UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DETECTION STATISTICS & RANDOM FINITE SETS IN SIMULTANEOUS LOCALIZATION AND MAPPING

TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA

FELIPE IGNACIO INOSTROZA FERRARI

PROFESOR GUÍA:
MARTIN ADAMS

MIEMBROS DE LA COMISIÓN:
MARCOS ORCHARD CONCHA
BA-NGU VO
REZA HOSEINNEZHAD

Esta tesis fue financiada por Becas Conicyt - Doctorado Nacional, 2014,
CONICYT/PIA AFB180004 y por CONICYT FONDECYT project 1190979.

SANTIAGO DE CHILE

2023

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELECTRICA
POR: Felipe Ignacio Inostroza Ferrari
FECHA: 2023
PROF. GUÍA: Martin Adams

Estadísticas de detección y conjuntos aleatorios finitos en la construcción de mapas y localización simultánea

El uso de conjuntos aleatorios finitos (RFS) en la construcción de mapas y localización simultánea (SLAM) permite incorporar estadísticas de detección y evitar las heurísticas de manejo del mapa y asociación de datos, incluyendo estos problemas en la estimación Bayesiana. Esta tesis tiene dos objetivos: Incluir modelos de estadísticas de detección más precisos y modelos de descriptores en algoritmos de SLAM basadas en RFS, e introducir un algoritmo de SLAM basado en RFS que utilice un enfoque de optimización. El primero se cumple modelando las estadísticas de detección y un descriptor de un detector de círculos para LIDAR 2D. Estos modelos se aplican a Rao-Blackwellized (RB)-probability hypothesis density (PHD)-SLAM y multiple hypothesis (MH)-factored solution to SLAM (FastSLAM), ambos algoritmos muestran mejor desempeño al usar los modelos propuestos. El segundo objetivo se cumple al proponer una distribución híbrida conjunto-vector, que se usa para proponer un algoritmo de SLAM que aprovecha el enfoque de optimización. Resultados de simulación y usando datos públicos de cámaras estéreo muestran un desempeño competitivo con los algoritmos del estado del arte, mientras que resuelven el problema de asociación de datos usando el teorema de Bayes.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELECTRICA
POR: Felipe Ignacio Inostroza Ferrari
FECHA: 2023
PROF. GUÍA: Martin Adams

Detection Statistics & Random Finite Sets in Simultaneous Localization and Mapping

The use of random finite sets (RFSs) in simultaneous localization and mapping (SLAM) enables the incorporation of detection and clutter statistics and the circumvention of data association and map management heuristics by including these problems in the Bayesian estimation. The goal of this thesis is twofold: To include more accurate models of detection statistics, as well as descriptor information, into existing RFS-based and vector-based SLAM algorithms. And to introduce a batch estimation RFS-based SLAM algorithm. Modeling detection statistics and a descriptor for a 2D lidar-based circle detector achieved the former goal. These models are applied to Rao-Blackwellized (RB)-probability hypothesis density (PHD)-SLAM and multiple hypothesis (MH)-factored solution to SLAM (FastSLAM), and both methods show improved performance when using the models. The latter goal was achieved by introducing a joint vector-set distribution, which is used to introduce a SLAM algorithm that takes advantage of the state-of-the-art least squares approach to SLAM. Results on both simulated data and publicly available stereo-vision data show performance on par with state-of-the-art solutions while jointly solving the data association problem and the state estimation problem using Bayes theorem.

ACKNOWLEDGEMENTS

Primero me gustaría agradecer a mis padres María Teresa y Fernando por hacer todo esto posible, siempre creer en mí y apoyarme. A mi esposa Genesis, te conocí, me enamoré, vivimos juntos y nos casamos, todo durante este proceso.

También me gustaría agradecer a mi profesor guía, Martin Adams, que me ayudó a entender todo el proceso de investigación, desde tener una idea hasta completar la tan necesaria publicación. Me gustaría también agradecer a todos mis compañeros del laboratorio, por hacer cada día más ameno ya sea compartiendo interés por cosas nerd, o simplemente memes. Especialmente a Keith, Pablo, Leo, Mauricio, Paul e Isao.

También agradezco a CONICYT, ahora ANID, por su financiamiento a través de "Becas Conicyt - Doctorado Nacional, 2014" y proyecto CONICYT/PIA AFB180004 y CONICYT FONDECYT project 1190979.

Table of Content

List of Figures

Chapter 1

Introduction

1.1 Motivation

Random finite set (RFS)-based filters have been shown to outperform traditional vector-based filters when exposed to a significant amount of clutter measurements (i.e., false alarms), both in target tracking and simultaneous localization and mapping (SLAM) applications. However, the probability hypothesis density (PHD) filter, which was the only RFS-based method that had been applied to SLAM using experimental data at the time of starting this thesis, has been described as having "poor memory"; i.e., it tends to discard old information in favor of new measurements [1]. It has also been found to have lower performance than vector-based filters under low amounts of clutter in SLAM applications. Some of the methods that attempt to solve the problems of the PHD filter are the cardinalized PHD (CPHD) and cardinality-balanced multi-target multi-Bernoulli (CB-MemBer) filters, and more recently the labelled multi Bernoulli (LMB) and Generalized Labeled Multi Bernoulli (GLMB) filters[1, 2, 3, 4]. The LMB has been adapted to SLAM and has been shown to outperform PHD-SLAM in simulation [5].

Concurrently with the development of RFS-based filters in the tracking community, the robotics SLAM community has moved away from filtering-based solutions to the batch estimation approach, which uses non-linear optimization methods to obtain either a Maximum Likelihood (ML) or a Maximum a posteriori (MAP) solution [6, 7]. Such methods include graphSLAM [8], iSAM [9] and iSAM2 [6]. These algorithms, which are usually based on non-linear least squares optimization, provide more accurate solutions over larger datasets than their filtering counterparts. These methods rely on external routines to perform data association and map management, usually based on either maximum likelihood or place recognition algorithms. The RFS formulation of the problem in this form complicates the optimization process considerably, even when assuming a known number of landmarks. Developing an effective batch estimation algorithm using RFSs that includes detection statistics is a primary objective of this thesis.

The facts presented above provide a compelling reason to derive batch RFS-based SLAM algorithms, which should result in a superior performance both for large datasets and under high clutter.

1.2 Problem definition

In this thesis, the SLAM problem will be addressed. Also, the method for extracting detection statistics proposed during the student's Master's thesis [10] will be extended and complemented with descriptor information. Additionally, a way of calculating detection statistics for visual Keypoints, for which ground truth information is not available, will be presented. Further, RFS theory will be used to include data association and map management into the batch estimation objective function. In summary, the contributions of this thesis are:

- A demonstration of the importance of including detection statistics into current RFS-based SLAM algorithms.
- The introduction of a new batch SLAM estimation algorithm based on RFS theory.
- The application of this new RFS batch estimation algorithm to visual keypoint data, for which ground truth information cannot be obtained.

A brief introduction of the SLAM problem now follows.

1.2.1 SLAM Process and Measurement Model Dynamics

SLAM is a state estimation problem, in which the best estimate of the robot trajectory and map feature positions is sought over time, using all sensor measurements. In general, the underlying stochastic system representing the robot's pose component of SLAM using non-linear discrete-time equations, is represented as:

$$\boldsymbol{x}_k = \boldsymbol{g}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{\delta}_k) \tag{1.1}$$

$$\boldsymbol{z}_k^i = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{m}_k^j, \boldsymbol{\epsilon}_k) \ , \tag{1.2}$$

where:

- $\boldsymbol{x}_k$ represents the robot pose (spatial coordinates and orientation) at time-step k,
- $\boldsymbol{g}$ is the robot motion model,

- $\boldsymbol{u}_k$ is the the odometry measurement at time-step k,
- $\boldsymbol{\delta}_k$ is the process noise at time-step k,
- $\boldsymbol{z}_k^i$ is the i-th measurement vector at time-step k,
- $\boldsymbol{h}$ is the sensor-specific measurement model relating measurement and state in the spatial domain,
- $\boldsymbol{m}_k^j$ is a random vector for the position of landmark j,
- $\boldsymbol{\epsilon}_k$ is the spatial measurement noise

Traditional vector-based approaches to SLAM concatenate random state vectors that model the robot and landmarks, in a single vector that is used for the estimation process:

$$\boldsymbol{M}_k \equiv \left[\boldsymbol{m}_k^{1\top} \ \boldsymbol{m}_k^{2\top} \ \cdots \ \boldsymbol{m}_k^{m\top} \right]^\top \ . \tag{1.3}$$

Similarly, multiple measurements, corresponding to feature detections are also concatenated as:

$$\boldsymbol{z}_k \equiv \left[\boldsymbol{z}_k^{1\top} \ \boldsymbol{z}_k^{2\top} \ \cdots \ \boldsymbol{z}_k^{n\top} \right]^\top \ . \tag{1.4}$$

Furthermore, the generally complex data association problem needs to be solved so that measurement i and feature j correspond to the same landmark linked by the vector-based measurement model

$$\boldsymbol{z}_k^i \sim \boldsymbol{h}(\cdot|\boldsymbol{m}_k^j, \boldsymbol{x}_k) \ , \tag{1.5}$$

which incorporates the spatial noise $\boldsymbol{\epsilon}_k$ implicitly.

Within the RFS approach, the observed landmarks up to and including time-step k, are defined as

$$\mathcal{M}_k \equiv \{\boldsymbol{m}_k^1, \boldsymbol{m}_k^2, ..., \boldsymbol{m}_k^m\} \ , \tag{1.6}$$

where the number of landmarks, $|\mathcal{M}_k| = m$, is also a random variable. In general, the landmark from which a measurement is generated is unknown. Furthermore, there is a probability of detection, P_D, associated with every landmark, implying that it may be undetected with probability $1 - P_D$. Measurements may also be generated from sensor noise or objects of non-interest (clutter), with assumed known distributions. The set of all n measurements at time-step k is defined as

$$\mathcal{Z}_k \equiv \{\boldsymbol{z}_k^1, \boldsymbol{z}_k^2, ..., \boldsymbol{z}_k^n\} \ , \tag{1.7}$$

where $|\mathcal{Z}_k| = n$ is a random variable. One could question why the use of RFSs to model the map is necesary, when traditionally the map is considered static and therefore the map size is fixed. However, in the SLAM definition even though the map can be assumed static it is unknown. In a manner similar to the way in which the uncertainty in the landmark positions should be represented via a distribution, even though they are assumed static, the uncertainty in the size of the map also needs to be modeled even though the map size does not change. RFS distributions jointly model the uncertainty both in the map size and spatial uncertainty in the map elements, ie. the landmarks. In addition to this, map elements coming into the FoV of the robot for the first time can be modeled as new map elements being born. With these definitions, a new, set-based, measurement model can be defined:

$$\mathcal{Z}_k \equiv \mathcal{H}(\boldsymbol{x}_k, \mathcal{M}_k, \boldsymbol{\epsilon}_k) \cup \mathcal{E}_k \ , \tag{1.8}$$

where $\mathcal{E}_k$ represents the clutter and $\boldsymbol{\epsilon}_k$ represents the spatial noise of the measurements. Using a probabilistic framework and a filtering approach, the probability density functions (PDFs)

$$p\left(\boldsymbol{x}_{0:k}, \mathcal{M}_k \middle| \mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}\right) \ , \tag{1.9}$$

$$p\left(\boldsymbol{x}_{0:k}, M_k \middle| Z_{0:k}, \boldsymbol{u}_{1:k}\right) \tag{1.10}$$

are sought by RFS and vector approaches respectively. In both cases, the estimates are made relative to the initial robot's pose, at each time step.

1.3 Hypotheses

- The modeling and incorporation of detection statistics into the SLAM problem can improve the performance of SLAM algorithms.
- An RFS formulation of the batch (non-filtering) estimation SLAM problem is possible and can be used to solve map management and data association using Bayes theorem, instead of heuristic-based methods.
- Useful detection statistics can be obtained for non-semantic features, for example, Oriented FAST Rotated Brief (ORB) or scale-invariant feature transform (SIFT) features, allowing the use of RFS-based methods, even if the detection statistics are not physically meaningful from a human perspective.

1.4 Objectives

1.4.1 General Objective

The general objective of this Ph.D. thesis is to propose modified SLAM algorithms, which account for detection statistical models and descriptor models, resulting in improved SLAM performance, and to derive the first batch RFS-based solution to SLAM.

1.4.2 Specific Objectives

In particular, the specific objectives are:

- to further develop the application of detection statistics with particle filter-based SLAM, presented during the Master's thesis, to also include descriptor likelihoods;
- to compare the performance of filter-based SLAM systems with and without the models for detection statistics and descriptor likelihoods;
- to propose an RFS-based batch estimation (i.e., smoothing) SLAM algorithm;
- to compare the performance of the proposed batch estimation SLAM algorithm, which does not require external data association routines, with methods which are given the correct data association and methods which use heuristic data association decisions. This comparison is carried out in simulation;
- to adapt the batch RFS-SLAM algorithm to use real stereo visual data;
- to compare the performance of the proposed batch estimation SLAM algorithm with state-of-the-art algorithms, using a publicly available stereo visual dataset.

1.5 Thesis structure

This thesis is structured as follows.

- Chapter 2 presents a review of the state-of-the-art in SLAM, both vector-based and using RFS.
- Chapter 3 explains the importance of including detection statistics in SLAM and introduces the concepts for using RFSs in batch SLAM
- Chapter 4 explains the work carried out and advances made in filter-based RFS-SLAM.

- Chapter 5 introduces a batch SLAM concept based on general optimization methods.
- Chapter 6 proposes a new batch SLAM concept based on a hydrid vector-set distribution, with extensive simulations and real results.
- Chapter 7 presents the conclusions and possible avenues of future research.
- Annex A presents a list of publications by the author related to this thesis.

Chapter 2

Literature review

2.1 Introduction

This chapter presents the concepts of random finite sets (RFSs) and then introduces the state of the art in SLAM methods, both RFS-based and vector-based.

2.2 Random Finite Sets

Traditional Bayesian state estimation has previously focused on vector states. In this paradigm, the problem is presented as having a prior distribution for the state vector $\boldsymbol{x}$:

$$\boldsymbol{x} \sim p(\boldsymbol{x}) \ , \tag{2.1}$$

and the user is provided with a measurement vector $\boldsymbol{z}$ and its corresponding measurement model:

$$\boldsymbol{z} \sim g(\boldsymbol{z}|\boldsymbol{x}) \ . \tag{2.2}$$

Using both of these and Bayes Theorem, the posterior is obtained as

$$p(\boldsymbol{x}|\boldsymbol{z}) = \frac{p(\boldsymbol{x})g(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} \ . \tag{2.3}$$

To apply this theorem when either the size of the state vector or data association are unknown, heuristics with varying degrees of rigor are usually applied. In fact,in the case of random vectors, Bayes theorem can only be strictly applied when the dimension of the measurement and state vectors, and the data association are known. When this is not the case, map management heuristics are required to associate the state vector elements with the corresponding elements in the measurement vector, so that Bayes theorem can be applied to the model with assumed known data association.

