement an EKF SLAM algorithm. Jensfelt and Christensen [11] extract rectangular shapes from the Lidar measurements, as this algorithm is designed for indoor environments where a lot of rectangles are usually visible. On the other hand, Guivant and Nebot [12] designed an algorithm for an outdoor, tree-filled environment and model landmarks as circles. Both of these methods use prior knowledge about the environment, which is crucial for the landmark models. These algorithms perform well in their specific environments, but are not suitable for other environments.

Nieto *et al.* [13, 14] proposed a method of modelling landmarks as the point cloud measurements obtained from the landmark. In their approach the Lidar scan is segmented into clusters of measurements that come from the same landmark and each cluster is evaluated separately. Each landmark is defined in a local reference frame and its reference frame's global pose is added to the SLAM state space. New measurements are associated with landmarks by using the iterated closest point (ICP) [15] algorithm to align the measurements with the landmark. The distances between the closest points between scans after scan alignment are evaluated against a threshold to determine if the shapes match. Since the landmarks are represented by the raw point measurements, they can represent any shape. The landmarks in this approach cannot be updated when new parts of a landmark is seen.

Other methods to extract general features from Lidar scans have been proposed as well. Bosse and Zlot [16] proposed three methods to extract keypoints from Lidar scans and creates a descriptor model of the area around each keypoint. These keypoints are used as landmarks in the SLAM application and the descriptor models are used to associate landmarks. In the first method to extract keypoints, the Lidar scan is segmented into connected components, similar to Nieto *et al.* [13, 14], and the centroid of each segment is calculated. The second method searches for points of high curvature in the scan by computing the second derivative of the scan range measurements and the third method computes a locally weighted mean of the measurements. The descriptor models create a discrete grid around the keypoint to describe the region. The extraction of these keypoints and associated descriptors are used to identify and associate landmarks to perform SLAM.

Himstedt *et al.* [17] also uses the curvature in a scan to extract features, similar to Bosse and Zlot [16]. The features in the scan are mapped to a pose-invariant histogram representing the Euclidean distance and angle between features. An approximate nearest neighbour approach is used to make initial associations to features and the histograms are used to validate these associations.

An alternative approach to feature extraction is to use image processing techniques on Lidar data. Li and Olson [18] proposed a method to rasterise the Lidar measurements to an image using a Gaussian kernel. This rasterisation method also takes into account occlusion boundaries and adds lines on the image where occlusion boundaries occur. An image processing corner detector is used to detect corners in the scan to use as landmarks. The landmarks in this approach is simply associated using a nearest neighbour approach, which could lead to bad associations when the robot pose uncertainty is big.

A number of different approaches have been proposed to extract features and model landmarks using Lidar measurements. All of these methods have their advantages, but they are lacking in some aspects. Some of the methods that use simple models only make nearest neighbour associations, which could easily result in faulty associations when the pose uncertainty is large. Other methods with more descriptive models can make more robust associations, but they typically lack the ability to update or extend the landmark model using new measurements. The methods that are able to update their landmark models do this in a separate step and do not consider the new information about the landmark model for pose correction.

2.3 Problem Statement

From this literature review, we conclude that existing approaches do not create very descriptive models of the environment that measurements could reliably be associated with and methods with more descriptive models cannot be updated with new measurements. We will therefore design a method to model landmarks, using Lidar measurements, that describe the objects that generate these measurements. The modelling method will describe the general shape of these objects, similar to Nieto *et al.* [13, 14], but in contrast to their approach that creates a point cloud description of the object, our model will create a parametric description of the object. This parametric model will be designed with the goal of being able to update the model with new measurements and to extend the model to create a better description of the object.

The goal of the landmark modelling is for it to be used in a SLAM application. Therefore, in addition to the modelling method, we will implement a SLAM algorithm and develop a novel measurement update to incorporate the update of our parametric landmark model.

Chapter 3

Landmark Modelling

In landmark-based SLAM, the modelling of landmarks is an essential part of the process. As stated previously, we approach landmark modelling by describing the shape of objects in the environment. This approach creates descriptive models of the environment so that new measurements can reliably be associated with landmarks. In addition, a method is designed to update the landmark models when new measurements are received.

The measurements, $\boldsymbol{z}$, that the Lidar takes are the only information the robot has about its environment. These measurements, which are often noisy, have to be used to model the landmarks in the environment. We choose to use Gaussian random variables to represent the uncertainty in the measurements as well as the model.

For a specific landmark model, we need to define a set of parameters, $\boldsymbol{v}$, to describe the model. Due to the uncertainty of the measurements, these parameters are also uncertain and are thus represented by Gaussian random variables. The transformation between the measurements and model parameters is often nonlinear and needs to be linearised to obtain a Gaussian distribution over the parameters.

In this chapter the proposed landmark model is discussed, including creating the model, detecting the model and updating the model. The unscented transform is explained first, since it is used as a linearisation method in many of the other processes.

3.1 Unscented Transform

The unscented transform described in this section is an existing method used to linearise a non-linear function. The material in this section is adapted from Thrun *et al.* [1, p. 64-71].

Suppose we have the nonlinear equation

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}), \tag{3.1.1}$$

and a Gaussian distribution over the variable $\boldsymbol{x}$,

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x). \tag{3.1.2}$$

The function $\boldsymbol{f}$ should be linearised to approximate the distribution over $\boldsymbol{y}$ as a Gaussian distribution,

$$p(\boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y). \tag{3.1.3}$$

The unscented transform is one method to linearise a function by drawing a set of sigma points, $\mathcal{X}$, from $p(\boldsymbol{x})$. Each of these sigma points, $\mathcal{X}_i$, are passed through the function $\boldsymbol{f}$ to obtain a mapped sigma point, $\mathcal{Y}_i$, where

$$\mathcal{Y}_i = \boldsymbol{f}(\mathcal{X}_i). \tag{3.1.4}$$

For an n-dimensional input vector, there are $2n+1$ sigma points that are calculated with the following equations:

$$\begin{aligned} \mathcal{X}_0 &= \boldsymbol{\mu}_x \\ \mathcal{X}_i &= \boldsymbol{\mu}_x + \sqrt{n+\lambda}\left(\sqrt{\boldsymbol{\Sigma}_x}\right)_i \qquad \text{for} \quad i = 1, ..., n \\ \mathcal{X}_i &= \boldsymbol{\mu}_x - \sqrt{n+\lambda}\left(\sqrt{\boldsymbol{\Sigma}_x}\right)_{i-n} \qquad \text{for} \quad i = n+1, ..., 2n, \end{aligned} \tag{3.1.5}$$

where $\lambda = \alpha^2(n+\kappa) - n$, with α and κ being scaling parameters. The subscript in $\left(\sqrt{\boldsymbol{\Sigma}_x}\right)_i$ indicates the i^{th} column of the matrix $\sqrt{\boldsymbol{\Sigma}_x}$.

There are also two weights associated with each sigma point. The one, $\omega_{m,i}$, is used to reconstruct the mean vector and the other, $\omega_{c,i}$, is used to reconstruct the covariance matrix. The weights are calculated as follows:

$$\begin{aligned} \omega_{m,0} &= \frac{\lambda}{n+\lambda} \\ \omega_{c,0} &= \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ \omega_{m,i} = \omega_{c,i} &= \frac{1}{2(n+\lambda)} \qquad \text{for} \quad i = 1, ..., 2n, \end{aligned} \tag{3.1.6}$$

where the parameter β can encode additional information about the underlying distribution.

The approximated Gaussian distribution, $p(\boldsymbol{y})$, is calculated by using the weights, ω_m and ω_c, and the mapped sigma points, $\mathcal{Y}$:

$$\begin{aligned} \boldsymbol{\mu}_y &= \sum_i \omega_{m,i}\mathcal{Y}_i \\ \boldsymbol{\Sigma}_y &= \sum_i \omega_{c,i}(\mathcal{Y}_i - \boldsymbol{\mu}_y)(\mathcal{Y}_i - \boldsymbol{\mu}_y)^T. \end{aligned} \tag{3.1.7}$$

In our application of the unscented transform, we made the standard choices of $\kappa = 0$ and $\beta = 2$. We also chose $\lambda = 1 - n$ so that the sigma points lie on the one standard deviation boundary, which results in $\alpha^2 = \frac{1}{n}$. An example of this process of linearisation through the unscented transform, for a scalar non-linear function, is shown in Figure 3.1. Since the output of the unscented transform is a Gaussian distribution, there is a linear transform that yield the same result as the unscented transform. The unscented transform can therefore be seen as a linearisation of the nonlinear model.

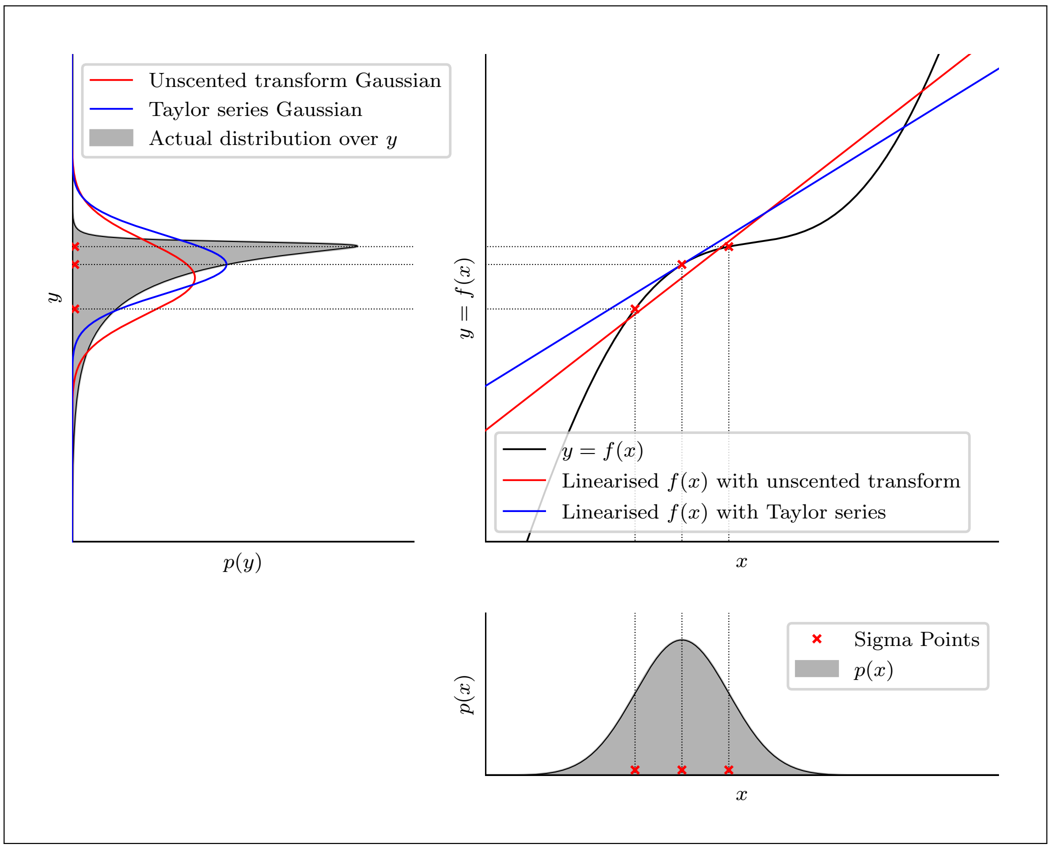


Figure 3.1 – Example of linearisation done with the unscented transform and the Taylor series expansion on a scalar non-linear function. The distribution $p(x)$ is transformed using $y = f(x)$ to form $p(y)$. The unscented transform and Taylor series expansion are used to linearise the function to obtain a Gaussian approximation of $p(y)$

Since we use the unscented transform (UT) frequently, we define the notation

$$p(\boldsymbol{y}) = \mathrm{UT}(p(\boldsymbol{x}); \boldsymbol{x} \mapsto \boldsymbol{y}) \tag{3.1.8}$$

to indicate the use of the unscented transform to linearise the transformation $\boldsymbol{x} \mapsto \boldsymbol{y}$, where $p(\boldsymbol{x})$ is the Gaussian distribution over $\boldsymbol{x}$ and $p(\boldsymbol{y})$ is the approximated Gaussian distribution over $\boldsymbol{y}$, obtained through the linearisation.

In Figure 3.1 the unscented transform is compared with linearisation using the Taylor series expansion. With the Taylor series expansion, the derivative of the non-linear function at the mean of the input distribution is calculated. This derivative is the slope of linearised function and the mean of the input distribution is substituted to obtain the offset of the linearised function. For multivariate functions, the derivative of the nonlinear function is called a Jacobian matrix.

It can be shown that the unscented transform performs better on average than the Taylor series expansion, which only uses the mean of the input distribution to linearise the function [1, p. 67]. The Taylor series expansion also requires the Jacobian of the function to be calculated, which is not the case for the unscented transform. This makes the implementation of the unscented transform more general since it does not need to be adapted for each separate function. The unscented transform and Taylor series expansion linearisation methods are compared in Figure 3.1.

The unscented transform is used to linearise different transformations in this project, especially in the landmark modelling process.

3.2 Vertex Parameter Modelling

As stated previously, the landmarks should be recognisable parts of the environment. The method we propose attempt to describe the shapes of these landmarks by approximating them as sets of straight lines, which can be parameterised by the vertices of these lines. Throughout this section, it is assumed that the robot and Lidar pose is known. It becomes clear in subsequent chapters why this assumption is used.

3.2.1 Landmark Model Creation

To create a landmark model, the robot has to make use of the Lidar measurements, $\boldsymbol{z}_p$, which are the only information it has about the environment. To model each individual landmark, these measurements have to be segmented into sets, where the measurements in each set come from the same landmark. In this subsection we assume that these measurement segmentation is done already and that $\boldsymbol{z}_p$ only refers to the measurements in a single set.

Vertex Parameterisation

We aim to parameterise the landmark by a set of vertices, $\boldsymbol{v}$, between straight lines describing the model. The process of obtaining the vertices from the measurements is discussed below. The case for noiseless measurements are first considered and then the process is extended to the case with noisy measurements.

The Lidar measurements are given as a list of n range and angle measurements, $\boldsymbol{z}_p = \begin{bmatrix}\boldsymbol{z}_{p,0}^T & \cdots & \boldsymbol{z}_{p,n-1}^T\end{bmatrix}^T$, with $\boldsymbol{z}_{p,i} = \begin{bmatrix}r_i & \phi_i\end{bmatrix}^T$. These measurements are in polar coordinates in a Lidar-fixed reference frame, with the pose of the Lidar, $\boldsymbol{x}_{\text{pose}} = \begin{bmatrix}x & y & \theta\end{bmatrix}^T$, as the origin of this coordinate system, as shown in Figure 3.2. The position of the Lidar is also defined as $\boldsymbol{x}_{\text{pos}} = \begin{bmatrix}x & y\end{bmatrix}^T$ in Cartesian coordinates in the inertial reference frame. These measurements are sorted in descending order with respect to the angle measurement so that $\phi_i > \phi_{i+1}$. This means the measurements are sorted from left to right from the Lidar's perspective.

In order to fit lines to these measurements, they first need to be transformed to Cartesian coordinates in the inertial reference frame, $\boldsymbol{z}_c = \begin{bmatrix}\boldsymbol{z}_{c,0}^T & \cdots & \boldsymbol{z}_{c,n-1}^T\end{bmatrix}^T$, where $\boldsymbol{z}_{c,i} = \begin{bmatrix}z_{x,i} & z_{y,i}\end{bmatrix}^T$. This transformation is done with the following equation:

$$\boldsymbol{z}_{c,i} = \boldsymbol{x}_{\text{pos}} + r_i \begin{bmatrix}\cos(\theta + \phi_i) \\ \sin(\theta + \phi_i)\end{bmatrix}. \tag{3.2.1}$$

These measurements are split into m groups of consecutive measurements, not to be confused with the measurement sets, and a single line is fitted to each group

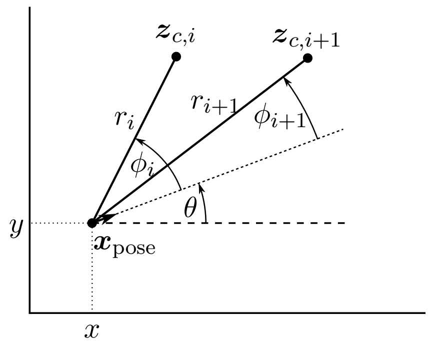


Figure 3.2 – Diagram of Lidar measurements in the inertial reference frame. The Lidar measurements are obtained in polar coordinates in a Lidar-fixed reference frame, centred at $\boldsymbol{x}_{\text{pose}}$ and transformed to the inertial reference frame.

of measurements. The measurements grouped together are indicated with $\boldsymbol{\gamma} = \begin{bmatrix}\boldsymbol{\gamma}_0 & \cdots & \boldsymbol{\gamma}_{m-1}\end{bmatrix}$, where $\boldsymbol{\gamma}_i = (\gamma_{i,0}, \gamma_{i,1})$ denotes the first and last indices of group i respectively, so that $\boldsymbol{z}_{c,\gamma_{i,0}:\gamma_{i,1}} = \begin{bmatrix}\boldsymbol{z}_{c,\gamma_{i,0}} & \cdots & \boldsymbol{z}_{c,\gamma_{i,1}}\end{bmatrix}^T$ are the measurements grouped together. The method of choosing these groups is discussed later in this section. For now we assume that the grouping is known.

We assume that the measurements in each of these groups are generated by a straight line, thus a single line is fitted to each group of Cartesian measurements. Each line is parameterised by a pair of points, $\boldsymbol{p}_0 = \begin{bmatrix}x_0 & y_0\end{bmatrix}^T$ and $\boldsymbol{p}_1 = \begin{bmatrix}x_1 & y_1\end{bmatrix}^T$, through which the line goes. This parameterisation is used to allow lines that lie in a vertical direction to be parameterised, which is not possible if the line is parameterised with a slope and an offset, $y = mx + c$. The equation for the line is

$$\boldsymbol{p}_\alpha = (1 - \alpha)\boldsymbol{p}_0 + \alpha\boldsymbol{p}_1, \tag{3.2.2}$$

where $\boldsymbol{p}_\alpha$ is a point on the line and α is the fraction that $\boldsymbol{p}_\alpha$ is between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$. This parameterisation has an infinite number of possible parameters to describe a specific line, as it can be parameterised by any two points on the line.

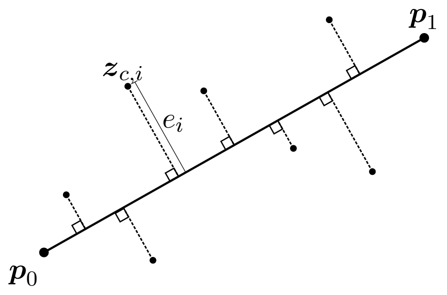


Figure 3.3 – Diagram of perpendicular errors between measurements and a line.

A line is fitted to the group of measurements by minimising the perpendicular square error, e_i^2, between the line and the measurements, as shown in Figure 3.3.

The derivation of Equations 3.2.3 to 3.2.6 are given in Appendix A, which goes through the process of minimising the square error. These equations result in the best fitted line, given the noiseless measurements. Once we introduce uncertainty to the measurements, uncertainty over the line arises as well. We omit the indices of the group, $\boldsymbol{z}_{c,\gamma_{i,0}:\gamma_{i,1}}$, here for simplicity and write it as $\boldsymbol{z}_c$. The two point parameters are calculated with the following equations:

$$\begin{aligned} \boldsymbol{p}_0 &= \overline{\boldsymbol{z}}_c = \frac{1}{n}\sum_i \boldsymbol{z}_{c,i} \\ \boldsymbol{p}_1 &= \boldsymbol{p}_0 + \beta \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}, \end{aligned} \tag{3.2.3}$$

where β is a chosen constant which equals the distance between $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$, which we choose as $\beta = 1$. ψ is the angle of the line,

$$\begin{aligned} \cos\psi &= \frac{b}{c} \\ \sin\psi &= \frac{a + \sqrt{a^2 + b^2}}{c}, \end{aligned} \tag{3.2.4}$$

where

$$\begin{aligned} a &= \sum_i (z_{y,i} - y_0)^2 - \sum_i (z_{x,i} - x_0)^2 \\ b &= 2\sum_i (z_{y,i} - y_0)(z_{x,i} - x_0) \\ c &= \sqrt{2\left(a^2 + b^2 + a\sqrt{a^2 + b^2}\right)}. \end{aligned} \tag{3.2.5}$$

For the case where $c = 0$, which can only occur if $b = 0$ and $a \leq 0$, the angle of the line is $\psi = 0°$, resulting in

$$\begin{aligned} \cos\psi &= 1 \\ \sin\psi &= 0. \end{aligned} \tag{3.2.6}$$

These values are substituted into Equation 3.2.3 to obtain the parameters for the line.

The parameters calculated here are just two possible points on the line which can be used to parameterise the same line. The exact choices of these parameters are, however, not as important here since it is only an intermediate step to finding the vertices between the different lines, which become the new parameters for the line. The choices for the parameters here are thus made to keep the equations as simple as possible, with $\boldsymbol{p}_0$ being the mean of the measurements and $\boldsymbol{p}_1$ a distance of β away from $\boldsymbol{p}_0$.

Once a line is fitted to every group of measurements, the vertices of the model is calculated. We now have m lines, $\boldsymbol{l} = \begin{bmatrix} \boldsymbol{l}_0^T & \cdots & \boldsymbol{l}_{m-1}^T \end{bmatrix}^T$, each parameterised by a pair of points, $\boldsymbol{l}_i = \begin{bmatrix} \boldsymbol{p}_{i,0}^T & \boldsymbol{p}_{i,1}^T \end{bmatrix}^T$. Using these m lines, $m - 1$ vertices can be calculated by finding the intersections between adjacent lines. Two vertices are also created at the two ends of the model, which results in $m+1$ vertices in total. This is done by adding a line at each end of the model, which is the line that goes through the Lidar position and the first and last measurement respectively, as shown in Figure 3.4. These two lines are parameterised by

$$\begin{aligned} \boldsymbol{l}_{-1} &= \begin{bmatrix} \boldsymbol{p}_{-1,0}^T & \boldsymbol{p}_{-1,1}^T \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{x}_{\text{pos}}^T & \boldsymbol{z}_{c,0}^T \end{bmatrix}^T \\ \boldsymbol{l}_m &= \begin{bmatrix} \boldsymbol{p}_{m,0}^T & \boldsymbol{p}_{m,1}^T \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{x}_{\text{pos}}^T & \boldsymbol{z}_{c,n-1}^T \end{bmatrix}^T . \end{aligned} \tag{3.2.7}$$

The vertices are now calculated by finding the intersections between adjacent lines. This is done by setting the the equations describing the two lines, Equation 3.2.2, equal to each other, finding α of the one line and then substituting α back into the equation of that line to find the intersection point $\boldsymbol{p}_\alpha$. This leads to the following equation, where the intersection point is assigned to the vertex, or $\boldsymbol{v}_i := \boldsymbol{p}_\alpha$:

$$\boldsymbol{v}_i = \frac{(\boldsymbol{p}_{i,0} - \boldsymbol{p}_{i-1,0})^T \boldsymbol{R} (\boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,0})}{(\boldsymbol{p}_{i-1,1} - \boldsymbol{p}_{i-1,0})^T \boldsymbol{R} (\boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,0})} (\boldsymbol{p}_{i-1,1} - \boldsymbol{p}_{i-1,0}) + \boldsymbol{p}_{i-1,0} \quad \text{for } i = 0, ..., m, \tag{3.2.8}$$

where

$$\boldsymbol{R} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} . \tag{3.2.9}$$

The method of finding these intersections is visualised in Figure 3.4.

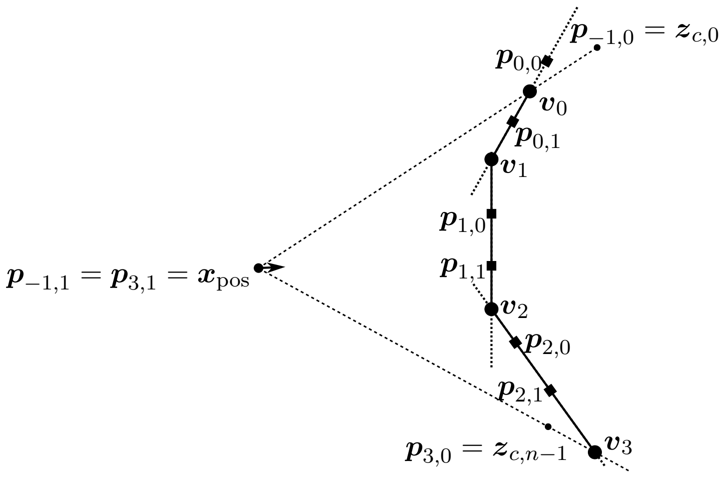


Figure 3.4 – Diagram of vertices found at line intersections. The first and last vertices are found by adding lines between the Lidar position and the first and last measurements.

Equations 3.2.1 to 3.2.8 describe the process of obtaining a set of vertices from a set of noiseless Lidar measurements, given the measurements that should be joined into lines. In order to take a probabilistic approach, we need to incorporate the uncertainty of the measurements. Each Lidar measurement, in polar coordinates, is assumed to have Gaussian distributed noise, $\boldsymbol{q}_i$, added from the underlying measurement, $\boldsymbol{z}'_{p,i}$, or

$$\boldsymbol{z}_{p,i} = \boldsymbol{z}'_{p,i} + \boldsymbol{q}_i, \quad \boldsymbol{q}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \tag{3.2.10}$$

where

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}. \tag{3.2.11}$$

σ_r^2 and σ_ϕ^2 are the variances of the range and angle measurements respectively.

The noise of the measurements is assumed to be independent and identically distributed (i.i.d). The set of underlying measurements can therefore be described by a Gaussian distribution,

$$p(\boldsymbol{z}_p') = \mathcal{N}(\boldsymbol{z}_p, \boldsymbol{\Sigma}_{z_p}), \tag{3.2.12}$$

where

$$\boldsymbol{\Sigma}_{z_p} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q} \end{bmatrix}. \tag{3.2.13}$$

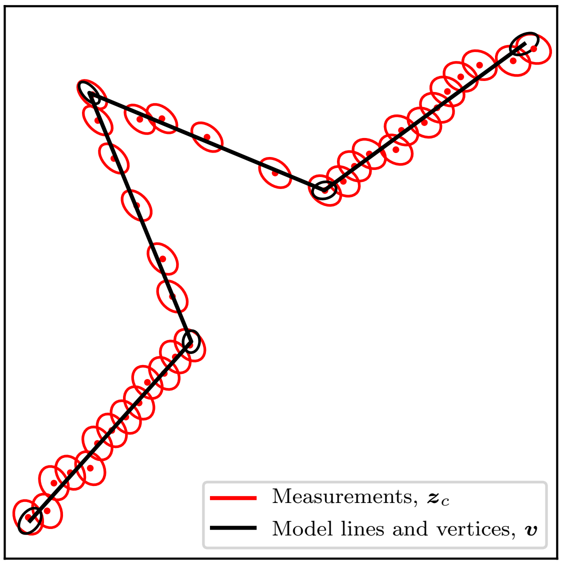


Figure 3.5 – Example of lines (black) being fit on a set of measurements (red). A distribution over the vertices is obtained. The 3-σ confidence ellipses of the measurements (red) and vertices (black) are shown.

Since we assume that the measurements are generated by a set of underlying straight lines, we use the unscented transform to obtain a Gaussian distribution over the vertices using the distribution over the measurements,

$$\mathcal{N}(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v) = \text{UT}(\mathcal{N}(\boldsymbol{z}_p, \boldsymbol{\Sigma}_{z_p}); \boldsymbol{z}_p' \mapsto \boldsymbol{v}). \tag{3.2.14}$$

An illustration of lines fitted to measurements are shown in Figure 3.5. The Cartesian measurements, $\boldsymbol{z}_c$, are shown, as well as its uncertainty, rather than the polar measurements, $\boldsymbol{z}_p$.

Likelihood Function

The distribution over the vertices allow us to define a function relating to the likelihood that a set of measurements is obtained from a landmark. For each measurement we calculate an expected range measurement, $r_{\text{exp},i,j}$, from the j^{th} line, given the line parameters and angle measurement. By performing an axis rotation, as in Figure 3.6, where the orientation of the robot is in line with the x-axis, we derive an equation for this expected measurement. The expected measurement is calculated as

$$r_{\text{exp},i,j} = v'_{j,x} - \frac{\Delta v'_{j,x} v'_{j,y}}{\Delta v'_{j,y}}, \tag{3.2.15}$$

where

$$\begin{aligned} \Delta \boldsymbol{v}'_j &= \begin{bmatrix} \Delta v'_{j,x} \\ \Delta v'_{j,y} \end{bmatrix} = \boldsymbol{R}(\boldsymbol{v}_j - \boldsymbol{v}_{j+1}), \\ \boldsymbol{v}'_j &= \begin{bmatrix} v'_{j,x} \\ v'_{j,y} \end{bmatrix} = \boldsymbol{R}(\boldsymbol{v}_j - \boldsymbol{x}_{\text{pos}}) \end{aligned} \tag{3.2.16}$$

and

$$\boldsymbol{R} = \begin{bmatrix} \cos(\theta + \phi_i) & \sin(\theta + \phi_i) \\ -\sin(\theta + \phi_i) & \cos(\theta + \phi_i) \end{bmatrix}. \tag{3.2.17}$$

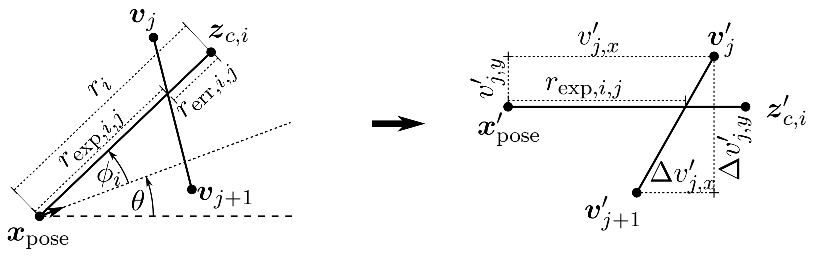


Figure 3.6 – Diagram of the expected measurement and error in measurement. The expected measurement is calculated by rotating the axis by $\theta + \phi_i$.

The error between the range measurement, r_i, and the expected range measurement can be calculated as the difference between the two,

$$r_{\text{err},i,j} = r_i - r_{\text{exp},i,j}. \tag{3.2.18}$$

By using the distributions over the line vertices and an underlying point being measured, the distribution over the error between the point and line can be obtained with the unscented transform,

$$\begin{aligned} p(r_{\text{err},i,j}) &= \mathcal{N}(\mu_{r_{\text{err},i,j}}, \Sigma_{r_{\text{err},i,j}}) \\ &= \text{UT}\left(\mathcal{N}\left(\begin{bmatrix} \boldsymbol{z}_{p,i} \\ \boldsymbol{\mu}_{v,j} \\ \boldsymbol{\mu}_{v,j+1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{v,j,j} & \boldsymbol{\Sigma}_{v,j,j+1} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{v,j+1,j} & \boldsymbol{\Sigma}_{v,j+1,j+1} \end{bmatrix} \right); \begin{bmatrix} \boldsymbol{z}'_{p,i} \\ \boldsymbol{v}_j \\ \boldsymbol{v}_{j+1} \end{bmatrix} \mapsto r_{\text{err},i,j} \right). \end{aligned} \tag{3.2.19}$$

The likelihood that the measurement is measured from the line is the likelihood that this error equals 0,

$$\begin{aligned}\log\left(p(\boldsymbol{z}_{p,i}|\boldsymbol{v}_j,\boldsymbol{v}_{j+1})\right) &= \log\left(p(r_{\mathrm{err},i,j}=0)\right)\\ &= -0.5\left(\log(2\pi\sigma^2_{r_{\mathrm{err},i,j}}) + \frac{\mu_{r_{\mathrm{err},i,j}}}{\sigma^2_{r_{\mathrm{err},i,j}}}\right).\end{aligned} \tag{3.2.20}$$

To calculate the likelihood that the whole set of measurements comes from the landmark, each measurements has to be associated with a specific line. This is indicated by the correspondence vector, $\boldsymbol{c} = \begin{bmatrix} c_0 & \cdots & c_{n-1}\end{bmatrix}^T$, where c_i is the index of the line that the i^{th} measurement is associated with. These correspondences is assumed to be known in this section.

The likelihood that the set of measurements is measured from the landmark, is the sum of the all log likelihoods of the individual measurements with their corresponding lines,

$$\log\left(p(\boldsymbol{z}_p|\boldsymbol{v})\right) = \sum_i \log\left(p(r_{\mathrm{err},i,c_i}=0)\right). \tag{3.2.21}$$

This likelihood is used to compare different models and in the process of associating a set of measurements with a landmark.

Model Selection

Up to this point we have assumed that the measurement grouping is known, but to create a landmark model from a set of measurements, we have to decide which measurements should be grouped together and how many lines the model should consist of. There are a lot of different possible models for a set of measurements and we preferably want to use the best one. We need to compare different models to each other with the following criteria: the data needs to fit the model well and the model should be as simple as possible.

A good probabilistic approach to this problem is to compare models with this likelihood equation,

$$p(\boldsymbol{z}_p|M) = \int p(\boldsymbol{z}_p|\boldsymbol{\theta},M)p(\boldsymbol{\theta}|M)\mathrm{d}\boldsymbol{\theta}. \tag{3.2.22}$$

M is the model with m lines, without the specific parameters, and $\boldsymbol{\theta}$ is the parameters of this model.