Mahler [1] proposed a new paradigm for Bayesian estimation in these cases, by replacing the state vector $\boldsymbol{x}$ with an RFS $\mathcal{X}$. In this context, it is natural to think of the size of the set as a random variable. Also, by having no particular order, sets are well suited to handle the data association rigorously. To solve this problem, Mahler [1] introduced new tools to allow the propagation of random sets through the Bayes theorem, called Finite Set Statistics. By using these concepts a variety of new filters have been introduced based on various assumptions. These include the PHD, the CPHD [1], CB-MemBer [2], the LMB [11], and GLMB [12] filters.

2.2.1 Probability Hypothesis Density filter

In this section, a simplified version of the PHD filter will be introduced. In the PHD filter the state is assumed to be a Poisson RFS. This means that the elements are independent and identically distributed and that the cardinality of the set follows a Poisson distribution:

$$|\mathcal{X}| = r \sim \frac{\lambda^r}{r!} e^{-\lambda} \ , \tag{2.4}$$

where r is the cardinality of the set $\mathcal{X}$ and λ corresponds to both the mean and variance of the distribution. The distribution of the set can then be modeled by its first moment $v_k(\boldsymbol{m})$, called its intensity or PHD. This intensity function is similar to a probability distribution but, instead of integrating to unity, it integrates to the mean of the number of randomly varying elements within the set.

Using this intensity, the PHD filter's prediction equation propagates the intensity through time:

$$\begin{aligned} v_k^-(\boldsymbol{m}) = & \int p_{s,k}(\boldsymbol{\chi}) f_{k|k-1}(\boldsymbol{m}|\boldsymbol{\chi}) v_{k-1}^+(\boldsymbol{\chi}) d\boldsymbol{\chi} \\ & + \int \beta_{k|k-1}(\boldsymbol{m}|\boldsymbol{\chi}) v_{k-1}^+(\boldsymbol{\chi}) d\boldsymbol{\chi} + v_k^b(\boldsymbol{m}) \ , \end{aligned} \tag{2.5}$$

where:

- $v_{k-1}^+(\boldsymbol{m})$ is the intensity of features at $\boldsymbol{m}$ at time $k-1$ given all measurements up to time $k-1$,
- $v_k^-(\boldsymbol{m})$ is the intensity of features at $\boldsymbol{m}$ at time k given all measurements up to time $k-1$,
- $p_{s,k}(\boldsymbol{\chi})$ is the probability of a feature at state $\boldsymbol{\chi}$ to survive to time k,

- $f_{k|k-1}(\boldsymbol{m}|\boldsymbol{\chi})$ is the state transition density from state $\boldsymbol{\chi}$ to state $\boldsymbol{m}$,
- $\beta_{k|k-1}(\boldsymbol{m}|\boldsymbol{\chi})$ is the probability that a feature at state $\boldsymbol{\chi}$ will spawn another feature at $\boldsymbol{m}$,
- $v_k^b(\boldsymbol{m})$ is the intensity of the set of new features that might appear at $\boldsymbol{m}$.

In SLAM, however, features are usually considered to be static, permanent, and incapable of spawning new features, so the prediction equation simplifies to:

$$v_k^-(\boldsymbol{m}) = v_{k-1}^+(\boldsymbol{m}) + v_k^b(\boldsymbol{m}) \ . \tag{2.6}$$

The measurement set, $\mathcal{Z}$, is distributed according to a multi-target measurement model, in which each element, $\boldsymbol{m}$, in the state set $\mathcal{X}$ generates a measurement with probability $P_D(\boldsymbol{m})$ and generates no measurement with probability $(1 - P_D(\boldsymbol{m}))$. Clutter measurements are modeled as a Poisson RFS with given (known) intensity function $\kappa(\boldsymbol{z}_i)$. Therefore, the measurement set $\mathcal{Z}$ is the union of the detection and the clutter sets. Using these assumptions, the PHD update equation

$$v_k^+(\boldsymbol{m}) = v_k^-(\boldsymbol{m}) \left[1 - P_D(\boldsymbol{m}) + \sum_{i=1}^{|\mathcal{Z}_k|} \frac{P_D(\boldsymbol{m})\boldsymbol{h}(\boldsymbol{z}_i|\boldsymbol{m})}{\kappa(\boldsymbol{z}_i) + \int P_D(\boldsymbol{m}')\boldsymbol{h}(\boldsymbol{z}_i|\boldsymbol{m}')v_k^-(\boldsymbol{m}')d\boldsymbol{m}'} \right] \tag{2.7}$$

is derived in [1]. In Equation (2.7), $\kappa(\boldsymbol{z}_i)$ is the intensity of the false alarm set at the position where measurement $\boldsymbol{z}_i$ was obtained. In this update equation, $v_k^-(\boldsymbol{m})(1 - P_D(\boldsymbol{m}))$ represents the possibility that features were misdetected while the second term represents detections. To implement this update equation the intensity can be represented either using sequential Monte-Carlo methods or a Gaussian mixture (GM) [1]. Sequential Monte Carlo methods have the advantage of being able to deal with highly non-linear measurement models while the GM-based implementation has a considerably reduced computational complexity, but is only able to deal with mildly non-linear measurement models (by using the Extended Kalman Filter to update means and covariances of each Gaussian Component). Because of the potentially large size of the SLAM state, which can range from hundreds to millions of features, only the GM-based solution will be used in this thesis.

2.2.2 The CB-MemBer filter

This section will show the main steps of the cardinality-balanced multi-target multi-Bernoulli (CB-MemBer) filter as introduced by Vo et. al. in [2]. The CB-MemBer filter models the map using a multi-Bernoulli RFS. A multi-Bernoulli RFS is defined as the union of N_k Bernoulli RFSs, where each Bernoulli RFS i has a probability $1-r_k^i$ of being empty and a probability r_k^i of being a singleton $\boldsymbol{m}$ with probability distribution $p_k^i(\boldsymbol{m})$. The multi-Bernoulli RFS for the map $\mathcal{M}_{k-1}$ is then represented by its N_{k-1} probabilities of existence r_{k-1}^i and spatial distributions $p_{k-1}^i(\boldsymbol{m})$:

$$\mathcal{M}_{k-1} = \left\{\left(r_{k-1}^i, p_{k-1}^i(\boldsymbol{m})\right)\right\}_{i=1}^{N_{k-1}} \ . \tag{2.8}$$

The predicted map at time k for the CB-MemBer filter is then:

$$\mathcal{M}_{k|k-1} = \left\{\left(r_{k-1}^i, p_{k-1}^i(\boldsymbol{m})\right)\right\}_{i=1}^{N_{k-1}} \cup \left\{\left(r_{B,k|k-1}^i, p_{B,k|k-1}^i(\boldsymbol{m})\right)\right\}_{i=1}^{N_{B,k-1}} \ , \tag{2.9}$$

where $\left\{\left(r_{B,k|k-1}^i, p_{B,k|k-1}^i(\boldsymbol{m})\right)\right\}_{i=1}^{N_{B,k-1}}$ are the parameters of the multi-Bernoulli RFS of new targets appearing at time k (i.e. the RFS of births). The update step can then be written as:

$$\mathcal{M}_{k} = \left\{\left(r_{L,k}^i, p_{L,k}^i(\boldsymbol{m})\right)\right\}_{i=1}^{N_{k|k-1}} \cup \left\{\left(r_{U,k|k-1}^i, p_{U,k|k-1}^i(\boldsymbol{m})\right)\right\}_{i=1}^{|\mathcal{Z}_k|} \ , \tag{2.10}$$

where

$$r_{L,k}^i = r_{k|k-1}^i \frac{1-\int p_{k|k-1}^i(\boldsymbol{m}) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}}{1 - r_{k|k-1}^i \int p_{k|k-1}^i(\boldsymbol{m}) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}} \ , \tag{2.11}$$

$$p_{L,k}^i(\boldsymbol{m}) = p_{k|k-1}^i(\boldsymbol{m}) \frac{1 - P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k)}{1 - \int p_{k|k-1}^i(\boldsymbol{m}') P_{\mathrm{D}}(\boldsymbol{m}'|\boldsymbol{x}_k) d\boldsymbol{m}'} \ , \tag{2.12}$$

$$r_{U,k|k-1}^i = \frac{\sum_{i=1}^{N_{k|k-1}} \frac{r_{k|k-1}^i(1-r_{k|k-1}^i)\int p_{k|k-1}^i(\boldsymbol{m}) \boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m},\boldsymbol{x}_k) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}}{\left(1-r_{k|k-1}^i \int p_{k|k-1}^i(\boldsymbol{m}) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}\right)^2}}{\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) + \sum_{i=1}^{N_{k|k-1}} \frac{r_{k|k-1}^i \int p_{k|k-1}^i(\boldsymbol{m}) \boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m},\boldsymbol{x}_k) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}}{1-r_{k|k-1}^i \int p_{k|k-1}^i(\boldsymbol{m}) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) d\boldsymbol{m}}} \ , \tag{2.13}$$

$$p_{U,k|k-1}^i(\boldsymbol{m}) = \frac{\sum_{i=1}^{N_{k|k-1}} \frac{r_{k|k-1}^i}{1-r_{k|k-1}^i} p_{k|k-1}^i(\boldsymbol{m}) \boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m},\boldsymbol{x}_k) P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k)}{\sum_{i=1}^{N_{k|k-1}} \frac{r_{k|k-1}^i}{1-r_{k|k-1}^i} \int p_{k|k-1}^i(\boldsymbol{m}') \boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m}',\boldsymbol{x}_k) P_{\mathrm{D}}(\boldsymbol{m}'|\boldsymbol{x}_k) d\boldsymbol{m}'} \ . \tag{2.14}$$

Using these equations the CB-MemBer filter can then be implemented using either a Gaussian mixture or a particle filter to approximate the spatial distributions $p(\boldsymbol{m})$.

2.2.3 The labelled multi Bernoulli (LMB) filter

Recently, Reuter et. al. [11] introduced a new filter based on multi-Bernoulli RFSs by adding unique track labels. The addition of labels allows for an exact estimate of the first posterior moment, as opposed to the approximate estimate provided by the CB-MemBer, which assumes a high signal-to-noise ratio.

The LMB filter uses a LMB RFS to model the map. A LMB RFS is a modified multi-

Bernoulli RFS in which every Bernoulli component has a unique label $l \in L$ and can be similarly described by its parameters

$$\mathcal{M}_{k|k-1} = \{(r_{k-1}(l), \pi_{k-1}(\boldsymbol{m}, l))\}_{l \in L} \ . \tag{2.15}$$

Analogous to Equation 2.9, the predicted map at time k can be written as:

$$\mathcal{M}_{k|k-1} = \{(r_{k-1}(l), \pi_{k-1}(\boldsymbol{m}, l))\}_{l \in L} \cup \left\{\left(r_{B,k|k-1}(l), \pi_{B,k|k-1}(\boldsymbol{m}, l)\right)\right\}_{l \in B} \ , \tag{2.16}$$

$$\mathcal{M}_{k|k-1} = \left\{\left(r_{k|k-1}(l), \pi_{k|k-1}(\boldsymbol{m}, l)\right)\right\}_{l \in L^+} \ , \tag{2.17}$$

where B is the label space of the new (birth) map features appearing at time k, and $L^+ = L \cup B$. The updated LMB RFS can then be approximated as:

$$\mathcal{M}_k = \{(r_k(l), \pi_k(\boldsymbol{m}, l))\}_{l \in L^+} \ , \tag{2.18}$$

$$\tag{2.19}$$

where

$$r_k(l) = \sum_{(I_+,\theta) \in F(L_+) \times \Theta_{I_+}} \omega^{(I_+,\theta)}(\mathcal{Z}_k) 1_{I_+}(l) \ , \tag{2.20}$$

$$\pi_k(\boldsymbol{m}, l) = \frac{1}{r_k(l)} \sum_{(I_+,\theta) \in F(L_+) \times \Theta_{I_+}} \omega^{(I_+,\theta)}(\mathcal{Z}_k) 1_{I_+}(l) \pi^{\theta}(\boldsymbol{m}, l | \mathcal{Z}_k) \ , \tag{2.21}$$

$$\omega^{(I_+,\theta)}(\mathcal{Z}_k) \propto \omega_+(I_+) \prod_{l \in I_+} \int_{\boldsymbol{m}} \pi_{k|k-1}(\boldsymbol{m}, l) \psi_{\mathcal{Z}_k}(\boldsymbol{m}, l, \theta) d\boldsymbol{m} \ , \tag{2.22}$$

$$\psi_{\mathcal{Z}_k}(\boldsymbol{m}, l, \theta) = \begin{cases} \frac{P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) \boldsymbol{h}(\boldsymbol{z}_k^{\theta(l)}|\boldsymbol{m}, \boldsymbol{x}_k)}{\kappa(\boldsymbol{z}_k^{\theta(l)}|\boldsymbol{x}_k)} & \text{if } \theta(l) > 0 \\ 1 - P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k) & \text{if } \theta(l) = 0 \end{cases} \ , \tag{2.23}$$

$$\omega_+(I_+) = \prod_{j \in L_+} \left(1 - r_{k|k-1}(j)\right) \prod_{l \in I_+} \frac{1_{L_+}(l) r_{k|k-1}(l)}{1 - r_{k|k-1}(l)} \ , \tag{2.24}$$

where $F(L_+)$ is the space of all finite subsets of L_+, and Θ_{I_+} is the space of all associations $\theta : I_+ \to \{0, 1, ..., |\mathcal{Z}_k|\}$, such that $\theta(l) = j > 0$ means that label l is associated with measurement z_k^j, and $\theta(l) = 0$ means that label l is not associated (missed detection).

2.2.4 The GLMB filter

Vo and Vo [4, 12] introduced a new labeled RFS distribution, namely the GLMB distribution, which is a conjugate prior when using the standard multitarget tracking measurement models.

With this model as a prior and the multitarget RFS sensor model, the posterior also has the form of a GLMB distribution.

The GLMB distribution $\pi(\mathcal{M})$ is a density over a labeled set $\mathcal{M}$, with label space $\mathbb{L}$ and state space $\mathbb{M}$

$$\pi(\mathcal{M}) = \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M})) \, [\pi^c]^{\mathcal{M}} \ , \tag{2.25}$$

where $[\pi^c]^{\mathcal{M}}$ is the set exponent, defined as

$$[f]^{\mathcal{X}} = \prod_{x \in \mathcal{X}} f(x) \ , \tag{2.26}$$

for any function $f(\cdot)$ and set $\mathcal{X}$, $\mathcal{C}$ is a discrete index set, $\Delta(\mathcal{M})$ is the distinct label indicator, defined as

$$\Delta(\mathcal{M}) = \delta_{|\mathcal{M}|}(|\mathcal{L}(\mathcal{M})|) = \begin{cases} 1 & \text{for no repeated labels in } \mathcal{M} \\ 0 & \text{else} \ , \end{cases} \tag{2.27}$$

which is zero if there are any repeated labels in the set $\mathcal{M}$, and unity otherwise. $w^c(L)$ satisfies

$$\sum_{L \subseteq \mathbb{L}} \sum_{c \in \mathcal{C}} w^c(L) = 1 \ , \tag{2.28}$$

$$\int \pi^c(\boldsymbol{m}, \boldsymbol{l}) = 1 \ . \tag{2.29}$$

In [12] the GLMB filter is introduced and the corresponding GLMB distribution is shown to be a conjugate prior when using the labeled RFS sensor model. To prove this the GLMB distribution from (2.25) is written in a special form to help with the implementation, called the δ Generalized Labeled Multi Bernoulli (δ-GLMB)

$$\pi(\mathcal{M}) = \Delta(\mathcal{M}) \sum_{(I,c) \in \mathcal{F}(\mathbb{L}) \times \mathcal{C}} w^{(I,c)} \delta_I(\mathcal{L}(\mathcal{M})) \prod_{\boldsymbol{m}, l \in \mathcal{M}} \pi^c(\boldsymbol{m}, l) \ , \tag{2.30}$$

where $\delta_I(\mathcal{L}(\mathcal{M}))$ is the set-valued Kronecker delta which discards all sets $\mathcal{M}$ whose labels are not equal to I, $w^{(c,I)}$ is a real-valued weight, and $\mathcal{F}(\mathbb{L})$ is the set of all possible subsets of the label space $\mathbb{L}$.

The δ-GLMB prediction is then shown to be

$$\pi_+(\mathcal{M}_+) = \Delta(\mathcal{M}_+) \sum_{(I_+,c) \in \mathcal{F}(\mathbb{L}_+) \times \mathcal{C}} w_+^{(I_+,c)} \delta_{I_+}(\mathcal{L}(\mathcal{M}_+)) \prod_{\boldsymbol{m}, l \in \mathcal{M}_+} \pi_+^c(\boldsymbol{m}, l) \ , \tag{2.31}$$

where

$$w_{+}^{(I_{+},c)} = w_{S}^{c}(I \cap \mathbb{L}) w_{B}(I_{+} \cap \mathbb{B}) \ , \tag{2.32}$$

$$w_{S}^{c}(L) = [\eta_{S}^{c}]^{L} \sum_{I \supseteq L} [1 - \eta_{S}^{c}]^{I-L} \, w^{(I,c)} \ , \tag{2.33}$$

$$\eta_{S}^{c}(l) = \langle p_{S}(\cdot, l), \pi^{c}(\cdot, l) \rangle \ , \tag{2.34}$$

$$\pi_{+}^{c}(\boldsymbol{m}, l) = 1_{\mathbb{L}}(l) p_{S}(\boldsymbol{m}, l) + 1_{\mathbb{B}}(l) p_{B}(\boldsymbol{m}, l) \ , \tag{2.35}$$

$$p_{S}(\boldsymbol{m}, l) = \frac{\langle p_{S}(\cdot, l) f(\boldsymbol{m}|\cdot, l), \pi^{c}(\cdot, l) \rangle}{\eta_{S}^{c}(l)} \ , \tag{2.36}$$

where $f(\boldsymbol{m}_{+}|\boldsymbol{m}, l)$ is the transition density from state $\boldsymbol{m}$ to $\boldsymbol{m}_{+}$ for surviving elements, $p_S(\boldsymbol{m}, l)$ is the survival probability, $p_B(\boldsymbol{m}, l)$ is the birth spatial distribution and $w_B(\cdot)$ are the birth weights. Note that because in this thesis the set-valued variable to be estimated is the map $\mathcal{M}$, which is assumed static, then these equations are not necessary and the prediction merely becomes

$$\pi_{+}(\mathcal{M}_{+}) = \pi(\mathcal{M}) \ . \tag{2.37}$$

The δ-GLMB update is then

$$\pi(\mathcal{M}|\mathcal{Z}) = \Delta(\mathcal{M}) \sum_{(I,c) \in \mathcal{F}(\mathbb{L}) \times \mathcal{C}} \sum_{\theta \in \Theta(I)} w^{(I,c,\theta)}(\mathcal{Z}) \delta_{I}(\mathcal{L}(\mathcal{M})) \left[\pi^{(c,\theta)}(\cdot|\mathcal{Z}) \right]^{\mathcal{M}} \ , \tag{2.38}$$

where $\Theta(I)$ is the space of possible associations with domain I, i.e. with existing labels I, and

$$w^{(I,c,\theta)}(\mathcal{Z}) \propto w^{(I,c)} \left[\eta_{\mathcal{Z}}^{(c,\theta)} \right]^{I} \ , \tag{2.39}$$

$$\eta_{\mathcal{Z}}^{(c,\theta)}(l) = \langle \pi^{c}(\cdot, l), \psi_{\mathcal{Z}}(\cdot, l; \theta) \rangle \ , \tag{2.40}$$

$$\pi^{(c,\theta)}(\boldsymbol{m}, l|\mathcal{Z}) = \frac{\pi^{(c)}(\boldsymbol{m}, l) \psi_{\mathcal{Z}}(\cdot, l; \theta)}{\eta_{\mathcal{Z}}^{(c,\theta)}(l)} \ , \tag{2.41}$$

$$\psi_{\mathcal{Z}}(\boldsymbol{m}, l; \theta) = \begin{cases} \dfrac{P_{\mathrm{D}}(\boldsymbol{m}, l) p\left(\boldsymbol{z}^{a_{\theta}(j)} \middle| \boldsymbol{m}\right)}{\kappa(\boldsymbol{z}^{a_{\theta}(j)})} & \text{if } (\boldsymbol{m}, l) \in \mathcal{M}^{A_{\theta}} \\ 1 - P_{\mathrm{D}}(\boldsymbol{m}, l) & \text{otherwise} \end{cases} \ , \tag{2.42}$$

where $\boldsymbol{z}^{a_{\theta}(j)}$ is the measurement associated to map label l according to θ, $\mathcal{M}^{A_{\theta}}$ is the subset of $\mathcal{M}$ that has been associated to a measurement according to θ.

2.3 Simultaneous Localization and Mapping

The SLAM problem is one of the fundamental problems in autonomous navigation and has been considered by some to be the "Holy Grail" of mobile robotics [13]. A robot equipped with exteroceptive sensors (and optionally proprioceptive) is tasked with estimating a map of its environment as it transverses it, while concurrently estimating its pose within that map. During the last two decades, remarkable progress has been made on this problem and this section reviews the SLAM problem and some of the most important advances.

2.3.1 Observability of SLAM

The first question that should be asked after formulating the SLAM problem is whether it is actually solvable. In control theory, observability is defined as the ability to estimate the state of a system from its inputs and outputs. In 2001, Dissanayake, Newman, Clark, Durrant-Whyte, and Csorba [13] provided a solution to the linear version of the SLAM problem, with known initial pose, and proved its convergence. This implies that the problem is observable. However, the observability of the linear version of the problem, cannot be applied to the real SLAM definition, which is highly non-linear. In 2006, Lee, Wijesoma, and Ibanez Guzman [14] showed that the two-dimensional non-linear version of SLAM with range bearing measurements and unknown initial robot pose, is observable only if at least two landmarks have known positions. Wang and Dissanayake [15] used Fisher's linear discriminant to show a general way to calculate the observability of different non-linear SLAM formulations. They used their method to confirm the results in [14] and show that the solution to two-dimensional SLAM, with known initial pose and a range bearing measurement model, is observable without the need for known feature positions.

It is important to note that all the results on the observability of SLAM have been obtained using the vector-based formulation of SLAM, with known feature number and data association. Therefore these results cannot be directly applied to the set-based definition of the SLAM problem.

2.3.2 Vector-based SLAM

Dissanayake, Newman, Clark, Durrant-Whyte, and Csorba [13] introduced a solution based on the extended Kalman filter (EKF), EKF-SLAM, given known target number and data association. In the case of linear measurement and robot motion models (i.e., the Kalman filter (KF)) they proved that it will converge monotonically to the correct solution, for both the map estimate and robot trajectory. The complexity of this algorithm is $O(|\boldsymbol{z}|\ |\boldsymbol{m}|^2)$, where $|\boldsymbol{z}|$ and

$|\boldsymbol{m}|$ are the number of measurements and number of landmarks respectively.

Thrun, Liu, Koller, Ng, Ghahramani, and Durrant-Whyte [16] introduced a solution to SLAM by modifying the dual of the EKF, called the extended information filter (EIF). In EIF-SLAM instead of representing the posterior using its mean $\boldsymbol{\mu}$ and covariance Σ it is represented using the information matrix H and information vector $\boldsymbol{b}$, defined as:

$$H \equiv \Sigma^{-1} \tag{2.43}$$

$$\boldsymbol{b} \equiv \boldsymbol{\mu}^T H \ . \tag{2.44}$$

In [16], the authors showed that the solution to EIF-SLAM in this form is naturally sparse. By enforcing this sparseness the authors reduced the complexity of both the prediction and the update steps to constant time. This considers the addition of a single measurement at a time, making the complexity of the algorithm $O(|Z|)$. However, significant computation is required to recover the mean and covariance from the information matrix and vector. It was shown that this new sparse EIF (SEIF) is more likely to become inconsistent (i.e. overconfident) than its non-sparse counterpart, but this did not seem to impact the results of the experiments.

Montemerlo, Thrun, Koller, Wegbreit, et al. [17] introduced a Rao-Blackwellized (RB)-particle filter (PF) solution to SLAM, called a factored solution to SLAM (FastSLAM), by factoring the Bayes posterior into:

$$p\left(\boldsymbol{x}_{0:k}, M_k \middle| Z_k, \boldsymbol{u}_{0:k}\right) = p\left(\boldsymbol{x}_{0:k} \middle| Z_k, \boldsymbol{u}_{0:k}\right) p\left(M_k \middle| \boldsymbol{x}_{0:k}, Z_k, \boldsymbol{u}_{0:k}\right) \tag{2.45}$$

$$= p\left(\boldsymbol{x}_{0:k} \middle| Z_k, \boldsymbol{u}_{0:k}\right) \prod_{i=0}^{|M_k|} p\left(m_i \middle| \boldsymbol{x}_{0:k}, Z_k, \boldsymbol{u}_{0:k}\right) \ . \tag{2.46}$$

This implies that landmarks in the map are conditionally independent given the robot's trajectory. Therefore, instead of using a single EKF with a state vector of dimension $|M_k|$, each particle uses a collection of $|M_k|$ fixed dimension EKFs to track individual landmarks separately. To do this the full trajectory has to be estimated, for which a particle filter is used. Finally, by storing the features in a K-D Tree they managed to reduce the computational complexity of the algorithm to $O(|Z_k| \ \log(|M_k|))$

Montemerlo, Thrun, Koller, and Wegbreit [18] proposed an improved version of FastSLAM, called FastSLAM 2.0, in which the measurements are included in the proposal distribution of the particle filter. This allows for more efficient use of the particles, permitting either the use of a reduced number of particles or the potential closure of longer loops.

Nieto, Guivant, Nebot, and Thrun [19] introduced a different modification to the FastSLAM algorithm by applying the Multiple Hypothesis Tracking concepts to deal with the data association problem in SLAM. Multiple Hypothesis Tracking is a deferred decision strategy in which multiple data association hypotheses are kept in anticipation that future measurements will disambiguate the data association uncertainty. In this algorithm, called multiple hypothesis (MH)-FastSLAM, when a particle determines that multiple hypotheses are possible it splits into several particles with identical pose values to its parent, one representing each possible data association, including a new feature and false alarm hypotheses. After this, the regular weighting and re-sampling of particles are expected to resolve the uncertainty at a future time.

Thrun and Montemerlo [20] proposed a different strategy to solve the SLAM problem, called graph SLAM. This is also referred to in the literature as batch estimation and bundle adjustment from computer vision. Graph SLAM consists of progressively building a graph of soft constraints (referred to in [20] as information constraints). In this graph each measurement $\boldsymbol{z}_k^i$ introduces a constraint between the landmark it represents and the robot's pose at time k, $\boldsymbol{x}_k$:

$$f_k(\boldsymbol{z}_k^i, \boldsymbol{x}_k) \equiv (\boldsymbol{z}_k^i - \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{m}_k^{j(i)}))^\top \boldsymbol{Q}_k^{-1} (\boldsymbol{z}_k^i - \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{m}_k^{j(i)})) \ , \tag{2.47}$$

where $\boldsymbol{Q}_k$ is the covariance of the measurement model, which is assumed to have additive Gaussian noise. Similarly, each movement of the robot introduces a constraint between two contiguous poses of the robot $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$:

$$g_k(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}) \equiv (\boldsymbol{x}_k - \boldsymbol{g}(\boldsymbol{u}_{k-1}, \boldsymbol{x}_{k-1}))^\top \boldsymbol{R}_k^{-1} (\boldsymbol{x}_k - \boldsymbol{g}(\boldsymbol{u}_{k-1}, \boldsymbol{x}_{k-1})) \ , \tag{2.48}$$

where $\boldsymbol{R}_k$ is the covariance of the motion model, which is also assumed to have additive Gaussian noise. Using this graph the solution is obtained by minimizing the cost function (2.49).

$$\min_{\boldsymbol{x}_{0:k}} \left(\boldsymbol{x}_0^\top \boldsymbol{\Omega}_0 \boldsymbol{x}_0 + \sum_k \sum_i f_k(\boldsymbol{z}_k^i, \boldsymbol{x}_k) + \sum_k g_k(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}) \right) \ , \tag{2.49}$$

where $\boldsymbol{x}_0$ is the initial pose of the robot and the $\boldsymbol{x}_0^\top \boldsymbol{\Omega}_0 \boldsymbol{x}_0$ term is called an anchoring constraint, where $\boldsymbol{\Omega}_0$ should be a positive definite matrix, and it anchors the absolute coordinates of the map by locking the first pose of the trajectory to $[0, 0, 0]$, or some other known pose. The minimization problem can be solved in many ways, but in the paper, it was solved by linearising it and solving it in the information space. Then, not only the mode is recovered but also its associated covariance.