The integral in Equation 3.2.22 cannot, however, be evaluated exactly. We therefore make use of the Bayes information criterion (BIC), which approximates this likelihood with the following equation:

$$\log p(\boldsymbol{z}_p|M) \approx \log p(\boldsymbol{z}_p|\boldsymbol{\theta},M) - \frac{K}{2}\log N, \tag{3.2.23}$$

where $K = \dim(\boldsymbol{v})$ and $N = \dim(\boldsymbol{z}_p)$ [19, p. 282-284]. In this equation, the first term is the likelihood of the measurements given the model and parameters, where

$$\log p(\boldsymbol{z}_p|\boldsymbol{\theta},M) = \log p(\boldsymbol{z}_p|\boldsymbol{v}) \tag{3.2.24}$$

from Equation 3.2.21, and the second term penalises the model complexity.

For n measurements, $N = 2n$, since each measurement consists of a range and angle. The dimensionality of $\boldsymbol{\theta}$ is more complex since the parameters consist of a mean vector, $\boldsymbol{\mu}_v$, and a covariance matrix, $\boldsymbol{\Sigma}_v$. For a model with k vertices, we calculate the dimensionality as

$$\begin{aligned} K &= \dim(\boldsymbol{\theta}) \\ &= \dim(\boldsymbol{\mu}_v) + \dim(\boldsymbol{\Sigma}_v) \\ &= 2k + \frac{2k(2k+1)}{2} \\ &= 2k^2 + 3k. \end{aligned} \tag{3.2.25}$$

Since the covariance matrix is symmetrical, only the upper triangular matrix is considered when calculating its dimensionality.

In this this model selection, the BIC penalises the complexity of the model too much. Since the penalty term is an approximation, we introduce a scaling factor, α, to it, which can be adjusted for better results in this application:

$$\log p(\boldsymbol{z}_p|M) \approx \log p(\boldsymbol{z}_p|\boldsymbol{v}) - \alpha\frac{K}{2}\log N. \tag{3.2.26}$$

This likelihood in Equation 3.2.26 can now be used to compare different models and parameters with each other. To obtain the best model, all possible models need to be evaluated and the highest likelihood model should be chosen. This is, however, computationally intractable, since all possible measurement grouping combinations has to be considered. We therefore propose another method, which attempts to choose the best grouping that results in the best model. This method, however, does not guarantee that the best model is selected, but selects a sufficient model.

The proposed method starts by fitting a line to each pair of adjacent measurements, essentially making each Cartesian measurement a vertex. The initial grouping vector is $\boldsymbol{\gamma} = \begin{bmatrix}(0,1) & (1,2) & \cdots & (n-2, n-1)\end{bmatrix}$, resulting in $n-1$ lines. The number of lines are then iteratively reduced by joining two adjacent lines that results in the lowest decrease in model likelihood. This is done until one line is fitted to all the measurements, with $\boldsymbol{\gamma} = \begin{bmatrix}(0 & n-1)\end{bmatrix}$. These $n-1$ models are then be compared using Equation 3.2.26 and the highest likelihood model is chosen.

A visualisation of this method is shown in Figure 3.7, where the model with four lines is clearly the best model. The model likelihoods of these models are also shown in Figure 3.8, which are calculated with Equation 3.2.26. Here the maximum likelihood occur at $m = 4$ for all values of α except $\alpha = 0$, since the latter has no penalty term.

It is not clear from this example that the BIC penalises the model complexity too much. It can be argued, however, that it is better to have a model with too many lines, which is still a good definition of the landmark, than to penalise complexity severely and end up with a model that does not define the landmark well. This argument can be aided by comparing the models for $m = 3$ and $m = 5$ in Figure 3.7.

This method is presented in more detail in Algorithm 1 and Algorithm 2. The implementation of these algorithms differs in certain aspects from the pseudocode

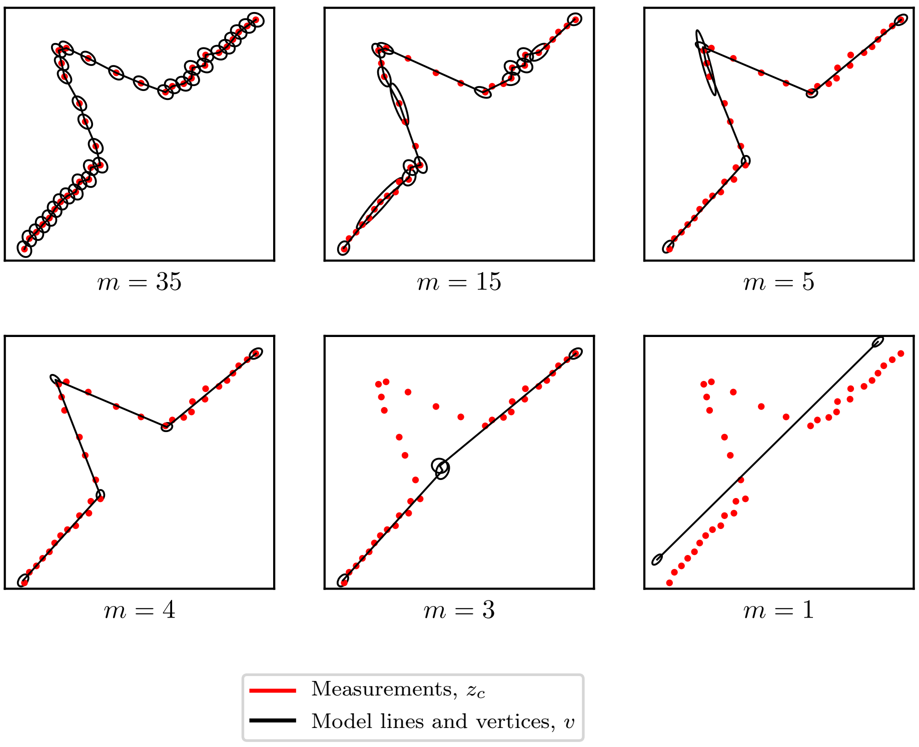


Figure 3.7 – Results of different steps in the proposed model selection algorithm. In each figure, m is the number of line fitted to the measurements.

to make it more optimal. This is mostly done to avoid recomputing the same parameters or likelihoods.

The purpose of Algorithm 1 is fit different models to the measurements and select the best model. Lines 3 to 14 in Algorithm 1 initialises the first model where all the measurements are connected as vertices and in line 18 the BIC likelihood of the initial model is calculated. The likelihood in line 10 is the lowest of the likelihoods of the measurements that are used to fit two lines. These likelihoods are subtracted from the total model likelihood in line 18, as well as line 28 and 9 in Algorithm 2. Since these measurements are used to fit two lines, when these two lines' likelihoods are added together, the effect of these measurements are considered twice. Therefore, the lowest of these two likelihoods are subtracted from the joint likelihood. In Lines 19 to 23 the number of lines is iteratively decreased using Algorithm 2, which also returns the BIC likelihood for each model. The index of the model with the maximum BIC likelihood is obtained in line 24 and this model is returned by the algorithm.

Algorithm 2 is a function used in Algorithm 1, which goes through the process of decreasing the number of lines in a model. In Lines 3 to 11, each pair of adjacent lines are joined and the difference between the likelihoods of the new line and the two separate lines are calculated. In Line 12, the best pair of lines to join is selected

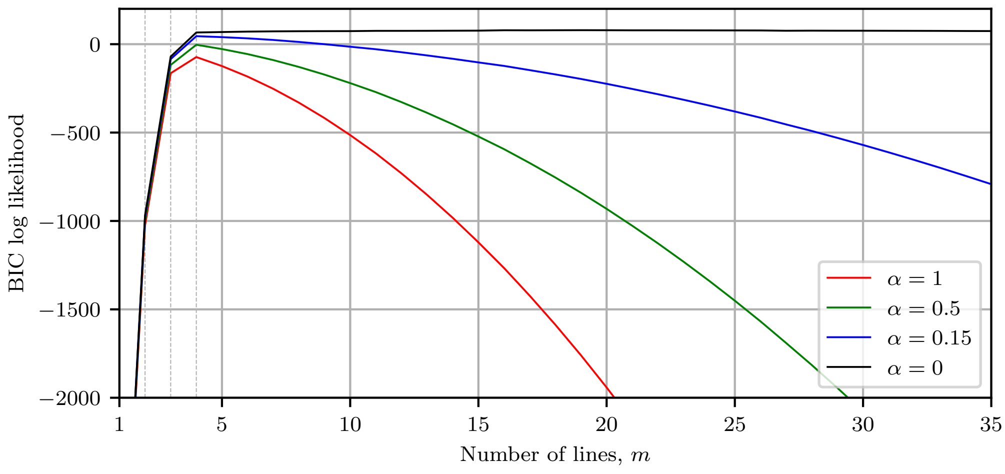


Figure 3.8 – Model likelihoods of the example in Figure 3.7 with different α values for Equation 3.2.26.

by choosing the pair that leads to the lowest decrease in likelihood. In the rest of this algorithm, the parameters of the new model are calculated, along with the likelihoods and grouping vector needed to perform this line decrease operation iteratively.

The best model selected by this method often have a lot fewer lines than the number of measurements, which is clear from Figure 3.8. This also makes intuitive sense, because if we have more measurements on a straight line, we have a better accuracy of this line.

The method used takes a lot of time to evaluate the first number of iterations, which seldom generate the best model. We thus want to initialise this model with fewer lines, which is closer to the optimal number of lines. The idea is to group consecutive measurements together as long as a straight line can be drawn so that the Mahalanobis distance between each Cartesian measurement and the closest point on the line is less than 1. This is done by grouping the first three measurements together and evaluating the above condition. The grouping is then iteratively extended by one until the condition does not hold any longer. A new grouping is then started from the last valid measurement in the previous group. This method is illustrated in Figure 3.9, where every line goes through all the 1-σ confidence ellipses of the measurements in its grouping.

To calculate the Mahalanobis distance between a measurement and a line, the distribution over the equivalent Euclidean distance first needs to be calculated. The distance between a Cartesian measurement, $\boldsymbol{z}_{c,i}$, and the closest point on a line, parameterised by $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$, can be described by the Gaussian distribution, $p(d_i) = \mathcal{N}(\mu_{d_i}, \sigma^2_{d_i})$. The mean and variance can be calculated with the following equations:

$$\begin{aligned} \mu_{d_i} &= \boldsymbol{r}^T(\boldsymbol{\mu}_{z_{c,i}} - \boldsymbol{p}_0) \\ \sigma^2_{d_i} &= \boldsymbol{r}^T\boldsymbol{\Sigma}_{z_{c,i}}\boldsymbol{r}, \end{aligned} \tag{3.2.27}$$

Algorithm 1 Model Selection

```
1: function ModelSelection($p(\boldsymbol{z}_p)$)
2:     $n :=$ number of measurements
3:     $\boldsymbol{\gamma} := \begin{bmatrix}(0,1) & (1,2) & \cdots & (n-2,n-1)\end{bmatrix}$        ▷ Initial grouping indices
4:     $p(\boldsymbol{v}) := \mathrm{UT}(p(\boldsymbol{z}_p); \boldsymbol{z}_p \mapsto \boldsymbol{z}_c)$        ▷ Create first set of vertices
5:     $\Psi_0 := \log(p(\boldsymbol{z}_{p,0}|\boldsymbol{v}_0, \boldsymbol{v}_1))$   ▷ Likelihood of first measurement given first line
6:     for $i := 0$ to $n-3$ do        ▷ Loop through all initial lines
7:         $\Psi_1 := \log(p(\boldsymbol{z}_{p,i+1}|\boldsymbol{v}_i, \boldsymbol{v}_{i+1}))$        ▷ Likelihood of second $\boldsymbol{z}_p$ on line
8:         $\Phi_i := \Psi_0 + \Psi_1$        ▷ Total log likelihood of measurements on line
9:         $\Psi_0 := \log(p(\boldsymbol{z}_{p,i+1}|\boldsymbol{v}_{i+1}, \boldsymbol{v}_{i+2}))$        ▷ Likelihood of first $\boldsymbol{z}_p$ on next line
10:        $\Phi_{\mathrm{dup},i} := \min(\Psi_0, \Psi_1)$   ▷ Minimum likelihood of $\boldsymbol{z}_{p,i+1}$ on the two lines
11:    end for
12:    $\Psi_1 := \log(p(\boldsymbol{z}_{p,n-1}|\boldsymbol{v}_{n-1}, \boldsymbol{v}_n))$   ▷ Likelihood of last measurement given last
   line
13:    $\Phi_{n-2} := \Psi_0 + \Psi_1$        ▷ Total log likelihood of measurements on last line
14:    $m := n-1$        ▷ Number of lines
15:    $M_m := p(\boldsymbol{v})$        ▷ Initial model with $m$ lines
16:    $K := 2n^2 + 3n$
17:    $N := 2n$
18:    $\mathrm{BIC}_m = \sum_i \Phi_i - \sum_i \Phi_{\mathrm{dup},i} - \alpha\frac{K}{2}\log N$        ▷ Equation 3.2.26
19:    while $m > 1$ do        ▷ Reduce number of lines up to 1
20:        $m := m-1$
21:        $p(\boldsymbol{v}), \boldsymbol{\gamma}, \boldsymbol{\Phi}, \boldsymbol{\Phi}_{\mathrm{dup}}, \mathrm{BIC}_m :=$ DecreaseLines($p(\boldsymbol{z}_p), p(\boldsymbol{v}), \boldsymbol{\gamma}, \boldsymbol{\Phi}, \boldsymbol{\Phi}_{\mathrm{dup}}$)
                                                                ▷ Algorithm 2
22:        $M_m := p(\boldsymbol{v})$
23:    end while
24:    $i := \arg\max(\mathbf{BIC})$        ▷ Choose best model with highest BIC likelihood
25:    $p(\boldsymbol{v}) := M_i$
26:    return $p(\boldsymbol{v}), \boldsymbol{\gamma}$
27: end function
```

where

$$\boldsymbol{r} = \frac{1}{d_p}\begin{bmatrix}-\Delta p_y \\ \Delta p_x\end{bmatrix} \tag{3.2.28}$$

is a unit vector in the direction normal to the line, with

$$\begin{aligned}\begin{bmatrix}\Delta p_x \\ \Delta p_y\end{bmatrix} &= \Delta\boldsymbol{p} = \boldsymbol{p}_1 - \boldsymbol{p}_0 \\ d_p &= ||\boldsymbol{p}_1 - \boldsymbol{p}_0||\end{aligned} \tag{3.2.29}$$

The Mahalanobis distance, d_{mhl}, is a normalised distance between a vector, $\boldsymbol{x}$, and a Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, over the same vector space. It can be calculated with the following equation:

$$d_{\mathrm{mhl}} = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}. \tag{3.2.30}$$

Algorithm 2 Decrease Lines for Model Selection

```
1: function DecreaseLines($p(\boldsymbol{z}_p), p(\boldsymbol{v}), \boldsymbol{\gamma}, \boldsymbol{\Phi}, \boldsymbol{\Phi}_{\text{dup}}$)
2:     $m :=$ current number of lines
3:     for $i := 0$ to $m-2$ do                              ▷ Join each pair of adjacent lines
4:         $a := \gamma_{i,0}$                                ▷ Index of first measurement on line
5:         $b := \gamma_{i,1}$                                ▷ Index of measurement shared by lines
6:         $c := \gamma_{i+1,1}$                              ▷ Index of last measurement on next line
7:         $p(\boldsymbol{p}_0, \boldsymbol{p}_1) := \text{UT}(p(\boldsymbol{z}_{p,a:c}); \boldsymbol{z}_p \mapsto \boldsymbol{p}_0, \boldsymbol{p}_1)$   ▷ Fit a line to measurements a-c
8:         $\Psi_i := \sum_{j=a}^{c} \log(p(\boldsymbol{z}_{p,j} | \boldsymbol{p}_0, \boldsymbol{p}_1))$    ▷ Equation (3.2.20)
9:         $\Phi'_i := \Phi_i + \Phi_{i+1} - \Phi_{\text{dup},i}$     ▷ Sum of likelihoods - duplicate likelihood
10:        $\Delta\Psi_i := \Phi'_i - \Psi_i$     ▷ Difference between current and new line likelihoods
11:    end for
12:    $i := \arg\min(\Delta\boldsymbol{\Psi})$                 ▷ Best pair of lines to join: line $i$ and $i+1$
13:    $m := m - 1$
14:    $\gamma_{i,1} := \gamma_{i+1,1}$
15:    remove element $i+1$ from $\boldsymbol{\gamma}$
16:    $p(\boldsymbol{v}) := \text{UT}(p(\boldsymbol{z}_p), \boldsymbol{\gamma}; \boldsymbol{z}_p \mapsto \boldsymbol{v})$       ▷ Create new set of vertices
17:    $\boldsymbol{\Phi} := \begin{bmatrix} \Phi_0 & \cdots & \Phi_{i-1} & \Psi_i & \Phi_{i+2} & \cdots & \Phi_m \end{bmatrix}^T$   ▷ Update line log likelihoods
18:    $a := \gamma_{i,0}$                                    ▷ Index of first measurement on joined line
19:    $b := \gamma_{i,1}$                                    ▷ Index of last measurement on joined line
20:    if $i > 0$ then                                       ▷ Update likelihood of left shared measurement
21:        $\Phi_{\text{dup},i-1} := \min(\log(p(\boldsymbol{z}_{p,a} | \boldsymbol{v}_i, \boldsymbol{v}_{i+1})), \log(p(\boldsymbol{z}_{p,a} | \boldsymbol{v}_{i-1}, \boldsymbol{v}_i)))$
22:    end if
23:    if $i < m-1$ then                                     ▷ Update likelihood of right shared measurement
24:        $\Phi_{\text{dup},i} := \min(\log(p(\boldsymbol{z}_{p,b} | \boldsymbol{v}_i, \boldsymbol{v}_{i+1})), \log(p(\boldsymbol{z}_{p,b} | \boldsymbol{v}_{i+1}, \boldsymbol{v}_{i+2})))$
25:    end if
26:    remove element $i+1$ from $\boldsymbol{\Phi}_{\text{dup}}$     ▷ Shared measurement of two lines joined
27:    $K := 2m^2 + 7m + 5$             ▷ Dimensionality of model parameters: $k = m + 1$
28:    $\text{BIC} := \sum_i \Phi_i - \sum_i \Phi_{\text{dup},i} - \alpha \frac{K}{2} \log N$          ▷ Equation 3.2.26
29:    return $p(\boldsymbol{v}), \boldsymbol{\gamma}, \boldsymbol{\Phi}, \boldsymbol{\Phi}_{\text{dup}}, \text{BIC}$
30: end function
```

The distribution over the distance between a measurement and a line is given by Equation 3.2.27. We want to calculate the Mahalanobis distance between 0 and this distribution, since for measurement to originate from the line this true distance should be zero. The resulting Mahalanobis distance for this scalar can thus be calculated as

$$d_{\text{mhl},i} = \sqrt{\frac{\mu_{d_i}^2}{\sigma_{d_i}^2}}. \tag{3.2.31}$$

The algorithm to find an initial grouping is presented in Algorithm 3. In Lines 5 and 6, the indices of the first group of measurements are initialised. The indices of the current group are maintained through the algorithm. Lines 7 to 21 loop through all the measurements and fit a line to the measurements in the current group in Line 8. In Lines 9 to 13 the Mahalanobis distance between the measurements in the group and the fitted line are calculated. If all these distances fall within a threshold,

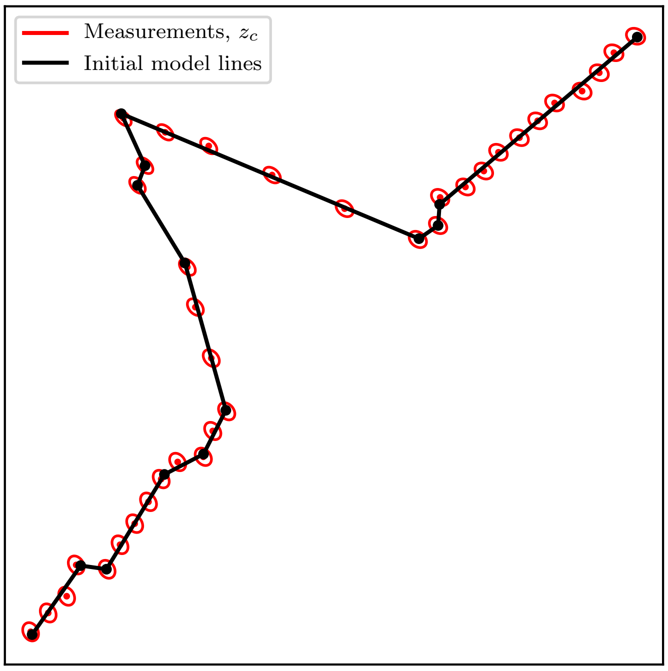


Figure 3.9 – Example of method to get an initial model. All the lines (black) goes through the 1-σ confidence ellipses of the measurements (red) of its grouping.

the group is extended and the next measurement is tested. If all the Mahalanobis distances do not fall within the threshold, the group is ended by adding its indices, without the latest measurement, to the grouping vector in Line 17 and a new group is started containing the latest two measurement. Once the loop is finished, the indices of the last group is added to the grouping vector in Line 22 and the initial model parameters are calculated in Line 23.

Once the initial model is obtained, the initial likelihoods can be computed for this model using Equation 3.2.21. Lines 3 to 14 in Algorithm 1 can now be replaced by the parameters of this new initial model. This method aims to reduce the number of loops of Line 19 to 23 in Algorithm 1 significantly, since the initial number of lines can be a lot lower than the number of measurements.

Unknown Lines

The landmark model created thus far uses the Lidar measurements generated from the landmark object, which models object boundaries. By investigating occlusion boundaries around the object, we can reason about expected measurements around the object from regions that are not currently observed. These occlusion boundaries are obtained using the measurements adjacent to the ones on the object. We aim to use this information to extend the model of the landmark. We still assume the Lidar pose is known.

The lines in the landmark model developed so far are fitted to the measurements

Algorithm 3 Initialise Model Selection Algorithm

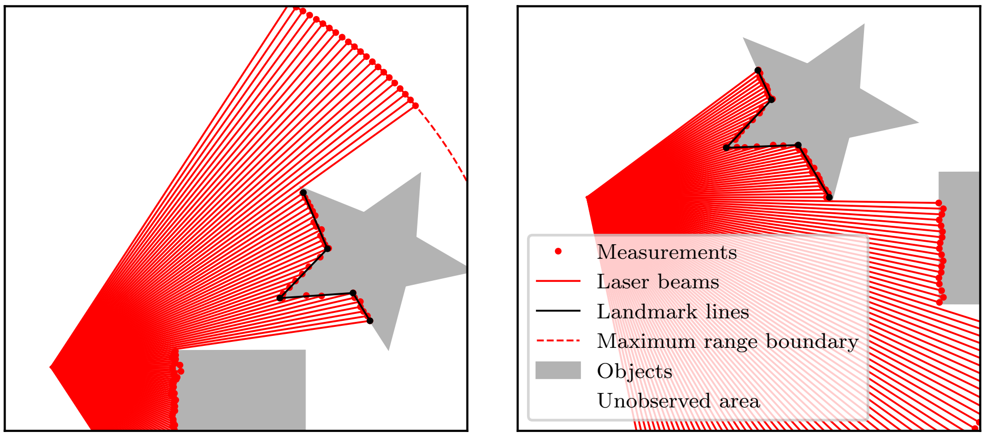
```
1: function InitialModel($p(\boldsymbol{z}_p)$)
2:     $n$ := number of measurements
3:     $\boldsymbol{\gamma} := []$                                   ▷ Initialise empty grouping vector
4:     $p(\boldsymbol{z}_c) := \mathrm{UT}(p(\boldsymbol{z}_p); \boldsymbol{z}_p \mapsto \boldsymbol{z}_c)$       ▷ Calculate the Cartesian measurements
5:     $a := 0$                                   ▷ Index of first measurement in current group
6:     $b := 2$                                   ▷ Index of last measurement in current group
7:     while $b < n$ do
8:         $\boldsymbol{p}_0, \boldsymbol{p}_1 := \boldsymbol{f}(\boldsymbol{\mu}_{\boldsymbol{z}_{c,a:b}}) : \boldsymbol{z}_c \mapsto \boldsymbol{p}_0, \boldsymbol{p}_1$     ▷ Fit a line to measurements $a$-$b$
                                                                  ▷ Equation 3.2.3
9:         for $i := a$ to $b$ do               ▷ Loop through all measurements in group
10:            $\mu_{d_i} = \boldsymbol{r}^T(\boldsymbol{\mu}_{z_{c,i}} - \boldsymbol{p}_0)$
11:            $\sigma^2_{d_i} = \boldsymbol{r}^T \boldsymbol{\Sigma}_{z_{c,i}} \boldsymbol{r}$
12:            $d_{\mathrm{mhl},i} := \sqrt{\frac{\mu^2_{d_i}}{\sigma^2_{d_i}}}$                  ▷ Calculate Mahalanobis distance
13:        end for
14:        if all $d_{\mathrm{mhl},i} \leq 1$ for i = [a,b] then     ▷ Measurements form straight line
15:            $b := b + 1$                               ▷ Increase the group size
16:        else                                  ▷ Measurements don't form straight line
17:            append $(a, b-1)$ to $\boldsymbol{\gamma}$               ▷ Add previous valid group's indices
18:            $a := b - 1$                                   ▷ Start a new group
19:            $b := a + 2$                     ▷ Initialise new group with 3 measurements
20:        end if
21:    end while
22:    append $(a, n-1)$ to $\boldsymbol{\gamma}$                          ▷ Add final group's indices
23:    $p(\boldsymbol{v}) := \mathrm{UT}(p(\boldsymbol{z}_p), \boldsymbol{\gamma}; \boldsymbol{z}_p \mapsto \boldsymbol{v})$                   ▷ Create initial model
24:    return $p(\boldsymbol{v}), \boldsymbol{\gamma}$
25: end function
```

of the landmark. These are boundaries between empty space and occupied space. We now want to use the measurements adjacent to the landmark to define lines that are boundaries between open space and unknown space, which we refer to as unknown lines. These lines could aid in detecting and updating a landmark when new segments of it are observed.

The adjacent measurements referred to here are the measurements to the immediate left, $\boldsymbol{z}_l = \begin{bmatrix} r_l & \phi_l \end{bmatrix}^T$, and right, $\boldsymbol{z}_r = \begin{bmatrix} r_r & \phi_r \end{bmatrix}^T$, of the measurements on the landmark, where $\phi_l = \phi_0 + \phi_{\mathrm{res}}$ and $\phi_r = \phi_{n-1} - \phi_{\mathrm{res}}$. The angle ϕ_{res} is the angular resolution of the Lidar.

To create these lines, we have to know why no further measurements are obtained from the landmark, so that we can reason about where the empty space and unknown space are. We identify three reasons that the measurements of a landmark has ended. The first is that the edge of the landmark is reached and the rest of the landmark is hidden behind the part of the landmark that is visible to the Lidar. In this case we expect the measurement adjacent to the landmark to be further than the measurement on the landmark, or that there will be no measurement due to the

Lidar sensor's maximum range, r_{max}. The second reason is that the rest of the landmark is occluded by another object, in which case the adjacent measurement is closer than the measurement on the landmark. The last is that the landmark is at the edge of the Lidar sensor's field-of-view (FOV), in which case the measurement on the landmark is at the Lidar sensor's minimum or maximum angle. An illustration of these occlusions and FOV boundaries is shown in Figure 3.10.



(a) (b)

Figure 3.10 – Illustration of landmark measurements, showing occlusions, maximum range boundaries and FOV boundaries. The red lines show the laser beams in open space and the white areas indicate unobserved areas. In (a) the measurements on the left of the star-like landmark show that the rest of the landmark is hidden behind the observed landmark and the measurements on the right shows that part of the star-like landmark is occluded by another object. In (b) the star-like landmark is on the left edge of the Lidar's FOV.

The unknown lines will form part of the landmark model and the creation thereof is now explained. These unknown lines will add a number of vertices to the left, $\boldsymbol{v}_l$, and to the right, $\boldsymbol{v}_r$, of the landmark model. The creation of the unknown lines at the two sides of the landmark model is explained together for simplicity.

In this explanation, we assume that maximum range measurements are filtered out and in such a case the sensor returns no measurement. If there is a measurement adjacent to the landmark measurements, it either means there is an object in front of the landmark, as seen on the right side measurements of Figure 3.10(a), or behind the landmark, as seen on the right side measurements of Figure 3.10(b). In both cases we want to create an unknown line from the edge of the landmark toward this adjacent measurement, $\boldsymbol{z}_{p,l}$ or $\boldsymbol{z}_{p,r}$ for the left and right adjacent measurement respectively, as seen in Figure 3.11(a) and (b) on the right side measurements.

If there are no measurements adjacent to the landmark measurements, it either means that the edge of the landmark, from the Lidar's perspective, is reached (Figure 3.10(a)) or that the landmark is on the edge of the Lidar's FOV (Figure 3.10(b)).

For the first case, we want to add an unknown line from the edge of the landmark toward the maximum range of the Lidar and another one from there on the maximum range boundary away from the landmark to indicate that we have no information about the region further away than the maximum range, as seen in Figure 3.11(a). We do this by adding two artificial measurements to the left or right,

$$\begin{aligned} \boldsymbol{z}_{p,l} &= \begin{bmatrix} r_{\text{max}} & \phi_0 + 5\phi_{\text{res}} & r_{\text{max}} & \phi_0 + \phi_{\text{res}} \end{bmatrix}^T \\ \boldsymbol{z}_{p,r} &= \begin{bmatrix} r_{\text{max}} & \phi_{n-1} - \phi_{\text{res}} & r_{\text{max}} & \phi_{n-1} - 5\phi_{\text{res}} \end{bmatrix}^T, \end{aligned} \tag{3.2.32}$$

where r_{max} is the maximum range of the Lidar and ϕ_{res} is the angular resolution of the Lidar. The second added artificial measurement is chosen at 5 angular resolutions away on the maximum range boundary to approximate this boundary as a straight line.

For the second case, if the last measurement of the landmark is on the edge of the Lidar's FOV, we want to create an unknown line from the edge of the landmark toward the Lidar's position, as seen in Figure 3.11(b) on the left side measurements. We do this by adding an artificial zero range measurement to the left or right,

$$\begin{aligned} \boldsymbol{z}_{p,l} &= \begin{bmatrix} 0 & \phi_0 + \phi_{\text{res}} \end{bmatrix}^T \\ \boldsymbol{z}_{p,r} &= \begin{bmatrix} 0 & \phi_{n-1} - \phi_{\text{res}} \end{bmatrix}^T. \end{aligned} \tag{3.2.33}$$

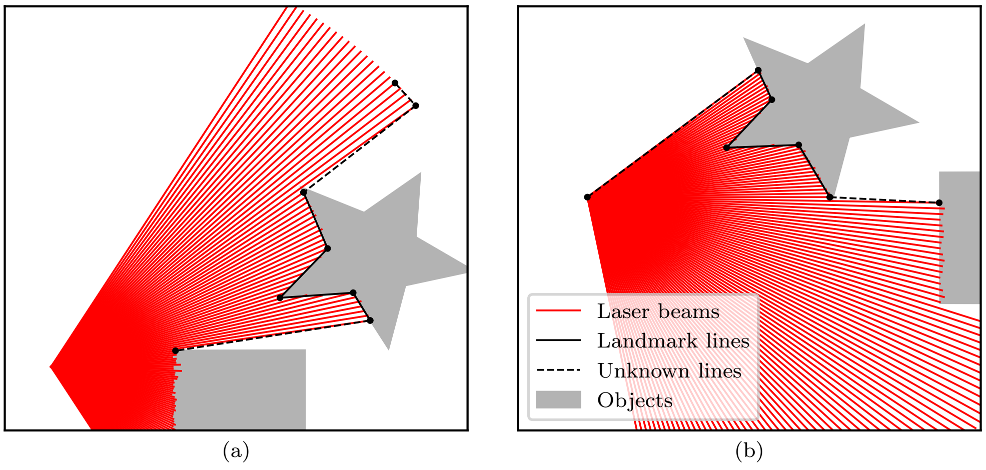


Figure 3.11 – Unknown lines added to the landmark model. The unknown lines are added to the landmark models of the example in Figure 3.10. The means of the unknown lines are displayed by the black dotted lines.

The unknown lines are created by drawing a line from the landmark's edge toward the adjacent measurements or artificial measurements in their Cartesian coordinates in the inertial reference frame. This is illustrated in Figure 3.11, for the example in Figure 3.10. We thus transform the distributions over the adjacent polar measurements to distributions over the Cartesian measurements and set them as the distributions of the left and right vertices,

$$\begin{aligned} p(\boldsymbol{v}_l) = p(\boldsymbol{z}_{c,l}) &= \mathrm{UT}(\mathcal{N}(\boldsymbol{z}_{p,l}, \boldsymbol{\Sigma}_{p,l}); \boldsymbol{z}_p \mapsto \boldsymbol{z}_c) \\ p(\boldsymbol{v}_r) = p(\boldsymbol{z}_{c,r}) &= \mathrm{UT}(\mathcal{N}(\boldsymbol{z}_{p,r}, \boldsymbol{\Sigma}_{p,r}); \boldsymbol{z}_p \mapsto \boldsymbol{z}_c), \end{aligned} \tag{3.2.34}$$

where $\boldsymbol{\Sigma}_{p,l}$ and $\boldsymbol{\Sigma}_{p,r}$ are populated with $\boldsymbol{Q}$ as in Equation 3.2.13. Once the distributions over these vertices are calculated, they are added to the landmark model,

$$p(\boldsymbol{v}') = \mathcal{N}(\boldsymbol{\mu}_v', \boldsymbol{\Sigma}_v') = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{v,l} \\ \boldsymbol{\mu}_v \\ \boldsymbol{\mu}_{v,r} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{v,l} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_v & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{v,r} \end{bmatrix} \right), \tag{3.2.35}$$

where we assume that the existing vertices and added vertices are statistically independent. From now on, we will use $\boldsymbol{v}$ to refer to the model with the unknown lines added, omitting the accent of Equation 3.2.35.