Kümmerle, Grisetti, Strasdat, Konolige, and Burgard [21] introduced a library, named g2o, to solve general least squares optimization such as the one introduced above. In this library, the optimization function is simplified as (2.50).

$$F(\boldsymbol{x}) = \sum_{<i,j>\in\mathcal{C}} e(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})^\intercal \boldsymbol{\Omega}_{ij} e(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij}) \tag{2.50a}$$

$$\boldsymbol{x}^* = \arg\min_x F(\boldsymbol{x}) \tag{2.50b}$$

where $\mathcal{C}$ is the set of measurements, in which measurement $\boldsymbol{z}_{ij}$ relates state variable $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$. $e(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{ij})$ is the error function and $\boldsymbol{\Omega}_{ij}$ is the information matrix. In this formulation, both odometry measurements, as well as landmark measurements, simply relate different states to each other. g2o utilizes the sparse structure of this function, particularly the sparse structure of the hessian matrix of $F(\boldsymbol{x})$, to efficiently solve the optimization function. For this, it uses sparse linear solvers and non-linear optimization algorithms, both of which can be selected. Further, it is capable of using the special structure of the landmarks in SLAM, which do not have measurements that relate them to each other to increase the efficiency of the solver.

Kaess, Ranganathan, and Dellaert [9] introduced incremental Smoothing and Mapping (iSAM) which solves SLAM using a special structure, called a Bayes tree which enables iSAM to solve the optimization formulation in an incremental fashion. The Bayes tree formulation is equivalent to the Cholesky square root of the Hessian matrix. This allows the solver to partially use the previous solution to incrementally solve the next optimization when adding new poses and measurements to the optimization problem. iSAM2 [6] is a follow-up to iSAM and it uses the Bayes tree representation while checking dynamically whether the linearization point of each measurement needs to be updated. Additionally, it reorders the variables in the estimate to reduce the fill-in of the Cholesky decomposition.

Mur-Artal, Montiel, and Tardós [22] proposed a complete monocular visual SLAM solution, using g2o [21] as a backend and the ORB keypoint detector and descriptor pair [23] as is basis. In this system ORB keypoints are extracted on each image and are matched with each other using their descriptors. Importantly this system uses three separate threads to accomplish real-time robust performance, *Tracking*, *Local Mapping*, and *Loop Closing.* In the *Tracking* thread only the latest camera pose is estimated using the set of already estimated keypoints as a known map. This thread is capable of running at the camera framerate, providing a real-time estimation of the current camera pose. The *Tracking* thread decides on whether the current frame should be a Keyframe and therefore be optimized as a part of the trajectory, based on how many of the detections in the current frame are seen by Keyframes already in the trajectory estimate,. The *Local Mapping* thread inserts the Keyframes given by *Tracking* into the g2o optimizer and performs the map management, i.e. creating new Keypoints and removing existing Keypoints from the map, while it runs a limited trajectory optimization only optimizing the last keyframe and the other Keyframes connected to it through the spanning tree. The *Loop Closing* thread takes the latest Keyframe from the *Local Mapping* thread and tries to detect loop closures between it and any Keypose in the trajectory estimate. To achieve this, the ORB features are compared using bag of words approach.

In [24], the ORBSLAM algorithm is extended to use stereo or RGB-D cameras and to be able to run the algorithm in localization-only mode with a fixed map. Finally, in [25], the authors further extend the algorithm to multisession SLAM in which the robot can start in the previously mapped area and is able to localize itself and continue the SLAM process.

Zhang and Singh [26] introduced another solution to the SLAM problem, called Lidar Odometry And Mapping (LOAM). The LOAM algorithm is based on 3D lidar measurements. This algorithm takes advantage of the asymmetric resolution of modern 3D lidar sensors, which usually have a limited number of high-resolution layers. Therefore features are extracted in each layer separately and then they can be associated across layers. These features are planar regions and corners. Neither of them are strictly Keypoints since they do not have a point location and a single feature can result in multiple detections.

Shan and Englot [27] created a similar solution based on LOAM called Lightweight and Ground Optimized LOAM (LeGO-LOAM). In this version of the algorithm the floor/ground is not discarded, and an iterative closest point (ICP)-based loop closure method.

Collectively all the methods previously described are vector-based solutions to SLAM. This means that they have to rely on heuristics to solve both the data association and map management problems. For this reason, this thesis will focus on set-based methods which rely on the Bayes theorem to solve these problems jointly with the conventional state estimation problem.

2.3.3 Random Finite Set-based SLAM

Mullane, Vo, and Adams [28] introduced the concept of Random Finite Sets into the SLAM problem. By recognizing that the SLAM state is more naturally represented by a set, instead of a vector, they were able to include the data association and map management problems into the Bayesian estimation paradigm. Previous solutions to the SLAM problem, such as the ones described in the previous section, resolved these problems using heuristic approaches.

Leung, Inostroza, and Adams [29] showed that the RB-PHD-SLAM is closely related to FastSLAM, and that RB-PHD-SLAM is equivalent to FastSLAM under ideal conditions.

Deusch, Reuter, and Dietmayer [5] applied the LMB filter to SLAM and showed that it has better performance than PHD-SLAM in simulation [5].

Moratuwage, Adams, and Inostroza [30] applied the δ-GLMB filter to SLAM with a Rao Blackbellized formulation, similar to the one in the other RFS SLAM filters in this section. [1]

[1]This work was carried out during the course of this thesis and in collaboration with the author.

Gao, Battistelli, and Chisci [31] improved on the RB-PHD-SLAM by using a proposal distribution that is closer to the posterior distribution on the latest robot pose, in a manner similar to the way that FastSLAM-2 improves FastSLAM.

2.3.4 Evaluating SLAM performance

Any SLAM solution will be composed of a trajectory estimate $\mathbf{x}^{\text{est}}_{0:K}$ and a map estimate $\mathcal{M}^{\text{est}}$. Metrics to evaluate SLAM performance have usually focused on the trajectory. This is because it is possible to measure the sensor's pose using outside-in methods, thus producing a ground truth trajectory $\mathbf{x}^{\text{gt}}_{0:K}$ against which different algorithms can be measured.

Metrics to evaluate trajectory error include Absolute Trajectory Error (ATE) and Relative Position Error (RPE) [32]. Sturm, Engelhard, Endres, Burgard, and Cremers [32] introduced the ATE and RPE metrics. In the ATE metric both trajectories are first aligned by rotating, translating, and optionally scaling the estimated trajectory to match the ground truth one. After alignment the individual positions $\mathbf{x}^{\text{aligned}}_{0:K}$ are compared and the RMS value of the error is calculated, producing the ATE value

$$ATE = \left(\frac{1}{K+1}\sum_{k=0}^{K}\left\|trans(\mathbf{x}^{\text{aligned}}_{k}) - trans(\mathbf{x}^{\text{gt}}_{k})\right\|^{2}\right)^{1/2}, \tag{2.51}$$

where $trans(\cdot)$ is a function that returns the translation part of a pose, i.e. the position.

In the RPE metric, the trajectories need not be aligned and only the differences between poses are measured. The error metric is then calculated by estimating the relative pose with a fixed time difference Δ

$$APE = \left(\frac{1}{K+1-\Delta}\sum_{k=\Delta}^{K}\left\|trans(\left(\mathbf{x}^{\text{est}}_{k-\Delta}\right)^{-1}\mathbf{x}^{\text{est}}_{k}) - trans(\left(\mathbf{x}^{\text{gt}}_{k-\Delta}\right)^{-1}\mathbf{x}^{\text{gt}}_{k})\right\|^{2}\right)^{1/2}. \tag{2.52}$$

This metric is designed to evaluate the local consistency of the trajectory. Therefore, the behavior of this metric is particularly useful to evaluate odometry algorithms that perform no loop closures.

Metrics to evaluate the map estimate are much more rarely used in robotics. This is because most SLAM algorithms use landmarks that are non-semantic. Therefore obtaining a ground truth map is impossible for these kinds of algorithms. Furthermore, algorithms that don't use the exact same landmark type and algorithms that do not use landmarks at all cannot be compared using these metrics. However, when a ground truth map exists, metrics coming from the target tracking literature can be used, such as optimal sub-pattern assignment (OSPA) [33]. Schuhmacher, Vo, and Vo [33] introduced the OSPA metric to measure the difference between two sets $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$, of size m and $\mathcal{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$, of size n, the OSPA metric with distance parameter c and order p is defined as

$$OSPA(\mathcal{X}, \mathcal{Y}, c, d) = \left(\frac{1}{n} \min_{\theta} \sum_{i=1}^{n} d^{(c)}(\boldsymbol{x}_{\theta(i)}, \boldsymbol{y}_i)^p + (n - m) \right)^{1/p} , \tag{2.53}$$

if $n \geq m$ and $OSPA(\mathcal{X}, \mathcal{Y}, c, d) = OSPA(\mathcal{Y}, \mathcal{X}, c, d)$ otherwise, and $d^{(c)}(\boldsymbol{x}, \boldsymbol{y}) = \min(c, d(\boldsymbol{x}, \boldsymbol{y}))$.

2.4 Non-linear Optimization

2.4.1 General unconstrained optimization

Particle Swarm Optimization

One method to maximize a non-linear function, as needed by any Maximum Likelihood solution to SLAM, is Particle Swarm Optimization (PSO) [34]. Due to the heuristic nature of PSO, many variations exist. An explanation of one of the most common variations, known as Standard PSO 2007 (SPSO-2007), and used in this thesis, follows.

Let the optimization problem to solve be

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) , \tag{2.54}$$

where $\boldsymbol{x}$ is the variable to optimize with respect to function $f(\cdot)$. In PSO a set of particles with positions $\boldsymbol{x}_j^i$ and velocities $\boldsymbol{v}_j^i$ is defined with the expectation that, as the particles move according to their velocities

$$\boldsymbol{x}_{j+1}^i = \boldsymbol{x}_j^i + \boldsymbol{v}_j^i , \tag{2.55}$$

the positions of the particles will converge to the optimal solution $\boldsymbol{x}^*$. SPSO-2007 uses a fixed number of particles n_p according to

$$n_p = \lfloor 10 + 2\sqrt{D} \rfloor , \tag{2.56}$$

where D is the number of dimensions of $\boldsymbol{x}$. This formula considers a suggested particle swarm size but is not required for SPSO-2007 compatibility. Particle positions and velocities are initialized randomly following the uniform distribution within the search space $[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$

$$\boldsymbol{x}_0^i = U(\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}) \tag{2.57}$$

$$\boldsymbol{v}_0^i = 0.5(U(\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}) - \boldsymbol{x}_0^i) . \tag{2.58}$$

Where $U(a, b)$ is the uniform distribution in the interval $[a, b]$. Then each particle will calculate its new velocity as a function of its own position and current velocity and the positions of the particles within a neighborhood - i.e. for each component of particle i at iteration j, the velocity will be

$$\boldsymbol{v}_j^i = w\boldsymbol{v}_j^i + r_1\phi_p(\boldsymbol{l}_j^i - \boldsymbol{x}_j^i) + r_2\phi_g(\boldsymbol{g}_j^i - \boldsymbol{x}_j^i) \ , \tag{2.59}$$

where r_1 and r_2 are random numbers uniformly distributed in the interval $[0, 1]$, $\boldsymbol{l}_j^i$ is the best (lowest $f(\cdot)$) position visited by particle i and $\boldsymbol{g}_j^i$ is the best position visited by any of the particles in particle i's neighborhood. w, ϕ_p and ϕ_g are parameters typically set to the values[2]:

$$\begin{aligned} w &= \frac{1}{2\ln(2)} \simeq 0.721 \\ \phi_p &= \phi_g = 0.5 + \ln(2) \simeq 1.193 \ . \end{aligned} \tag{2.60}$$

This neighborhood is defined randomly by having each particle inform K other particles at random (i.e. adding themselves to the other particles' neighborhoods). Typically $K = 3$. If at any iteration the best solution found does not improve, then all neighborhoods are randomly redrawn using the same process [34].

2.4.2 Non-linear Least Squares

In the non-linear least squares formulation, the function to be minimized has the form

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) = (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}))^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})) \ , \tag{2.61}$$

where $\boldsymbol{h}(\boldsymbol{x})$ is a general non-linear function, $\boldsymbol{z}$ is a vector, typically containing the measurement values, which does not depend on the variable $\boldsymbol{x}$. $\boldsymbol{\Omega}$ is a weighting matrix that weights the different components of the error. Typically this is the measurement information matrix which is usually diagonal.

Based this form, a family of algorithms has been developed to efficiently find the least squared error solution. The most common ones are gradient descent, the Gauss-Newton method, and the Levenberg-Marquardt algorithm [35].

In all three algorithms, we start with an initial guess at the optimum value of $\boldsymbol{x}$, $\boldsymbol{x}_0$, and then we iteratively approximate the solution with estimates $\boldsymbol{x}_i$ which should converge to the optimum value of $\boldsymbol{x}$.

[2]As stated in [34], these parameters have been shown empirically to produce good results.

Gradient descent updates the current estimate using the gradient at the location

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha \frac{\partial f}{\partial \boldsymbol{x}}^{\intercal} (\boldsymbol{x}_i) \ , \tag{2.62}$$

with α being a scalar parameter. Calculating the gradient of $f(\boldsymbol{x})$ at the current approximation gives

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_i) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}))^{\top} \boldsymbol{\Omega}^{-1} (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})) \ , \tag{2.63}$$

$$\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_i) = -2(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}_i))^{\top} \boldsymbol{\Omega}^{-1} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\boldsymbol{x}_i) \ . \tag{2.64}$$

yielding the update formula for gradient descent

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha 2 J \boldsymbol{\Omega}^{-1} (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}_i)) \ , \tag{2.65}$$

where the gradient or Jacobian was given as $J = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\boldsymbol{x}_i)$ for brevity.

The Gauss-Newton method is an improvement on this, which assumes that the objective function can be approximated in the vicinity of $\boldsymbol{x}_i$ by a quadratic function as follows.

$$\boldsymbol{h}(\boldsymbol{x}_i + \Delta \boldsymbol{x}) \approx \boldsymbol{h}(\boldsymbol{x}_i) + J \Delta \boldsymbol{x} \ . \tag{2.66}$$

Using this approximation the solution to the optimization problem becomes a linear equation, resulting in the Gauss-Newton method

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta \boldsymbol{x}_i \ , \tag{2.67}$$

with $\Delta \boldsymbol{x}_i$ being calculated by solving the linear problem

$$J^{\intercal} \boldsymbol{\Omega}^{-1} J \Delta \boldsymbol{x}_i = J \boldsymbol{\Omega}^{-1} (z - \boldsymbol{h}(\boldsymbol{x}_i))) \ . \tag{2.68}$$

The previous two algorithms have the problem that their performances can vary significantly depending on the initialization point $\boldsymbol{x}_0$. The Levenberg-Marquardt algorithm improves on both these algorithms using the following intermediate solution.

$$(J^{\intercal} \boldsymbol{\Omega}^{-1} J \Delta + \lambda I) \boldsymbol{x}_i = J \boldsymbol{\Omega}^{-1} (z - \boldsymbol{h}(\boldsymbol{x}_i))) \ , \tag{2.69}$$

where λ is an adaptive parameter. It can be inferred from equation (2.69), that when λ is in the vicinity of zero, the update is equivalent to the Gauss-Newton algorithm and when λ is very high the update is equivalent to gradient descent. Usually, λ is initialized with a high value, and if the value after the update increases (i.e. we are moving away from the optimum) then λ is increased, else as the approximation gets closer and closer to the optimum, λ is lowered accelerating the convergence with the Gauss-Newton update step. The Levenberg-Marquardt algorithm is widely used in many least squares problems, including in SLAM batch solvers such as g2o[21] and GTSAM[36], both of which are used in this thesis.