After the unknown lines are added to the model, we create a vector, $\boldsymbol{u} = \begin{bmatrix} u_0 & \cdots & u_{m-1} \end{bmatrix}^T$, where u_i indicates whether the i^{th} line is an object boundary or an unknown boundary. A value of $u_i = 1$ is given to each unknown line and $u_i = 0$ is given to each object boundary.

The idea of the unknown boundary is also extended to the already existing lines of the landmark, which are fitted to the measurements. This is done to account for the uncertainty of what an object looks like between measurements, especially the chance that parts in the middle of the landmark could be occluded by the landmark itself, which is illustrated in Figure 3.12. Each element in the vector, $\boldsymbol{u}$, now denotes the probability that the line is an unknown boundary.

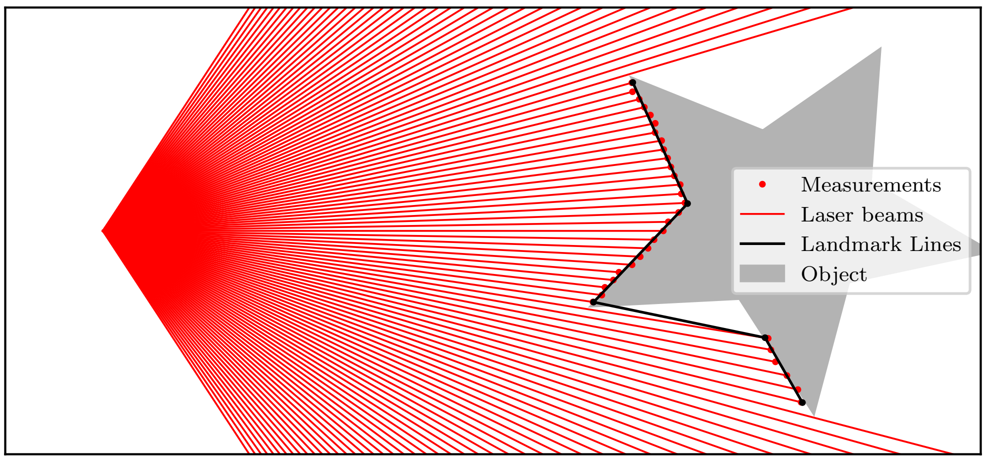


Figure 3.12 – Illustration of a landmark occluding parts of itself. The third line of the landmark model is not a line on the object, but rather a boundary between observed and unobserved areas.

We approximate this probability by evaluating the average distance on a line, d_i, between the measurements it is fitted to. This probability is approximated with a piecewise linear function,

$$u_i = \begin{cases} 0; & d_i \leq \epsilon_l \\ \frac{d_i - \epsilon_l}{\epsilon_u - \epsilon_l}; & \epsilon_l < d_i < \epsilon_u \\ 1; & d_i \geq \epsilon_u \end{cases}, \tag{3.2.36}$$

where d_i is the length of the line divided by the number of measurements the line is fitted to, minus one,

$$d_i = \frac{\left|\left|\boldsymbol{\mu}_{v,i+1} - \boldsymbol{\mu}_{v,i}\right|\right|}{\gamma_{i,1} - \gamma_{i,0}}. \tag{3.2.37}$$

The indices represented by the grouping vector, $\boldsymbol{\gamma}$, is used to determine the number of measurements a line is fitted to. ϵ_l and ϵ_u in Equation 3.2.36 are the lower and upper boundaries of the piecewise linear function. These values are chosen as $\epsilon_l = 0.2$ and $\epsilon_u = 0.8$, but can be adapted for different applications according to the Lidar sensor's angular resolution. With this method we assume that if the measurements are close together, the fitted line is an object boundary, but if they are too far apart the fitted line is an unknown boundary and the region behind the line is unknown.

Since an unknown line is not an object boundary, the likelihood function for measurements corresponding to this line is different than the likelihood function for object lines. To derive this likelihood function, we first look at what we expect the distribution of Lidar measurements would look like with no prior information of landmarks in the environment.

The Lidar sensor measures the closest object in a certain direction, therefore it is more likely to have a closer measurement than a farther one. Having no prior information about the environment, we assume that objects in the environment are distributed uniformly. The exponential distribution models the distance to the first object in a uniformly distributed environment, therefore we can approximate the prior distribution over global Lidar range measurements as an exponential distribution,

$$p(r) = \lambda \exp(-\lambda r)\mathrm{u}(r)\mathrm{u}(r_{\max} - r) + \exp(-\lambda r_{\max})\delta(r - r_{\max}), \tag{3.2.38}$$

where λ relates to the density of objects in the environment. The notation, $\mathrm{u}(x)$, is used for the step function. This distribution is cut off at $r_{\max}$ due to the fact that no measurement is greater than the Lidar's maximum range. A Dirac delta function, $\delta(x)$, is added at the maximum range, with a weight to normalise the distribution. This impulse is due to the fact that the Lidar sensor returns an $r_{\max}$ measurement if nothing is measured in that direction. This function is shown in Figure 3.13.

In Equation 3.2.38, λ is a parameter of the probability density function. This parameter needs to be set to use this equation in likelihood estimation. We choose to estimate this parameter with maximum likelihood estimation from the measurements obtained from the environment. The density parameter can be estimated as the number of measurements divided by the sum of all the measurements, which is the inverse of the mean. For the exponential distribution which is cut off at $r_{\max}$,

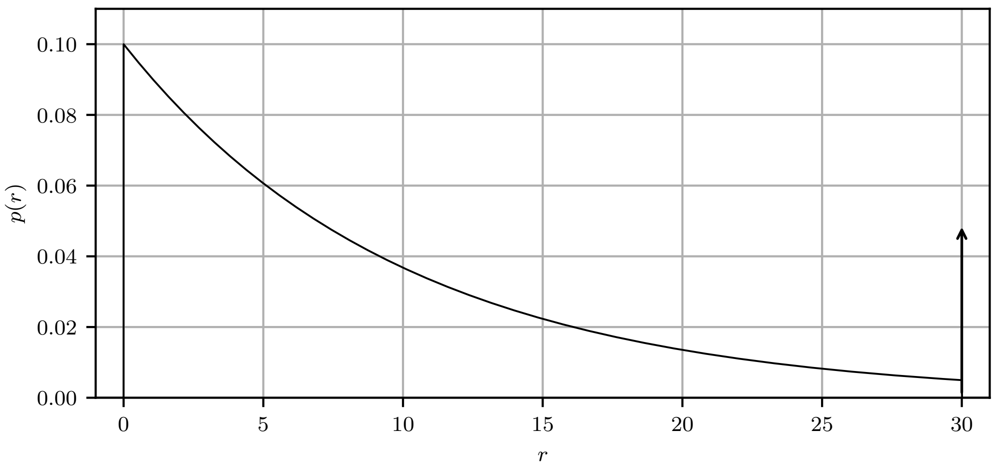


Figure 3.13 – Graph of the exponential distribution with a cut-off at the maximum range measurement, where the range on the x-axis is in metres ($\lambda = 0.1$ and $r_{\max} = 30$ m).

the number of $r_{\max}$ measurements also have an effect, which leads to the following equation:

$$\lambda = \frac{n_z}{n_z \overline{r} + n_r r_{\max}}. \tag{3.2.39}$$

where n_z is the number of measurements not having a maximum range measurement (an actual point measured), n_r is the number of measurements with a maximum range measurement (nothing measured) and $\overline{r}$ is the mean of the actual measurements.

Since the unknown lines are boundaries between empty and unobserved space, we can approximate the distribution of measurements from the region behind an unknown line with an exponential distribution. The zero point of this distribution is on the unknown line, since we know the area in front of the line is empty space.

These lines are, however, parameterised by two vertices with Gaussian distributed uncertainty over them and the measurements also have Gaussian distributed uncertainty over them. The distribution over the error between a measurement and the point that generated the measurement, $\mathcal{N}\left(\mu_{r_{\text{err}}}, \sigma^2_{\text{err}}\right)$, can be obtained with Equation 3.2.19, which uses the unscented transform to linearise this error. The likelihood function associated with these lines is a convolution between the Gaussian distribution over this error and the exponential distribution of the unknown line. However, we assume that the maximum range, $r_{\max}$, used in Equation 3.2.38 is far enough to have a trivial effect on the probability density function. We therefore model the distribution over measurements associated with an unknown line as a general exponential distribution without the cutoff and impulse at $r_{\max}$. The convolution between the Gaussian distribution and the general exponential distribution is calculated here:

$$\begin{aligned}p(x) =&\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right) * \lambda\exp(-\lambda x)\mathrm{u}(x)\\ =&\int_0^\infty \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\tau)^2}{2\sigma^2}\right)\lambda\exp(-\lambda\tau)\mathrm{d}\tau\\ =&\frac{\lambda}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2-\left(x-\lambda\sigma^2\right)^2}{2\sigma^2}\right)\int_0^\infty\exp\left(-\frac{\left(\tau-(x-\lambda\sigma^2)\right)^2}{2\sigma^2}\right)\mathrm{d}\tau\\ =&\frac{\lambda}{\sqrt{2\pi\sigma^2}}\exp\left(-\lambda x+\frac{\lambda^2\sigma^2}{2}\right)\left[\sqrt{\frac{\pi\sigma^2}{2}}\mathrm{erf}\left(-\frac{\tau-(x-\lambda\sigma^2)}{2\sigma^2}\right)\right]_{\tau=0}^{\tau=\infty}\\ =&\frac{\lambda}{2}\exp\left(-\lambda x+\frac{\lambda^2\sigma^2}{2}\right)\left(1-\mathrm{erf}\left(-\frac{x-\lambda\sigma^2}{\sqrt{2\sigma^2}}\right)\right),\end{aligned} \tag{3.2.40}$$

where $\mathrm{erf}(x)$ is the error function, defined as

$$\mathrm{erf}(x) = \frac{1}{\sqrt{\pi}}\int_{-x}^{x}\exp\left(-t^2\right)\mathrm{d}t. \tag{3.2.41}$$

The error function can be approximated with numerical methods and there exists implementations of it for many programming languages. The unknown distribution along with the exponential and Gaussian distributions are shown in Figure 3.14.

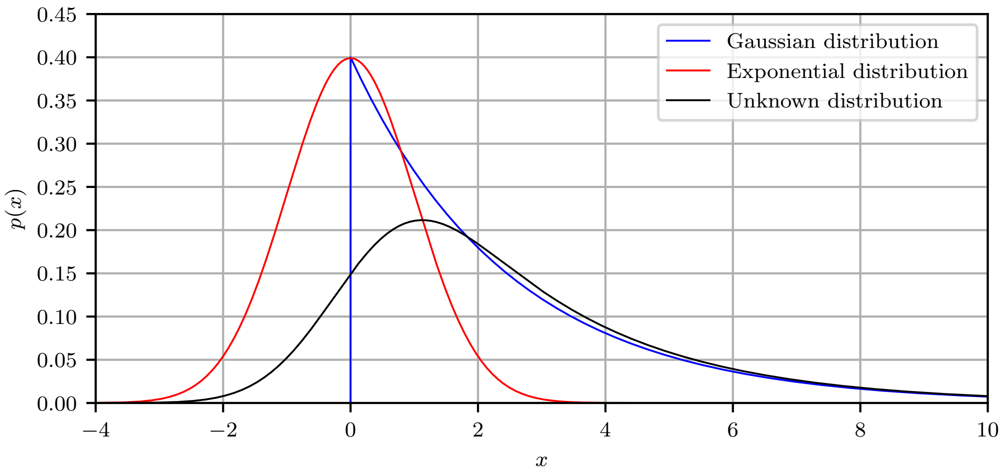


Figure 3.14 – Graph of the unknown distribution (black), which is a convolution of the exponential (blue) and Gaussian (red) distributions ($\lambda = 0.4$ and $\sigma^2 = 1$).

For the likelihood of a measurement obtained from an unknown line, we substitute $x := \mu_{r_{\mathrm{err},i,j}}$ and $\sigma := \sigma_{r_{\mathrm{err},i,j}}$ into Equation 3.2.40. The likelihood is then given by

$$p(\boldsymbol{z}_{p,i}|\boldsymbol{v}_j, \boldsymbol{v}_{j+1}, u_j = 1) = \frac{\lambda}{2}\exp\left(-\lambda\mu_{r_{\mathrm{err},i,j}} + \frac{\lambda^2\sigma^2_{r_{\mathrm{err},i,j}}}{2}\right) \times \left(1 - \mathrm{erf}\left(-\frac{\mu_{r_{\mathrm{err},i,j}} - \lambda\sigma^2_{r_{\mathrm{err},i,j}}}{\sqrt{2\sigma^2_{r_{\mathrm{err},i,j}}}}\right)\right). \tag{3.2.42}$$

For the case where the type of boundary a line describes is uncertain ($0 < u_j < 1$), the likelihood for a measurement corresponding to this line is obtained using the total probability, which takes into account the probability of the line's state and the probability density functions for each state, given by Equation 3.2.20 and 3.2.42. The likelihood function associated with a general line in the model is thus

$$p(\boldsymbol{z}_{p,i}|\boldsymbol{v}_j, \boldsymbol{v}_{j+1}) = (1-u_j)p(\boldsymbol{z}_{p,i}|\boldsymbol{v}_j, \boldsymbol{v}_{j+1}, u_j = 0) + u_j p(\boldsymbol{z}_{p,i}|\boldsymbol{v}_j, \boldsymbol{v}_{j+1}, u_j = 1). \tag{3.2.43}$$

These unknown lines can be used to obtain a likelihood that a set of measurements are from a certain landmark, even if parts of the landmark measured now have not been observed before. It can also be used to find out in what areas the landmark model can be extended by using new measurements.

Conditional Landmark Modelling

Throughout this section we assume that the pose of the Lidar is known, which means that the distribution over the vertices is actually a conditional distribution, given the Lidar pose, $p(\boldsymbol{v}|\boldsymbol{x}_{\mathrm{pose}})$. To perform SLAM with these landmark models, which is discussed in Chapter 4, we need to take the robot pose uncertainty into account. We still assume a known pose for now and discuss obtaining the parameters of this conditional distribution given the known pose. This allows us to obtain a joint distribution over the vertices and pose once the pose uncertainty is taken into account in Chapter 4.

This conditional Gaussian distribution can be approximated by a noisy linear relationship between $\boldsymbol{x}_{\mathrm{pose}}$ and $\boldsymbol{v}$, given by

$$\boldsymbol{v} = \boldsymbol{A}_{vx}\boldsymbol{x}_{\mathrm{pose}} + \boldsymbol{b}_{vx} + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_v), \tag{3.2.44}$$

where the noise covariance is the uncertainty over the vertices from the modelling process. The linear relationship between $\boldsymbol{x}_{\mathrm{pose}}$ and each vertex, $\boldsymbol{v}_i$ can similarly be written as

$$\boldsymbol{v}_i = \boldsymbol{A}_{vx,i}\boldsymbol{x}_{\mathrm{pose}} + \boldsymbol{b}_{vx,i} + \boldsymbol{n}_i, \quad \boldsymbol{n}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{v,i,i}). \tag{3.2.45}$$

To obtain this linear relationship, we need to first obtain the actual relationship between $\boldsymbol{x}_{\mathrm{pose}}$ and $\boldsymbol{v}_i$. We can define the vertices in polar coordinates in the Lidar's reference frame, with

$$\boldsymbol{v}_{p,i} = \begin{bmatrix} v_{r,i} \\ v_{\phi,i} \end{bmatrix} = \begin{bmatrix} ||\boldsymbol{v}_i - \boldsymbol{x}_{\mathrm{pos}}|| \\ \arctan\left(\frac{v_{y,i}-y}{v_{x,i}-x}\right) - \theta \end{bmatrix}, \tag{3.2.46}$$

which is the inverse function of Equation 3.2.1 and $\boldsymbol{x}_{\mathrm{pos}} = \begin{bmatrix} x & y \end{bmatrix}^T$ is the position of the Lidar. The vertices in Cartesian coordinates in the inertial reference frame can then be expressed as a function of $\boldsymbol{x}_{\text{pose}}$ and $\boldsymbol{v}_{p,i}$, given by Equation 3.2.1, which we rewrite here for $\boldsymbol{v}_i$,

$$\boldsymbol{v}_i = \boldsymbol{x}_{\text{pos}} + v_{r,i} \begin{bmatrix} \cos(\theta + v_{\phi,i}) \\ \sin(\theta + v_{\phi,i}) \end{bmatrix}. \tag{3.2.47}$$

Linearising Equation 3.2.47 leads to the linear relationship in Equation 3.2.45. We choose to use the Taylor series expansion in this linearisation, rather than the unscented transform, since we assume $\boldsymbol{x}_{\text{pose}}$ is known and we do not use the distribution over $\boldsymbol{x}_{\text{pose}}$ in the modelling process. $\boldsymbol{A}_{vx,i}$ is thus the Jacobian of Equation 3.2.47, with respect to $\boldsymbol{x}_{\text{pose}}$, which we can calculate as

$$\boldsymbol{A}_{vx,i} = \begin{bmatrix} 1 & 0 & -v_{r,i} \sin(\theta + v_{\phi,i}) \\ 0 & 1 & v_{r,i} \cos(\theta + v_{\phi,i}) \end{bmatrix}. \tag{3.2.48}$$

Since we have a distribution over $\boldsymbol{v}_i$, we use the mean thereof to calculate $\boldsymbol{v}_{p,i}$ using Equation 3.2.46.

$\boldsymbol{A}_{vx}$ can now be populated with all its elements, $\boldsymbol{A}_{vx} = \begin{bmatrix} \boldsymbol{A}_{vx,0}^T & \cdots & \boldsymbol{A}_{vx,m}^T \end{bmatrix}^T$, and $\boldsymbol{b}_{vx}$ can be calculated by substituting the mean of $\boldsymbol{v}$ and $\boldsymbol{n}$ into Equation 3.2.44,

$$\boldsymbol{b}_{vx} = \boldsymbol{\mu}_v - \boldsymbol{A}_{vx}\boldsymbol{x}_{\text{pose}}. \tag{3.2.49}$$

The parameters of this relationship, $\boldsymbol{A}_{vx}$, $\boldsymbol{b}_{vx}$ and $\boldsymbol{\Sigma}_v$, are used in the SLAM algorithm in Chapter 4.

3.2.2 Landmark Detection

When new measurements of the environment are taken, an important aspect in landmark-based SLAM is to know whether these measurements come from a landmark already modelled or if it is part of a previously unobserved part of the environment. We now discuss how to decide whether a set of measurements comes from a landmark or not, given a known or estimated Lidar pose.

We have already discussed the likelihood that a set of measurements comes from a landmark, but this likelihood requires that each measurement is associated with a line of the landmark model, denoted by a correspondence vector, $\boldsymbol{c}$. For vertices and measurements with no uncertainty, we can associate a measurement by extending the line between the Lidar and the measurement and finding the first line of the landmark it intersects with. This is done by transforming the vertices to polar coordinates in the Lidar's reference frame, $\boldsymbol{v}_p = \begin{bmatrix} \boldsymbol{v}_{p,0}^T & \cdots & \boldsymbol{v}_{p,m}^T \end{bmatrix}^T$, using Equation 3.2.46, and finding the two adjacent vertices of which the angles are respectively greater and smaller than the measurement angle. Each element of the correspondence vector can then be calculated accordingly,

$$c_i = j \quad \text{if} \quad v_{\phi,j+1} \leq \phi_i \leq v_{\phi,j}. \tag{3.2.50}$$

If multiple lines fit this criteria, the closest one is selected for correspondence, since the others are obstructed by the closest line.

Assuming that we have noiseless measurements and no uncertainty over the vertices could lead to bad associations. The uncertainty therefore needs to be taken into account when making these associations. We thus rather find the distribution over the vertices' polar coordinates,

$$p(\boldsymbol{v}_{p,j}) = \mathcal{N}(\boldsymbol{\mu}_{v,p,j}, \boldsymbol{\Sigma}_{v,p,j}) = \mathrm{UT}(p(\boldsymbol{v}); \boldsymbol{v}_j \mapsto \boldsymbol{v}_{p,j}). \tag{3.2.51}$$

By adding the measurement noise to a polar vertex, $\boldsymbol{v}_{p,j}$, we obtain the distribution of the point where we expect the measurement of each vertex to come from,

$$p(\boldsymbol{v}'_{p,j}) = \mathcal{N}(\boldsymbol{\mu}'_{v,p,j}, \boldsymbol{\Sigma}'_{v,p,j}) = \mathcal{N}(\boldsymbol{\mu}_{v,p,j}, \boldsymbol{\Sigma}_{v,p,j} + \boldsymbol{Q}), \tag{3.2.52}$$

where

$$\boldsymbol{\mu}'_{v,p,j} = \begin{bmatrix} \mu_{v,r,j} \\ \mu_{v,\phi,j} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}'_{v,p,j} = \begin{bmatrix} {\sigma'_{v,r,j}}^2 & \rho_j \sigma'_{v,r,j}\sigma'_{v,\phi,j} \\ \rho_j \sigma'_{v,r,j}\sigma'_{v,\phi,j} & {\sigma'_{v,\phi,j}}^2 \end{bmatrix}. \tag{3.2.53}$$

With these expected vertex measurements, a list of all lines a measurement possibly comes from, ς_i, is constructed. This is done by evaluating if the measurement angle is between the two vertices' expected angle measurement's 3-σ bound,

$$j \in \varsigma_i \quad \text{if} \quad \mu_{v,\phi,j+1} - 3\sigma'_{v,\phi,j+1} \leq \phi_i \leq \mu_{v,\phi,j} + 3\sigma'_{v,\phi,j}. \tag{3.2.54}$$

The likelihoods of each line in this list are evaluated with Equation 3.2.43 and the association is made with the line with the highest likelihood,

$$c_i = \arg\max_j p(\boldsymbol{z}_{p,i} | \boldsymbol{v}_j, \boldsymbol{v}_{j+1}) \quad \text{for} \quad j \in \varsigma_i. \tag{3.2.55}$$

An example of associations made to lines is shown in Figure 3.15.

If a measurement is not associated with any line using Equation 3.2.55, it is automatically associated with the first or last line of the model, depending if it is to the left or right of the model. This is done to ensure that measurements that partly fits the model well are rejected if the rest of the measurements do not fit the model.

With these associations, we can reason whether a set of measurements come from a specific landmark. The likelihood function is, however, not a good measure to reason whether a set of measurements come from a landmark, because it is not a normalised function and it depends heavily on the number of measurements and the uncertainty over the vertices. It is therefore very difficult to say what a good likelihood value is for a set of measurements. We therefore evaluate the Mahalanobis distance of the error between measurements and lines to decide whether the measurements should be associated with a landmark.

The Mahalanobis distance is already defined in Equation 3.2.30 for Gaussian distributions, but the measurements associated with the unknown lines in the landmark model are, however, not Gaussian distributed. We therefore need to define a generalised Mahalanobis distance equivalent, d_{GM}, for non-Gaussian distributions. A method for this generalisation has been proposed by Martos *et al.* [20], where

$$d_{\mathrm{GM}}^2 = \log\left(\frac{p(x_{\max})}{p(x)}\right). \tag{3.2.56}$$

Here $x_{\max}$ is the mode of p and can be calculated by finding the derivative of $p(x)$ and setting it to zero,

$$p(x_{\max}) = \max_x p(x) \quad \text{where} \quad \left.\frac{\mathrm{d}p(x)}{\mathrm{d}x}\right|_{x=x_{\max}} = 0. \tag{3.2.57}$$

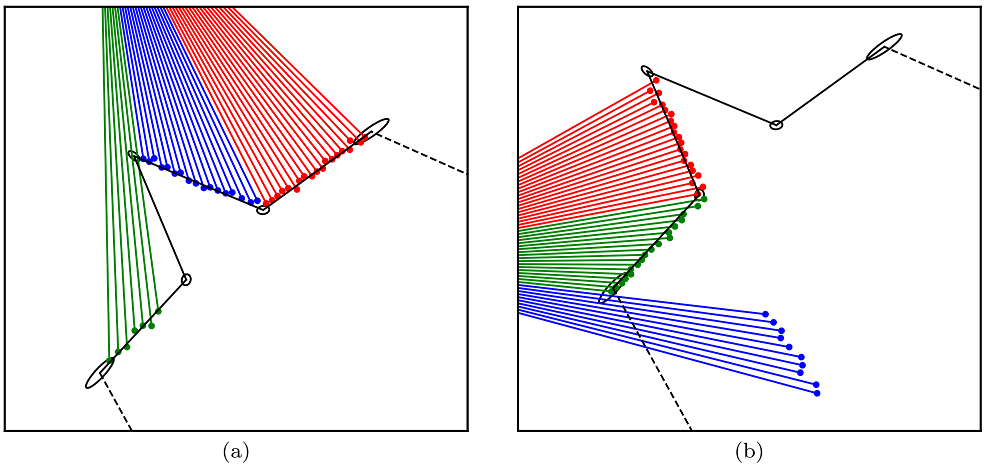


Figure 3.15 – Example of measurements associated with lines. The different colours indicate measurements associated with different lines. All the measurements of which the laser beam goes through the 3-σ confidence ellipse of the vertices are evaluated with the different lines and associated with the one with the highest likelihood.

If the distribution over x is a Gaussian distribution, this equation simplifies to the Mahalanobis distance of Equation 3.2.30.

We want to get the generalised Mahalanobis distance for the distribution in Equation 3.2.43 that we rewrite here in terms of x:

$$\begin{aligned} p(x) =&(1-u)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \\ &+u\frac{\lambda}{2}\exp\left(\frac{\lambda^2\sigma^2}{2}-\lambda x\right)\left(1-\operatorname{erf}\left(-\frac{x-\lambda\sigma^2}{\sqrt{2\sigma^2}}\right)\right). \end{aligned} \tag{3.2.58}$$

The derivative of this function can be calculated as

$$\begin{aligned} \frac{\mathrm{d}p(x)}{\mathrm{d}x} =&(1-u)\frac{x}{\sigma^2\sqrt{2\pi\sigma^2}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \\ &+u\lambda\exp\left(\frac{\lambda^2\sigma^2}{2}-\lambda x\right) \\ &\times\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(x-\lambda\sigma^2\right)^2}{2\sigma^2}\right)-\left(1-\operatorname{erf}\left(-\frac{x-\lambda\sigma^2}{\sqrt{2\sigma^2}}\right)\right)\right]. \end{aligned} \tag{3.2.59}$$

The zero-crossing of this function cannot be calculated analytically, thus we rather approximate it with a numerical method. Since Equation 3.2.58 has only one global maximum with no minima, Equation 3.2.59 only has one zero-crossing. The zero-crossing is always at $x \geq 0$ due to the properties of the exponential distribution. The numerical method we use performs a binary search between $x = 0$ and $x = 5\sigma$ if the function is greater than zero at the upper bound. If the function at the upper bound is less than zero, this value is used as the lower bound and the upper bound is doubled. This binary search process is repeated until the function at the upper and lower bounds are sufficiently close to zero.

The Mahalanobis distance can now be found for each measurement using the error between the measurement and its corresponding line, or $p(e_{\text{err}})$, calculated with Equation 3.2.19, substituting it into Equation 3.2.58 and calculating d_{GM} with Equation 3.2.56. The square of the Mahalanobis distances of the error between the set of measurements and the landmark lines are expected to be chi-squared distributed with a mean of 1 and variance of 2 [21]. For the set of Mahalanobis distances, we calculate the mean,

$$\mu_{\text{GM}} = \frac{1}{n} \sum_i d^2_{\text{GM},i} \tag{3.2.60}$$

and variance

$$\sigma^2_{\text{GM}} = \frac{1}{n} \sum_i \left(d^2_{\text{GM},i} - \mu_{\text{GM}} \right)^2, \tag{3.2.61}$$

where $d_{\text{GM},i}$ is the Mahalanobis distance of the i^{th} measurement. These mean and variance values are evaluated and if they fall below satisfactory thresholds, $\mu_{\text{GM}} \leq \mu_{\text{thres}}$ and $\sigma^2_{\text{GM}} \leq \sigma^2_{\text{thres}}$, we assume that the measurements come from the landmark. We choose these thresholds as $\mu_{\text{thres}} = 3$ and $\sigma^2_{\text{thres}} = 3$, which is not very strict on associating a set of measurements to a landmark. These choices are made especially because the Lidar pose is assumed to be known in this section, but in the complete SLAM application, this pose will only be an estimate of the true pose, which will have uncertainty over it.

3.2.3 Landmark Update

Once a new set of measurements is associated with a landmark, it can be used to update the landmark model. These measurements are used to update the observed part of the landmark, as well as to obtain new information about where unknown lines should be. In this subsection we still assume that the Lidar's pose for the new measurements as well as the pose at the time of model creation is known.

Observed Lines Update

To update the belief of the existing vertices, a new observed model is first created using only the new measurements, $\boldsymbol{z}_t$. This observed model is then combined with the existing model to create the updated model. The observed model should have lines and vertices similar to the existing model, therefore the observed model is not created by using the model selection algorithm, but rather by finding the line associations of each measurement, $\boldsymbol{c}$, described in Section 3.2.2, and fitting lines to the measurements associated with the same line. The distribution over the vertices between these lines, $p(\boldsymbol{v}|\boldsymbol{z}_t)$, is then calculated. The existing model distribution is also written as $p(\boldsymbol{v}|\boldsymbol{z}_{0:t-1})$, where $\boldsymbol{z}_{0:t-1}$ is all the previous measurements with which the model is created.

We assume that different measurements are conditionally independent given $\boldsymbol{v}$,

$$p(\boldsymbol{z}_{0:t-1}, \boldsymbol{z}_t|\boldsymbol{v}) = p(\boldsymbol{z}_{0:t-1}|\boldsymbol{v})p(\boldsymbol{z}_t|\boldsymbol{v}), \tag{3.2.62}$$

and that the prior distribution over $\boldsymbol{v}$, given no measurements, is uninformative and flat. The distribution over the $\boldsymbol{v}$, given measurements,

$$p(\boldsymbol{v}|\boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{v})p(\boldsymbol{v})}{p(\boldsymbol{z})}, \tag{3.2.63}$$

is proportional to $p(\boldsymbol{z}|\boldsymbol{v})$, since the prior distribution, $p(\boldsymbol{v})$, is flat and the measurement distribution, $p(\boldsymbol{z})$, is a constant for a given measurement. Therefore, the updated distribution over the vertices is calculated by multiplying the prior distribution with the observed distribution,

$$p(\boldsymbol{v}|\boldsymbol{z}_t, \boldsymbol{z}_{0:t-1}) = \eta\, p(\boldsymbol{v}|\boldsymbol{z}_{0:t-1})p(\boldsymbol{v}|\boldsymbol{z}_t), \tag{3.2.64}$$

where η is a normalising constant.

Here we assume that the new measurements observe the same part of the landmark that is already modelled. The algorithm should, however, have the ability to update the landmark if some parts are not seen again and to extend the model when new parts of the landmark are observed, as in Figure 3.16(a) and (b).

If we extend the model with new lines, it means that the existing model has no information about the new vertices. Similarly, if we do not observe certain lines with the new measurements, the new observed distribution has no information about their vertices.

The mean and covariance parameterisation used for the Gaussian distribution does not allow for a completely unknown or unobserved element to be parameterised, since it requires that the corresponding diagonal element in the covariance matrix is infinity. We therefore use the canonical parameterisation, which is parameterised by an information matrix, $\boldsymbol{K}$, and information vector, $\boldsymbol{h}$, in the landmark updating step. The information matrix is the inverse of the covariance matrix,

$$\boldsymbol{K} = \boldsymbol{\Sigma}^{-1}, \tag{3.2.65}$$

and the information vector is the covariance inverse multiplied by the mean vector,

$$\boldsymbol{h} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \tag{3.2.66}$$

so that the canonical form and the normal form represents the same distribution,

$$\mathcal{C}(\boldsymbol{K}, \boldsymbol{h}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.2.67}$$

The canonical form allows an element of a vector of random variables to have no information over it. The multiplication of distributions also simplifies to the addition of the canonical parameters. Details about operations in the canonical form can be found in Koller *et al.* [22, ch. 14]

The prior belief over the vertices can now be transformed to the canonical form,

$$p(\boldsymbol{v}|\boldsymbol{z}_{0:t-1}) = \mathcal{C}(\boldsymbol{K}_v, \boldsymbol{h}_v) = \mathcal{C}(\boldsymbol{\Sigma}_v^{-1}, \boldsymbol{\Sigma}_v^{-1}\boldsymbol{\mu}_v), \tag{3.2.68}$$

using the relationship in Equation 3.2.65 and Equation 3.2.66. The observed vertices are also parameterised in the canonical form, $p(\boldsymbol{v}|\boldsymbol{z}_t) = \mathcal{C}(\boldsymbol{K}_{v,\text{obs}}, \boldsymbol{h}_{v,\text{obs}})$.

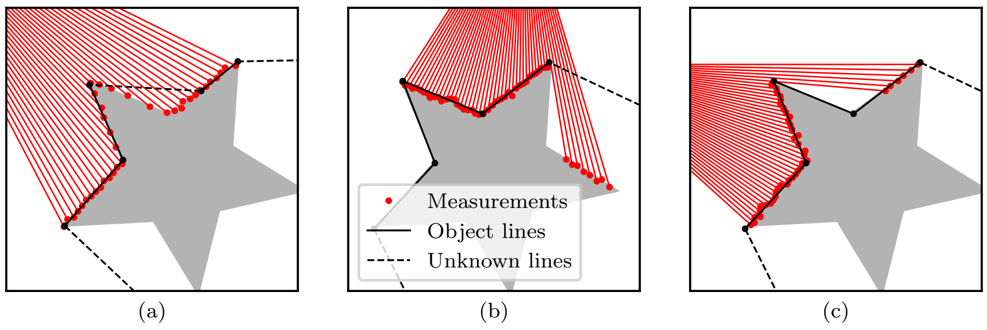


Figure 3.16 – Illustration of new measurements of an existing landmark. In (a) and (b), areas where unknown lines are defined are observed with new measurements. In (a), the entire region behind the unknown line is observed, but in (b), only a part of this region is observed and new unknown lines should be added as well. In (c), a part of the one line is observed, but its one vertex is not observed.