Chapter 3

Detection Statistics in SLAM

3.1 Introduction

In this Chapter, the importance of detection statistics in both vector and RFS-based SLAM is summarized. Additionally, a formulation of the RFS-based batch estimation approach to SLAM is presented.

3.2 Filtering-Based SLAM Techniques & Detection Statistics

Most vector-based solutions to SLAM which adopt features require a map management routine to manage the addition and removal of features from the map estimate. Such methods vary in mathematical rigor. While some are heuristic in nature, others use a binary Bayes filter [18, 37].

In the binary Bayes filter the probability of existence $P([\mathbf{m}^i]_{\mathrm{E}} | \boldsymbol{x}_{0:k}, \mathcal{Z}_{0:k})$ of the i-th map feature vector, $\boldsymbol{m}^i$, given the history of robot poses from discrete-time 0 to k, $\mathbf{x}_{0:k}$, and all feature measurement sets, $\mathcal{Z}_{0:k}$, is updated at each step. This is achieved using the probabilistic evidence provided by the current measurement set $\mathcal{Z}_k$, and assumed or known data association:

$$P(\left[\mathbf{m}^i\right]_{\mathrm{E}} | \mathbf{x}_{0:k}, \mathcal{Z}_{0:k}) = \frac{p(\mathcal{Z}_k | \mathbf{x}_k, [\mathbf{m}^i]_{\mathrm{E}}) P([\mathbf{m}^i]_{\mathrm{E}} | \mathbf{x}_{0:k-1}, \mathcal{Z}_{0:k-1})}{p(\mathcal{Z}_k | \mathcal{Z}_{0:k-1})}, \tag{3.1}$$

we can write the probability of non-existence of $\mathbf{m}^i$, $P([\mathbf{m}^i]_{\bar{\mathrm{E}}} | \boldsymbol{x}_{0:k}, \mathcal{Z}_{0:k})$ as

$$P(\left[\mathbf{m}^i\right]_{\bar{\mathrm{E}}} | \mathbf{x}_{0:k}, \mathcal{Z}_{0:k}) = \frac{p(\mathcal{Z}_k | \mathbf{x}_k, [\mathbf{m}^i]_{\bar{\mathrm{E}}}) P([\mathbf{m}^i]_{\bar{\mathrm{E}}} | \mathbf{x}_{0:k-1}, \mathcal{Z}_{0:k-1})}{p(\mathcal{Z}_k | \mathcal{Z}_{0:k-1})} . \tag{3.2}$$

Noticing that dividing the previous equations cancels out the $p(\mathcal{Z}_k | \mathcal{Z}_{0:k-1})$, we can express (3.1) in log-odds form as follows,

$$l_k(\mathbf{m}^i) = l_{k-1}(\mathbf{m}^i) + \log \frac{P([\mathbf{m}^i]_{\mathrm{E}}\,|\boldsymbol{x}_k, \mathcal{Z}_k)}{1 - P([\mathbf{m}^i]_{\mathrm{E}}\,|\boldsymbol{x}_k, \mathcal{Z}_k)} - \log \frac{P([\mathbf{m}^i]_{\mathrm{E}})}{1 - P([\mathbf{m}^i]_{\mathrm{E}})} \ , \tag{3.3}$$

where the log-odds $l_k(\boldsymbol{m}^i)$ are defined as:

$$l_k(\boldsymbol{m}^i) \equiv \log \frac{P([\mathbf{m}^i]_{\mathrm{E}}\,|\mathbf{x}_{0:k}, \mathcal{Z}_{0:k})}{1 - P([\mathbf{m}^i]_{\mathrm{E}}\,|\mathbf{x}_{0:k}, \mathcal{Z}_{0:k})} \ . \tag{3.4}$$

By setting the prior probability of existence $P([\mathbf{m}^i]_{\mathrm{E}})$ to a non-informative prior (0.5), the last term in Equation (3.3) is eliminated. Then, incorporating modeled or known probabilities of detection and false alarm, the probabilistic evidence provided by the measurements can be calculated as shown in [38]

$$P(\left[\mathbf{m}^i\right]_{\mathrm{E}}|\boldsymbol{x}_k, \mathcal{Z}_k) = \frac{(1 - P_{\mathrm{D}}(\mathbf{m}^i))P_{\mathrm{FA}}P([\mathbf{m}^i]_{\mathrm{E}}) + P_{\mathrm{D}}(\mathbf{m}^i)P([\mathbf{m}^i]_{\mathrm{E}})}{P_{\mathrm{FA}} + (1 - P_{\mathrm{FA}})P_{\mathrm{D}}(\mathbf{m}^i)P([\mathbf{m}^i]_{\mathrm{E}})} \ , \tag{3.5}$$

when $\mathbf{m}^i$ is associated to a measurement in $\mathcal{Z}_k$, and as

$$P(\left[\mathbf{m}^i\right]_{\mathrm{E}}|\boldsymbol{x}_k, \mathcal{Z}_k) = \frac{(1 - P_{\mathrm{D}}(\mathbf{m}^i))P([\mathbf{m}^i]_{\mathrm{E}})}{(1 - P([\mathbf{m}^i]_{\mathrm{E}})) + (1 - P_{\mathrm{D}}(\mathbf{m}^i))P([\mathbf{m}^i]_{\mathrm{E}})} \ , \tag{3.6}$$

when $\boldsymbol{m}^i$ is unassociated. $P_{\mathrm{D}}(\mathbf{m}^i)$ is the probability of detection of feature $\mathbf{m}^i$, P_{FA} is the probability of a measurement being a false alarm [38]. Both quantities are constants that must be substituted into (3.5) and (3.6) and then into (3.3) when an association or a non-association occurs. From Equation (3.3) it can be seen that a simple measurement counting heuristic can be interpreted as a log-odds binary Bayes filter with an implicitly assumed probability of detection and false alarm. Prior research which has adopted this approach includes [18].

3.2.1 Evaluating SLAM performance

Any SLAM solution will be composed of an estimated trajectory and estimated map. Metrics for evaluating the performance of this results have been mainly focused on the trajectory. This is because of the ease of measuring the trajectory by external means, producing a ground truth angainst which the estimated trajectory can be compared to. Metrics to evaluate the trajectory error include the

3.2.2 RFS SLAM and the Importance of Detection Statistics

Mullane et al. [28] introduced the concept of RFSs into the SLAM problem. By recognizing that the SLAM state can be naturally represented by a random set, they were able to include the data association and map management problems into the Bayesian estimation paradigm. Since an RFS implementation of SLAM will be used as the primary demonstration of the importance of determining the detection statistics in this thesis, an overview of Rao Blackwellized (RB)-PHD-SLAM now follows.

Using the Rao-Blackwellized Particle Filter as in FastSLAM, the SLAM posterior PDF (Equation (1.9)) can be factored into the form [17, 39]

$$p\left(\boldsymbol{x}_{0:k}\middle|\mathcal{Z}_k, \boldsymbol{u}_{0:k}\right) p\left(\mathcal{M}_k\middle|\boldsymbol{x}_{0:k}, \mathcal{Z}_k, \boldsymbol{u}_{0:k}\right) \ , \tag{3.7}$$

such that the first term in (3.7) is a PDF on the robot's trajectory and can be sampled using particles. The second term in (3.7) is the PDF of the map conditioned on the robot's trajectory. In the RFS-based approach, if the map RFS is assumed to follow a multi-object Poisson distribution, such that features are independent and identically distributed (IID) as[1]

$$p\left(\mathcal{M}_k = \{\boldsymbol{m}_k^1, \boldsymbol{m}_k^2, ..., \boldsymbol{m}_k^{\mathfrak{m}_k}\}\middle| \ |\mathcal{M}_k| = \mathfrak{m}_k\right) = \mathfrak{m}_k! \prod_{i=1}^{\mathfrak{m}_k} p_{\boldsymbol{m}}(\boldsymbol{m}_i) \ , \tag{3.8}$$

where $p_{\boldsymbol{m}}(\cdot)$ is the spatial distribution for the features in the map, then the PDF of the map RFS can be fully represented by its PHD, also referred to as an intensity function, v_k:

$$v_k = v_k(\boldsymbol{m}) = \lambda p_{\boldsymbol{m}}(\boldsymbol{m}). \tag{3.9}$$

The number of features is assumed Poisson distributed with parameter λ according to

$$p\left(|\mathcal{M}_k| = \mathfrak{m}_k\middle|\lambda\right) = \frac{\lambda^{\mathfrak{m}_k}\exp(-\lambda)}{\mathfrak{m}_k!} \ . \tag{3.10}$$

These assumptions allow the PDF of the map RFS to be approximated by a time-varying PHD. The map PDF is then approximated as

$$p(\mathcal{M}_k = \{\boldsymbol{m}_k^1, \boldsymbol{m}_k^2 \ ... \ \boldsymbol{m}_k^{\mathfrak{m}_k}\}) = \frac{\prod_{i=1}^{\mathfrak{m}_k} v_k(\boldsymbol{m}_k^i)}{\exp(\int v_k(\boldsymbol{m})d\boldsymbol{m})} \ . \tag{3.11}$$

In contrast to vector-based RB-PF approaches, which typically use the EKF to update the Gaussians for individual landmarks, a PHD filter is used to update the map intensity function

[1]Note that the $\mathfrak{m}_k!$ term is necessary since a set includes all possible permutations of its elements.

in RB-PHD-SLAM [39]. An overview of the main steps in the RB-PHD-SLAM filter now follows, highlighting the importance of detection statistics.

Particle Propagation

At time-step k, the particles representing the prior distribution,

$$\boldsymbol{x}_{k-1}^{[i]} \sim p\left(\boldsymbol{x}_{0:k-1} \middle| \mathcal{Z}_{1:k-1}, \boldsymbol{u}_{0:k-1}\right) \tag{3.12}$$

are propagated forward in time by sampling the motion noise, $\boldsymbol{\delta}_k^{[i]}$, and using the robot motion model (1.1):

$$\boldsymbol{x}_k^{[i]} \sim \boldsymbol{g}(\boldsymbol{x}_{k-1}^{[i]}, \boldsymbol{u}_{k-1}, \boldsymbol{\delta}_{k-1}^{[i]}) \longrightarrow p\left(\boldsymbol{x}_{0:k} \middle| \mathcal{Z}_{1:k-1}, \boldsymbol{u}_{0:k-1}\right) \ . \tag{3.13}$$

This step is common to vector-based Rao-Blackwellized solutions to SLAM, such as FastSLAM [17].

Prediction

For each particle, its map intensity from the previous update, $v_{k-1}^{+}(\boldsymbol{m})$, is augmented with an arbitrarily small *birth* intensity v_k^b, according to the PHD filter predictor equation:

$$v_k^{-}(\boldsymbol{m}) = v_{k-1}^{+}(\boldsymbol{m}) + v_k^b(\boldsymbol{m}) \tag{3.14}$$

The birth intensity $v_k^b(\boldsymbol{m})$ represents the number of new features that might appear at $\boldsymbol{m}$ and is usually heuristically determined. This intensity is required to model the appearance of new features and is similar to the proposal distribution concept in FastSLAM.

Map Update

The map intensity for each particle is updated with the latest measurements according to the PHD filter update equation

$$\begin{aligned} v_k^{+}(\boldsymbol{m}) = {} & v_k^{-}(\boldsymbol{m})(1 - P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k)) + \\ & v_k^{-}(\boldsymbol{m}) \sum_{i}^{|\mathcal{Z}_k|} \frac{P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k)\boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m}, \boldsymbol{x}_k)}{\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) + \int P_{\mathrm{D}}(\boldsymbol{m}'|\boldsymbol{x}_k)\boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m}', \boldsymbol{x}_k)v_k^{-}(\boldsymbol{m}')d\boldsymbol{m}'} \ , \end{aligned} \tag{3.15}$$

where $\boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m}, \boldsymbol{x}_k)$ is the i-th measurement's/detected feature's spatial likelihood and $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ is the intensity of the clutter RFS at time k. The first term in (3.15) is a copy of $v_k^{-}(\boldsymbol{m})$

reduced by the factor $(1 - P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k))$ to account for the possibility that the predicted features are undetected. In the second term, note that instead of determining data association based on an external algorithm, the PHD filter determines how much the landmark estimate is influenced by *each* and *every* measurement.

Importance Weighting and Re-sampling

The weighting and re-sampling of particles is the method used to update the robot trajectory PDF after propagation (also known as the proposal distribution). This is given by

$$p\left(\boldsymbol{x}_{0:k}\middle|\mathcal{Z}_{1:k-1}, \boldsymbol{u}_{0:k-1}\right) . \tag{3.16}$$

This has to be updated to become a new PDF representing the robot trajectory after measurement updates (or the target distribution),

$$p\left(\boldsymbol{x}_{0:k}\middle|\mathcal{Z}_{1:k}, \boldsymbol{u}_{0:k-1}\right) . \tag{3.17}$$

Bayes rule allows the weighting distribution to be expressed in terms of (3.16) and (3.17):

$$\frac{p\left(\boldsymbol{x}_{0:k}\middle|\mathcal{Z}_{1:k-1}, \boldsymbol{u}_{0:k-1}\right)}{p\left(\boldsymbol{x}_{0:k}\middle|\mathcal{Z}_{1:k}, \boldsymbol{u}_{0:k-1}\right)} = \theta p\left(\mathcal{Z}_k\middle|\boldsymbol{x}_{0:k}, \mathcal{Z}_{1:k-1}\right) , \tag{3.18}$$

in which θ is a normalizing constant. Since (3.16) and (3.17) are sampled using particles, the weighting distribution, defined as w_k, is also sampled such that a weight is calculated for each particle. To solve (3.18), Bayes theorem gives

$$\begin{aligned} w_k &\equiv p\left(\mathcal{Z}_k\middle|\boldsymbol{x}_{0:k}, \mathcal{Z}_{1:k-1}\right) \\ &= p\left(\mathcal{Z}_k\middle|\mathcal{M}_k, \boldsymbol{x}_{0:k}\right) \frac{p\left(\mathcal{M}_k\middle|\mathcal{Z}_{1:k-1}, \boldsymbol{x}_{0:k}\right)}{p\left(\mathcal{M}_k\middle|\mathcal{Z}_{1:k}, \boldsymbol{x}_{0:k}\right)} . \end{aligned} \tag{3.19}$$

Equation (3.19) can be solved because the map RFS is assumed to be multi-object, Poisson distributed. Note from (3.19) that the choice of the map, $\mathcal{M}_k$, for which the expression is evaluated in its general form is theoretically arbitrary since the right-hand side of the first line of (3.19) is independent of the map. This has led to multiple solutions that adopt the empty-set strategy, the single-feature strategy and multi-feature strategy in determining the particle weight w_k in (3.19) [38, 40]. Although w_k is theoretically independent of the map, because of the approximations involved in the PHD Filter, it has been shown that the choice of the

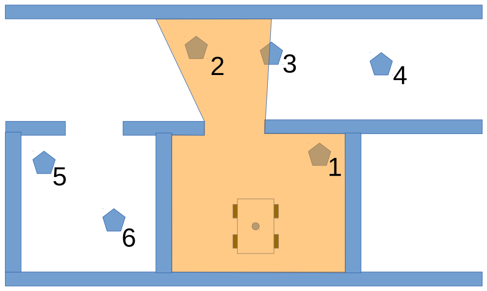


Figure 3.1: *A hypothetical pentagon detector, occluded by walls. The yellow area shows a zone where $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ is high and in the white zones, $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k) = 0$. From a geometric perspective, objects 1 and 2 should have high values of $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$, while object 3 should have a significantly reduced $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ due to its partial occlusion by the wall. Objects 4, 5 and 6 would be expected to have $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k) = 0$.*

map can have a significant effect on the performance of the filter and that the multi-feature strategy is superior to the others [38]. This is achieved at the cost of an increased computational complexity. The multi-feature strategy is adopted in this work. In [39, 41] the implementation of RB-PHD-SLAM Equations (3.13), (3.14) and (3.15) using Gaussian mixtures are shown.

Importantly, within the above four steps, the map update and particle weighting steps require the knowledge of both the probability of detection, $P_{\mathrm{D}}(\boldsymbol{m}|\boldsymbol{x}_k)$, of the feature detector and the distribution (defined by its first moment or PHD) of its false alarms, $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$.

3.2.3 How do Detection Statistics Affect SLAM?

In SLAM, the map feature probability of detection $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ (in (3.5) and (3.15)), the probability of false alarm $P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k)$ (in (3.5)) and the false alarm intensity $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ (in (3.15)), depend on the state of the robot and environment - i.e.

$$P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k) \longleftarrow P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k, \Gamma) \ , \tag{3.20}$$

$$P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k) \longleftarrow P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k, \Gamma) \ , \tag{3.21}$$

$$\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) \longleftarrow \kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k, \Gamma) \ . \tag{3.22}$$

Here, Γ represents the *total* state of the environment including, but not limited to, the subset of it being estimated (i.e., the vicinity of the feature locations) [37]. For example, if the sensor being used is affected by occlusions, then it may be the case that not all objects that can cause the occlusions are represented in the SLAM map $\mathcal{M}_k$. Therefore, theoretically, the total environmental state Γ, which includes these objects, would be necessary. As an example, Fig. 3.1 shows a hypothetical shape (pentagon) detector, which can be occluded by walls. The value of $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ associated with each pentagon is therefore highly dependent on the state of both the robot and the environment, in this case, the quantity and relative location of the walls. Even though the walls may not constitute map features of interest, they still affect the probability of detection of the pentagons. The spatial distribution of false alarms is complex to model since the source of false alarms can vary depending on the environment. Given this lack of knowledge about the false alarm distribution, approximating it by the least informative distribution, i.e. the uniform distribution is usually accepted as a reasonable assumption [42]. The expected number of false alarms can be estimated by utilizing a feature detector in an environment with a known number and location of features and analyzing the number of false alarms, or it can be left as a parameter of the algorithm to tune.

3.3 RFS-based batch estimation

SLAM solutions using the batch estimation approach are usually maximum likelihood strategies in which the likelihood function,

$$p\left(\mathcal{Z}_{1:k}, \boldsymbol{u}_{0:k-1} \middle| \boldsymbol{x}_{0:k}, \mathcal{M}\right) \ , \tag{3.23}$$

is maximized over all possible trajectories, $\boldsymbol{x}_{0:k}$, and maps, $\mathcal{M}$. Assuming that the number of landmarks is known, and using a known data association hypothesis, this likelihood can be expressed as:

$$\prod_{i=1}^{k} \boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}) \prod_{j}^{|\mathcal{Z}_i|} \boldsymbol{h}(\boldsymbol{z}_i^j|\boldsymbol{x}_i, \boldsymbol{m}^{a_i(j)}) \ , \tag{3.24}$$

where the function $a_i(j)$ represents the data association of measurement $\boldsymbol{z}_i^j$.

The objective function of (3.23) can be formulated using the RFS framework, by using the set-based measurement and motion models:

$$\prod_{i=1}^{k} \boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1}) p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}\right) \ . \tag{3.25}$$

As can be seen from (3.25), the likelihood is very similar to (3.24) but uses the RFS-based measurement model $p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_i, \mathcal{M}\right)$:

$$\begin{aligned} p\left(\mathcal{Z}_i \middle| \mathbf{x}_i, \mathcal{M}\right) &= p\left(\mathcal{Z}_i \middle| \mathbf{x}_i, \{\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^{|\mathcal{M}|}\}\right) \\ &= \sum_{\theta} p_{\kappa}\left(\mathcal{Z}_i^{\bar{A}_\theta}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_\theta}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j|\boldsymbol{x}_i\right)\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{A_\theta}} P_{\mathrm{D}}\left(\mathbf{m}^j|\boldsymbol{x}_i\right) p\left(\boldsymbol{z}_i^{a_\theta(j)} \middle| \mathbf{m}^j, \mathbf{x}_i\right) \end{aligned} \tag{3.26}$$

where θ is a possible association between the elements of $\mathcal{M}$ and $\mathcal{Z}_i$, $\mathcal{Z}_i^{\bar{A}_\theta}$ is the set of measurements in $\mathcal{Z}_i$ that is not associated with a landmark in the map according to θ. Similarly, $\mathcal{M}^{A_\theta}$ is the set of landmarks in $\mathcal{M}$ that are associated with a measurement in $\mathcal{Z}_i$, again according to θ. $p_\kappa\left(\mathcal{Z}_i^{\bar{A}_\theta}\right)$ is the probability of all measurements in $\mathcal{Z}_i^{\bar{A}_\theta}$ being clutter.

Now applying log() to (3.25) gives

$$\begin{aligned}
\log\left(p\left(\mathcal{Z}_{1:k}, \boldsymbol{u}_{0:k-1}\middle|\boldsymbol{x}_{0:k}, \mathcal{M}\right)\right) &= \sum_{i=1}^{k} \log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})\right) + \log\left(p\left(\mathcal{Z}_i\middle|\boldsymbol{x}_k, \mathcal{M}\right)\right) && (3.27)\\
&= \sum_{i=1}^{k} \log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})\right) + \log\left(\sum_{\theta} p\left(\mathcal{Z}_i\middle|\boldsymbol{x}_k, \mathcal{M}, \theta\right)\right) . && (3.28)
\end{aligned}$$

In [29], (3.26) is derived and shown to generalize its vector counterpart.

Using the RFS-based objective function from (3.26), the non-linear optimization can be performed over all possible map sizes $|\mathcal{M}|$. This function would also not require a data association hypothesis to perform the optimization. In Chapter 5 the optimization of this objective function is carried out with general optimization methods and evaluate the performance compared is compared to to state-of-the-art vector-based SLAM batch estimation algorithm, which have access to the ground truth data association. The partly integer nature of the optimization (i.e. the number of features in the map) will cause significant difficulties for the optimization method. Chapter 6 then uses a new joint vector-set distribution to find a Maximum a posteriori solution to SLAM, with much better results, comparable to state-of-the-art solutions to SLAM.

Chapter 4

Detection Statistics in Filter-Based SLAM

4.1 Introduction

In the field of target tracking, detection statistics are considered to be of prime importance. For example, it has long been recognized that a sensor's received signal amplitudes, corresponding to true targets, should be higher than those corresponding to false alarms, and that this information should be utilized [43]. However, this requires the Signal-to-Noise Ratio (SNR) corresponding to targets to be known from any sensor to target viewpoint. Since such information is typically unavailable, detection probabilities are usually naively considered to be constant (but not necessarily zero or unity), despite the fact that the relative positions of objects with respect to the sensor, and occlusions, have a large effect on those objects' detection probabilities [43].

Within the autonomous robotic, feature-based navigation literature, a vehicle's onboard sensors are used to obtain exteroceptive measurements. Measurements and feature estimate uncertainties are typically considered to lie solely in the spatial, rather than the detection, domain and often modeled with range and bearing uncertainties [37, 44]. The joint estimation of feature locations, and the trajectory of a robotic vehicle, which obtains measurements from these features, within a common coordinate frame is referred to as feature-based SLAM. In most SLAM algorithms it is considered the task of external map management, outlier rejection, and data association algorithms to minimize the problems of false alarms and missed detections, before map estimation takes place, so that mathematically, the probabilities of detection of features that have been associated are assumed to be unity and the probabilities that the associated measurements are false alarms are assumed to be zero. Similarly, the probabilities of detection of unassociated features are assumed to be zero, while the probabilities of false alarm of unassociated measurements are assumed to be unity [29].

In SLAM, the probability of detection is particularly important, since features that exit the field of view (FoV) of the sensor(s) are expected to remain in the map estimate, contrary to target tracking problems, in which the maintenance of tracks for targets that exit the FoV is usually not required. However, principled methods which currently provide such statistics calculate them based upon the *measurements* themselves, such as Constant False Alarm Rate (CFAR) processors [45]. In both the binary Bayes Filter and RFS-SLAM, feature state-based detection statistics are required both in the presence and absence of corresponding measurements, since the existence probabilities of feature states must be updated in both cases. Further, in RFS-SLAM, the detection statistics required are state-dependent, and there are no feature to measurement association decisions. Therefore, *measurement*-based detection statistics, such as those provided by CFAR processors, cannot be directly applied. This chapter, therefore, addresses methods to estimate these statistics on a per estimated feature basis. This takes into account feature descriptor information, which is not part of the spatial location of the measurement, to aid both in map-management and data association, within vector-based SLAM approaches, and for direct use within RFS-SLAM methods.

To apply this concept to spatial-based features, this work proposes the use of a feature descriptor, based on the estimated *number* of unoccluded range points sensed from that feature. This feature descriptor depends on the current best estimate of the SLAM state and the current range-based sensor scan, but not any detected features from that scan. This avoids the necessity for data association between detected and currently estimated features when applying the descriptor. In contrast to standard feature-based SLAM methods, which discard any data which has not contributed to a feature detection, a sub-set of this remaining data provides the extra information necessary to estimate the probability of detection of features, even when partially occluded. It will be demonstrated that this descriptor can provide an approximate sufficient statistic to define a SLAM state-dependent distribution on the probability of detection of a feature. Results will demonstrate superior SLAM performances than those obtained from the equivalent algorithms, with the usually assumed constant feature probabilities of detection within the sensor's FoV.

The work in this chapter was partially published in [46], the author's Master's degree thesis [10], and [47]. The work in [46] and the Master's thesis [10] shows the determination of the detection statistics for the detector used in a local park environment. Meanwhile, [47] shows via simulation, that the incorporation of a descriptor into RFS-based SLAM can be advantageous. This chapter extends [46, 47] by providing more detailed simulation analyses, demonstrating the applicability of the descriptor to standard detection concepts based on the random sample consensus (RANSAC) algorithm [48] and also by generating detection statistics for tree detection in a local park environment (Santiago, Chile), and successfully applying it to the publicly available Victoria park dataset, from Sydney, Australia.

This chapter has two main contributions. The first shows that including detection statistics into SLAM solutions can be advantageous. The second is the introduction of a descriptor, namely the number of points used by the detector, whose prediction can be used to calculate the detection statistics for features extracted from range sensor data, particularly lidar data. The contributions of this chapter are based on the derivation of a range-based feature descriptor. An algorithm capable of estimating the probability of detection for any range-based detector, based on the expected value of the descriptor, is then presented. This is followed by an evaluation of the effects of including both the probability of detection and the descriptor information into set-based RB-PHD-SLAM, and the conceptually similar vector-based SLAM algorithm MH-FastSLAM. Each framework is compared, both with and without the use of the described descriptor.

4.2 Estimating Feature Detection Statistics Based on Range Data

4.2.1 Estimating Probability of Detection $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$

This section provides a quantified model of the probability of detection of features, based on a descriptor related to range/bearing data for use in any robotic navigation formulation. Although the descriptor can be inspired by the chosen detector algorithm, a more general solution, which is independent of the detector used to extract those features, and which takes into account occlusions, is presented here. Therefore, the chosen descriptor model does not use any information about the feature detector itself and can be used with any detector that estimates both the position and shape attributes of a feature, such as RANSAC [48].

As shown in Fig. 4.1, given the robot's location, $\boldsymbol{x}_k$, and the location and other attributes, such as the shape, of features, $\boldsymbol{m}_k^j$, (i.e. a SLAM state), the number of range points that the feature is predicted to return (red and green points in the figure), can be calculated via ray casting [37]. These predicted range values are then compared with the actual range values from the sensor (black points). If the actual range values at particular bearing angles are considerably lower than predicted, then the predicted range points at those bearing angles are labelled as occluded (red points) and the number of remaining, unoccluded (green) points is defined as $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ will be examined as an approximate sufficient statistic, which determines

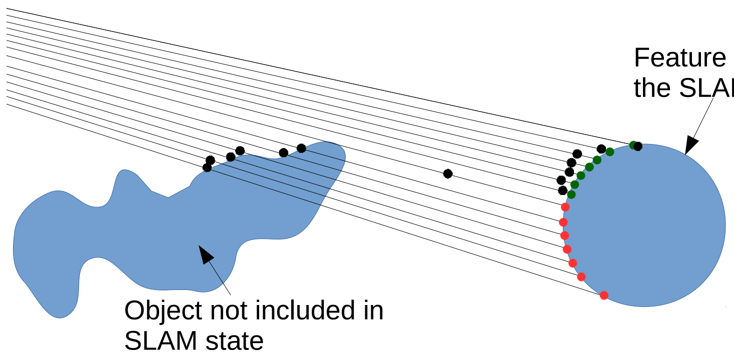


Figure 4.1: *Analysis of range data from a circular shaped feature. The number of predicted (green) points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, which are unoccluded, is used as a sufficient statistic of the probability of detection $P_D(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ of the feature.*

the probability of detection of the feature.[1] Since $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ is calculated using ray casting, its effectiveness will depend on the validity of using ray casting to predict the measurements of the range sensor. In its current form, this should apply to any narrow beam range sensor, being best suited to lidar sensors.