To update the belief over the vertices, we have to reason about which vertices are observed with the new measurements and where the landmark should be extended with new observed parts. An illustration of new measurements that observe new parts of the environment is shown in Figure 3.16. First we obtain the associations, $\boldsymbol{c}$, between the measurements and lines and check which lines are observed, considering a line observed if it has at least two measurements associated with it. For an observed object boundary, we fit a single line to all the measurements associated with it, but for an observed unknown boundary, the model selection algorithm, Algorithm 1, is executed with the measurements associated with it. This is done because measurements associated with an unknown line indicates that there is new information about what the landmark looks like in that region. Here we assume that line i is an unknown boundary if $u_i > 0.5$ and an object boundary otherwise.

If an unknown line is considered observed, it does not necessarily mean that both of the vertices of the unknown line are observed, as seen in Figure 3.16(b). We therefore check if the observed lines are likely to terminate at the vertices of the unknown line, or if the measurements observe only part of the unknown region. An example of this is shown in Figure 3.17, where the new fitted lines in (a) models the entire region behind the unknown line and the two vertices at the ends of the fitted lines are the observed vertices of the unknown line. In (b), however, only a part of the region behind the unknown line is observed and the new fitted line's vertices are not the vertices of the unknown line. Therefore both of these vertices should be added as new vertices in the model. To check if the observed vertices are the same as the existing unknown line's vertices, the Mahalanobis distances of the perpendicular error between the vertices of the unknown line and the first and last observed line are evaluated.

The perpendicular error between the vertex, $\boldsymbol{v}_i = \begin{bmatrix} v_{x,i} & v_{y,i} \end{bmatrix}^T$, and line, parameterised by $\boldsymbol{v}'_0$ and $\boldsymbol{v}'_1$, can be calculated with

$$e = (v_{x,i} - v'_{x,0}) \sin \psi - (v_{y,i} - v'_{y,0}) \cos \psi, \tag{3.2.69}$$

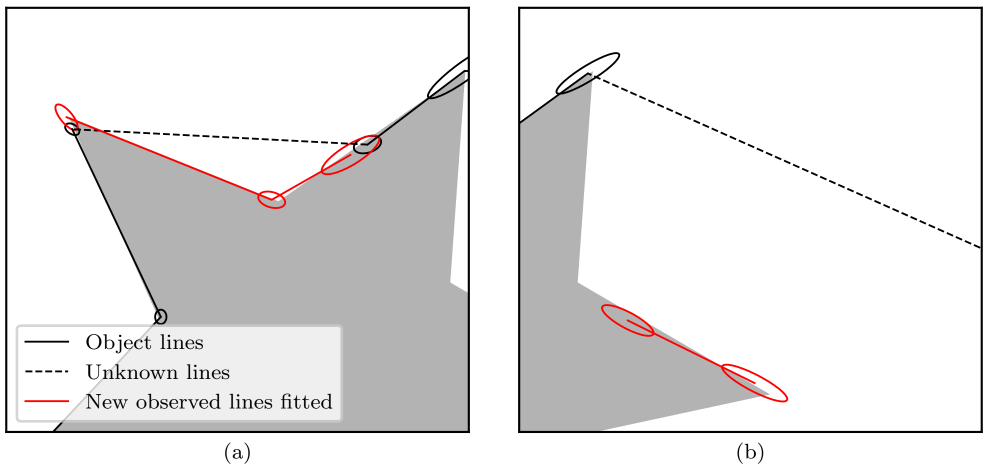


Figure 3.17 – Illustration of new lines observed in unknown regions. The new lines shown here is the lines fitted to the measurements associated to the unknown lines in Figure 3.16(a) and (b). In (a), the entire region behind the unknown line is observed, but in (b) only part of it is observed.

where

$$\psi = \arctan\left(\frac{v'_{y,1} - v'_{y,0}}{v'_{x,1} - v'_{x,0}}\right). \tag{3.2.70}$$

This equation is similar to Equation A.0.3 and can be derived from Figure A.1. The distribution over this error is found using the unscented transform,

$$\mathcal{N}(\mu_e, \sigma_e^2) = \text{UT}\left(\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{v,i}\\ \boldsymbol{\mu}'_{v,0}\\ \boldsymbol{\mu}'_{v,1}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{v,j,j} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}'_{v,0,0} & \boldsymbol{\Sigma}'_{v,0,1}\\ \mathbf{0} & \boldsymbol{\Sigma}'_{v,1,0} & \boldsymbol{\Sigma}'_{v,1,1}\end{bmatrix}\right); \begin{bmatrix}\boldsymbol{v}_i\\ \boldsymbol{v}'_0\\ \boldsymbol{v}'_1\end{bmatrix} \mapsto e\right), \tag{3.2.71}$$

and the Mahalanobis distance of the error can be calculated as,

$$e_{\text{mhl}} = \frac{\mu_e^2}{\sigma_e^2}. \tag{3.2.72}$$

If $e_{\text{mhl}} \leq 3$, it is considered that the line is likely to terminate at the vertex and the vertex is considered observed. If the vertex is not observed, as in Figure 3.16(b), an unobserved unknown line is created between the vertex and the observed lines.

To fit the observed model lines, we create an observed lines encoding list, $\boldsymbol{\zeta}_i$, for each existing line. For each existing line, this list contains m' values, where m' is the number of observed lines in the existing line's region and can only be larger than 1 if the existing line is an unknown line. If the existing line is an unknown line, it can be replaced by multiple new lines. Each value in this list, indicates whether the corresponding line in the observed model is observed or not. We also obtain the grouping vector, $\boldsymbol{\gamma}$, which indicates the indices of measurements that the new observed lines are fitted to. This process is presented in Algorithm 4.

Algorithm 4 loops through all the lines of the existing model. In Line 4 the number of associations is evaluated to assess if the line is observed. If it is not

Algorithm 4 Observed Lines Algorithm

```
function ObservedLines($p(\boldsymbol{v}), p(\boldsymbol{z}_p), \boldsymbol{c}$)
    $\boldsymbol{\gamma} = []$
    for $i := 0$ to $m-1$ do                                   ▷ loop through all existing lines
        if associations to $i^{\text{th}}$ line $\geq 2$ then           ▷ line considered observed
            $a, b :=$ 1st and last index of measurements associated with line
            if $u_i \leq 0.5$ then                      ▷ if existing line is an object boundary
                $\boldsymbol{\zeta}_i := (1)$                                  ▷ single line fitted
                Append $(a, b)$ to $\boldsymbol{\gamma}$
            else                                     ▷ if existing line is an unknown boundary
                $\boldsymbol{\mu}'_v, \boldsymbol{\Sigma}'_v, \boldsymbol{\gamma}' =$ ModelSelection($p(\boldsymbol{z}_{p,a:b})$)            ▷ Algorithm 1
                $\boldsymbol{\zeta}_i := (1, \cdots, 1)$ of length $m'$          ▷ $m'$ is number of lines in $\boldsymbol{\mu}'_v$
                if Line $p(\boldsymbol{v}'_0, \boldsymbol{v}'_1)$ terminate at vertex $p(\boldsymbol{v}_i)$ then
                    Insert 0 in front of $\boldsymbol{\zeta}_i$                    ▷ add unknown line left
                    Insert $(0, 0)$ in front of $\boldsymbol{\gamma}'$
                end if
                if Line $p(\boldsymbol{v}'_{m'}, \boldsymbol{v}'_{m'+1})$ terminate at vertex $p(\boldsymbol{v}_{i+1})$ then
                    Append 0 to $\boldsymbol{\zeta}_i$                           ▷ add unknown line right
                    Append $(0, 0)$ to $\boldsymbol{\gamma}'$
                end if
                Append $\boldsymbol{\gamma}'$ to $\boldsymbol{\gamma}$
            end if
        else                                                            ▷ unobserved line
            $\boldsymbol{\zeta}_i := (0)$
            Append $(0, 0)$ to $\boldsymbol{\gamma}$
        end if
    end for
    return $\boldsymbol{\zeta}, \boldsymbol{\gamma}$
end function
```

observed, $\boldsymbol{\zeta}_i$ is set to zero and placeholder values are added to the grouping vector, $\boldsymbol{\gamma}$, in Lines 23 and 24. If the line is observed, it is determined if the lines is an object or unknown boundary in Line 6. If the line is an object boundary, $\boldsymbol{\zeta}_i$ is set to zero and the indices of the measurement associated with the line are added to $\boldsymbol{\gamma}$ in Lines 7 and 8. If the line is an unknown boundary, however, a new partial model is created from the measurements associated with the line in Line 10. In Line 12 and 16 we determine if the two ends of the new partial model are existing vertices or new vertices. The grouping vector of the partial model is then added to $\boldsymbol{\gamma}$ and $\boldsymbol{\zeta}_i$ is populated with ones for all fitted lines and zeros where an unknown line is created between the existing vertices and the new vertices.

Once the observed line encoding, $\boldsymbol{\zeta}$, and the grouping vector, $\boldsymbol{\gamma}$, are obtained, new lines are fitted to the measurements to obtain the observed vertices. There could be lines in the middle of the model that are not observed with the new measurements, as shown in Figure 3.16(c). Therefore, the vertices of the observed model are obtained by fitting new lines to sets of adjacent observed lines, where all the lines in the set are observed. Figure 3.16(a) has only one set of adjacent observed lines, while (b) and (c) each has two sets of adjacent observed lines, since there is an unobserved line in the middle of the model. The distribution over the vertices of each set are then transformed to the canonical form and the canonical parameters are added to the corresponding part of the full canonical parameters of the complete observed model. The canonical parameterisation is useful here, since the vertices that are not observed, contain zeros in the information matrix and vector.

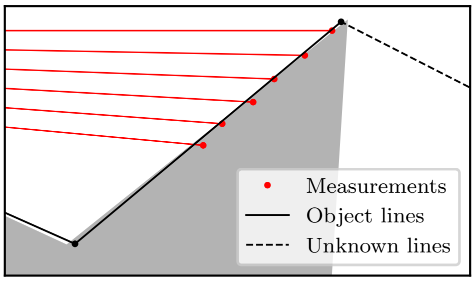


Figure 3.18 – Zoomed in figure of Figure 3.16(c). The line shown is only partly observed by the measurements, where the lower part of this line is not observed. Therefore, no information about the lower vertex should be added in the observed model.

The two end vertices of each set of adjacent observed lines are, however, not necessarily observed, as shown in Figure 3.16(c) and Figure 3.18. It is possible that part of a line is observed, but its vertex is occluded by another object or another part of the landmark, or that the end of the Lidar's FOV is reached. Therefore, for each set of adjacent observed lines, we check if the end vertices are occluded and if they are, the vertex is removed from the observed distribution. This check is done in the same manner that we check where the end unknown lines of a new landmark should be, visualised by Figure 3.10. If the adjacent measurement is closer to the Lidar than the measurement on the line or the measurement is at the Lidar's FOV limit, the vertex is considered occluded. In Figure 3.18, only the top part of the line is observed and the bottom part is occluded by another part of the landmark. Therefore only new information about the vertex at the top should be added in the observed model and no information about the vertex at the bottom should be added.

The method of finding the distribution over the vertices in the canonical form is shown in Algorithm 5. Here, a and b denote the indices of an observed set of lines, which are initialised as zeros in Line 3. Lines 7 to 28 loop through all the existing lines and Lines 8 to 26 loop through the observed line encoding $\boldsymbol{\zeta}_i$ of the existing line. Line 9 determines if this is the first line in an observed set and Line 11 determines if the current line is unobserved. If the line is observed, the observed set index, b, is increased and the loop continues. If the current line is not observed, the observed set is ended and lines are fit to all the measurements of the current observed set in Line 12. Lines 13 and 18 determine if the left and right vertices of this set are really observed and removes these vertices if they are not observed. In Line 22 the canonical parameters of the vertices of the observed set is added to the appropriate part of the full canonical parameters. In Line 23 the unknown probabilities of the

Algorithm 5 Observed Parameters Algorithm

```
1: function ObservedParameters($p(\boldsymbol{z}_p), \boldsymbol{\zeta}, \boldsymbol{\gamma}$)
2:     $\boldsymbol{\zeta}_m := (0)$        ▷ add artificial unobserved line
3:     $a := b := 0$        ▷ start and end indices of observed set
4:     $k := \dim(\boldsymbol{\gamma}) + 1$        ▷ number of posterior vertices
5:     $\mathcal{C}\left(\boldsymbol{K}_{v,\text{obs}}, \boldsymbol{h}_{v,\text{obs}}\right) = \mathcal{C}\left(\mathbf{0}_{[2k\times 2k]}, \mathbf{0}_{[2k]}\right)$        ▷ initialise zero information
6:     $\boldsymbol{u}_{\text{obs}} = \begin{bmatrix}1 & \cdots & 1\end{bmatrix}^T$ of length $k-1$
7:     for $i := 0$ to $m$ do        ▷ loop through existing lines and $\boldsymbol{\zeta}_m$
8:         for $j := 0$ to length($\boldsymbol{\zeta}_i$) do
9:             if $\zeta_{i,j} = 1$ and $b - a = 0$ then        ▷ first observed line in set
10:                $j_a := j$        ▷ save j value of first in set
11:            else if $\zeta_{i,j} = 0$ then        ▷ unobserved line
12:                $\boldsymbol{\mu}'_v, \boldsymbol{\Sigma}'_v := \text{UT}(p(\boldsymbol{z}_p), \boldsymbol{\gamma}_{a:b}; \boldsymbol{z}_p \mapsto \boldsymbol{v})$
13:                if ($\phi_{\gamma_{a,0}} = \phi_{\text{FOV}}$ or $r_{\gamma_{a,0}-1} < r_{\gamma_{a,0}}$) and $j_a = 0$ then
14:                    $a := a + 1$
15:                    Remove first vertex from $p(\boldsymbol{v}')$
16:                end if
17:                $t := b$
18:                if ($\phi_{\gamma_{a,0}} = -\phi_{\text{FOV}}$ or $r_{\gamma_{b,1}+1} < r_{\gamma_{b,1}}$) and $j = 0$ then
19:                    $t := t - 1$
20:                    Remove last vertex from $p(\boldsymbol{v}')$
21:                end if
22:                $\mathcal{C}\left(\boldsymbol{K}_{v,\text{obs},a:t+1,a:t+1}, \boldsymbol{h}_{v,\text{obs},a:t+1}\right) := \mathcal{C}\left({\boldsymbol{\Sigma}'_v}^{-1}, {\boldsymbol{\Sigma}'_v}^{-1}\boldsymbol{\mu}'_v\right)$
23:                $\boldsymbol{u}_{\text{obs},a:t} :=$ Equation 3.2.36 $\leftarrow p(\boldsymbol{v}'), \boldsymbol{\gamma}_{a:b}, \boldsymbol{z}_{p,a:b}$
                                                       ▷ calculate unknown probabilities
24:                $a := b + 1$        ▷ reset observed set
25:            end if
26:            $b := b + 1$
27:        end for
28:    end for
29:    return $\boldsymbol{K}_{v,\text{obs}}, \boldsymbol{h}_{v,\text{obs}}, \boldsymbol{u}_{\text{obs}}$
30: end function
```

observed lines are calculated using Equation 3.2.36. The artificial unobserved line in Line 2 ensures that the parameters of the final observed set are calculated.

The vertices that are observed, not observed and newly added are encoded in a list, $\boldsymbol{\vartheta}$, using the information matrix, $\boldsymbol{K}_{v,\text{obs}}$, and observed lines encoding, $\boldsymbol{\zeta}$. Each element, ϑ_i, relates to the state of the i^{th} vertex, with 0 being unobserved, 1 being a previously existing, observed vertex and 2 being a newly added vertex. This encoding for the observed vertices can be used to transform the distribution over the prior model to the state space of the posterior model.

The process of obtaining this encoding is shown in Algorithm 6. In Line 2 an artificial unobserved line is added again to ensure that the final vertex encoding is calculated. In Line 3 the entire vertex encoding vector is initialised as 0, which is unobserved. Lines 6 to 15 loop through the existing lines and Lines 8 to 14 loop

Algorithm 6 Observed Vertices Algorithm

```
1:  function ObservedVertices($\boldsymbol{\zeta}, \boldsymbol{K}_{v,\text{obs}}, \boldsymbol{u}$)
2:      $\boldsymbol{\zeta}_m := (0)$                              ▷ add artificial unobserved line
3:      $\boldsymbol{\vartheta} := \mathbf{0}_{[k]}$                          ▷ $k$ is number of posterior vertices
4:      $\boldsymbol{u}' := \begin{bmatrix}1 & \cdots & 1\end{bmatrix}^T_{[k-1]}$     ▷ initialise extended unknown probabilities
5:      $a := 0$                                   ▷ observed vertex iterator
6:      for $i := 0$ to $m$ do                      ▷ loop through existing lines and $\boldsymbol{\zeta}_m$
7:          $\boldsymbol{u}'_a := u_i$ if length$(\boldsymbol{\zeta}_i) = 1$          ▷ assign $\boldsymbol{u}'_a$ if line is not replaced
8:          for $j := 0$ to length$(\boldsymbol{\zeta}_i)$ do
9:              if $\boldsymbol{K}_{v,\text{obs},a,a} \neq \mathbf{0}$ then
10:                 $\vartheta_a := 1$ if $j = 0$                ▷ observed prior vertex
11:                 $\vartheta_a := 2$ if $j > 0$                ▷ new added vertex
12:             end if
13:             $a := a + 1$
14:         end for
15:     end for
16:     return $\boldsymbol{\vartheta}, \boldsymbol{u}'$
17: end function
```

through $\boldsymbol{\zeta}_i$. In Line 9 the appropriate information matrix elements are checked to see if it contains information. If it does it means that a vertex is observed. Lines 10 and 11 determine whether it is an existing vertex or a newly added vertex.

To add the existing model's canonical parameters to the observed canonical parameters, the vertices of the existing model need to be transformed to the state space of the updated model. For this transformation, we define a transformation matrix $\boldsymbol{F}$ to map the the existing model parameters to the appropriate locations in the higher-dimensional state space. The dimensions of $\boldsymbol{F}$ are $N \times M$, where N is the dimension of the existing model, $\boldsymbol{v}$, and M is the dimension of the observed model, $\boldsymbol{v}_{\text{obs}}$. $\boldsymbol{F}$ is populated by using $\boldsymbol{\vartheta}$, where

$$\boldsymbol{F}_{i,j} = \begin{cases} \boldsymbol{I}_{[2\times 2]}; & \vartheta_j < 2 \text{ and is the } i^{\text{th}} \text{ such element} \\ \mathbf{0}_{[2\times 2]}; & \text{otherwise} \end{cases} \tag{3.2.73}$$

and

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{0,0} & \cdots & \boldsymbol{F}_{0,m'} \\ \vdots & \ddots & \vdots \\ \boldsymbol{F}_{m,0} & \cdots & \boldsymbol{F}_{m,m'} \end{bmatrix}. \tag{3.2.74}$$

Here m is the number of lines of the existing model and m' is the number of lines of the updated model. Each element, $\boldsymbol{F}_{i,j}$, is a 2×2 matrix, since each vertex is 2-dimensional.

As an example, the observed line encoding for Figure 3.16(a) would be

$$\boldsymbol{\zeta} = ((0), (1), (1, 1), (1), (1), (0)). \tag{3.2.75}$$

The two zeroes at the start and end of this list indicate the two unobserved unknown lines at the ends of the model. Since the unknown line in the middle of the model is observed and should be replaced by two new lines, the third list contains two ones. The observed vertices encoding of this example would be

$$\boldsymbol{\vartheta} = (0, 1, 1, 2, 1, 1, 1, 0), \tag{3.2.76}$$

where the two vertices at the ends of the model is not observed (these vertices are also not visible is Figure 3.16(a)) and the 2 indicates the new vertex that will be added. The transformation matrix to extend the existing model to the dimensions of the observed model would be

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} \end{bmatrix}. \tag{3.2.77}$$

In this matrix, the fourth column, which only contains zeroes, will place an element with no information in the extended existing model where the new vertex will be added.

The existing model parameters can be extended to the higher-dimensional state space of the updated model parameters with the following transformation,

$$\mathcal{C}\left(\boldsymbol{K}'_v, \boldsymbol{h}'_v\right) = \mathcal{C}\left(\boldsymbol{F}^T \boldsymbol{K}_v \boldsymbol{F}, \boldsymbol{F}^T \boldsymbol{h}_v\right), \tag{3.2.78}$$

so that $p(\boldsymbol{v}|\boldsymbol{z}_{0:t-1}) = \mathcal{C}\left(\boldsymbol{K}'_v, \boldsymbol{h}'_v\right)$. This transformation maps every vertex in the existing model to the corresponding location in the extended state space, so that $\boldsymbol{v}'_i$ and $\boldsymbol{v}_{\text{obs},i}$ correspond to the same vertex. The posterior distribution of the vertices can now be calculated by adding the canonical parameters of the prior and observed distributions,

$$\mathcal{C}\left(\boldsymbol{K}_{v,\text{new}}, \boldsymbol{h}_{v,\text{new}}\right) = \mathcal{C}\left(\boldsymbol{K}'_v + \boldsymbol{K}_{v,\text{obs}}, \boldsymbol{h}'_v + \boldsymbol{h}_{v,\text{obs}}\right). \tag{3.2.79}$$

An illustration of the observed models, created from the example in Figure 3.16, and the updated models is shown in Figure 3.19.

Although neither of the information matrices of the extended existing vertices, $\boldsymbol{K}'_v$, or the observed vertices, $\boldsymbol{K}_{v,\text{obs}}$, are necessarily invertible, the updated distribution's information matrix, $\boldsymbol{K}_{v,\text{new}}$, is invertible. This means that the updated distribution can be transformed back to the normal form, $p(\boldsymbol{v}|\boldsymbol{z}_t, \boldsymbol{z}_{0:t-1}) = \mathcal{N}\left(\boldsymbol{\mu}_{v,\text{new}}, \boldsymbol{\Sigma}_{v,\text{new}}\right)$.

The unknown boundary probabilities of the observed lines, $\boldsymbol{u}_{\text{obs}}$, are also calculated in Algorithm 5. The unknown boundary probabilities of the updated lines, $\boldsymbol{u}_{\text{new}}$, are approximated by taking the minimum of the existing probabilities, $\boldsymbol{u}$, and the observed probabilities, $\boldsymbol{u}_{\text{new}}$. The minimum is taken since it comes from the observation with the most information or measurements of that line. The existing model's unknown boundary probabilities should, however, be extended to the dimensions of the updated model before this can be done. This extended unknown

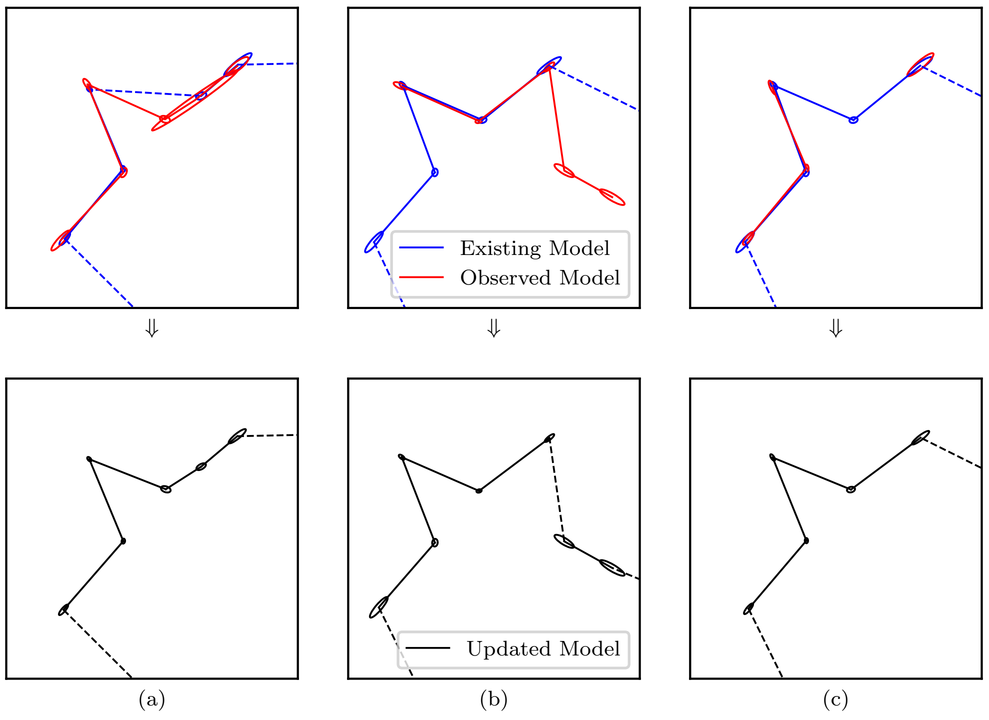


Figure 3.19 – Illustration of updated model. The existing and observed models are shown in the top row, which is created from Figure 3.16. These distributions are multiplied to obtain the updated model in the bottom row.

boundary probabilities, $\boldsymbol{u}'$, are all initialised as ones, as in Line 4 of Algorithm 6. In Line 7, the elements of $\boldsymbol{u}$ are then assigned to the appropriate location in $\boldsymbol{u}'$ only if the line is not replaced by a new set of lines. This ensures that u_i is at the location in $\boldsymbol{u}'$ corresponding to line i of the existing model and that the unknown probabilities of unknown lines that are replaced with multiple new lines are not considered. A value of 1 is given to all new lines, since this is the maximum value it can have and will not influence the new value when taking the minimum. The new unknown boundary probabilities are now calculated by

$$u_{\text{new},i} = \min\left(u'_i, u_{\text{obs},i}\right). \tag{3.2.80}$$

Similar to Equation 3.2.44, where a linear relationship between two parameters is used to describe the conditional distribution between them, we can describe the relationship between $\boldsymbol{x}_{\text{pose}}$ and $\boldsymbol{v}_{\text{obs}}$ as

$$\boldsymbol{v}_{\text{obs}} = \boldsymbol{A}_{vx}\boldsymbol{x}_{\text{pose}} + \boldsymbol{b}_{vx} + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{C}(\boldsymbol{K}_{v,\text{obs}}, \boldsymbol{0}). \tag{3.2.81}$$

The noise is, however, parameterised in the canonical form, since $\boldsymbol{K}_{v,\text{obs}}$ is possibly not invertible. Due to the possible singular nature of $\boldsymbol{K}_{v,\text{obs}}$, the mean of the full observed distribution can also not be calculated in this form. We therefore want create a transformation matrix, $\boldsymbol{G}$, to extract only the observed vertices from $\boldsymbol{v}_{\text{obs}}$.

With this transformation matrix we can obtain

$$\mathcal{C}\left(\boldsymbol{K}'_{v,\text{obs}}, \boldsymbol{h}'_{v,\text{obs}}\right) = \mathcal{C}\left(\boldsymbol{G}\boldsymbol{K}_{v,\text{obs}}\boldsymbol{G}^T, \boldsymbol{G}\boldsymbol{h}_{v,\text{obs}}\right), \tag{3.2.82}$$

where $\boldsymbol{K}'_{v,\text{obs}}$ is invertible. $\boldsymbol{G}$ is populated similar to $\boldsymbol{F}$ in Equation 3.2.73, but rather checking where $\vartheta_j > 0$, which indicates vertices observed at the current timestep,

$$\boldsymbol{G}_{i,j} = \begin{cases} \boldsymbol{I}_{[2\times2]}; & \vartheta_j > 0 \text{ and is the } i^{\text{th}} \text{ such element} \\ \boldsymbol{0}_{[2\times2]}; & \text{otherwise} \end{cases} \tag{3.2.83}$$

and

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{0,0} & \cdots & \boldsymbol{G}_{0,m} \\ \vdots & \ddots & \vdots \\ \boldsymbol{G}_{m',0} & \cdots & \boldsymbol{G}_{m',m} \end{bmatrix}, \tag{3.2.84}$$

where m' is the number of vertices for which $\vartheta_j > 0$.

We can now obtain the mean of these observed vertices,

$$\boldsymbol{\mu}'_{v,\text{obs}} = \boldsymbol{K}'^{-1}_{v,\text{obs}}\boldsymbol{h}'_{v,\text{obs}}, \tag{3.2.85}$$

and the mean of their polar coordinates, $\boldsymbol{\mu}'_{v,p,\text{obs}}$, with

$$\boldsymbol{\mu}'_{v,p,\text{obs},i} = \begin{bmatrix} \mu'_{v,r,i} \\ \mu'_{v,\phi,i} \end{bmatrix} = \begin{bmatrix} \left|\left|\boldsymbol{\mu}'_{v,\text{obs}} - \boldsymbol{x}_{\text{pos}}\right|\right| \\ \arctan\left(\frac{\mu_{v,y,i}-y}{\mu_{v,x,i}-x}\right) - \theta \end{bmatrix}, \tag{3.2.86}$$

as in Equation 3.2.46. The transformation matrix for the observed part, $\boldsymbol{A}'_{vx}$, is calculated as in Equation 3.2.48, where

$$\boldsymbol{A}'_{vx,i} = \begin{bmatrix} 1 & 0 & -\mu'_{v,r,i}\sin(\theta + \mu'_{v,\phi,i}) \\ 0 & 1 & \mu'_{v,r,i}\cos(\theta + \mu'_{v,\phi,i}) \end{bmatrix}, \tag{3.2.87}$$

and subsequently, calculate

$$\boldsymbol{b}'_{vx} = \boldsymbol{\mu}'_{v,p,\text{obs}} - \boldsymbol{A}'_{vx}\boldsymbol{x}_{\text{pose}}. \tag{3.2.88}$$

We now have the linear relationship,

$$\boldsymbol{v}'_{\text{obs}} = \boldsymbol{A}'_{vx}\boldsymbol{x}_{\text{pose}} + \boldsymbol{b}'_{vx} + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{C}(\boldsymbol{K}'_{v,\text{obs}}, \boldsymbol{0}). \tag{3.2.89}$$

For the unobserved vertices, with no observed information about them, it does not matter what their means are, since the uncertainty over them is infinity. We can therefore populate their corresponding locations in $\boldsymbol{A}_{vx}$ and $\boldsymbol{b}_{vx}$ with zeros. These parameters are thus reconstructed using the transformation matrix $\boldsymbol{G}$,

$$\begin{aligned} \boldsymbol{A}_{vx} &= \boldsymbol{G}^T\boldsymbol{A}'_{vx}\boldsymbol{G} \\ \boldsymbol{b}_{vx} &= \boldsymbol{G}^T\boldsymbol{b}'_{vx}. \end{aligned} \tag{3.2.90}$$

This parameterisation of the conditional distribution over the observed vertices is used in the SLAM algorithm in Chapter 4.

Unknown Lines Update

Once the landmark parameters are updated with the observed vertices, the unknown lines at the end of the model, $\boldsymbol{v}_l$ and $\boldsymbol{v}_r$, can also be updated by finding the observed unknown lines, $\boldsymbol{v}_{l,\text{obs}}$ and $\boldsymbol{v}_{r,\text{obs}}$. This is done with the same method of finding the original unknown lines, Equation 3.2.34, by assessing the measurements adjacent to the one on the model.

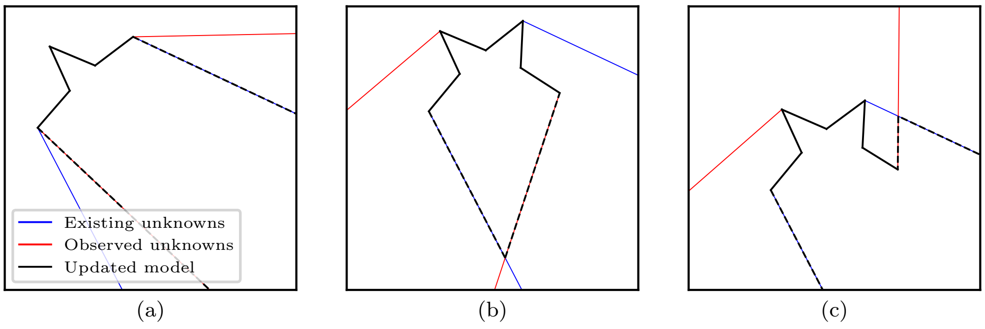


Figure 3.20 – Illustration of updated unknown lines. The existing unknown lines (blue) and observed unknown lines (red) are shown, as well as the updated model with updated unknown lines (black).

Once these observed unknown lines are obtained, they are compared to the existing unknown lines. The updated unknown lines, $\boldsymbol{v}_{l,\text{new}}$ and $\boldsymbol{v}_{r,\text{new}}$, are selected as the combination of the existing and observed unknown lines that contain the most information; so that the new unknown lines are not in any of the existing or observed unknown lines' empty regions. This is visualised in Figure 3.20, where the existing, observed and updated unknown lines are shown for different cases.