It should be noted that $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ is expressed only as a function of the SLAM state ($\boldsymbol{x}_k$ and $\boldsymbol{m}_k^j$). This is despite the fact that the current range scan is necessary to determine which, if any, of the predicted points determined via ray casting are occluded by any objects not included within the SLAM map state $\mathcal{M}_k$. Although the current scan is necessary to determine this, the detected features themselves $\mathcal{Z}_k$ are not required, meaning that no data association between detections $\boldsymbol{z}_k^i \in \mathcal{Z}_k$ and $\hat{\boldsymbol{m}}_k^j \in \widehat{\mathcal{M}}_k$, where $\widehat{\mathcal{M}}_k$ is the estimated map at time k, is necessary. In fact, it is a sub-set of the range values from each scan, which are not necessarily used by the SLAM feature state detector, which are used to determine $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. In contrast to standard feature-based SLAM methods, which discard such data within each scan, in this work, this data provides the extra information necessary to estimate the probability of detection of features, even when partially occluded. Therefore, this chapter analyses the feasibility of assuming $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ to be a sufficient statistic of $P_\text{D}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ - i.e. It poses the question of whether

$$P_\text{D}(\boldsymbol{m}_k^j|\boldsymbol{x}_k, \Gamma, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) \approx P_\text{D}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)). \tag{4.1}$$

[1]Note that $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ can be theoretically calculated for any general values of $\boldsymbol{m}_k^j$ and $\boldsymbol{x}_k$. For implementation purposes however, if the SLAM estimate is modelled in vector form, using Normal distributions to model the robot and feature states (e.g. standard EKF SLAM), then $\boldsymbol{x}_k$ and $\boldsymbol{m}_k^j$ would be replaced with $\hat{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{m}}_k^j$ representing the estimated means of the Normal distributions representing the vehicle and feature states respectively. Alternatively, if an RB solution to SLAM is used, in which a Gaussian model of the features is assumed (e.g. FastSLAM or RB-GM-PHD-SLAM), then $\boldsymbol{x}_k$ would be replaced with the state of the i-th particle $\boldsymbol{x}_k^{[i]}$ under consideration, and again $\boldsymbol{m}_k^j$ would be replaced with $\hat{\boldsymbol{m}}_k^j$.

It should also be noted that, the analysis presented here could be applied in tracking approaches such as [42], allowing the inclusion of unmodelled objects within the calculation of the detection probabilities. In particular, $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ could be used to estimate the mean of the Poisson RFS used in [42], representing the number of measurements per extended feature.

As carried out in [49] the dependency of $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ on $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ could be calculated by integrating the variables used by the detector over the detectable area:

$$P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \int_{\boldsymbol{\theta} \in \boldsymbol{\theta}_{vol}} p_D(\boldsymbol{\theta}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))d\boldsymbol{\theta}, \tag{4.2}$$

where $\boldsymbol{\theta}$ is the descriptor used by the detector to make its decision, $\boldsymbol{\theta}_{vol}$ is the multi-dimensional volume in the descriptor space where the detector will decide to return a feature detection, and $p_D()$ is a multivariate distribution corresponding to detections. Unlike typical radar detectors, which use only the returned power for detection decisions, a general feature detector can use several quantities to make its decision. (4.2) generalizes the probability of detection of a range-based feature extractor, given occlusion information $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$).

One method to estimate the probability of detection is to estimate $p_D(\boldsymbol{\theta}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$, and then use (4.2) to determine $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$. The potentially high dimensionality of both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{vol}$ however, means that a large data set containing a significant number of descriptor $\boldsymbol{\theta}$, $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ pairs would be required. A simpler method is to learn $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ directly from measurements within a test data set as follows.

Experimental Determination of Detection Probabilities

This section demonstrates that feature probabilities of detection can be adequately and experimentally quantified based on $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, via statistical analyses on range data sets. Initially, a dataset is required from an environment where the ground truth positions of features are known, via independent means. A way to achieve this is through the use of features identifiable by humans - i.e. semantic features. (4.1) approximates the probability of detection of such a feature, which should be based on *all the necessary information*, Γ, by the probability of detection given the value of the *single parameter* $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$.

In the test data set, measurements manually associated with known ground truth features were used to determine the probability of detection of those features, conditioned on the number of unoccluded points. Intuitively, the more "representative" the test data is of the actual environment in which SLAM is to be achieved, the better the detection probability estimates would be expected to be. However, if the number of unoccluded points is a good approximation of a sufficient statistic of the detection probability, then it should have an impact on range data based SLAM performance in general environments.

Therefore, within the test data set, the predicted number of unoccluded points in the scan recorded at time k, $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, for each feature, $\boldsymbol{m}_k^j \in \mathcal{M}_k$, - robot pose, $\boldsymbol{x}_k$, pair, was calculated. For every possible number i of calculated unoccluded points, the actual number of times $\hat{n}_{\text{act}}$ that a particular value of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ was calculated within the dataset as follows

$$\hat{n}_{\text{act}} = \sum_{k=0}^{N_s} \sum_{\boldsymbol{m}_k^j \in \mathcal{M}_k} \delta(\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k), i) \quad \forall i \geq 0, \tag{4.3}$$

where N_s is the total number of scans in the dataset and $\delta(\cdot, \cdot)$ is the Kronecker delta function.

The number of times that a feature, with a predicted number of unoccluded points, $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, equal to i, produced a valid detection, $\hat{n}_{\text{det}}$ is given by:

$$\hat{n}_{\text{det}} = \sum_{k=0}^{N_s} \sum_{\boldsymbol{m}_k^j \in \mathcal{M}_k} \delta(\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k), i) \; c(\boldsymbol{m}_k^j, k) \quad \forall i \geq 0 \tag{4.4}$$

where $c(\boldsymbol{m}_k^j, k)$ is an indicator function, which equals unity if feature $\boldsymbol{m}_k^j \in \mathcal{M}_k$ was detected in the scan recorded at time k, and zero otherwise. Then, the probability of detection for each feature with a predicted number of points, $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, equal to i, can be approximated as the ratio of (4.4) and (4.3):

$$P_{\text{D}}(\boldsymbol{m}_k^j | \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) = i) \approx \hat{n}_{\text{det}} / \hat{n}_{\text{act}}. \tag{4.5}$$

The estimated probabilities of detection, utilizing the above concept, will be shown graphically in Section 4.6.1.

4.2.2 Estimating Probabilities of False Alarm

A full analysis of the probability of false alarm would require a model for every possible range scan, which does *not* contain any of the semantic features of interest. In practise this is infeasible. Importantly, the statistical representation of the number of false alarms in RB-PHD-SLAM[2] is a Poisson random set, which only requires an estimate of their expected number. When using the binary Bayes filter, the probability of false alarm can be approximated by dividing the total number of false alarms by the total number of detected features in the dataset,

$$P_{\text{FA}}(\boldsymbol{z}_k^l | \boldsymbol{x}_k) \approx \left(\sum_{k=0}^{N_s} N_{\text{FA}}(k) \right) / \left(\sum_{k=0}^{N_s} N_m(k) \right) , \tag{4.6}$$

[2]and indeed in many target tracking formulations.

where $N_m(k)$ is the total number of features detected in the scan recorded at time k and $N_{\mathrm{FA}}(k)$ is the total number of detected features that could not be associated with any feature in the scan recorded at time k. In a manner similar to the probabilities of detection outlined above, the statistical analysis of range based data, known to not contain the chosen semantic features, can yield an informative estimate of the probability of false alarm. This data can be obtained by either taking a scan in an environment known not to contain the features of interest or by identifying and removing detections which are manually associated with actual features from a sufficiently large dataset, as carried out here. The clutter distribution was modelled as Poisson in number, with mean value equal to the average number of false alarms per scan. Its spatial distribution was modelled as uniform, resulting in the following clutter intensity function for use in (3.15):

$$\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) \approx \left(\sum_{k=0}^{N_s} N_{\mathrm{FA}}(k)\right) \Bigg/ \left(N_s \int\limits_{\boldsymbol{z}_{\mathrm{FoV}}} d\boldsymbol{z}\right) . \tag{4.7}$$

4.3 Including Descriptor Information into SLAM

4.3.1 Including Descriptor Information into PHD-SLAM

In the field of target tracking, Clark et. al. [49] proposed a modification to the PHD filter that uses RADAR measurement *amplitude information* together with each accompanying range value. The idea behind this paper is to use this theory and replace the target amplitude with a general feature descriptor, by changing the likelihoods accordingly. Then the extended measurement vectors $\mathring{\boldsymbol{z}}_k^i$ are defined as

$$\mathring{\boldsymbol{z}}_k^i \equiv [\boldsymbol{z}_k^i \ a_i]^\top , \tag{4.8}$$

where $\boldsymbol{z}_k^i$ corresponds to the spatial part of the measurement (i.e., what used to be the entire measurement), and a_i is the *amplitude* information (which in general could be replaced by a *descriptor*). Hence, $\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ and $\mathring{\kappa}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{x}_k)$ account for the *joint* likelihood of target state and amplitude. In [49], the distributions of this amplitude, under false alarm and detection hypotheses, were modelled as Swerling type I and II models [50], which provide probabilistic (Rayleigh) models of received power fluctuations when the RADAR-to-target viewing aspect changes. The dependency on the environment is modelled by a single parameter d, where the expected (mean) SNR from a target was $1 + d$, where $p_{\mathrm{FA}}(a)$ and $p_{\mathrm{D}}(a|d)$ are the distributions of a for false alarms and targets of interest respectively. The modified PHD update equation used in [49] is then

$$
\begin{aligned}
{v_k}^{+}(\boldsymbol{m}) = v_k^{-}(\boldsymbol{m})(1 - P_{\mathrm{D}}(d)) + v_k^{-}(\boldsymbol{m}) \\
\times \sum_{i}^{|\mathcal{Z}_k|} \frac{P_{\mathrm{D}}(d)\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m}, \boldsymbol{x}_k)}{\mathring{\kappa}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{x}_k) + \int P_{\mathrm{D}}(d)\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m}, \boldsymbol{x}_k)v_k^{-}(\boldsymbol{m})d\boldsymbol{m}} ,
\end{aligned}
\tag{4.9}
$$

where

$$
\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m}, \boldsymbol{x}_k) = g_a^{\tau}(a_i|d)\boldsymbol{h}(\boldsymbol{z}_k^i|\boldsymbol{m}, \boldsymbol{x}_k) \tag{4.10}
$$

represents the product of the feature detection probability, the amplitude measurement likelihood and the spatial measurement likelihood, and

$$
\mathring{\kappa}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{x}_k) = \kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)g_{\mathrm{FA}}^{\tau}(a_i) \tag{4.11}
$$

represents the extended clutter intensity, including the amplitude likelihood. $g_a^{\tau}(a_i|d)$ and $g_{\mathrm{FA}}^{\tau}(a_i)$ are the measurement and false alarm likelihoods of the amplitude a_i, related to measurement z_i, based on a detection threshold τ. And $P_{\mathrm{D}}(d)$ is the probability of detection given an object SNR. This is

$$
P_{\mathrm{D}}(d) = \int_{a>\tau} p_{\mathrm{D}}(a|d)da , \tag{4.12}
$$

$$
g_{\mathrm{FA}}^{\tau}(a_i) = \begin{cases} \frac{p_{\mathrm{FA}}(a_i)}{\int_{a_i>\tau} p_{\mathrm{FA}}(a_i)da_i}, & a_i \geq \tau \\ 0.0, & a_i < \tau \end{cases} , \tag{4.13}
$$

$$
g_a^{\tau}(a_i|d) = \begin{cases} \frac{p_{\mathrm{D}}(a_i)}{\int_{a_i>\tau} p_{\mathrm{D}}(a_i|d)da_i}, & a_i \geq \tau \\ 0.0, & a_i < \tau \end{cases} , \tag{4.14}
$$

where the value of $p_{\mathrm{D}}(a|d)$ and $p_{\mathrm{FA}}(a_i)$ and their integrals are determined by the model of the received signal amplitude, which is modelled as a Swerling model [49]. The difference between (4.9) and the standard PHD update (3.15) is the inclusion of the measurement and false alarm likelihoods. Measurements which are more likely to be true detections rather than false alarms, will receive higher weights.

In this chapter, instead of using the Swerling-based Rayleigh distributions of the received signal amplitude, adopted for radar measurement likelihoods in [49], a generic descriptor model is applied. The extended measurement vectors $\mathring{\boldsymbol{z}}_k^i$ here are redefined as

$$
\mathring{\boldsymbol{z}}_k^i \equiv [\boldsymbol{z}_k^i \ \boldsymbol{\theta}_k^i]^{\top} , \tag{4.15}
$$

where $\boldsymbol{\theta}_k^i$ is a descriptor vector accompanying measurement $\boldsymbol{z}_k^i$. Along with this, a variable $\boldsymbol{d}_{\boldsymbol{\theta}}$ equivalent to the target SNR is used. A general probability distribution $p_D(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})$ for true detections can be used to define a feature likelihood equivalent to $g_a^{\tau}(a_i|d)$ in (4.14), for the general feature descriptor $\boldsymbol{\theta}$ i.e.

$$g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}}) = \begin{cases} \dfrac{p_D(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})}{\int\limits_{\boldsymbol{\theta}_{\text{vol}}} p_D(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})d\boldsymbol{\theta}} & \text{if } \boldsymbol{\theta} \in \boldsymbol{\theta}_{\text{vol}} \\ 0.0 & \text{if } \boldsymbol{\theta} \notin \boldsymbol{\theta}_{\text{vol}} \end{cases}. \tag{4.16}$$

An example of such a descriptor vector will be given in Section 4.6.3. In (4.16), $p_D(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})$ is a distribution on $\boldsymbol{\theta}$, with known parameter $\boldsymbol{d}_{\boldsymbol{\theta}}$ being equivalent to the SNR in the RADAR implementation. A different distribution $p_{\text{FA}}(\boldsymbol{\theta})$ has to be used to model false alarms in the detectable volume.

$$g_{\text{FA}}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta}) = \begin{cases} \dfrac{p_{\text{FA}}(\boldsymbol{\theta})}{\int\limits_{\boldsymbol{\theta}_{\text{vol}}} p_{\text{FA}}(\boldsymbol{\theta})d\boldsymbol{\theta}} & \text{if } \boldsymbol{\theta} \in \boldsymbol{\theta}_{\text{vol}} \\ 0.0 & \text{if } \boldsymbol{\theta} \notin \boldsymbol{\theta}_{\text{vol}} \end{cases}. \tag{4.17}$$

These likelihoods can be used with the detection statistics $P_{\text{D}}(\boldsymbol{m}|\hat{n}(\boldsymbol{m},\boldsymbol{x}_k))$ and $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ in a modified PHD filter update equation equivalent to (4.9), now given by

$$\begin{aligned} v_k^+(\boldsymbol{m}) &= v_k^-(\boldsymbol{m})(1 - P_{\text{D}}(\boldsymbol{m}|\hat{n}(\boldsymbol{m},\boldsymbol{x}_k))) + v_k^-(\boldsymbol{m})\times \\ &\sum_i^{|\mathcal{Z}_k|} \frac{P_{\text{D}}(\boldsymbol{m}|\hat{n}(\boldsymbol{m},\boldsymbol{x}_k))\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m},\boldsymbol{x}_k)}{\mathring{\kappa}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{x}_k) + \int P_{\text{D}}(\boldsymbol{m}|\hat{n}(\boldsymbol{m},\boldsymbol{x}_k))\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i|\boldsymbol{m},\boldsymbol{x}_k)v_k^-(\boldsymbol{m})d\boldsymbol{m}}. \end{aligned} \tag{4.18}$$

In principle, $\boldsymbol{\theta}$ can be any descriptor-based on the measurement. It is desirable that its distribution for detections $p_D(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})$ and for false alarms $p_{\text{FA}}(\boldsymbol{\theta})$ are as separated in the $\boldsymbol{\theta}$ space as possible, so that the measurements that are more likely to be false alarms will have lower weights in the modified PHD update Equation (4.18). If the likelihoods $g_{\text{FA}}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta})$ and $g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})$ are approximately equal, the modified PHD filter update in (4.18) will tend to its traditional form, given in (3.15). In this chapter the descriptor $\boldsymbol{\theta}$ will be the number of points n_{d} used to detect the feature. The variable $\boldsymbol{d}_{\boldsymbol{\theta}}$, which is equivalent to the target SNR in Clark's work, is the predicted value of n_{d}, i.e. $\boldsymbol{d}_{\boldsymbol{\theta}} = \hat{n}(\boldsymbol{m},\boldsymbol{x}_k)$.