Algorithm 7 is implemented to achieve this for the left unknown lines; the right unknown lines' update is performed similarly. Firstly, the algorithm finds the first vertex of the existing model's object lines (Line 2) and draws the existing unknown lines from this vertex (Line 7). Similarly, the first observed vertex is obtained (Line 3) and the observed unknown vertices are drawn from this vertex (Line 4). The two unknown lines closest to the model from both of these sets are evaluated (Lines 8 to 17) to obtain the one with the most information about the environment; the line that lies in the other's unknown region. Line 8 simply checks for the leftmost of these two lines, and if they start at the same location, the line that makes the smallest angle with the first object line is selected (Lines 11 to 17). The selected line is the first line in the updated unknown line and the set it comes from is selected as the current set (Line 18). In Line 20 the updated unknown vertices are initialised with the last vertex of the selected set. From here the lines in this set are followed (Lines 21 to 34) until one intersects with a line from the other set, which is checked for in Line 23. If an intersection occurs, the intersection point is added to the updated unknown lines' vertices (Line 24) and the other set is followed by switching the selected set (Line 26). In Line 24 the intersection distribution is calculated using

Algorithm 7 Update Left Unknown Lines Algorithm

```
 1: function UpdateUnknownLinesLeft($p(\boldsymbol{v}_{\text{new}}), \boldsymbol{u}_{\text{new}}, p(\boldsymbol{z}_p), \boldsymbol{\vartheta}$)
 2:     $i_p :=$ first prior line not unknown, $\vartheta_{i_p} < 2$ and $u_{i_p} < 1$
 3:     $i_o :=$ first observed vertex, $\vartheta_{i_o} > 0$
 4:     $\boldsymbol{v}_{l,\text{obs}} :=$ find observed unknown vertices and append $\boldsymbol{v}_{\text{new},i_o}$
 5:     $n_o :=$ number of vertices in $\boldsymbol{v}_{l,\text{obs}} - 1$
 6:     $n_p :=$ number of prior left unknown lines
 7:     $\boldsymbol{v}_{l,\text{pre}} := \boldsymbol{v}_{0:n_p-1}$ and append $\boldsymbol{v}_{\text{new},i_p}$
 8:     $i := \min(i_o, i_p)$
 9:     $c :=$ obs if $i_o < i_p$                     ▷ observed lines are farther to the left
10:     $c :=$ pre if $i_p < i_o$                     ▷ existing lines are farther to the left
11:     if $i_o = i_p$ then          ▷ left existing and observed vertices are the same
12:         $\theta_i :=$ angle of line $(\boldsymbol{v}_i, \boldsymbol{v}_{i+1})$
13:         $\theta_o :=$ angle of line $(\boldsymbol{v}_i, \boldsymbol{v}_{l,\text{obs},n_o-1}) - \theta_i$
14:         $\theta_p :=$ angle of line $(\boldsymbol{v}_i, \boldsymbol{v}_{l,\text{pre},n_p-1}) - \theta_i$
15:         $c :=$ obs if $\theta_o \leq \theta_p$
16:         $c :=$ pre if $\theta_p < \theta_o$
17:     end if
18:     $\boldsymbol{v}_a, n_a :=$ selected $\boldsymbol{v}_{l,c}$ and $n_c$          ▷ line selected by $c$, obs or pre
19:     $\boldsymbol{v}_b, n_b :=$ other $\boldsymbol{v}_{l,\dots}$ and $n_{\dots}$          ▷ line opposite of $c$, obs or pre
20:     $\boldsymbol{v}_l := \left[\boldsymbol{v}_{a,n_a}\right]$
21:     while $n_a > 0$ and $n_b > 0$ do     ▷ continue both sets have line to evaluate
22:         for $j := n_b$ decrease to 1 do
23:             if lines $(\boldsymbol{v}_{l,0}, \boldsymbol{v}_{a,n_a-1})$ and $(\boldsymbol{v}_{b,j}, \boldsymbol{v}_{b,j-1})$ intersect then
24:                 Insert intersection distribution in front of $\boldsymbol{v}_l$      ▷ UT(Eq. 3.2.8)
25:                 $n_b := j - 1$
26:                 Switch values of $\boldsymbol{v}_a, n_a$ and $\boldsymbol{v}_b, n_b$          ▷ switch selected sets
27:                 Exit loop
28:             end if
29:         end for
30:         if no intersection then                         ▷ end of line is reached
31:             Insert $\boldsymbol{v}_{a,n_a-1}$ in front of $\boldsymbol{v}_l$   ▷ add vertex to updated unknown lines
32:             $n_a := n_a - 1$
33:         end if
34:     end while
35:     Remove last vertex of $\boldsymbol{v}_l$
36:     Replace $\boldsymbol{v}_{\text{new},0:n_p-1}$ with $\boldsymbol{v}_l$
37:     return $p(\boldsymbol{v}_{\text{new}})$
38: end function
```

the unscented transform and Equation 3.2.8. Each time the end of the current line is reached without an intersection occurring, the vertex at the end is added to the updated unknown lines (Line 31). If the loop reaches the end of the current set, the updated unknown lines are completely defined. In Line 35 the initial vertex of the updated unknown lines is removed, because it is already contained in the model's object vertices. The existing unknown lines' vertices are then removed from the model and replaced with the updated unknown lines' vertices (Line 36).

In Figure 3.20(a), it is seen that the updated unknown lines follow the unknown lines to the "inside" of the model. In (b), It is seen that the existing and observed unknown lines are drawn from vertices in the middle of the updated model when new segments are observed. The unknown line connected to the end of the model are then selected as the start of the updated unknown lines. The intersection of the existing and observed unknown lines in (c) shows that once the updated unknown lines reach this intersection, they follow the other set of unknown lines.

Once the left and right unknown lines are updated, they are evaluated against each other for intersections. If the left and right unknown lines intersect, as in Figure 3.20(b), they are both terminated at the intersection point, since the added information of these unknown lines indicates that the region further on is empty. In the case of multiple intersections, the intersection closest to the rest of the model is used.

Landmark Wrapping

In our modelling technique, we want to be able to enclose a landmark once the robot has observed it from all sides. An enclosed landmark has no unknown lines at the ends of the model, since the entire area is observed and the first and last vertices are also connected with a line.

After the landmark vertices are updated, the first vertex, $\boldsymbol{v}_l$, and last vertex, $\boldsymbol{v}_r$, of the landmark's object lines are evaluated to reason whether they model the same underlying vertex. The joint distribution over these vertices is

$$p(\boldsymbol{v}_l, \boldsymbol{v}_r) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{v,l}\\ \boldsymbol{\mu}_{v,r}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{v,ll} & \boldsymbol{\Sigma}_{v,lr}\\ \boldsymbol{\Sigma}_{v,rl} & \boldsymbol{\Sigma}_{v,rr}\end{bmatrix}\right), \tag{3.2.91}$$

which is extracted from the full model parameter distribution. A distribution over the difference between these two vertices are obtained, where the mean and covariance are calculated as

$$\begin{aligned}\boldsymbol{\mu}_\delta &= \boldsymbol{\mu}_{v,l} - \boldsymbol{\mu}_{v,r}\\ \boldsymbol{\Sigma}_\delta &= \boldsymbol{\Sigma}_{v,ll} + \boldsymbol{\Sigma}_{v,rr} - \boldsymbol{\Sigma}_{v,lr} - \boldsymbol{\Sigma}_{v,rl}.\end{aligned} \tag{3.2.92}$$

The Mahalanobis distance of this distribution is calculated,

$$d_{\text{mhl}} = \boldsymbol{\mu}_\delta^T \boldsymbol{\Sigma}_\delta^{-1} \boldsymbol{\mu}_\delta, \tag{3.2.93}$$

and if it is smaller than 3, the vertices are assumed to model the same underlying vertex.

If these vertices model the same vertex, the unknown lines are removed from the model and the information about these vertices is combined. Since there are dependencies between vertices in the model, this could affect other vertices in the model. The distribution over the entire model should thus be updated. In the canonical form, the current distribution over the vertices can be written as

$$\mathcal{C}\left(\boldsymbol{K}_v, \boldsymbol{h}_v\right) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{v}_l\\ \boldsymbol{v}_c\\ \boldsymbol{v}_r\end{bmatrix}\begin{bmatrix}\boldsymbol{K}_{ll} & \boldsymbol{K}_{lc} & \boldsymbol{K}_{lr}\\ \boldsymbol{K}_{cl} & \boldsymbol{K}_{vc} & \boldsymbol{K}_{cr}\\ \boldsymbol{K}_{rl} & \boldsymbol{K}_{rc} & \boldsymbol{K}_{rr}\end{bmatrix}, \begin{bmatrix}\boldsymbol{h}_l\\ \boldsymbol{h}_c\\ \boldsymbol{h}_r\end{bmatrix}\right), \tag{3.2.94}$$

where $\boldsymbol{v}_c$ is the model vertices excluding the left and right object vertices. To combine the information of the left and right vertex, we remove the right vertex and add the information to the left vertex, which leads to the following distribution,

$$\mathcal{C}\left(\boldsymbol{K}'_v, \boldsymbol{h}'_v\right) = \mathcal{C}\left(\begin{bmatrix} \boldsymbol{v}'_l \\ \boldsymbol{v}' \end{bmatrix} \begin{bmatrix} \boldsymbol{K}_{ll} + \boldsymbol{K}_{rr} + \boldsymbol{K}_{lr} + \boldsymbol{K}_{rl} & \boldsymbol{K}_{lc} + \boldsymbol{K}_{rc} \\ \boldsymbol{K}_{cl} + \boldsymbol{K}_{cr} & \boldsymbol{K}_{cc} \end{bmatrix}, \begin{bmatrix} \boldsymbol{h}_l + \boldsymbol{h}_r \\ \boldsymbol{h}_c \end{bmatrix}\right). \tag{3.2.95}$$

This is the updated distribution and can be transformed back to the normal form to obtain the mean and covariance of the model.

If the landmark is enclosed, an object line also connects the first and last vertices of the model. The vertex that is first is not important any more, only the order of vertices is important. This means that a number of vertices at the end of the model could be moved to the front, and still produce the same model.

The discussion of the landmark modelling, as well as associating measurement with these landmark models and updating these models are now concluded. The modelling method in this chapter is essential in the implementation of the SLAM algorithm discussed in the following chapter.

Chapter 4

SLAM with Vertex Parameter Models

The aim of the landmark modelling method developed in Chapter 3 is to be used in SLAM. We now discuss the development of a SLAM algorithm that uses this landmark modelling method.

As discussed in Chapter 2, there are a number of SLAM algorithms that use a landmark-based map. EKF-SLAM is a commonly-used algorithm for landmark-based maps and therefore we base our SLAM algorithm on the EKF SLAM algorithm. The basic EKF SLAM algorithm assumes point landmarks and updates the belief over the robot states and landmark positions each time a landmark is observed. This algorithm is a Bayes filter approach, which only calculates the belief over the latest robot states and the landmark positions. Although we use EKF SLAM, it is possible to use our landmark modelling method with any landmark-based SLAM algorithm.

The EKF SLAM motion update is used directly in our approach, but we derive a novel measurement update to simultaneously update the complete landmark model and robot states when a landmark is observed. This is in contrast to other approaches [11; 12; 13; 14] that estimate the location of each observed landmark and update the belief over the robot states and landmark locations simultaneously. In these other methods, the landmark model is updated separately.

In this chapter we discuss the details of the motion update, as well as all aspects of the measurement update, which include measurement segmentation, landmark correspondence, updating the belief over the full state space and adding new landmarks to the map. An overview of the EKF is given first, since it is used in the subsequent sections.

4.1 Overview of the EKF

In this section we give an overview of the EKF algorithm, based on the description by Thrun *et al.* [1] that is used in the motion update in the next section. The EKF is a recursive state estimator that calculates a Gaussian belief over the current states of a system, $\boldsymbol{s}_t$, using the belief over the previous states, $\boldsymbol{s}_{t-1}$. To ensure that the belief stays Gaussian, all nonlinear models are linearised. The EKF state estimation is done in two steps: a motion update that uses controls, $\boldsymbol{u}_t$, to calculate a predicted belief of the new states, and an measurement update that uses measurements, $\boldsymbol{z}_t$, to calculate a posterior belief of the states.

At the start of the EKF algorithm, a Gaussian belief over the previous states, $p(\boldsymbol{s}_{t-1}|\boldsymbol{u}_{0:t-1}, \boldsymbol{z}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{s,t-1}, \boldsymbol{\Sigma}_{s,t-1})$, given all previous controls and measurements is assumed to be available. In the motion update, the predicted belief over the current states, $p(\boldsymbol{s}_t|\boldsymbol{u}_{0:t}, \boldsymbol{z}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}'_{s,t}, \boldsymbol{\Sigma}'_{s,t})$, given all controls and *previous* measurements is calculated. The state transition from the previous states to the current states is given by the noisy, nonlinear function

$$\boldsymbol{s}_t = \boldsymbol{g}(\boldsymbol{s}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{n}_r, \quad \boldsymbol{n}_r \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}). \tag{4.1.1}$$

The EKF is applicable to state transitions that can be described by this equation, which comes from the modelling process of the problem. This function is linearised using the first two terms of the Taylor series expansion. The partial derivative of $\boldsymbol{g}$ with respect to $\boldsymbol{s}_{t-1}$ is calculated at the linearisation point, which is chosen as its value at the mean of the previous belief, $\boldsymbol{\mu}_{s,t-1}$, which results in the Jacobian

$$\boldsymbol{G} = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{s}_{t-1}, \boldsymbol{u}_t)}{\partial \boldsymbol{s}_{t-1}}\right|_{\boldsymbol{s}_{t-1}=\boldsymbol{\mu}_{s,t-1}}. \tag{4.1.2}$$

The mean of the predicted belief is calculated by passing the mean of the prior belief through $\boldsymbol{g}$,

$$\boldsymbol{\mu}'_{s,t} = \boldsymbol{g}(\boldsymbol{\mu}_{s,t-1}, \boldsymbol{u}_t) \tag{4.1.3}$$

and the covariance matrix is calculated by using the Jacobian and the noise covariance,

$$\boldsymbol{\Sigma}'_{s,t} = \boldsymbol{G}\boldsymbol{\Sigma}_{s,t-1}\boldsymbol{G}^T + \boldsymbol{R}. \tag{4.1.4}$$

Next, the measurement update is performed to calculate the posterior belief of the states, $p(\boldsymbol{s}_t|\boldsymbol{u}_{0:t}, \boldsymbol{z}_{0:t}) = \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\Sigma}_{s,t})$, given all controls and measurements, where the current measurements are also now included. The measurements received are described by the noisy, nonlinear function

$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{s}_t) + \boldsymbol{n}_q, \quad \boldsymbol{n}_q \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \tag{4.1.5}$$

which is obtained in the modelling process of the problem. Similar to the motion update, the partial derivative of $\boldsymbol{h}$ with respect to $\boldsymbol{s}_t$ is calculated at the linearisation point, which is chosen as its value at the mean of the predicted belief, $\boldsymbol{\mu}'_{s,t}$, resulting in the Jacobian

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{s}_t)}{\partial \boldsymbol{s}_t}\right|_{\boldsymbol{s}_{t-1}=\boldsymbol{\mu}'_{s,t-1}}. \tag{4.1.6}$$

Using this Jacobian, the Kalman gain matrix is defined as

$$\boldsymbol{K} = \boldsymbol{\Sigma}'_t\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{\Sigma}'_t\boldsymbol{H}^T + \boldsymbol{Q})^{-1}. \tag{4.1.7}$$

By substituting the mean of the predicted belief into Equation 4.1.5, a predicted measurement can be calculated and the mean of the posterior distribution can be calculated as

$$\boldsymbol{\mu}_{s,t} = \boldsymbol{\mu}'_{s,t} + \boldsymbol{K}\left(\boldsymbol{z}_t - \boldsymbol{h}(\boldsymbol{\mu}'_{s,t})\right). \tag{4.1.8}$$

The covariance matrix of the posterior belief can also be calculated as

$$\boldsymbol{\Sigma}_{s,t} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{\Sigma}'_{s,t}. \tag{4.1.9}$$

The posterior belief of the states, $p(\boldsymbol{s}_t|\boldsymbol{u}_{0:t}, \boldsymbol{z}_{0:t})$, given all available controls and measurements have now been calculated. The derivations of these calculations are given by Thrun *et al.* [1]. The motion update of the EKF described here is used in our SLAM motion update in the next section, but the EKF measurement update is not used in our SLAM measurement update.

4.2 Motion Update

In the SLAM context, the motion update involves updating the belief over the robot pose at each timestep. This update makes use of control inputs given to the robot's actuators or sensor information of the robot's motion to predict the next pose of the robot. During this step, the uncertainty over the robot's pose usually increases due to the added noise of the control inputs or sensor information.

At the start of the motion update we have a state vector, $\boldsymbol{s}_{t-1} = \begin{bmatrix}\boldsymbol{x}_{t-1}^T & \boldsymbol{m}^T\end{bmatrix}^T$, which includes the previous robot pose, $\boldsymbol{x}_{t-1}$, and a map vector of all landmarks, $\boldsymbol{m}$. We assume that we have a prior belief over this state vector, given all previous evidence, $p(\boldsymbol{s}_{t-1}|\mathcal{E}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{s,t-1}, \boldsymbol{\Sigma}_{s,t-1})$, where the evidence, $\mathcal{E}_{0:t-1}$, includes all previous Lidar measurements, $\boldsymbol{z}_{0:t-1}$, and all previous odometry measurements, $\boldsymbol{u}_{0:t-1}$. The odometry measurements are used to describe the state transition from the previous robot pose to the next pose, therefore they are used as the controls discussed in the previous section. The goal of the motion update is to use the new odometry measurements, $\boldsymbol{u}_t$, to obtain a predicted belief over the new state vector, $p(\boldsymbol{s}_t|\mathcal{E}_{0:t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\Sigma}_{s,t})$, where $\boldsymbol{s}_t = \begin{bmatrix}\boldsymbol{x}_t^T & \boldsymbol{m}^T\end{bmatrix}^T$ includes the new robot pose and not the previous robot pose.

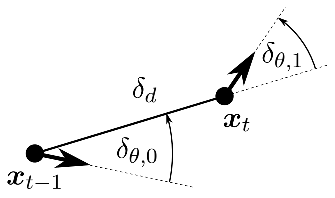


Figure 4.1 – Diagram of odometry motion model.

In this project the odometry motion model, as discussed in Thrun *et al.* [1, p. 132-139] and visualised in Figure 4.1, is used to predict the robot's pose. The odometry motion model uses measurements of the incremental motion of the robot, from sensors such as wheel encoders, in contrast with using robot control inputs for the same purpose. These measurements are then converted to an initial rotation, $\delta_{\theta,0}$, a forward translation, δ_d, and a final rotation, $\delta_{\theta,1}$, that describes the robot's incremental motion over the sampling period. The current pose of the robot, $\boldsymbol{x}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$, can be calculated using the previous pose, $\boldsymbol{x}_{t-1} = \begin{bmatrix} x_{t-1} & y_{t-1} & \theta_{t-1} \end{bmatrix}^T$, and the odometry measurements, $\boldsymbol{u}_t = \begin{bmatrix} \delta_{\theta,0} & \delta_d & \delta_{\theta,1} \end{bmatrix}^T$, where

$$\boldsymbol{x}_t = \boldsymbol{g}_x(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) = \boldsymbol{x}_{t-1} + \begin{bmatrix} \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_{\theta,0} + \delta_{\theta,1} \end{bmatrix}. \tag{4.2.1}$$

These odometry measurements are typically noisy,

$$\boldsymbol{u}_t = \boldsymbol{u}'_t + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_n), \tag{4.2.2}$$

where each measurement contains additive noise that we assume to be zero-mean Gaussian distributed. Here $\boldsymbol{u}'_t$ is the noiseless odometry measurements. The covariance of the noise is defined as

$$\boldsymbol{\Sigma}_n = \begin{bmatrix} \sigma^2_{\theta,0} & 0 & 0 \\ 0 & \sigma^2_d & 0 \\ 0 & 0 & \sigma^2_{\theta,1} \end{bmatrix}, \tag{4.2.3}$$

where $\sigma_{\theta,0}$ and $\sigma_{\theta,1}$ are the standard deviations of the rotations and σ_d is the standard deviation of the translation.

In this explanation we first give the calculation of the predicted belief over the current pose alone, $p(\boldsymbol{x}_t|\mathcal{E}_{0:t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{x,t}, \boldsymbol{\Sigma}_{x,t})$, and then extend this to the calculation of the predicted belief of the entire state space. We approximate all states as Gaussian random variables, therefore the transformation in Equation 4.2.1 needs to be linearised. We use the standard EKF SLAM motion update, similar to the one explained in Thrun *et al.* [1, p. 312-322], but using the odometry motion model instead of the velocity motion model. This motion update involves calculating the Jacobian of Equation 4.2.1 with respect to $\boldsymbol{x}_{t-1}$, which is calculated as

$$\boldsymbol{G}_x = \frac{\partial \boldsymbol{g}_x(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}{\partial \boldsymbol{x}_{t-1}} = \boldsymbol{I} + \begin{bmatrix} 0 & 0 & -\delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) \\ 0 & 0 & \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.2.4}$$

Since the process noise of this model is not additive to the state transition function, we also need to calculate the Jacobian of Equation 4.2.1 with respect to the odometry measurements, $\boldsymbol{u}_t$, as

$$\boldsymbol{H}_x = \frac{\partial \boldsymbol{g}_x(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)}{\partial \boldsymbol{u}_t} = \begin{bmatrix} -\delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) & \cos(\theta_{t-1} + \delta_{\theta,0}) & 0 \\ \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) & \sin(\theta_{t-1} + \delta_{\theta,0}) & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4.2.5}$$

Assuming we have a prior belief over the previous pose, $p(\boldsymbol{x}_{t-1}|\mathcal{E}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_{x,t-1}, \boldsymbol{\Sigma}_{x,t-1})$, we can calculate the mean of the predicted pose using Equation 4.2.1,

$$\boldsymbol{\mu}_{x,t} = \boldsymbol{\mu}_{x,t-1} + \begin{bmatrix} \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_{\theta,0} + \delta_{\theta,1} \end{bmatrix}. \tag{4.2.6}$$

The covariance matrix of the predicted pose, can also be calculated using the Jacobians,

$$\boldsymbol{\Sigma}_{x,t} = \boldsymbol{G}_x \boldsymbol{\Sigma}_{x,t-1} \boldsymbol{G}_x{}^T + \boldsymbol{H}_x \boldsymbol{\Sigma}_n \boldsymbol{H}_x{}^T. \tag{4.2.7}$$

The state vector in the SLAM scope includes the robot pose, as well as the map vector, $\boldsymbol{m}$, therefore the transformation from the complete previous state vector, $\boldsymbol{s}_{t-1}$, to the current state vector, $\boldsymbol{s}_t$, needs to be calculated. In order to do this we define a matrix,

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_{[3\times 3]} & \boldsymbol{0}_{[M\times 3]} \end{bmatrix}, \tag{4.2.8}$$

where M is the dimensions of $\boldsymbol{m}$, to expand the 3-dimensional pose vector to the full state space. Since the map remains the same between timesteps, $\boldsymbol{F}$ can be used to transform the pose transformation in Equation 4.2.1 to the higher-dimensional state space of the complete state vector. Using the matrix $\boldsymbol{F}$, Equation 4.2.1 is expanded to the full state space:

$$\boldsymbol{s}_t = \boldsymbol{s}_{t-1} + \boldsymbol{F}^T \begin{bmatrix} \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) \\ \delta_{\theta,0} + \delta_{\theta,1} \end{bmatrix}. \tag{4.2.9}$$

The Jacobian of this transformation with respect to $\boldsymbol{s}_{t-1}$ is

$$\boldsymbol{G} = \boldsymbol{I} + \boldsymbol{F}^T \begin{bmatrix} 0 & 0 & -\delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) \\ 0 & 0 & \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{F}, \tag{4.2.10}$$

and the Jacobian with respect to $\boldsymbol{u}_t$ is

$$\boldsymbol{H} = \boldsymbol{F}^T \begin{bmatrix} -\delta_d \sin(\theta_{t-1} + \delta_{\theta,0}) & \cos(\theta_{t-1} + \delta_{\theta,0}) & 0 \\ \delta_d \cos(\theta_{t-1} + \delta_{\theta,0}) & \sin(\theta_{t-1} + \delta_{\theta,0}) & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4.2.11}$$

We can now calculate the predicted belief over the full state space, $p(\boldsymbol{s}_t | \mathcal{E}_{0:t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\Sigma}_{s,t})$, using the prior belief, $p(\boldsymbol{s}_t | \mathcal{E}_{0:t-1}, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\Sigma}_{s,t})$, and odometry measurements, $\boldsymbol{u}_t$. The mean of the predicted belief is calculated as

$$\boldsymbol{\mu}_{s,t} = \boldsymbol{\mu}_{s,t-1} + \boldsymbol{F}^T \begin{bmatrix} \delta_d \cos(\mu_{\theta,t-1} + \delta_{\theta,0}) \\ \delta_d \sin(\mu_{\theta,t-1} + \delta_{\theta,0}) \\ \delta_{\theta,0} + \delta_{\theta,1} \end{bmatrix} \tag{4.2.12}$$

and the covariance matrix is calculated as

$$\boldsymbol{\Sigma}_{s,t} = \boldsymbol{G} \boldsymbol{\Sigma}_{s,t-1} \boldsymbol{G}^T + \boldsymbol{H} \boldsymbol{\Sigma}_n \boldsymbol{H}^T. \tag{4.2.13}$$

Once the robot's pose is updated and the predicted belief, $p(\boldsymbol{s}_t | \mathcal{E}_{0:t-1}, \boldsymbol{u}_t)$, is obtained, the measurements taken by the robot's sensors can be used to perform the measurement update.

4.3 Measurement Update

In this section we discuss the measurement update used in this SLAM application, which is executed every time the Lidar sensor takes new measurements. We first discuss segmenting the measurements into sets, then associating these sets to landmarks. Next, the update of the state space variables is discussed, followed by adding new landmarks to the state space.

4.3.1 Measurement Segmentation

When new measurements are taken, they are first segmented into different sets that are likely to come from the same landmark. Each of these sets are evaluated separately in the landmark modelling, correspondence and updating steps.

We take a simple approach to the segmentation, similar to the approach in Nieto *et al.* [14]. We only evaluate the difference in angle and the distance between measurements, $\boldsymbol{z}_{p,i} = \begin{bmatrix} r_i & \phi_i \end{bmatrix}^T$, which are defined as

$$\Delta\phi_{i,j} = |\phi_i - \phi_j| \tag{4.3.1}$$

and

$$d_{i,j} = \sqrt{r_i^2 + r_j^2 - 2r_i r_j \cos(\Delta\phi_{i,j})}. \tag{4.3.2}$$

If both of these values lie under certain thresholds, ϕ_{thres} and d_{thres}, the two measurements are assumed to come from the same landmark. These measurements are then grouped in the same measurement set, $\mathcal{Z}_{p,k}$,

$$\boldsymbol{z}_{p,i}, \boldsymbol{z}_{p,j} \in \mathcal{Z}_{p,k} \quad \text{if} \quad \Delta\phi_{i,j} \leq \phi_{\text{thres}} \quad \text{and} \quad d_{i,j} \leq d_{\text{thres}}, \tag{4.3.3}$$

where each measurement is only part of one set and a measurement can only be part of a set if the conditions hold true with at least one other measurement in the set.

We choose these thresholds as $\phi_{\text{thres}} = 2\phi_{\text{res}}$, where ϕ_{res} is the angular resolution of the Lidar, and $d_{\text{thres}} = 1$. The value for ϕ_{thres} is chosen so that a single outlier or faulty measurement will not split a measurement set and the choice for d_{thres} assumes that measurements that are closer than 1 m from each other come from the same landmark. These choices can be adjusted for different applications, with different sensors and prior information about the environment. Figure 4.2 shows an example of this segmentation. Measurements with gaps between them (red and green) as well as measurements far apart (green and blue) are separated into different segments. A straight line section, as seen at the bottom of this figure is separated into different sets when the measurements on this line are far apart.

Once these measurements are segmented into sets, each set, $\mathcal{Z}_{p,k}$, is evaluated to determine whether it should be used in the measurement update. The reason for this evaluation is that measurements that do not give sufficient information about a landmark should not be used in the measurement update. The set will be evaluated on the number of measurements and the linearity of the measurements. The number of measurements are only compared to a threshold value, n_{thres}, and if the set size is less than this value, the set is not used in the measurement update. If the set has enough measurements, the algorithm to choose initial lines for model selection,

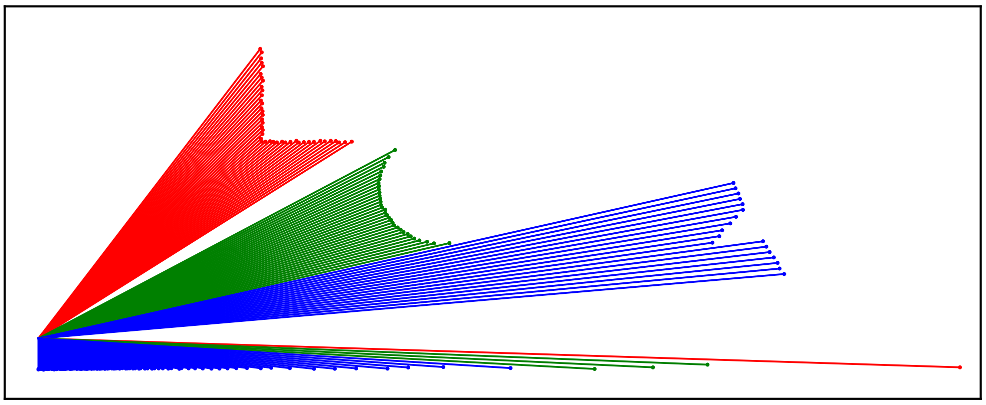


Figure 4.2 – Segmentation of measurements into sets. Different measurement sets are shown in alternating colours.

Algorithm 3, is executed using these measurements. If this algorithm returns only one line, the set will not be used in the measurement update, since it is deemed that it does not contain enough information for landmark correspondence or update.

All the valid measurement sets are used further in the measurement update, checking for associations to existing landmarks or using the measurements to create new landmarks. The discarded sets are, however, not used further, but their measurements can be used to calculate unknown lines of other landmarks.

4.3.2 Landmark Correspondence

A crucial part of doing SLAM with landmarks is associating new measurements to specific landmarks. In the previous section, we discussed how to retrieve measurements from a single landmark and in Section 3.2.2 we discussed a way of measuring if a set of measurements originate from a landmark. The approach in Section 3.2.2, however, assumes that the robot pose is known, which is not the case in SLAM. To use this method, we select a likely pose, given the measurements and the landmark and then evaluate the measurements against the landmark.

To select this likely pose, we align the measurements with the landmark. The iterated closest point (ICP) algorithm, originally proposed by Besl and McKay [15], is a commonly-used scan-matching technique designed to align two point clouds to each other. We now discuss the ICP algorithm and the adaptation we make to use it with ou landmark model.

Given two point clouds, $\mathcal{P} = \{\boldsymbol{p}_0, \cdots, \boldsymbol{p}_{n-1}\}$ and $\mathcal{Q} = \{\boldsymbol{q}_0, \cdots, \boldsymbol{q}_{m-1}\}$, we want to obtain a rotation and translation, given by the transformation parameters $\boldsymbol{t} = \begin{bmatrix} t_x & t_y & t_\theta \end{bmatrix}^T$, to transform $\mathcal{Q}$ to $\mathcal{Q}'$, where $\mathcal{Q}'$ is aligned with $\mathcal{P}$. The transformation from $\mathcal{Q}$ to $\mathcal{Q}'$ is given by

$$\boldsymbol{q}'_i = \boldsymbol{R}\boldsymbol{q}_i + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{4.3.4}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \cos t_\theta & -\sin t_\theta \\ \sin t_\theta & \cos t_\theta \end{bmatrix} \tag{4.3.5}$$

and $\boldsymbol{q}'_i \in \mathcal{Q}'$. The ICP algorithm associates every point in $\mathcal{Q}$ to the closest point in $\mathcal{P}$, resulting in an association

$$c_i = \arg\min_j \left|\left|\boldsymbol{q}_i - \boldsymbol{p}_j\right|\right|, \tag{4.3.6}$$

so that $\boldsymbol{p}_{c_i}$ is the closest point to $\boldsymbol{q}_i$ and the distance between these points is

$$\boldsymbol{d}_i = \boldsymbol{q}_i - \boldsymbol{p}_{c_i}. \tag{4.3.7}$$

Once these associations are made, the algorithm attempts to minimise the mean square error between all point pairs by optimising the transformation parameters $\boldsymbol{t}$. The distance between the transformed point, $\boldsymbol{q}'_i$, and its associated point in $\mathcal{P}$ is

$$\boldsymbol{d}'_i = \boldsymbol{q}'_i - \boldsymbol{p}_{c_i} \tag{4.3.8}$$

and the Jacobian of this equation with regards to $\boldsymbol{t}$, linearised around $\boldsymbol{t} = \boldsymbol{0}$, is

$$\boldsymbol{J}_i = \begin{bmatrix} 1 & 0 & -q_{y,i} \\ 0 & 1 & q_{x,i} \end{bmatrix}. \tag{4.3.9}$$

The transformation vector is then calculated with the following equations,

$$\begin{aligned} \boldsymbol{H} &= \sum_i \boldsymbol{J}_i^T \boldsymbol{J}_i \\ \boldsymbol{b} &= \sum_i \boldsymbol{J}_i^T \boldsymbol{d}_i \\ \boldsymbol{t} &= \boldsymbol{H}^{-1}\boldsymbol{b}, \end{aligned} \tag{4.3.10}$$

and the transformed set $\mathcal{Q}$ is calculated with Equation 4.3.4.

This process of finding the corresponding points and calculating $\boldsymbol{t}$ is repeated iteratively by replacing $\mathcal{Q}$ with $\mathcal{Q}'$ until $\boldsymbol{t}$ converges to $\boldsymbol{0}$. This results in a $\mathcal{Q}'$ which is aligned with $\mathcal{P}$. It should be noted that $\boldsymbol{t}$ calculated here only describes the transformation of the last iteration and not the transformation from the original set to the aligned set. The complete transformation vector, $\boldsymbol{t}_f$, can be obtained by maintaining the transformation vector of the original set to the current set in each iteration of the ICP. This is done by initialising $\boldsymbol{t}_f := \boldsymbol{0}$ and updating it after each iteration, by applying the new transformation to it,

$$\boldsymbol{t}_f := \begin{bmatrix} t_{f,x}\cos t_\theta - t_{f,y}\sin t_\theta + t_x \\ t_{f,x}\sin t_\theta + t_{f,y}\cos t_\theta + t_y \\ t_{f,\theta} + t_\theta \end{bmatrix}. \tag{4.3.11}$$

It should also be noted that the ICP algorithm uses all points in $\mathcal{Q}$ once, but not necessarily all point in $\mathcal{P}$. Some points in $\mathcal{P}$ could also be used multiple times. This means that aligning $\mathcal{Q}$ to $\mathcal{P}$ and aligning $\mathcal{P}$ to $\mathcal{Q}$ could result in different alignments.

We want to align a measurement set, $\mathcal{Z}_c = \{\boldsymbol{z}_{c,0}, \cdots, \boldsymbol{z}_{c,n-1}\}$, defined in the inertial Cartesian reference frame, to a landmark described by lines. Both of these have uncertainty over them, but for the purpose of alignment we only consider the means of the distributions. The measurements are already in a point cloud form, but the landmark model is not. To use the ICP algorithm with the landmark model, the perpendicular distance of each measurement to the closest line in the model is calculated. The distance from measurement $\boldsymbol{z}_{c,i} = \begin{bmatrix} z_{x,i} & z_{y,i} \end{bmatrix}^T$ to the j^{th} line, parameterised by $\boldsymbol{v}_j = \begin{bmatrix} v_{x,j} & v_{y,j} \end{bmatrix}^T$ and $\boldsymbol{v}_{j+1} = \begin{bmatrix} v_{x,j+1} & v_{y,j+1} \end{bmatrix}^T$, is calculated by

$$e_{i,j} = (z_{x,i} - v_{x,j}) \sin \psi_j - (z_{y,i} - v_{y,j}) \cos \psi_j, \tag{4.3.12}$$

where ψ_j is the angle of the line, calculated as

$$\psi_j = \arctan \left(\frac{v_{y,j+1} - v_{y,j}}{v_{x,j+1} - v_{x,j}} \right). \tag{4.3.13}$$

This equation is derived in Appendix A. As in Section 3.2.2, only the visible lines are considered in this calculation, given the current pose estimate.

Each measurement is associated with its closest line,

$$c_i = \arg \min_j e_{i,j}. \tag{4.3.14}$$

If it is an unknown line, $u_{c_i} > 0.5$, and the measurement lies behind the line, the measurement is not used in this iteration of the ICP, since we have no information about that part of the environment. However, if a measurement lies in front of an unknown line it is in the empty region of the line and should be used in the alignment to move it behind the line. If this is the case or the line is an object boundary, $u_{c_i} \leq 0.5$, the closest point on that line is considered for the ICP algorithm and the difference vector of Equation 4.3.7 is replaced with

$$\boldsymbol{d}_i = e_{i,c_i} \begin{bmatrix} -\sin \psi_j \\ \cos \psi_j \end{bmatrix}. \tag{4.3.15}$$

The ICP algorithm is adapted to use this closest distance to calculate $\boldsymbol{t}$ and $\mathcal{Z}'_c$, the transformed measurement set, for each iteration.

The estimated pose, $\boldsymbol{x}_{\text{icp}}$, after each iteration of the ICP algorithm can be calculated by applying the complete transformation, $\boldsymbol{t}_f$, to the original pose, $\boldsymbol{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$,

$$\boldsymbol{x}_{\text{icp}} = \begin{bmatrix} x \cos t_{f,\theta} - y \sin t_{f,\theta} + t_{f,x} \\ x \sin t_{f,\theta} + y \cos t_{f,\theta} + t_{f,y} \\ \theta + t_\theta \end{bmatrix}. \tag{4.3.16}$$

During the execution of the ICP algorithm, this pose estimate is used to reason whether lines of the model are visible or not. The final pose estimate, when the ICP algorithm has converged, is then used as the pose estimate to reason about correspondence with the landmark. In Figure 4.3, two examples of the ICP algorithm are shown where misaligned measurements are successfully aligned to the landmark.

In (b) it can be seen that the measurements in front of the unknown line is moved behind it in the alignment. When the measurements are behind the unknown line, they have no further effect on the alignment and the part measurement associated with the object boundaries can be successfully aligned to the model.

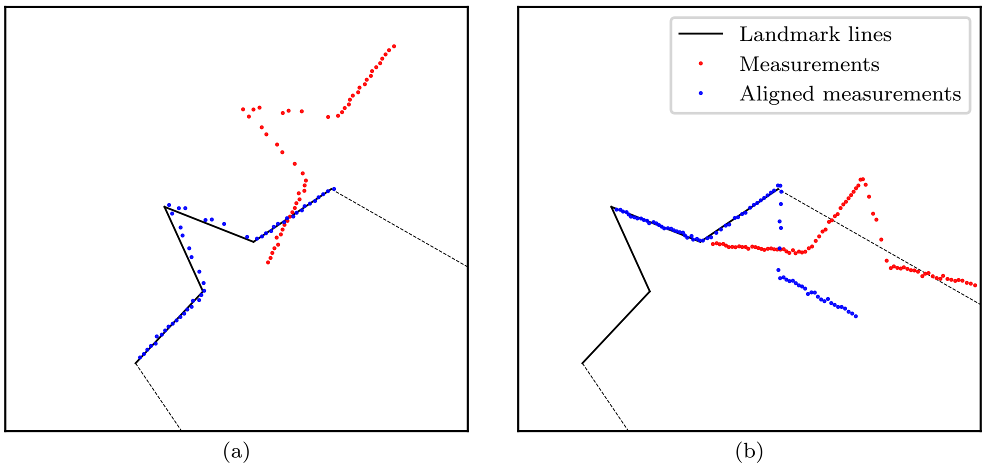


Figure 4.3 – Measurement to landmark alignment using ICP.

Once this pose estimate is obtained, the method discussed in Section 3.2.2 can be used to decide whether the measurements come from the landmark. However, we do not want to perform the ICP algorithm between each measurement set and landmark as this can be computationally very expensive. A lot of landmarks will be far from the measurement set, which we can ignore beforehand. To ignore these landmarks, we evaluate the distance between the centroid of the measurement set and the reference points of the landmarks and ignore landmarks if this distance is above a threshold. Landmarks are also only considered if they fall in the robot's field of view.

For each measurement set, an ICP pose estimate is obtained for each landmark in its vicinity. Using the method discussed in Section 3.2.2, we can reason whether the measurements can come from the landmark if measured from the estimated pose. This method can, however, associate multiple landmarks to a measurement set. To select a single landmark from these possible matches we use the maximum likelihood correspondence. The likelihood that is evaluated is the combined likelihood that the measurements fit the model and that the robot pose is at the estimated pose. The log likelihood in Equation 3.2.21, $\log\left(p(\boldsymbol{z}_p|\boldsymbol{v},\boldsymbol{x}_{\text{icp}})\right)$, is obtained, taking into account the unknown likelihoods as in Equation 3.2.43, and the log likelihood that the robot pose is at the ICP pose estimate is calculated as

$$\log(p(\boldsymbol{x}=\boldsymbol{x}_{\text{icp}})) = -0.5\left(\log(2\pi|\boldsymbol{\Sigma}_x|) + (\boldsymbol{x}_{\text{icp}}-\boldsymbol{\mu}_x)^T\boldsymbol{\Sigma}_x^{-1}(\boldsymbol{x}_{\text{icp}}-\boldsymbol{\mu}_x)\right), \quad (4.3.17)$$

where $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ is the distribution over the robot pose. These two log likelihoods are added together to obtain the likelihood that the measurements taken at the estimated pose match the landmark,

$$\log(p(\boldsymbol{z}_p, \boldsymbol{x} = \boldsymbol{x}_{\text{icp}}|\boldsymbol{v})) = \log\left(p(\boldsymbol{z}_p|\boldsymbol{v}, \boldsymbol{x}_{\text{icp}})\right) + \log(p(\boldsymbol{x} = \boldsymbol{x}_{\text{icp}})). \tag{4.3.18}$$

The measurement set is then associated with the landmark resulting in the highest likelihood. If it is not corresponded to a landmark, the measurements are considered to be added to the map as a new landmark, which is explained in Section 4.3.4.

Once the measurements are associated with the landmark, we evaluate if the measurements contain enough information about the landmark to perform a useful update. To do this we find the measurement-to-line correspondences, $\boldsymbol{c}$. We consider a line observed if at least two measurements are associated with it. If at least two object boundary lines, with unknown probability $u_i \leq 0.5$, are observed, the information is deemed sufficient. If this check is passed, the landmark and measurement set are used in the state space update discussed in the next subsection. The estimated pose, $\boldsymbol{x}_{\text{icp}}$, associated with this correspondence is also used in the landmark updating step discussed in Section 3.2.3 and used in the state space update.

4.3.3 State Space Update

Once the measurement sets are associated with landmarks, the measurements are used to simultaneously update the landmark parameters and other variables in the SLAM state space.

In this approach of SLAM, each landmark is represented by a single reference point, $\boldsymbol{r}_i$, in the classical EKF SLAM state space and the line model of the landmark is conditionally independent of the robot states and other landmarks, given its reference point. The chosen structure contains a distribution over the robot states, $\boldsymbol{x}$, and all the landmark references, $p(\boldsymbol{x}, \boldsymbol{r}_0, \cdots, \boldsymbol{r}_{n-1})$, for n landmarks, as well as a distribution over each landmark model, given its reference, $p(\boldsymbol{v}_i|\boldsymbol{r}_i)$. A factor graph representation [22] of this structure is shown in Figure 4.4. This chosen structure aims to decrease the complexity of the state space significantly.

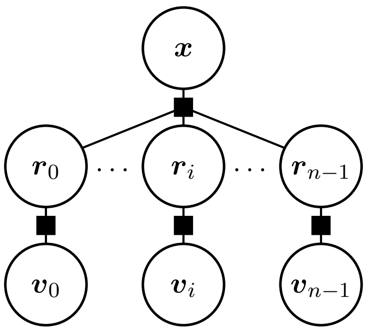


Figure 4.4 – Factor graph representation of the proposed state space structure [22]. The landmark models are only connected to its references, which are then connected to the robot states and other references.

The measurements obtained from the landmark are transformed to an observed distribution over the landmark parameters, as explained in Section 3.2.3, using the estimated pose obtained from the previous subsection. The reference point of the landmark is thus never measured directly and the EKF SLAM measurement update cannot be used directly to simultaneously update the line model parameters, landmark reference points and robot states with this structure. In contrast to other methods that use reference points of the landmarks in the state space update, we derive a new state space update to update all states, including the line model parameters, simultaneously.

We now only consider the i^{th} landmark, represented by $\boldsymbol{r}_i$ and $\boldsymbol{v}_i$, and group all other reference points in a map vector, $\boldsymbol{m}_i = \begin{bmatrix}\boldsymbol{r}_0^T & \cdots & \boldsymbol{r}_{i-1}^T & \boldsymbol{r}_{i+1}^T & \cdots & \boldsymbol{r}_{n-1}^T\end{bmatrix}^T$. We omit the index i for the rest of this subsection for simplicity.

We assume that we have a joint distribution, given all previous evidence, $\mathcal{E}_{0:t-1}$, which is factorised as

$$p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}, \boldsymbol{v}|\mathcal{E}_{0:t-1}) = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\mathcal{E}_{0:t-1})\, p(\boldsymbol{v}|\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}, \mathcal{E}_{0:t-1}), \tag{4.3.19}$$

where the evidence includes all previous Lidar measurements, $\boldsymbol{z}_{0:t-1}$, and odometry measurements, $\boldsymbol{u}_{0:t}$. The first factor is denoted as $\Phi_1 = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\mathcal{E}_{0:t-1})$ and in the second factor $\boldsymbol{v}$ is assumed to be conditionally independent of $\boldsymbol{x}$ and $\boldsymbol{m}$ given $\boldsymbol{r}$ and is denoted as $\Phi_2 = p(\boldsymbol{v}|\boldsymbol{r}, \mathcal{E}_{0:t-1})$

When new measurements, $\boldsymbol{z}_t$, are received from this landmark, a distribution over the landmark parameters can be obtained, given the robot states and new measurements, introducing a new factor, $\Phi_3 = p(\boldsymbol{v}|\boldsymbol{x}, \boldsymbol{z}_t)$. This distribution is the observed distribution discussed in Section 3.2.3. These three factors are visualised in the first graph of Figure 4.5.

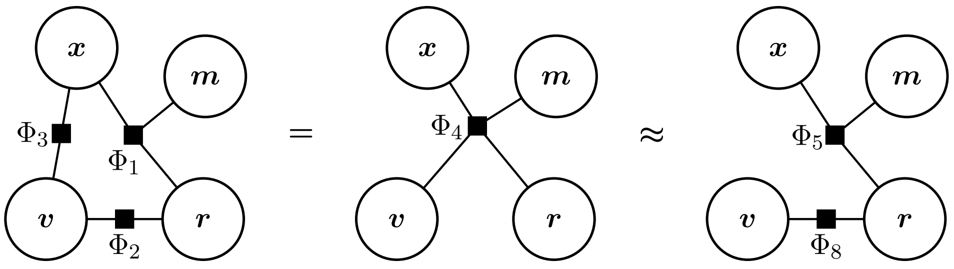


Figure 4.5 – Factor graph representation of the measurement update derivation. The first graph shows the prior factors, Φ_1 and Φ_2, as in Figure 4.4 and the new factor, Φ_3, obtained from the new measurements. A joint factor over all states is created in the second graph and then in the last graph approximated as the structure in Figure 4.4.

These three factors are combined to produce a joint factor over all states,

$$\Phi_4 = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}, \boldsymbol{v}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\mathcal{E}_{0:t-1})p(\boldsymbol{v}|\boldsymbol{r}, \mathcal{E}_{0:t-1})p(\boldsymbol{v}|\boldsymbol{x}, \boldsymbol{z}_t), \tag{4.3.20}$$

visualised in the second graph of Figure 4.5. From this joint factor, we want to obtain the structure in the last graph of Figure 4.5, which is similar to Figure 4.4, given by

$$p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}, \boldsymbol{v}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) \approx p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1})p(\boldsymbol{v}|\boldsymbol{r}, \boldsymbol{z}_t, \mathcal{E}_{0:t-1}). \tag{4.3.21}$$

This is, however, an approximation, since the model parameters, $\boldsymbol{v}$, actually depend on its reference, $\boldsymbol{r}$ and the robot states, $\boldsymbol{x}$, where the exact equation is

$$p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1}) = p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})p(\boldsymbol{v}|\boldsymbol{x},\boldsymbol{r},\boldsymbol{z}_t,\mathcal{E}_{0:t-1}). \tag{4.3.22}$$

The marginal distribution over the robot states and references is obtained by marginalising out the landmark parameters from the joint distribution, Φ_4,

$$\Phi_5 = p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1}) = \int p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})\mathrm{d}\boldsymbol{v}. \tag{4.3.23}$$

The approximate conditional distribution over the landmark parameters is calculated by marginalising out $\boldsymbol{x}$ and $\boldsymbol{m}$ from Φ_4,

$$\Phi_6 = p(\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1}) = \int\int p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{m}, \tag{4.3.24}$$

and then reducing it to $\boldsymbol{v}$,

$$\Phi_8 = p(\boldsymbol{v}|\boldsymbol{r},\boldsymbol{z}_t,\mathcal{E}_{0:t-1}) = \frac{p(\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})}{p(\boldsymbol{r}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})}, \tag{4.3.25}$$

where

$$\Phi_7 = p(\boldsymbol{r}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1}) = \int p(\boldsymbol{r},\boldsymbol{v}|\boldsymbol{z}_t,\mathcal{E}_{0:t-1})\mathrm{d}\boldsymbol{v}. \tag{4.3.26}$$

The process of calculating these factors in the canonical form is now explained. The first three factors are first defined in the canonical form and then the other factors are subsequently derived. The distribution over the robot states and landmark references given all previous measurements is defined as

$$\Phi_1 = p(\boldsymbol{x},\boldsymbol{m},\boldsymbol{r}|\mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\\boldsymbol{m}\\\boldsymbol{r}\end{bmatrix};\begin{bmatrix}\boldsymbol{K}_{xx} & \boldsymbol{K}_{xm} & \boldsymbol{K}_{xr}\\\boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} & \boldsymbol{K}_{mr}\\\boldsymbol{K}_{rx} & \boldsymbol{K}_{rm} & \boldsymbol{K}_{rr}\end{bmatrix},\begin{bmatrix}\boldsymbol{h}_x\\\boldsymbol{h}_m\\\boldsymbol{h}_r\end{bmatrix}\right). \tag{4.3.27}$$

This factor is obtained after the motion update by converting the state space to the canonical form.

The conditional distribution of Φ_2 can be represented by a noisy linear relationship between $\boldsymbol{r}$ and $\boldsymbol{v}$, given by

$$\boldsymbol{v} = \boldsymbol{A}_2\boldsymbol{r} + \boldsymbol{b}_2 + \boldsymbol{n}_2, \quad \boldsymbol{n}_2 \sim \mathcal{C}(\boldsymbol{K}_2, \boldsymbol{0}), \tag{4.3.28}$$

and is expressed in the canonical form as

$$\Phi_2 = p(\boldsymbol{v}|\boldsymbol{r},\mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{r}\\\boldsymbol{v}\end{bmatrix};\begin{bmatrix}\boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{A}_2 & -\boldsymbol{A}_2^T\boldsymbol{K}_2\\-\boldsymbol{K}_2\boldsymbol{A}_2 & \boldsymbol{K}_2\end{bmatrix},\begin{bmatrix}-\boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{b}_2\\\boldsymbol{K}_2\boldsymbol{b}_2\end{bmatrix}\right). \tag{4.3.29}$$

This factor is created by using the values of $\boldsymbol{A}_{vr}$, $\boldsymbol{b}_{vr}$ and $\boldsymbol{K}_v$ that are obtained from the previous time this landmark was observed and updated, calculated at the end of the measurement update. These parameters, however, need to be extended to the dimensions of the updated model to be used in this update. The extended parameters are calculated as

$$\begin{aligned}\boldsymbol{A}_2 &= \boldsymbol{F}^T\boldsymbol{A}_{vr},\\\boldsymbol{b}_2 &= \boldsymbol{F}^T\boldsymbol{b}_{vr},\\\boldsymbol{K}_2 &= \boldsymbol{F}^T\boldsymbol{K}_v\boldsymbol{F},\end{aligned} \tag{4.3.30}$$

where $\boldsymbol{F}$ is the transformation matrix described in Section 3.2.3.

Similar to Equation 4.3.28, the conditional distribution of Φ_3 can be represented by a noisy linear relationship between $\boldsymbol{x}$ and $\boldsymbol{v}$, given by

$$\boldsymbol{v} = \boldsymbol{A}_3\boldsymbol{x} + \boldsymbol{b}_3 + \boldsymbol{n}_3, \quad \boldsymbol{n}_3 \sim \mathcal{C}(\boldsymbol{K}_3, \boldsymbol{0}), \tag{4.3.31}$$

and is expressed in the canonical form as

$$\Phi_3 = p(\boldsymbol{v}|\boldsymbol{x}, \boldsymbol{z}_t) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{v}\end{bmatrix}; \begin{bmatrix}\boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & -\boldsymbol{A}_3^T\boldsymbol{K}_3\\ -\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_3\end{bmatrix}, \begin{bmatrix}-\boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3\\ \boldsymbol{K}_3\boldsymbol{b}_3\end{bmatrix}\right). \tag{4.3.32}$$

$\boldsymbol{A}_3$, $\boldsymbol{b}_3$ and $\boldsymbol{K}_3$ is obtained with the landmark updating step in Section 3.2.3, where $\boldsymbol{A}_3 = \boldsymbol{A}_{vx}$, $\boldsymbol{b}_3 = \boldsymbol{b}_{vx}$ and $\boldsymbol{K}_3 = \boldsymbol{K}_{v,\text{obs}}$. We note that the full information matrices of Φ_2 and Φ_3 are both singular, but they are never inverted.

To obtain the joint distribution over all states, all the factors are multiplied together, as in Equation 4.3.20, which means that the information matrices and vectors can be added together [22]. The joint distribution is thus expressed as

$$\Phi_4 = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}, \boldsymbol{v}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{m}\\ \boldsymbol{r}\\ \boldsymbol{v}\end{bmatrix}; \boldsymbol{K}_4, \boldsymbol{h}_4\right), \tag{4.3.33}$$

where the information matrix is given by

$$\boldsymbol{K}_4 = \begin{bmatrix}\boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_{xm} & \boldsymbol{K}_{xr} & -\boldsymbol{A}_3^T\boldsymbol{K}_3\\ \boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} & \boldsymbol{K}_{mr} & \boldsymbol{0}\\ \boldsymbol{K}_{rx} & \boldsymbol{K}_{rm} & \boldsymbol{K}_{rr} + \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{A}_2 & -\boldsymbol{A}_2^T\boldsymbol{K}_2\\ -\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0} & -\boldsymbol{K}_2\boldsymbol{A}_2 & \boldsymbol{K}_2 + \boldsymbol{K}_3\end{bmatrix} \tag{4.3.34}$$

and the information vector is given by

$$\boldsymbol{h}_4 = \begin{bmatrix}\boldsymbol{h}_x - \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3\\ \boldsymbol{h}_m\\ \boldsymbol{h}_r - \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{b}_2\\ \boldsymbol{K}_2\boldsymbol{b}_2 + \boldsymbol{K}_3\boldsymbol{b}_3\end{bmatrix}. \tag{4.3.35}$$

We have now calculated the updated joint distribution, given all measurements up to the current timestep, visualised by the middle graph of Figure 4.5. Next, the approximate factorisation of this distribution, as shown in the last graph of Figure 4.5, is calculated. This requires that the marginal distribution, $p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1})$, and the conditional distribution, $p(\boldsymbol{v}|\boldsymbol{r}, \mathcal{E}_{0:t-1})$, are calculated.

The marginal distribution over $\boldsymbol{x}$, $\boldsymbol{m}$ and $\boldsymbol{r}$ is calculated by marginalising out $\boldsymbol{v}$ from Φ_4, as in Equation 4.3.23. This factor is expressed in the canonical form as

$$\Phi_5 = p(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{r}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{m}\\ \boldsymbol{r}\end{bmatrix}; \boldsymbol{K}_5, \boldsymbol{h}_5\right), \tag{4.3.36}$$

where the information matrix is given by

$$\boldsymbol{K}_5 = \begin{bmatrix} \boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_{xm} & \boldsymbol{K}_{xr} \\ \boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} & \boldsymbol{K}_{mr} \\ \boldsymbol{K}_{rx} & \boldsymbol{K}_{rm} & \boldsymbol{K}_{rr} + \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{A}_2 \end{bmatrix} - \begin{bmatrix} -\boldsymbol{A}_3^T\boldsymbol{K}_3 \\ \boldsymbol{0} \\ -\boldsymbol{A}_2^T\boldsymbol{K}_2 \end{bmatrix} (\boldsymbol{K}_2 + \boldsymbol{K}_3)^{-1} \begin{bmatrix} -\boldsymbol{A}_3^T\boldsymbol{K}_3 \\ \boldsymbol{0} \\ -\boldsymbol{A}_2^T\boldsymbol{K}_2 \end{bmatrix}^T \tag{4.3.37}$$

and the information vector is given by

$$\boldsymbol{h}_5 = \begin{bmatrix} \boldsymbol{h}_x - \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3 \\ \boldsymbol{h}_m \\ \boldsymbol{h}_r - \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{b}_2 \end{bmatrix} - \begin{bmatrix} -\boldsymbol{A}_3^T\boldsymbol{K}_3 \\ \boldsymbol{0} \\ -\boldsymbol{A}_2^T\boldsymbol{K}_2 \end{bmatrix} (\boldsymbol{K}_2 + \boldsymbol{K}_3)^{-1}(\boldsymbol{K}_2\boldsymbol{b}_2 + \boldsymbol{K}_3\boldsymbol{b}_3). \tag{4.3.38}$$

Next, the conditional distribution, $p(\boldsymbol{v}|\boldsymbol{r}, \mathcal{E}_{0:t-1})$, should be calculated. To do this we first calculate the marginal distribution over $\boldsymbol{r}$ and $\boldsymbol{v}$, as in Equation 4.3.24. This distribution is expressed as

$$\Phi_6 = p(\boldsymbol{r}, \boldsymbol{v}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix}; \boldsymbol{K}_6, \boldsymbol{h}_6\right), \tag{4.3.39}$$

where the information matrix is given by

$$\boldsymbol{K}_6 = \begin{bmatrix} \boldsymbol{K}_{rr} + \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{A}_2 & -\boldsymbol{A}_2^T\boldsymbol{K}_2 \\ -\boldsymbol{K}_2\boldsymbol{A}_2 & \boldsymbol{K}_2 + \boldsymbol{K}_3 \end{bmatrix} - \begin{bmatrix} \boldsymbol{K}_{rx} & \boldsymbol{K}_{rm} \\ -\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_{xm} \\ \boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{K}_{xr} & -\boldsymbol{A}_3^T\boldsymbol{K}_3 \\ \boldsymbol{K}_{mr} & \boldsymbol{0} \end{bmatrix} \tag{4.3.40}$$

and the information vector is given by

$$\boldsymbol{h}_6 = \begin{bmatrix} \boldsymbol{h}_r - \boldsymbol{A}_2^T\boldsymbol{K}_2\boldsymbol{b}_2 \\ \boldsymbol{K}_2\boldsymbol{b}_2 + \boldsymbol{K}_3\boldsymbol{b}_3 \end{bmatrix} - \begin{bmatrix} \boldsymbol{K}_{rx} & \boldsymbol{K}_{rm} \\ -\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_{xm} \\ \boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{h}_x - \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3 \\ \boldsymbol{h}_m \end{bmatrix}. \tag{4.3.41}$$

We define the matrix being inverted here as

$$\boldsymbol{K}' = \begin{bmatrix} \boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{K}_{xm} \\ \boldsymbol{K}_{mx} & \boldsymbol{K}_{mm} \end{bmatrix} \tag{4.3.42}$$

and by using blockwise inversion [23], the inverse of this matrix is calculated as

$$\boldsymbol{K}'^{-1} = \begin{bmatrix} \boldsymbol{K}'^{-1}_{xx} & -\boldsymbol{K}'^{-1}_{xx}\boldsymbol{K}_{xm}\boldsymbol{K}^{-1}_{mm} \\ -\boldsymbol{K}^{-1}_{mm}\boldsymbol{K}_{mx}\boldsymbol{K}'^{-1}_{xx} & \boldsymbol{K}^{-1}_{mm} + \boldsymbol{K}^{-1}_{mm}\boldsymbol{K}_{mx}\boldsymbol{K}'^{-1}_{xx}\boldsymbol{K}_{xm}\boldsymbol{K}^{-1}_{mm} \end{bmatrix}, \tag{4.3.43}$$

where

$$\boldsymbol{K}'_{xx} = \boldsymbol{K}_{xx} + \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{A}_3 - \boldsymbol{K}_{xm}\boldsymbol{K}^{-1}_{mm}\boldsymbol{K}_{mx} \tag{4.3.44}$$

is the Schur complement of $\boldsymbol{K}_{mm}$ in Equation 4.3.42.

The joint distribution over $\boldsymbol{r}$ and $\boldsymbol{v}$ is sufficient to calculate the canonical form parameters of the conditional distribution over $\boldsymbol{v}$ given $\boldsymbol{r}$, given by Equation 4.3.25. Although the marginal distribution over $\boldsymbol{r}$ is also present in Equation 4.3.25, its canonical parameters has no effect on the parameterisation of Φ_8. Therefore, Φ_7 does not need to be calculated.

Similar to Equation 4.3.28, the conditional distribution of Φ_8 can be represented by a noisy linear relationship between $\boldsymbol{r}$ and $\boldsymbol{v}$, given by

$$\boldsymbol{v} = \boldsymbol{A}_8\boldsymbol{r} + \boldsymbol{b}_8 + \boldsymbol{n}_8, \quad \boldsymbol{n}_8 \sim \mathcal{C}(\boldsymbol{K}_8, \boldsymbol{0}) \tag{4.3.45}$$

and is expressed in the canonical form as

$$\Phi_8 = p(\boldsymbol{v}|\boldsymbol{r}, \mathcal{E}_{0:t-1}) = \mathcal{C}\left(\begin{bmatrix}\boldsymbol{r}\\ \boldsymbol{v}\end{bmatrix}; \begin{bmatrix}\boldsymbol{A}_8^T\boldsymbol{K}_8\boldsymbol{A}_8 & -\boldsymbol{A}_8^T\boldsymbol{K}_8\\ -\boldsymbol{K}_8\boldsymbol{A}_8 & \boldsymbol{K}_8\end{bmatrix}, \begin{bmatrix}-\boldsymbol{A}_8^T\boldsymbol{K}_8\boldsymbol{b}_8\\ \boldsymbol{K}_8\boldsymbol{b}_8\end{bmatrix}\right). \tag{4.3.46}$$

We now calculate the values of $\boldsymbol{A}_8$, $\boldsymbol{b}_8$ and $\boldsymbol{K}_8$ to parameterise this conditional distribution. Since $\Phi_8 = \frac{\Phi_6}{\Phi_7}$, the information matrix of Φ_8 can be calculated by subtracting the information matrix of Φ_7 from the information matrix of Φ_6,

$$\begin{bmatrix}\boldsymbol{A}_8^T\boldsymbol{K}_8\boldsymbol{A}_8 & -\boldsymbol{A}_8^T\boldsymbol{K}_8\\ -\boldsymbol{K}_8\boldsymbol{A}_8 & \boldsymbol{K}_8\end{bmatrix} = \boldsymbol{K}_6 - \begin{bmatrix}\boldsymbol{K}_7 & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} \tag{4.3.47}$$

and similarly the information vectors can be subtracted,

$$\begin{bmatrix}-\boldsymbol{A}_8^T\boldsymbol{K}_8\boldsymbol{b}_8\\ \boldsymbol{K}_8\boldsymbol{b}_8\end{bmatrix} = \boldsymbol{h}_6 - \begin{bmatrix}\boldsymbol{h}_7\\ \boldsymbol{0}\end{bmatrix}. \tag{4.3.48}$$

Since Φ_7 has no information about $\boldsymbol{v}$, the lower right of the expanded matrix in Equation 4.3.47 is zero. Therefore, $\boldsymbol{K}_8$ can be calculated using only $\boldsymbol{K}_6$. By substituting in the values of $\boldsymbol{K}_6$ in Equation 4.3.47, $\boldsymbol{K}_8$ is calculated as

$$\begin{aligned}\boldsymbol{K}_8 &= \boldsymbol{K}_2 + \boldsymbol{K}_3 - \begin{bmatrix}-\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0}\end{bmatrix}\boldsymbol{K}'^{-1}\begin{bmatrix}-\boldsymbol{A}_3^T\boldsymbol{K}_3\\ \boldsymbol{0}\end{bmatrix}\\ \therefore \boldsymbol{K}_8 &= \boldsymbol{K}_2 + \boldsymbol{K}_3 - \boldsymbol{K}_3\boldsymbol{A}_3\boldsymbol{K}'^{-1}_{xx}\boldsymbol{A}_3^T\boldsymbol{K}_3.\end{aligned} \tag{4.3.49}$$

Similarly, the lower left of the expanded matrix of Φ_7 in Equation 4.3.47 is zero. Therefore, the lower left of $\boldsymbol{K}_6$ is equal to the lower left of the information matrix of Φ_8,

$$-\boldsymbol{K}_8\boldsymbol{A}_8 = -\boldsymbol{K}_2\boldsymbol{A}_2 - \begin{bmatrix}-\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0}\end{bmatrix}\boldsymbol{K}'^{-1}\begin{bmatrix}\boldsymbol{K}_{xr}\\ \boldsymbol{K}_{mr}\end{bmatrix} \tag{4.3.50}$$

and $\boldsymbol{A}_8$ is calculated as

$$\therefore \boldsymbol{A}_8 = \boldsymbol{K}_8^{-1}\left(\boldsymbol{K}_2\boldsymbol{A}_2 - \boldsymbol{K}_3\boldsymbol{A}_3\boldsymbol{K}'^{-1}_{xx}\left(\boldsymbol{K}_{xr} - \boldsymbol{K}_{xm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{K}_{mr}\right)\right). \tag{4.3.51}$$

Similar to the information matrix, the lower part of the extended information vector of Φ_7 is zero and $\boldsymbol{b}_8$ is calculated by substituting in the lower part of $\boldsymbol{h}_6$ into

Equation 4.3.48,

$$\boldsymbol{K}_8\boldsymbol{b}_8 = \boldsymbol{K}_2\boldsymbol{b}_2 + \boldsymbol{K}_3\boldsymbol{b}_3 - \begin{bmatrix} -\boldsymbol{K}_3\boldsymbol{A}_3 & \boldsymbol{0} \end{bmatrix} \boldsymbol{K}'^{-1} \begin{bmatrix} \boldsymbol{h}_x - \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3 \\ \boldsymbol{h}_m \end{bmatrix}$$

$$\therefore \boldsymbol{b}_8 = \boldsymbol{K}_8^{-1}\left(\boldsymbol{K}_2\boldsymbol{b}_2 + \boldsymbol{K}_3\boldsymbol{b}_3 + \boldsymbol{K}_3\boldsymbol{A}_3\boldsymbol{K}'^{-1}_{xx}\left(\boldsymbol{h}_x - \boldsymbol{K}_{xm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{h}_m - \boldsymbol{A}_3^T\boldsymbol{K}_3\boldsymbol{b}_3\right)\right). \tag{4.3.52}$$

The matrices $\boldsymbol{K}_8$ and $\boldsymbol{A}_8$ and vector $\boldsymbol{b}_8$ are sufficient to describe factor Φ_8, which is the updated belief of the landmark parameters. The parameters are assigned according to $\boldsymbol{A}_{vr} := \boldsymbol{A}_8$, $\boldsymbol{b}_{vr} := \boldsymbol{b}_8$ and $\boldsymbol{K}_v := \boldsymbol{K}_8$, which are used to calculate the parameters of Φ_2 when the landmark is observed again. The updated covariance matrix of this landmark's vertices is also obtained by inverting the information matrix, $\boldsymbol{\Sigma}_v = \boldsymbol{K}_v^{-1}$. The parameters of Φ_5 and Φ_8 have now been calculated in terms of the parameters of Φ_1, Φ_2 and Φ_3, which concludes the update step depicted in Figure 4.5.

This state space update process, calculating the parameters of Φ_5 and Φ_8, is repeated for each landmark observed in a timestep. The updated distribution after each iteration, Φ_5, forms the first factor of the next iteration, Φ_1. After all the iterations, the updated distribution, Φ_5, is transformed back to the mean and covariance parameterisation,

$$p(\boldsymbol{x}, \boldsymbol{r}_0, \cdots, \boldsymbol{r}_{n-1}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1}) = \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s) = \mathcal{N}\left(\boldsymbol{K}_5^{-1}\boldsymbol{h}_5, \boldsymbol{K}_5^{-1}\right), \tag{4.3.53}$$

where $\boldsymbol{\mu}_s = \begin{bmatrix} \boldsymbol{\mu}_x^T & \boldsymbol{\mu}_{r,0}^T & \cdots & \boldsymbol{\mu}_{r,n-1}^T \end{bmatrix}^T$ is the mean vector of the robot states and all the reference points. This posterior distribution, $p(\boldsymbol{x}, \boldsymbol{r}_0, \cdots, \boldsymbol{r}_{n-1}|\boldsymbol{z}_t, \mathcal{E}_{0:t-1})$, in the normal form is then used in the next motion update.

Although this state space update is developed for the specific SLAM algorithm and landmark parameterisation that we implement, it can be used with other probabilistic landmark parameterisations as well as different robot states and reference dimensions.

Once all the iterations of the state space update are performed, the means of all the landmarks, in the inertial reference frame, are recalculated with

$$\boldsymbol{\mu}_{v,i} = \boldsymbol{A}_{vr,i}\boldsymbol{\mu}_{r,i} + \boldsymbol{b}_{vr,i}, \tag{4.3.54}$$

where $\boldsymbol{\mu}_{v,i}$ is the means of the vertices of the i^{th} landmark and $\boldsymbol{A}_{vr,i}$ and $\boldsymbol{b}_{vr,i}$ are the transformation parameters of this landmark. The mean and covariance of the landmark can now again be used for the purposes of detecting and updating the landmark.

After the landmark means are recalculated, each observed landmark's unknown lines are updated, as discussed in Section 3.2.3. This update is performed with the updated mean of the robot pose, $\boldsymbol{\mu}_x$.

4.3.4 New Landmark

Each set of measurements that is not associated with an existing landmark in Section 4.3.2, is used to create a new landmark, explained in Section 3.2.1, using the mean of the robot pose, $\boldsymbol{\mu}_x$, as the estimated pose for landmark creation. These landmarks are added after the state space update is done for the observed landmarks. This results in a more accurate estimate of the pose and better linearisation.

We now consider a single new landmark, with vertices $\boldsymbol{v}$, where $\boldsymbol{v}_i = \begin{bmatrix} v_{x,i} & v_{y,i} \end{bmatrix}^T$. The new landmark is, however, evaluated for detectability before it is added to the map. This evaluation is done before the unknown lines are added at the ends the model, but the unknown probabilities, $\boldsymbol{u}$, of the fitted lines are calculated. A measure for the detectability of the landmark is obtained by evaluating the deviation in the angles of consecutive lines and the number of object boundaries of a model, where we assume a line is an object boundary if $u_i \leq 0.5$. The angle of each line is calculated,

$$\psi_i = \arctan\left(\frac{v_{y,i+1} - v_{y,i}}{v_{x,i+1} - v_{x,i}}\right), \tag{4.3.55}$$

then the absolute difference between each pair of consecutive lines is calculated,

$$\Delta\psi_i = |\psi_{i+1} - \psi_i| \tag{4.3.56}$$

and the sum of these angles is obtained,

$$\psi_{\text{sum}} = \sum_i \Delta\psi_i. \tag{4.3.57}$$

If ψ_{sum} is greater than a threshold, ψ_{thres}, and if the the model consists of at least two object boundaries, we assume that the landmark's detectability is good enough and it can be added to the map.

To add a landmark to the map, a reference point for the landmark is created and appended to the state space, $\boldsymbol{s}$. The reference point, $\boldsymbol{r}$, is chosen as the mean of the landmark vertices, which is calculated with

$$\boldsymbol{\mu}_r = \overline{\boldsymbol{\mu}}_v = \frac{1}{k}\sum_i \boldsymbol{v}_i, \tag{4.3.58}$$

where k is the number of vertices.

The unknown lines at the ends of the model are now added, as explained in Section 3.2.1, and $\boldsymbol{v}$ now refers to the complete model with unknown lines added. With the landmark creation, a conditional distribution over the landmark parameters is obtained, given the estimated robot pose,

$$p(\boldsymbol{v}|\boldsymbol{x} = \boldsymbol{\mu}_x) = \mathcal{N}\left(\boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v\right). \tag{4.3.59}$$

The transformation parameters, $\boldsymbol{A}_{vx}$ and $\boldsymbol{b}_{vx}$, as well as the information matrix, $\boldsymbol{K}_v = \boldsymbol{\Sigma}_v^{-1}$, are obtained with model creation, as discussed in Section 3.2.1. These parameters describe the relationship between $\boldsymbol{x}$ and $\boldsymbol{v}$, therefore the state space update is performed to represent the states as in Figure 4.4, decoupling $\boldsymbol{x}$ and $\boldsymbol{v}$. The parameters above thus define factor Φ_3 in Section 4.3.3.

The linear relationship between $\boldsymbol{v}$ and $\boldsymbol{r}$ is given by

$$\boldsymbol{v} = \boldsymbol{A}_{vr}\boldsymbol{r} + \boldsymbol{b}_{vr} + \boldsymbol{n}, \quad \boldsymbol{n} \sim \mathcal{C}(\boldsymbol{K}_v, \boldsymbol{0}), \tag{4.3.60}$$

where

$$\begin{aligned}
\boldsymbol{K}_v &= \boldsymbol{\Sigma}_v^{-1} \\
\boldsymbol{A}_{vr} &= \begin{bmatrix} \boldsymbol{I}_{[2\times2]} & \cdots & \boldsymbol{I}_{[2\times2]} \end{bmatrix}^T \\
\boldsymbol{b}_{vr} &= \boldsymbol{\mu}_v - \boldsymbol{A}_{vr}\boldsymbol{\mu}_r,
\end{aligned} \tag{4.3.61}$$

which define the factor Φ_2 in Section 4.3.3. Since $\boldsymbol{r}$ is an artificial point, which is placed at the mean of the vertices, we choose the transformation matrix, $\boldsymbol{A}_{vr}$, so that each vertex, $\boldsymbol{v}_i$, is just an offset from the reference.

Since the reference point is created with regards to the model parameters, there is no prior information about where the reference is in the global frame. The parameters associated with $\boldsymbol{r}$ in Φ_1 are thus all populated with zeros,

$$\begin{aligned}
\boldsymbol{K}_{rr} &= \mathbf{0}_{[2\times2]} \\
\boldsymbol{K}_{rx} = \boldsymbol{K}_{xr}^T &= \mathbf{0}_{[2\times3]} \\
\boldsymbol{K}_{rm} = \boldsymbol{K}_{mr}^T &= \mathbf{0}_{[2\times2n]} \\
\boldsymbol{h}_r &= \mathbf{0}_{[2]},
\end{aligned} \tag{4.3.62}$$

where n is the number of existing landmarks. These parameters for Φ_1, Φ_2 and Φ_3 are now used in Section 4.3.3 to update the complete state space for each new landmark and obtain the state space representation as in Figure 4.4.

When the robot is placed in an unknown environment and starts to observe it, the robot creates models of new landmarks and adds them to its map of the environment. As the robot moves through the environment, its pose estimate is first updated with the motion update and subsequently the measurement update is performed, associating measurements to landmarks, updating the belief of the robot's pose and its environment and adding new landmarks to the map. These updates are typically performed at timesteps when measurements are received and odometry information is available and are repeated for each such timestep.

In this chapter the development of the SLAM algorithm was discussed. This SLAM algorithm is a crucial part of this project, since the aim is to identify landmarks for SLAM. A novel measurement update was therefore created to incorporate the landmark modelling method of Chapter 3.

Chapter 5

Results

The SLAM algorithm discussed in Chapter 4 is implemented using the landmark modelling discussed in Chapter 3. The algorithms are implemented in Python, due to the simplicity of this high-level programming language. The use of the numpy library also simplifies the implementation of matrix operations significantly. Python is, however, much slower than languages such as C++, but the tests are more focussed on the precision of these methods, rather than the execution time. In this chapter we analyse this method by executing it in designed simulated environments as well as two real-world datasets.

5.1 Simulations

We create multiple environments in which the simulated robot moves around to show different aspects of the designed algorithms. In all the simulations we approximate the robot as a single point, with the Lidar sensor at the same position.

In these simulations we simulate a typical SICK LMS Lidar sensor with a field of view of $\phi_{\text{FOV}} = 180°$, an angular resolution of $\phi_{\text{res}} = 1°$ and maximum range of $r_{\max} = 30\,\text{m}$. The standard deviation of the noise of the range and angle measurements are $\sigma_r = 0.03\,\text{m}$ and $\sigma_\phi = 0.2°$ respectively. The motion noise differs for each simulation.

Single Landmark

In the first simulation, a single object is placed in the environment. The robot moves around the landmark and observes new parts of the landmark. In each step, the measurements are aligned to the landmark model and then associated with the landmark. Once the measurements are associated with the landmark, they are used to update the landmark model and the robot pose.

In Figure 5.1(a), it is shown that the algorithm is able to align the measurements and associate them with the landmark when only part of the landmark is observed and new measurements lies behind an unknown line. As the robot moves around the landmark, the new vertices that are added to the model have higher uncertainty over them, which is clear in (b) and (c). This is due to the robot's uncertainty

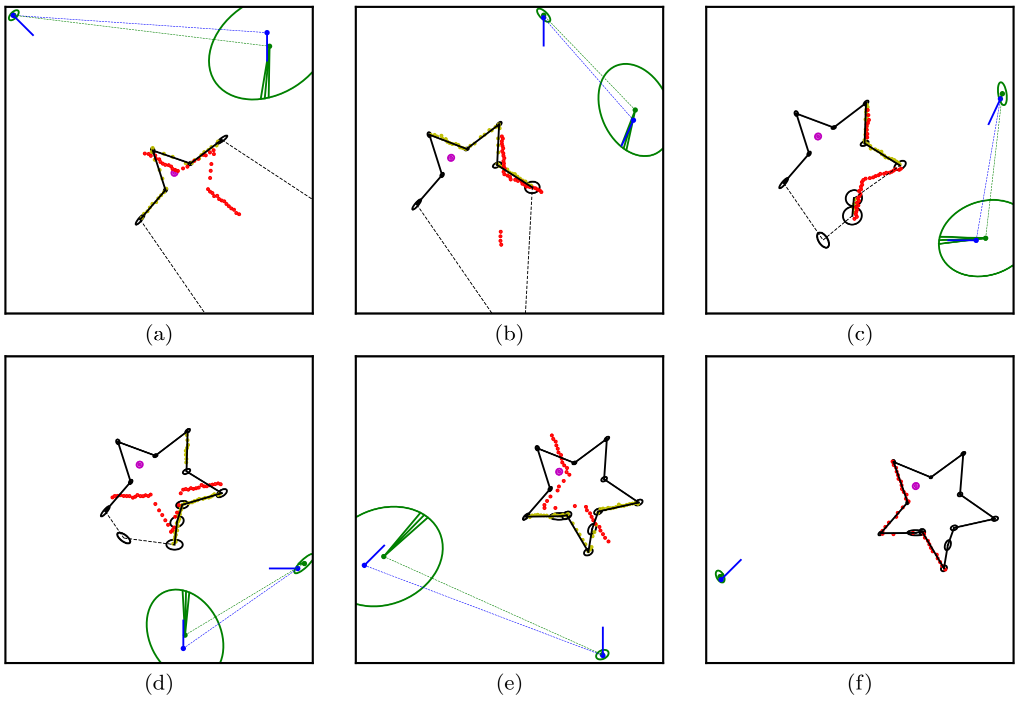


Figure 5.1 – Results with a single landmark. Each figure shows the belief of the robot before the motion update with the smaller green ellipse, as well as its true pose in blue. The corresponding measurements of the previous step is shown in yellow. The belief of the robot after the motion update, prior to the measurement update, is shown with the bigger green ellipse, along with the new true robot pose in blue. The measurements corresponding to the mean of this robot pose are shown in red and the landmark model is shown in black. The final figure (f) only shows the robot pose, measurements and updated landmark model after the final measurement update.

increasing and subsequently the uncertainty over the new observed vertices increase as well.

Once the initial vertices are observed again, together with the last added vertices, as in Figure 5.1(e), the uncertainty over the last vertices decreases significantly due to the new correlation with the initial vertices, which can be seen in (f). The uncertainty of the unobserved vertices also decrease after this measurement update, due to its correlation with the observed vertices. This is the effect of loop closure in SLAM, which is shown here for the parameters of a single landmark.

It is also shown that as the landmark model gets updated, its unknown lines are updated as well. Once the unknown lines from the two sides intersect, it is terminated at that intersection point, which is clear in Figure 5.1(c). When the robot completes a full loop around the landmark, the algorithm has the ability to close the landmark and connect the first and last vertices of the landmark, as in (e) and (f).

Note that the algorithm is unable to join two line segments that can possibly be represented by a single line. In (c) the measurements could indicate that the bottom observed line should be extended, but another line segment is created instead. These two line segments, which can be seen in (d), describe different sections of the same underlying line, but are modelled as two lines.

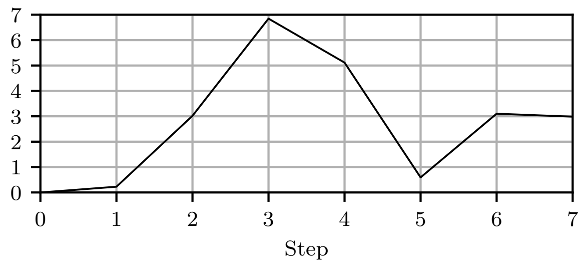


Figure 5.2 – Mahalanobis distances of posterior poses around the single landmark. Each pose is evaluated against the true pose after the measurement update is performed.

The Mahalanobis distance, given by Equation 3.2.30 and discussed in Section 3.2.1, is used to evaluate the pose estimates during the simulation. The Mahalanobis distance is a normalised distance between a vector and a Gaussian distribution over the vector space. This distance is a measure of the number of standard deviations the vector is away from the mean of the distribution. The square of the Mahalanobis distance is expected to be chi-squared distributed with a mean of 3 for the 3-dimensional pose.

The Mahalanobis distances between the posterior pose distributions and the true poses are shown in Figure 5.2, where the poses 1-6 relate to the posteriors in (a)-(f) in Figure 5.2. It is seen here that as the robot moves around the object, the robot becomes overconfident in its pose estimate and the Mahalanobis distance is very high in steps 3 and 4. This overconfidence could be caused by overconfidence of the vertex measurements or effects of linearisation. When the initial vertices are observed again, the Mahalanobis distance improves significantly. This is due to errors in the calculation of the pose estimate as the robot moves away from its initial pose, but as it observes some of the initial parts of the landmark again, some of these error are corrected.

In Figure 5.3, the final model is shown against the real object. The model is not perfect, but the 3-σ confidence ellipses capture the corners of the actual model in most cases. The vertices modelled when the Mahalanobis distance of the pose is high are less accurate. The model created depends heavily on the measurements it obtained and the robot's uncertainty at that time.

Small Environment

The next simulation has four objects placed in the environment. The robot observes these objects at different timesteps at first and then observes the first landmark again to perform loop closure.

In the first three steps of Figure 5.4, the robot only observes new landmarks and no landmarks already modelled. Thus, the uncertainty over the robot's pose does not decrease when observing these new landmarks and subsequently the uncertainties

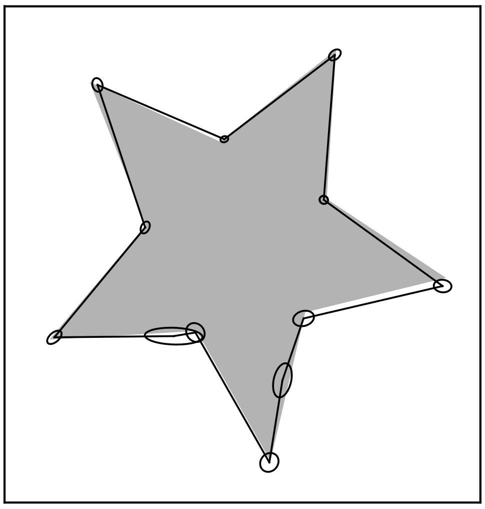


Figure 5.3 – Final model of a single landmark. The real object is shown in grey and the landmark vertices after the last measurement update are shown in black.

over the landmark references in (b) and (c) are high. The correlation between the two landmarks observed in the third step is higher than the rest, since they are observed together. This is, however, not visible in the figure.

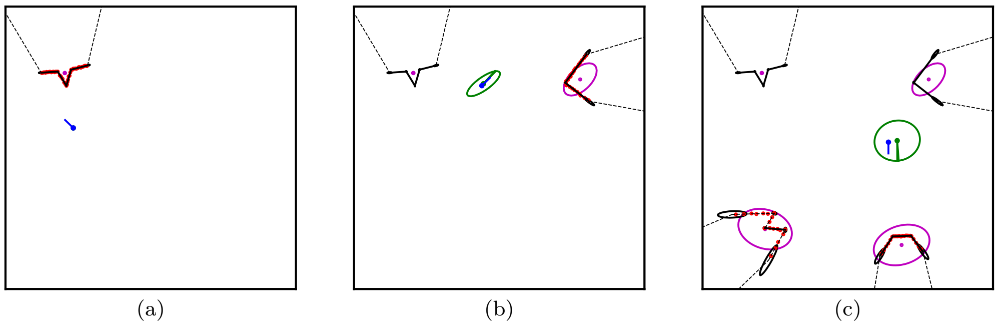


Figure 5.4 – First three steps in a small environment. The belief over the robot pose (green), along with the landmark references (purple) and landmark models (black) are shown for each step. The measurements (red) and actual pose (blue) are also shown.

In the fourth step, seen in Figure 5.5, the robot observes the first landmark again, along with the landmark added last. This causes loop closure, where all the landmark reference estimates are improved significantly and consequently the landmark models are better aligned with the actual environment.

Curved Landmarks

The previous simulations only consisted of objects constructed out of straight lines. In the next simulation, we use some landmarks with curved edges to show how

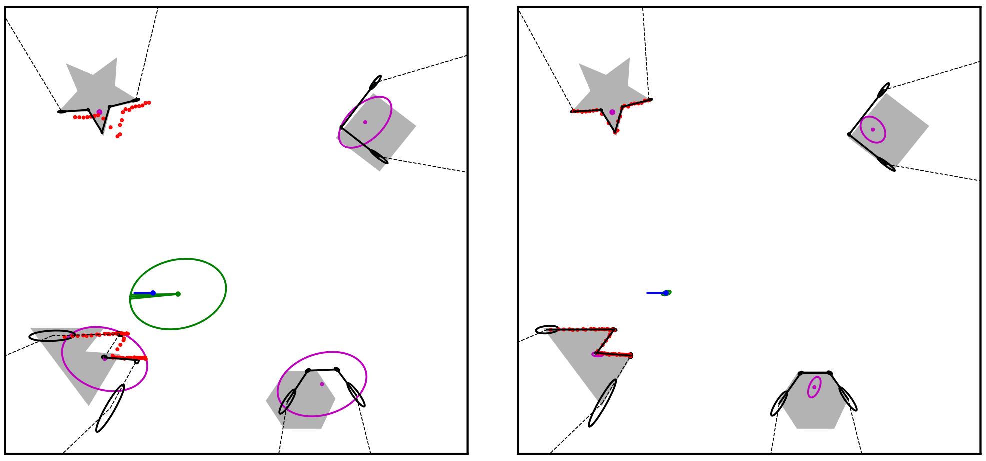


Figure 5.5 – Loop closure in a small environment. This is the step following that of Figure 5.4. The first figure shows the beliefs prior to the measurement update and the second figure shows the updated beliefs. The environment objects are shown in grey.

the algorithm handles these landmarks. The robot takes a larger loop through this environment and takes more steps.

In Figure 5.6 the simulated environment is shown, as well as the true robot poses through the environment and the poses estimated only using the odometry sensors. The completed simulation is shown, with the landmarks created and the robot pose estimates at each timestep. It is seen that the models of the curved landmarks do resemble the objects sufficiently well: although these straight-line approximations seem crude, the measurements of these landmarks are associated with them and the landmarks are updated with measurements from different perspectives.

Another observation is that the landmark at the bottom left of the map is modelled by two landmarks. This is caused by measurements that are not associated with the landmark, but used instead to model a new landmark. The missed association can be caused by the specific viewpoint of the robot or failure to align the measurements with the landmark. Although later information can suggest that these to landmarks model the same object, the algorithm is unable to join these landmarks into one.

The robot poses estimated by the SLAM algorithm are very close to the real robot poses. The confidence ellipses of these poses are, however, barely visible, which means that the robot is very certain about its pose and this could be problematic. Although the mean of the pose belief is very close to the actual pose, the robot is still more certain than it should be in some timesteps. This is more easily seen when evaluating the Mahalonobis distances as shown in Figure 5.7. The Mahalanobis distance is very high for many poses and although the mean of the final pose belief is very close to the true pose, the Mahalanobis distance for this pose is extremely high. This overconfidence of the pose could be a result of approximations made in our landmark modelling and updating approach. Another possibility is that it is due to the EKF motion update and other linearisation applied in the SLAM algorithm.

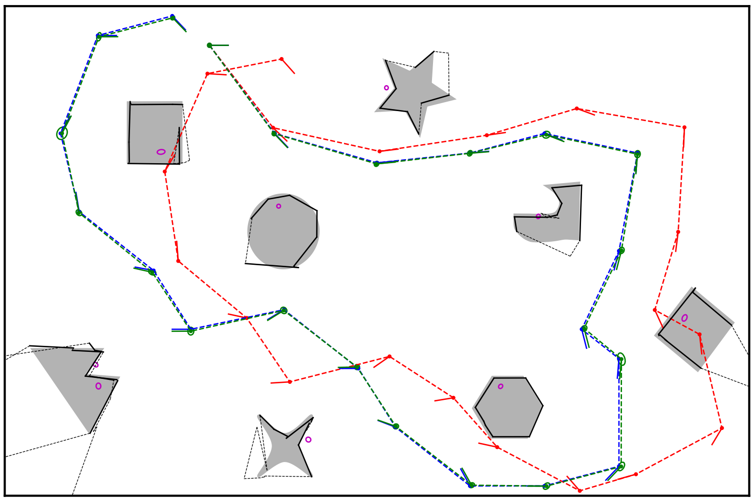


Figure 5.6 – Environment with curved landmarks. The true poses of the robot (blue) are shown along with the poses only estimated using the odometry (red). The resulting landmarks (black), after the entire simulation is completed, are shown along with the 3-σ confidence ellipses of the references (purple). The estimated pose after the measurement update at each timestep is also shown (green) with its 3-σ confidence ellipses, as well as the true pose (blue).

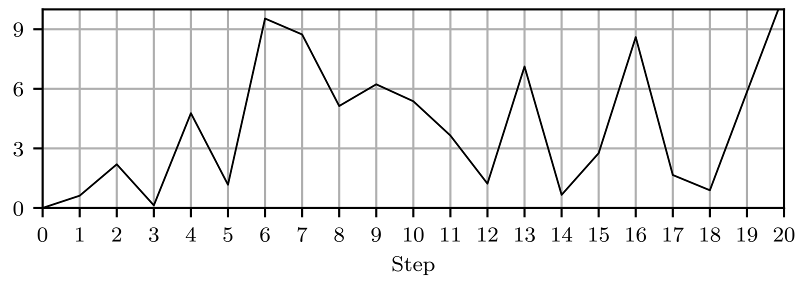


Figure 5.7 – Mahalanobis distances of posterior poses in the curved landmark environment. Each pose is evaluated against the true pose after the measurement update is performed.

The timesteps during this simulation took an average execution time of 1.8 s to perform the motion and measurement update. Although this execution time is high, the implementation is done in Python, which is a high-level language that does not get compiled. Implementing these algorithm in a faster programming language such as C++, would result in much faster execution times.

Realistic Environment

In the final simulation, we aim to create a more realistic environment by simulating a house plan with objects in it. The map of this environment is shown in Figure 5.8.

The scale of this environment is larger than the previous simulations and the robot attempts to make a larger loop within the environment. The robot also takes 511 steps in this environment, where the previous simulation only has 21 steps.

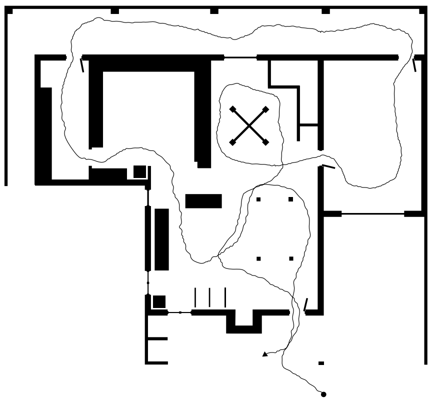


Figure 5.8 – Map of simulated house with true robot path.

The objects in this simulation differ in size and the walls have long straight sections, which are not in the other simulations. All the objects in this environment exist of straight lines with a lot of 90° angles between lines, but some of the objects have very small cornered segments, that could be missed by Lidar measurements. Associating measurements to the correct landmarks is thus more challenging in this environment.

In Figure 5.9, the robot path estimated by the odometry sensors is shown, as well as the path estimated by the SLAM algorithm. The measurements taken by the robot at all timesteps are also shown, as seen from the true robot pose. From this figure, it is clear that the SLAM algorithm corrects the path significantly from odometry the estimate, but the updated SLAM estimate contains some drift that is not corrected by the end of the simulation.

Some of the timesteps, after the measurement update, are shown in Figure 5.10. During this simulation 79 landmarks are created. In (a) it is seen that the robot maintains a good estimate of its pose and the landmarks created is a sensible model of the environment. When the robot moves around to (b), however, the robot's pose estimate is offset from the true pose. This drift can also be seen in the top line of the map, which should be straight, but starts to curve downwards. In (c) the robot corrects its position estimate, but its orientation is off. This can be seen when looking at the measurements, which are not aligned with landmarks. The cause of this offset in orientation is the three vertical landmarks at the bottom of the map, where the robot associates measurements from the leftmost landmark to the landmark in the centre. The faulty associations cause the robot to become overconfident about its pose and cannot be corrected in this simulation. In (d) it is shown that the pose estimate at the end of the simulation is significantly off and new landmarks are created with measurements that actually come from landmarks already modelled.

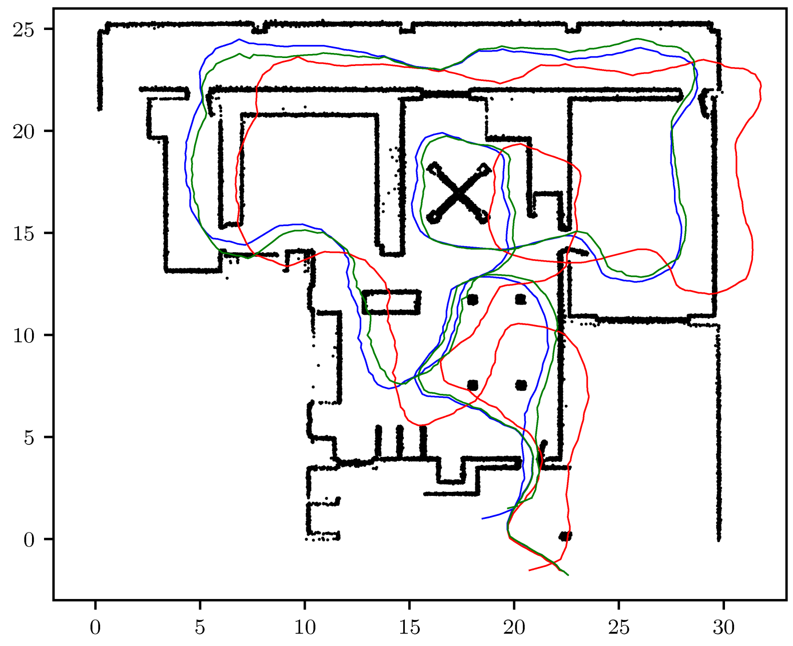


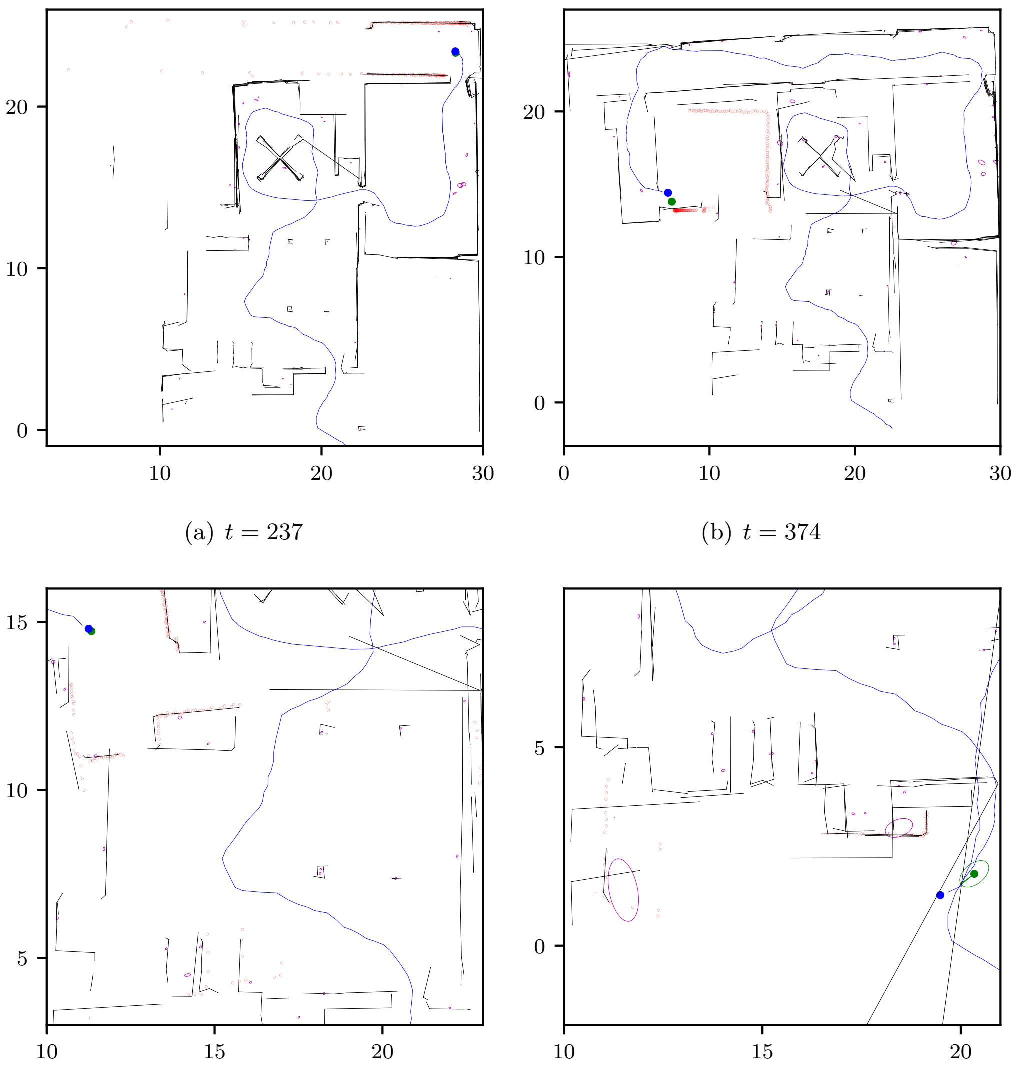
Figure 5.9 – Estimated robot paths through simulated house. The true path of the robot through the environment is shown (blue), along with the path estimated only with the odometry sensors (red) and the path estimated by the SLAM algorithm (green). The measurements, as seen from the true path, are also shown (black) and a scale of the axis is given in metres.

Although the high frequency of measurements, with small motion steps between measurements, should give the robot a more accurate estimate of its pose, the uncertainties over the landmark models, and subsequently the pose, become very small. These small uncertainties result in the robot becoming overconfident and not being able to close the loop. The association errors during the simulation also introduces false information, which causes the landmark and pose estimates to become confident of the wrong values.

This concludes the test of the algorithms in simulation. In these simulations most of the effects modelled are ideal. In the next section we will test the algorithms in practical datasets with real Lidar and odometry measurements, which introduces other difficulties.

5.2 Practical Datasets

During the simulations, many of the effects being modelled are ideal and known. Firstly, the noise simulated is Gaussian distributed and the parameters of this Gaussian distribution are known. This is not necessarily the case for real sensors, but the Gaussian noise assumption is commonly made. The parameters of the noise, however, need to be estimated. These parameters can often be found in the datasheets of sensors, if available, or can be estimated by a statistical evaluation.



(a) $t = 237$ (b) $t = 374$

(c) $t = 390$ (d) $t = 506$

Figure 5.10 – Different steps during the house simulation. All the modelled landmarks (black) are shown for the each timestep, along with the robot pose belief (green), true pose (blue) and the measurements (red). The true path history is also shown for each timestep. Note that the scales, given in metres on the axis, of the four figures differ from each other.

Another assumption made is that the odometry information and Lidar measurements are obtained at the same time. With real sensors, however, the odometry information and Lidar measurements are often out of sync and obtained at different frequencies, but both typically have timestamps associated with them. This problem can be overcome by extrapolation of the odometry measurements at the time when the Lidar measurements are obtained. We do this by assuming the robot motion is constant between two odometry measurements and using a fraction of the latest odometry reading, according to the fraction of time when the Lidar measurements are received.

The simulated environment is also assumed to be flat and the measurements are assumed to be taken at the same height. This is usually the case for indoor environments, but outdoor environments often contain uneven terrain, which causes the robot to pitch and roll. The true path and poses of the robot in real data are also not always available. In the simulations, it is assumed that the Lidar sensor and the centre of the robot's odometry is at the same location, which is usually not the case with real robot's. Therefore a transformation between the odometry pose and Lidar pose needs to be performed between the motion and measurement updates.

Taking these non-ideal effects into account, we now test our landmarking method on two datasets, and indoor dataset set taken at the Università di Milano-Bicocca in Milan, Italy, and an outdoor dataset taken at Victoria Park in Sydney, Australia.

5.2.1 The Bicocca Dataset

The Bicocca dataset [24; 25] is an indoor multi-sensory dataset taken in two buildings connected with bridges. Features in the environment include hallways, chairs, tables and a library. The terrain is smooth for the largest part of this environment, with the exception being the two sloped bridges that connect the buildings.

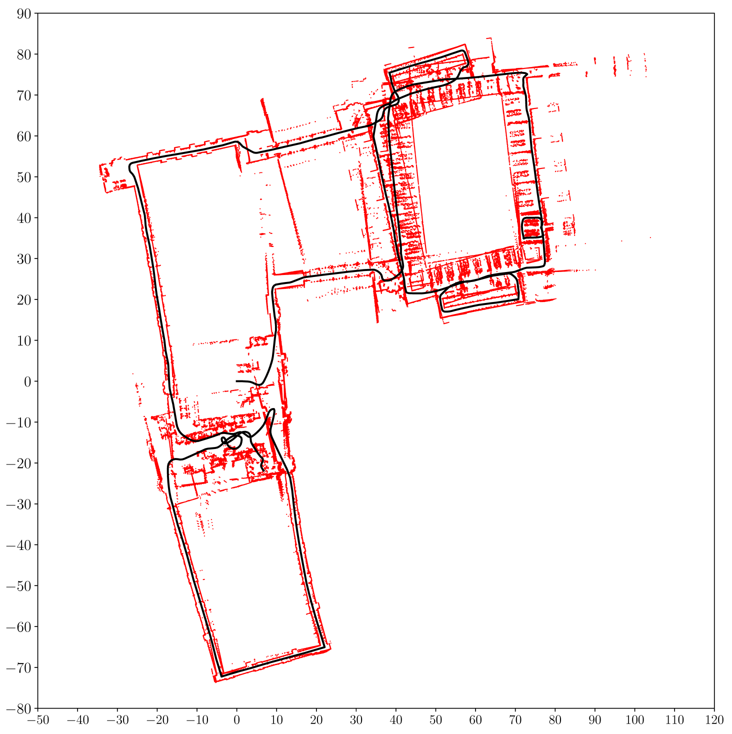


Figure 5.11 – Estimated robot path (black) and measurements (red) in the Biccoca dataset prior to any measurement updates. The scale of this environment is given in metres on the axes.

Multiple datasets have been taken in this environment of which we only use the "`Bicocca_2009-02-25b`" dataset. The robot is equipped with multiple sensors, such as cameras, Lidar, sonar, IMU and odometry sensors. We, however, only make use of the odometry sensor and the front-facing SICK LMS Lidar sensor. The odometry information is obtained at a 50 Hz frequency and the Lidar measurements at 75 Hz. The Lidar sensor also has a FOV of $\phi_{\text{FOV}} = 180°$, an angular resolution of $\phi_{\text{res}} = 1°$ and a maximum range of $r_{\max} = 80\,\text{m}$. Since measurements that are very far away, at this angular resolution, have little information about landmarks, we assume the maximum range is $r_{\max} = 30\,\text{m}$ and ignore any measurement farther than this range.

In Figure 5.11, all the measurements obtained from the dataset as well as the odometry estimate of the robot's path are shown. The measurements in this dataset are taken at a high frequency of 75 Hz, but we run the tests using subsampled measurements. The subsampled frequencies of 1 Hz and 5 Hz are chosen for the tests. Although this subsampling discards useful information, the multitude of measurements from the 75 Hz measurement rate can cause the robot to become very overconfident.

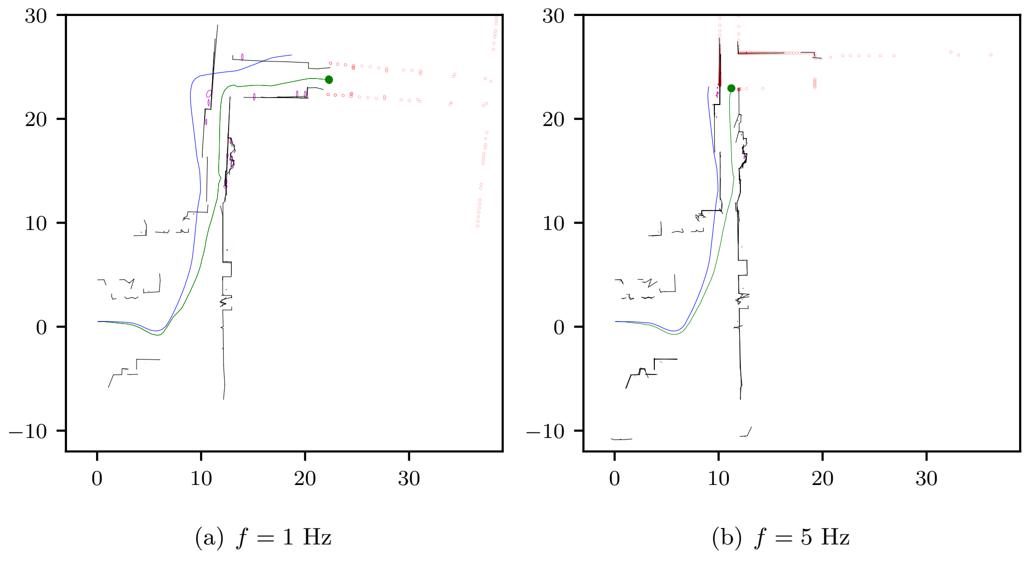


Figure 5.12 – Results with the Bicocca dataset using different scan frequencies. The landmarks created (black), as well as the SLAM estimate of the robot path (green) are shown. The odometry estimate of the robot path is also shown (blue) as well as the latest measurements (red). The scales and locations of these figures are also given in metres on the axes.

In Figure 5.12, the results of the start of the Bicocca dataset is shown for the 1 Hz and 5 Hz test. In the 1 Hz test, the robot creates a decent model of the environment, but it drifts as it moves along. When the robot reaches the bridge section, which is located where the robot stops in Figure 5.12(a), it struggles to find good landmarks due to a lot of pillar-like structures, which causes it to drift more and make faulty associations. For the 5 Hz test, the model of the map seems to be more accurate than the 1 Hz test case. Due to faulty associations around the corner, however, the robot's estimate becomes completely wrong.

Although the implementation of our method failed to complete a full loop through the entire dataset, the initial results show that the model describes the environment well. If a more robust method to associate landmarks is designed, our modelling approach could be useful in an environment like this.

5.2.2 The Victoria Park Dataset

The Victoria Park dataset [12] is an outdoor dataset taken in a park filled with trees and bushes. The terrain in this environment is uneven and could cause variation in measurements taken at different heights on objects and measurements hitting the ground.

The robot is equipped with an odometry sensor and a *SICK LMS* Lidar sensor. This Lidar operates at an angular resolution of $\phi_{\text{res}} = 0.5°$ and a scanning frequency of 50 Hz. The data in this dataset is already in a synchronised format, where the odometry information describes the motion between Lidar scans. The other measurement parameters are the same as for the Bicocca dataset.

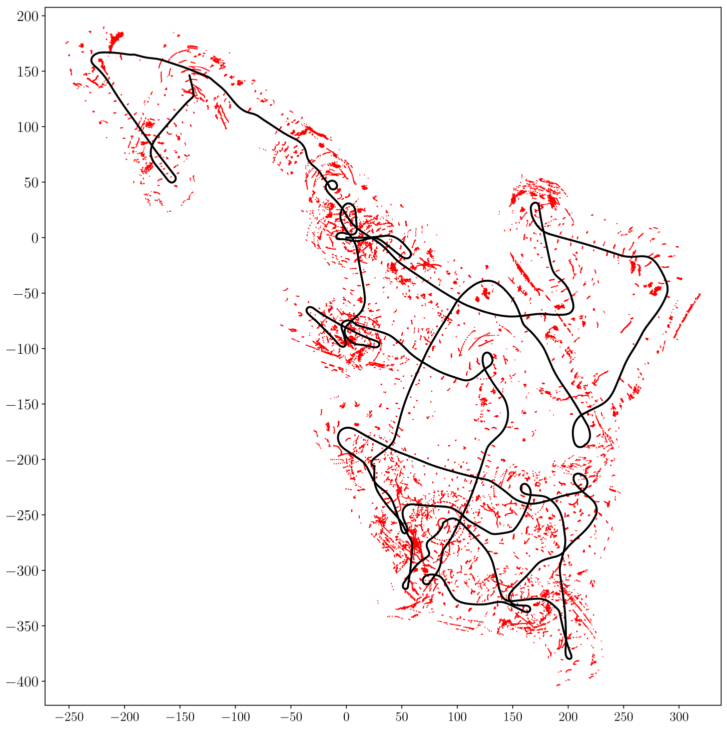
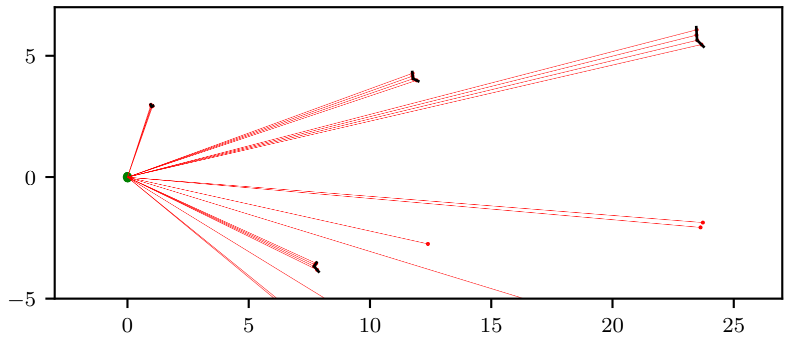
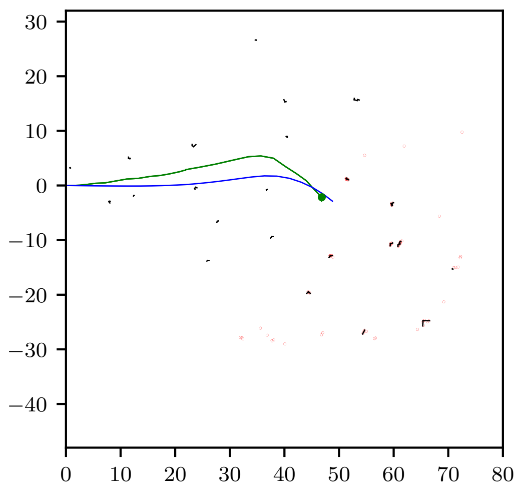


Figure 5.13 – Estimated robot path (black) and measurements (red) in the Victoria Park dataset prior to any measurement updates. The scale of this environment is given in metres on the axes.

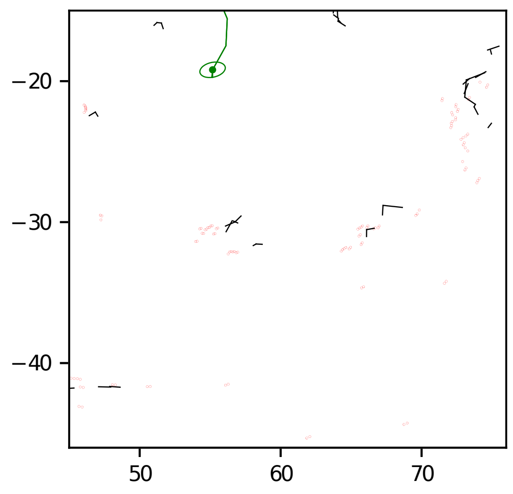
In Figure 5.13, all the measurements obtained from the dataset as well as the odometry estimate of the robot's path are shown. From this figure, it is seen that the environment is a lot less dense than in the Bicocca dataset, which is expected for an outdoor environment. The objects in this environment are often far from the robot and relatively small, which means that a low number of measurements are obtained from each object. Therefore, to be able to model landmarks, the number of measurements needed for an informative set of landmarks in the segmentation process is adjusted for this test to $n_{\text{thres}} = 4$.



(a) $t = 0$



(b) $t = 50$



(c) $t = 58$

Figure 5.14 – Results with the Victoria Park dataset at different timesteps. The landmarks created (black) as well as the SLAM estimate of the robot path (green) are shown. The odometry estimate of the robot path is also shown (blue) as well as the latest measurements (red). In (a) the measurement beams are shown to visualise the amount of measurement of each landmark. The scales and locations of these figures are also given in metres on the axes.

Figure 5.14 shows results of different steps in the Victoria Park dataset. In (a), which is the start of the dataset, it can be seen that the robot receives a maximum of 4 measurements from any single landmark. The robot moves along to timestep 50, which is shown in (b), and identifies a number of landmark in this environment, while maintaining a good estimate of its pose. At timestep 58, however, the difference in angle between the real robot and its mean is big, causing the landmarks and measurements to be misaligned, as seen in (c). Since the landmarks in this environment are not very descriptive, these measurements are not associated with their correct landmarks, which causes the robot to drift further.

These results show that in its current stage, our modelling method is not suitable for outdoor environments. With some adaptations, such as using multiple measurement sets to align data or to group multiple objects as a landmark, the method could be improved for outdoor applications.

Some promising results are shown in this chapter, but in larger environments, the robustness of the methods has to be improved significantly to make it a viable solution in practical applications.

Chapter 6

Conclusions

The main focus of this project was to develop a method to model and identify landmarks from 2D Lidar measurements to be used in SLAM. In this chapter we give an overview of the development and draw some conclusions from the results.

We first did a study of existing SLAM algorithms and approaches to Lidar-based SLAM from the literature in Chapter 2. From this study we identified possible gaps in current research as well as methods that are useful for our approach.

In our approach, we chose to use objects as landmarks and model these objects by a set of straight lines. Although this approximation seems crude, the probabilistic approach used to model these lines is expected to allow a sensible model of curved segments as well. The results show that curved objects can indeed be modelled, associated and updated with this approach. This approximation could, however, be a cause of the overconfidence of the robot seen in the results.

The probabilistic landmark model allowed us to derive a likelihood function of obtaining measurements from the landmark. This likelihood function was then used to compare different models with the BIC and select the model that represents the measurements with the highest likelihood. Since the computation of all possible models is intractable, a method was developed to iteratively reduce the number of lines in between models and only compare these models. Although this method does not guarantee the selection of the best model, it proved to still result in a good model describing the measurements. From the observation that the models created of landmarks often have a lot fewer vertices than the measurements they are modelled from, a method was developed to initialise the model selection algorithm with a lower number of lines fitted to the data, which speeds up the model selection process.

Once a method was developed to create a model from the measurements, the model was adapted to describe occlusion boundaries in the environment. These lines were added at the ends of a model after model selection and a likelihood function was derived for these lines. These lines proved useful when measuring parts of an object not seen before, both to associate these measurements to the correct landmark and to update the landmark in these previously occluded areas.

The next step was to develop a method to decide whether a set of measurements comes from a landmark. Although the likelihood function was already derived, it is not a good method to make an association decision. Instead, we calculated the

Mahalanobis distances of the error between the measurements and expected measurements and evaluated this against a threshold. This method, however, assumed a known pose and thus requires the measurements to be aligned to the landmark. Therefore, the ICP algorithm was adapted to align the measurements to the landmark model. This method could, however, be sensitive to the initial condition and converge to a local minimum instead of the global minimum, which could cause incorrect associations.

A method was also developed to update a landmark model when new measurements of the landmark are obtained. The method estimates the vertices of the model from the new measurements, using measurement-to-line correspondences from the existing model. Using the existing model and the new observed model, the method was able to update the distribution over the model parameters as well as extend the model with new observed sections.

After the development related to the modelling was completed, a SLAM algorithm was developed that uses this landmark modelling to create a map of the environment. The motion update of the EKF SLAM algorithm was used with an odometry motion model. However, a new measurement model was derived which uses the observed model parameters to simultaneously update the model parameters and robot pose. To reduce complexity, some approximations have been made in the derivation, removing direct correlations between the robot pose and the model parameters.

Finally, our method is tested with simulations as well as real-world datasets. The first simulations showed that good landmarks of the environment could be created and the landmarks could be associated and updated. Some correspondences were missed, but it did not prove to be critical. The modelling proved to work for objects with both straight and curved segments. These simulations, however, also showed that the robot tends to become overconfident about its pose. The final simulation aimed to be more realistic with the robot making a larger loop through the environment. Due to the overconfidence in the robot's pose and faulty correspondences, the robot failed to close the loop and created new landmarks of objects that were previously modelled, but at different locations.

In the tests performed on the real-world datasets, the robot failed to complete full loops through the entire environments. In the Bicocca dataset, the robot initially created a good representation of the environment, but drifted due to overconfidence and faulty correspondences. The Victoria Park dataset proved to be more difficult, with few measurements received from any single object, making it difficult to create models and make associations. Other environmental effects, such as the uneven terrain, also created challenges in this dataset. Despite these challenges, the robot was able to create landmarks and keep a good estimate of its path during the start of the test. Once the robot's uncertainty became too large, however, it failed to associate measurements to the correct landmarks due to the low recognisability of the landmarks.

The developed landmark modelling method seems to be a good proposal for modelling landmarks from Lidar measurements. In its current form, however, it is not robust enough to be implemented in practice.

6.1 Contributions

In this section we discuss the original contributions made in this project. This is mainly focussed on the landmark modelling method and the new measurement update we developed.

The landmark modelling method we developed describes the shape of objects in the environment. This method create probabilistic, parametric models of these shapes were developed, which differs from similar approaches that created a point cloud model of the objects. The probabilistic, parametric models of landmarks allowed us to develop a method to update the models of landmarks when new segments of landmarks are observed. Other existing methods to model landmarks from Lidar measurements are unable to update the landmark models when new segments of landmarks are observed. This ability to update the landmark models are very useful to maintain a good description of the environment as the robot moves through the environment.

Secondly, we developed a novel measurement update for the SLAM algorithm which incorporates the probabilistic, parametric landmark models. Other approaches to landmark-based SLAM with Lidar measurements only uses classical SLAM algorithms, such as EKF SLAM, and only estimate the locations of landmarks in the environment. Our measurement update, however, is developed to simultaneously update the distributions over the robot states, landmark references and landmark models, which was not previously possible.

These two contributions allow a robot to maintain a better description of a new environment it observes. Further possible improvements to these models are discussed in the following section.

6.2 Future Work

In this section we discuss some possible improvements and adaptations which could be made to the methods described throughout this thesis. From evaluating the results, it is clear that the main problems in the tests arise from overconfidence in the robot pose and faulty correspondences. The suggested improvements are mostly focussed to address these issues.

A possible improvement to the current model is to introduce a “coarseness” to the straight lines. This proposed coarseness aims to capture the variation in the distance of measurements from the lines, creating more accurate models for curved models. The coarseness could also be used to increase the uncertainty of the observed vertex parameters, which could improve the issue of overconfidence.

Another suggestion is to look into using other features from Lidar measurements, such as the the curvature keypoints used by Bosse and Zlot [16], to obtain a lower bound for the model selection algorithm. This could possibly improve the consistency of models, which can lead to more robust correspondences. Pose-invariant information between data, similar to the method used in Himstedt *et al.* [17], could be useful for making correspondences or to get an initial alignment between measurements and landmarks.

To make correspondences more robust, a post-measurement update verification could be developed to verify the correspondences between measurements and landmarks. If correspondences are not verified in this step, the measurement update could be reversed and executed again with only the verified correspondences.

A recommendation which could be useful in datasets such as Victoria Park, where a low number of measurements are obtained from a single object, is to group different objects together as a landmark. This could make correspondences to these landmarks more robust. This adaptation will, however, need significant changes from the current design. An alternative to this is to use multiple measurement sets and landmarks in the alignment step, which could also make correspondences more robust.

Other suggestions to improve the current model is to be able to join two line segments if they seem to be a straight line and to combine different landmarks that model the same object. Single-line landmarks could also be included to give information about the orientation and position normal to the line, but not about its position parallel to the line.

Implementing these recommendations could make the proposed modelling method more robust and useful in a larger variety of environments.

Appendix A

Line Fit Derivation

In this appendix we provide the derivation of the equation to fit a line parameterised by two points, $\boldsymbol{p}_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$ and $\boldsymbol{p}_1 = \begin{bmatrix} x_1 & y_1 \end{bmatrix}^T$, to a set of data points, $\mathcal{Z} = \{\boldsymbol{z}_{c,0}, \cdots, \boldsymbol{z}_{c,n-1}\}$, where $\boldsymbol{z}_{c,i} = \begin{bmatrix} z_{x,i} & z_{y,i} \end{bmatrix}^T$, by minimising the perpendicular square error between the data and the line. The equation of the line is

$$\boldsymbol{p}_\alpha = (1-\alpha)\boldsymbol{p}_0 + \alpha\boldsymbol{p}_1 \tag{A.0.1}$$

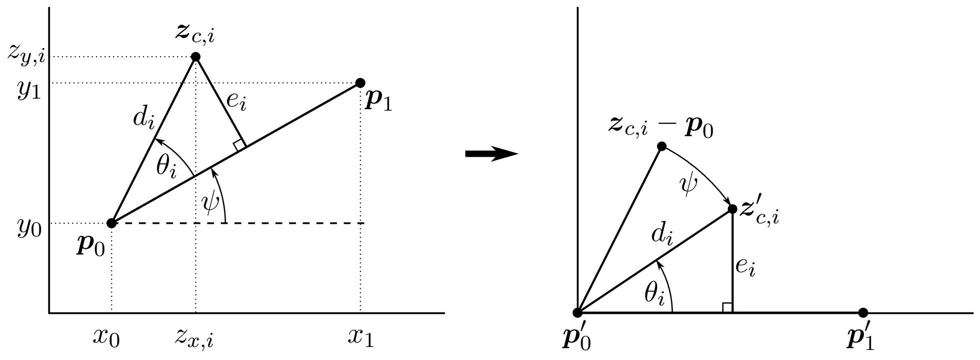


Figure A.1 – Diagram of the error between a single measurement and a line. A coordinate system transformation is done to obtain the second diagram with the line on the x-axis.

For a single data point, visualised in Figure A.1, the perpendicular error between the line and the point can be written as

$$e_i = d_i \sin\theta_i, \tag{A.0.2}$$

but this requires the calculation of d_i and θ_i. By transforming the coordinate frame so that the line lies on the x-axis and $\boldsymbol{p}_0$ is the origin, results in a simpler form of this equation,

$$e_i = (z_{x,i} - x_0)\sin\psi - (z_{y,i} - y_0)\cos\psi, \tag{A.0.3}$$

where ψ is the angle of the line. For the first part of this derivation we assume the line is parameterised by $\boldsymbol{p}_0$ and ψ and then we calculate $\boldsymbol{p}_1$ according to these parameters.

It is noted that with this parameterisation, an infinite number of parameters define the same line, we therefore choose $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ to be as simple as possible.

For the best line fitted to this data, we want to minimise the sum over the error of all the points,

$$\begin{aligned} E &= \sum_i e_i^2 \\ &= \sin^2\psi \sum_i (z_{x,i} - x_0)^2 - \sin\psi\cos\psi \sum_i (z_{x,i} - x_0)(z_{y,i} - y_0) + \cos^2\psi \sum_i (z_{y,i} - y_0)^2 \\ &= \sum_i \left((z_{x,i} - x_0)\sin\psi - (z_{y,i} - y_0)\cos\psi\right)^2 . \end{aligned} \tag{A.0.4}$$

To obtain the minimum, the derivatives of this function with respect to the different parameters is calculated and set to 0. First the function is derived with respect to $\boldsymbol{p}_0$,

$$\begin{aligned} \frac{\partial E}{\partial x_0} = -2\sum_i \left((z_{x,i} - x_0)\sin\psi - (z_{y,i} - y_0)\cos\psi\right)\sin\psi = 0 \\ (\overline{z}_x - x_0)\sin\psi = (\overline{z}_y - y_0)\cos\psi \\ \frac{\partial E}{\partial y_0} = -2\sum_i \left((z_{x,i} - x_0)\sin\psi - (z_{y,i} - y_0)\cos\psi\right)\cos\psi = 0 \\ (\overline{z}_x - x_0)\sin\psi = (\overline{z}_y - y_0)\cos\psi, \end{aligned} \tag{A.0.5}$$

where $\overline{z}_x$ and $\overline{z}_y$ is the mean of the data. Both of these derivatives result in the same equations and from these it can be seen that setting $\boldsymbol{p}_0$ equal to the mean of the data satisfy these equations.

$$\begin{aligned} x_0 &= \overline{z}_x \\ y_0 &= \overline{z}_y \end{aligned} \tag{A.0.6}$$

Next the derivative of Equation A.0.4 with respect to ψ is calculated,

$$\begin{aligned} \frac{\partial E}{\partial \psi} = 2\sum_i \left((z_{x,i} - x_0)\sin\psi - (z_{y,i} - y_0)\cos\psi\right) \\ \times \left((z_{x,i} - x_0)\cos\psi + (z_{y,i} - y_0)\sin\psi\right) = 0 \\ \sum_i (z_{x,i} - x_0)^2 \sin\psi\cos\psi + \sum_i (z_{x,i} - x_0)(z_{y,i} - y_0)(\sin^2\psi - \cos^2\psi) \\ - \sum_i (z_{y,i} - y_0)^2 \sin\psi\cos\psi = 0. \end{aligned} \tag{A.0.7}$$

By rearranging terms a with some trigonometric identities, this is simplified,

$$\begin{aligned} \frac{2\sin\psi\cos\psi}{\cos^2\psi - \sin^2\psi} &= \frac{2\sum_i (z_{x,i} - x_0)(z_{y,i} - y_0)}{\sum_i (z_{x,i} - x_0)^2 - \sum_i (z_{y,i} - y_0)^2} \\ \frac{2\tan\psi}{1 - \tan^2\psi} &= \frac{2\sum_i (z_{x,i} - x_0)(z_{y,i} - y_0)}{\sum_i (z_{x,i} - x_0)^2 - \sum_i (z_{y,i} - y_0)^2}. \end{aligned} \tag{A.0.8}$$

If we introduce a variable B as the negative inverse of this equation, we obtain the quadratic equation,

$$\begin{aligned} \frac{2\tan\psi}{1-\tan^2\psi} &= -\frac{1}{B} \\ \tan^2\psi - 2B\tan\psi - 1 &= 0 \\ \tan\psi &= B \pm \sqrt{B^2+1}, \end{aligned} \tag{A.0.9}$$

which can easily be solved if B is known. We make the following substitutions,

$$\begin{aligned} a &= \sum_i (z_{y,i}-y_0)^2 - \sum_i (z_{x,i}-x_0)^2 \\ b &= 2\sum_i (z_{x,i}-x_0)(z_{y,i}-y_0), \end{aligned} \tag{A.0.10}$$

and from Equation A.0.8 and Equation A.0.9 we obtain

$$\begin{aligned} -\frac{1}{B} &= \frac{b}{-a} \\ B &= \frac{a}{b} \\ \tan\psi &= \frac{a}{b} + \sqrt{\left(\frac{a}{b}\right)^2 + 1} \\ \tan\psi &= \frac{a+\sqrt{a^2+b^2}}{b}. \end{aligned} \tag{A.0.11}$$

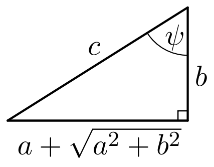


Figure A.2 – Diagram of triangle with ψ.

If we draw the triangle of ψ, as in Figure A.2, with the numerator and denominator of Equation A.0.11 as its sides, we obtain the following equation for its diagonal side,

$$\begin{aligned} c &= \sqrt{\left(a+\sqrt{a^2+b^2}\right)^2 + b^2} \\ &= \sqrt{2\left(a^2+b^2+a\sqrt{a^2+b^2}\right)}. \end{aligned} \tag{A.0.12}$$

With these we calculate the values for the sin and cos of ψ, without calculating the exact angle,

$$\begin{aligned} \cos\psi &= \frac{b}{c} \\ \sin\psi &= \frac{a+\sqrt{a^2+b^2}}{c}. \end{aligned} \tag{A.0.13}$$

Now that we have a equation for $\boldsymbol{p}_0$ and ψ, we can calculate any point on the line and use it as the second parameter of the line. We define this second point at a distance β from $\boldsymbol{p}_0$, which is calculated as

$$\begin{aligned} x_1 &= x_0 + \beta \cos \psi \\ y_1 &= y_0 + \beta \sin \psi. \end{aligned} \tag{A.0.14}$$

Equations A.0.6, A.0.10, A.0.12, A.0.13 and A.0.14 are now sufficient to obtain the parameters of the best fitted line from the data.

Bibliography

[1] Thrun, S., Burgard, W. and Fox, D.: *Probabilistic robotics*. MIT press, 2005.

[2] Elfes, A.: Using occupancy grids for mobile robot perception and navigation. *Computer*, , no. 6, pp. 46–57, 1989.

[3] Smith, R.C. and Cheeseman, P.: On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[4] Smith, R., Self, M. and Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In: *Autonomous Robot Vehicles*, pp. 167–193. Springer, 1990.

[5] Lu, F. and Milios, E.: Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[6] Thrun, S.: Towards programming tools for robots that integrate probabilistic computation and learning. In: *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 306–312. IEEE, 2000.

[7] Murphy, K.P.: Bayesian map learning in dynamic environments. In: *Advances in Neural Information Processing Systems*, pp. 1015–1021. 2000.

[8] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. *et al.*: FastSLAM: A factored solution to the simultaneous localization and mapping problem. *AAAI/IAAI*, vol. 593598, 2002.

[9] Mendes, E., Koch, P. and Lacroix, S.: ICP-based pose-graph SLAM. In: *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*, pp. 195–200. IEEE, 2016.

[10] Hess, W., Kohler, D., Rapp, H. and Andor, D.: Real-time loop closure in 2D LIDAR SLAM. In: *IEEE International Conference on Robotics and Automation*, pp. 1271–1278. IEEE, 2016.

[11] Jensfelt, P. and Christensen, H.I.: Pose tracking using laser scanning and minimalistic environmental models. *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 138–147, 2001.

[12] Guivant, J. and Nebot, E.: Simultaneous localization and map building: Test case for outdoor applications. In: *IEEE International Conference on Robotics and Automation*. 2002.

[13] Nieto, J., Bailey, T. and Nebot, E.: Scan-SLAM: Combining EKF-SLAM and scan correlation. In: *Field and Service Robotics*, pp. 167–178. Springer, 2006.

[14] Nieto, J., Bailey, T. and Nebot, E.: Recursive scan-matching SLAM. *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 39–49, 2007.

[15] Besl, P.J. and McKay, N.D.: Method for registration of 3-D shapes. In: *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, pp. 586–607. International Society for Optics and Photonics, 1992.

[16] Bosse, M. and Zlot, R.: Keypoint design and evaluation for place recognition in 2D lidar maps. *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.

[17] Himstedt, M., Frost, J., Hellbach, S., Böhme, H.-J. and Maehle, E.: Large scale place recognition in 2D LIDAR scans using geometrical landmark relations. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5030–5035. IEEE, 2014.

[18] Li, Y. and Olson, E.B.: Extracting general-purpose features from LIDAR data. In: *IEEE International Conference on Robotics and Automation*, pp. 1388–1393. IEEE, 2010.

[19] Barber, D.: *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[20] Martos, G., Muñoz, A. and González, J.: On the generalization of the Mahalanobis distance. In: *Iberoamerican Congress on Pattern Recognition*, pp. 125–132. Springer, 2013.

[21] Simon, M.K.: *Probability distributions involving Gaussian random variables: A handbook for engineers and scientists.* Springer Science & Business Media, 2007.

[22] Koller, D., Friedman, N. and Bach, F.: *Probabilistic Graphical Models: Principles and Techniques.* MIT press, 2009.

[23] Brookes, M.: The Matrix Reference Manual. 2011.
Available at: `http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html`

[24] Bonarini, A., Burgard, W., Fontana, G., Matteucci, M., Sorrenti, D.G. and Tardos, J.D.: RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets. In: *In proceedings of IROS'06 Workshop on Benchmarks in Robotics Research.* 2006.

[25] Ceriani, S., Fontana, G., Giusti, A., Marzorati, D., Matteucci, M., Migliore, D., Rizzi, D., Sorrenti, D.G. and Taddei, P.: Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, vol. 27, no. 4, pp. 353–371, 2009.