4.3.2 Including Descriptor Information into MH-FastSLAM (MH-FastSLAM)

Since $P_{\mathrm{D}}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ is used within the binary Bayes addition to MH-FastSLAM, the descriptor information can be included by replacing (1.5) with the extended measurement likelihood

$$\mathring{\boldsymbol{h}}(\mathring{\boldsymbol{z}}_k^i | \boldsymbol{m}_k^j, \boldsymbol{x}_k, \boldsymbol{d}_{\boldsymbol{\theta}}) = g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta} | \boldsymbol{d}_{\boldsymbol{\theta}}) \boldsymbol{h}(\boldsymbol{z}_k^i | \boldsymbol{m}_k^j, \boldsymbol{x}_k) \tag{4.19}$$

in the data association and particle weighting steps. In MH-FastSLAM there is no mechanism known to the authors, to incorporate $g_{\mathrm{FA}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta})$ and therefore it is not used.

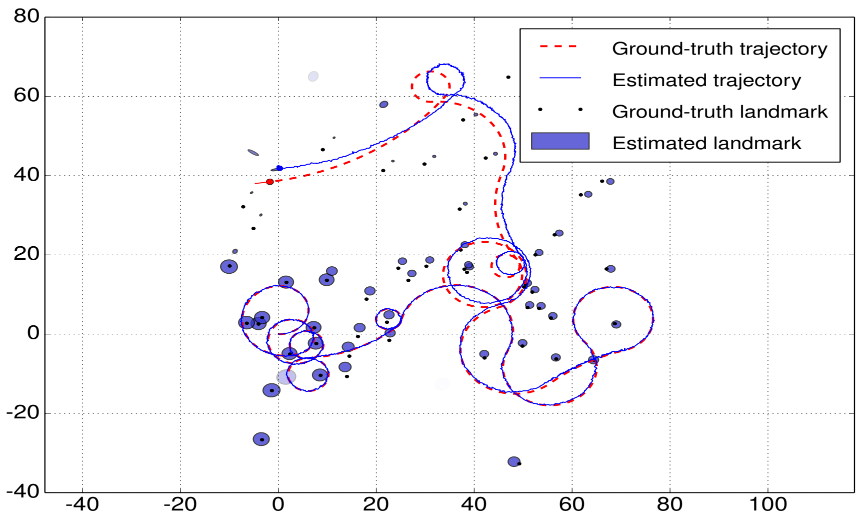
Note that (4.9) requires estimates of both $P_{\mathrm{D}}(d)$ and $g_a^\tau(a_i|d)$ as well as the clutter term $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) g_{\mathrm{FA}}^\tau(a)$. Similarly, (4.19) requires an estimate of $g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta}|\boldsymbol{d}_{\boldsymbol{\theta}})$. A similar concept is applied in the following sections, in which instead of received amplitude information, a range - bearing - based feature descriptor $\boldsymbol{\theta}$ is defined, which uses ray casting[3].

4.3.3 Adding Descriptor Information into RB-PHD-SLAM, a Proof of Concept

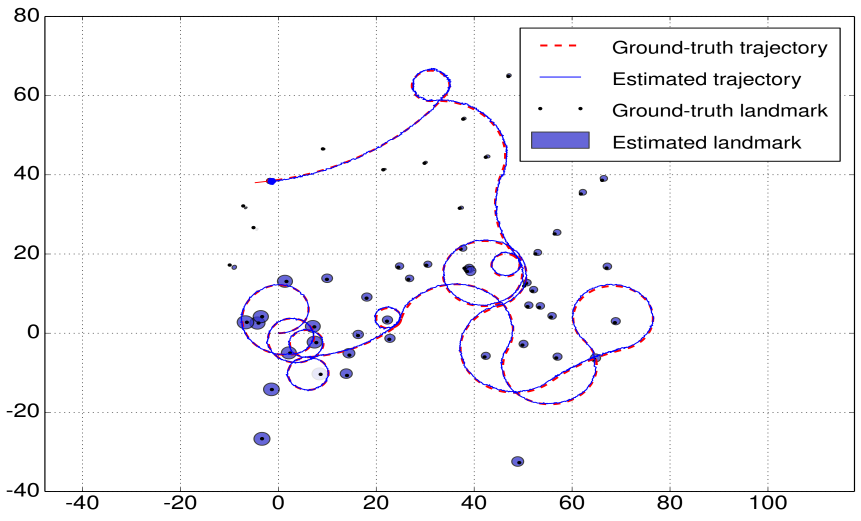
To demonstrate the applicability of [49] in RB-PHD-SLAM, a preliminary simulation was performed. Both the regular RB-PHD-SLAM algorithm from [51] and RB-PHD-SLAM with the modified PHD filter that includes target descriptor information (4.9) were executed in a simulated environment. Range-Bearing measurements were simulated along with a descriptor described completely by its detection and false alarm distributions ($g_a^\tau(a_i|d)$ and $g_{\mathrm{FA}}^\tau(a_i)$, respectively). The descriptor a_i for detections was modelled using a Normal distribution to represent $g_a^\tau(a_i|d)$ with a mean of 4 and a standard deviation of 2, while the descriptor a_i for false alarms was modelled using an exponential distribution to represent $g_{\mathrm{FA}}^\tau(a_i)$ with parameter $\lambda = 0.77$, corresponding to any reasonably well separated detection and false alarm distributions in the descriptor space. For example, the descriptor a_i could represent the received power, or contrast value, accompanying each range and bearing measurement. The false alarm rate was increased to the point where the unmodified RB-PHD-SLAM trajectory estimate diverged from its ground truth path by exceeding a minimum Euclidean error of 2.0m. This corresponded to an average of 50% of the detected features being false alarms (12.5 false features per time step). Results for both the unmodified and modified RB-PHD-SLAM algorithms, based on (3.15) and (4.9) respectively, are shown in Figs. 4.2a and 4.2b. As can be seen in the figures, the addition of target descriptor information improves the solution in terms of both the trajectory and map

[3]Ray casting, also referred to as ray tracing, is a method to calculate the ideal range that a range sensor would return in the absence of noise, given robot and feature location estimates.

estimates. The mapping error was also evaluated using the Optimal Sub-Pattern Assignment (OSPA) metric, which penalizes both spatial and cardinality map errors [33], as shown in Fig. 4.3. The metric shows an improved performance of the modified algorithm compared to the original RB-PHD-SLAM solution. This demonstrates that, in principle, adding a descriptor, at least to the RB-PHD-SLAM filter, can improve its performance.



(a) *RB-PHD-SLAM without target amplitude information.*



(b) *RB-PHD-SLAM with target amplitude information.*

Figure 4.2: *RB-PHD-SLAM simulation results. A 3 sigma ellipse is shown for every estimated landmark.*

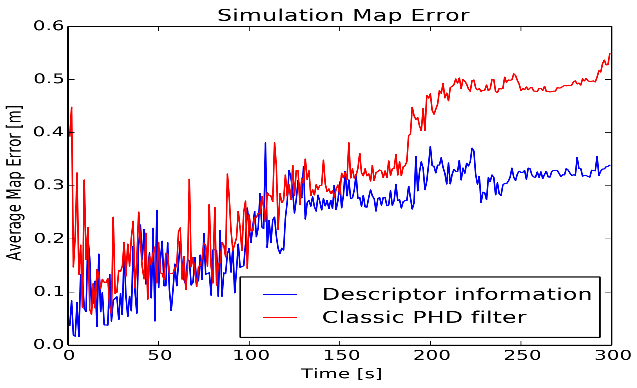


Figure 4.3: *The average OSPA distance from* (2.53) *between the ground truth and estimated maps. The errors where averaged over 5 independent simulations.*

4.4 Learning from Simulated Range Data

This section demonstrates the validity of the concept to determine the detection statistics of a feature detector, based on the number of unoccluded points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ described in section 4.2. It is applied in a simulated environment with a feature extraction method based on RANSAC. The environment was simulated with the Gazebo simulator [52], composed of cylinders, which are the features of interest, and cuboids, which are objects of non-interest. The SLAM state will be composed of the trajectory of the robot and the center coordinates of the cylinders in the vehicle's plane of motion. The simulated environment is shown in Fig. 4.4a and its plan view is also shown in the background of the SLAM results in Fig. 4.10.

A simple two-step detector is used in the simulation, composed of a segmentation method (based on a distance threshold of 0.3m), followed by RANSAC to detect circular objects in the segmented data. The RANSAC algorithm was implemented based on a required circular in-lier number of 13 range points and an in-lier distance threshold of 0.02m and was executed $12,500$ times per feature, corresponding to a RANSAC detection failure rate of 1.7%.

4.4.1 Estimating the Detection Probability $P_{\mathrm{D}}(\boldsymbol{m}_k^j | \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$

By using the known poses of the robot and features provided by the simulator, (4.5) can be applied to obtain an estimate of the probability of detection given the predicted number of points, $P_{\mathrm{D}}(\boldsymbol{m}_k^j | \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$. Fig. 4.5 shows the results. A 99% confidence interval was calculated based on the number of points available at each value of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, showing the uncertainty in the estimate. As can be seen from the figure, the probability of detection given a predicted number of points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ is very close to zero when $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ is lower than 7, then it tends to increase as $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ increases to 14. For values of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ higher than 14 the probability

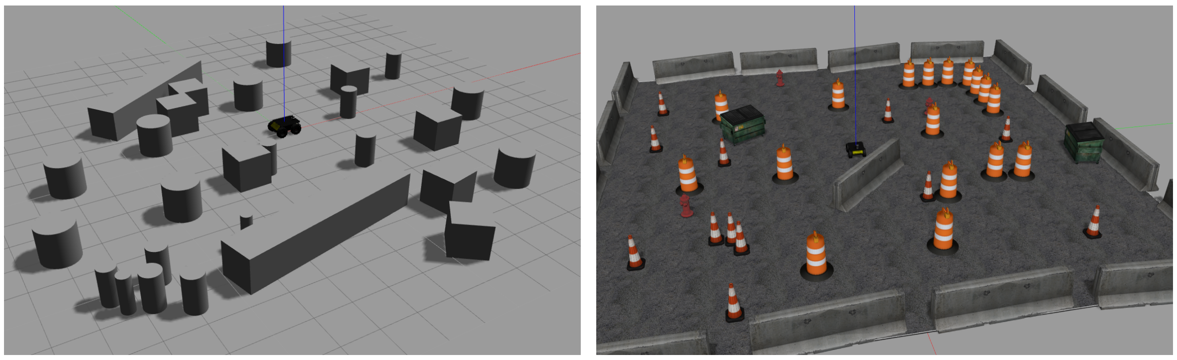

(a) (b)

Figure 4.4: *Simulated environments used to obtain the detection statistics. The same environment used in simulations (a) and a second dataset (b).*

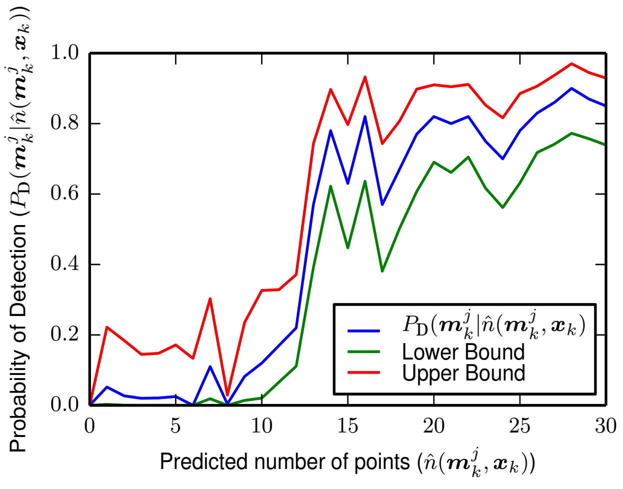


Figure 4.5: *The probability of detection as a function of the number of unoccluded points* $\hat{n}(\boldsymbol{m}_k^j,\boldsymbol{x}_k)$. *Based on the amount of data available, a 99% confidence interval is calculated, the bounds of which are shown.*

of detection fluctuates about a value of approximately 0.8. This could be interpreted as the intrinsic probability of detection of the feature detector being 0.8, once it is known not to be occluded.

4.4.2 Estimating $P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k)$ and Intensity Function $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$

To estimate the probability of false alarm and, for the case of the RB-PHD-SLAM filter, the false alarm intensity function, the same dataset can be used, labelling the unassociated measurements as false alarms and estimating the statistical parameters of those measurements. In particular, since the clutter distribution is assumed Poisson in the PHD filter, only the average number of false alarms per scan needs to be estimated. To assess the accuracy of the Poisson approximation, Fig. 4.6 shows a distribution of the number of false alarms per scan, and a Poisson distribution (red) with parameter $\lambda = 0.06$ calculated from the sampled data. $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ was then be determined from (4.7):

$$\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k) \approx \frac{\sum_{k=0}^{N_s} N_{\mathrm{FA}}(k)}{N_s \int_{\boldsymbol{z}_{\mathrm{FoV}}} d\boldsymbol{z}} = \frac{\lambda}{\int_{\boldsymbol{z}_{\mathrm{FoV}}} d\boldsymbol{z}} \tag{4.20}$$

while $P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k)$ was determined from (4.6) to be 2.34%

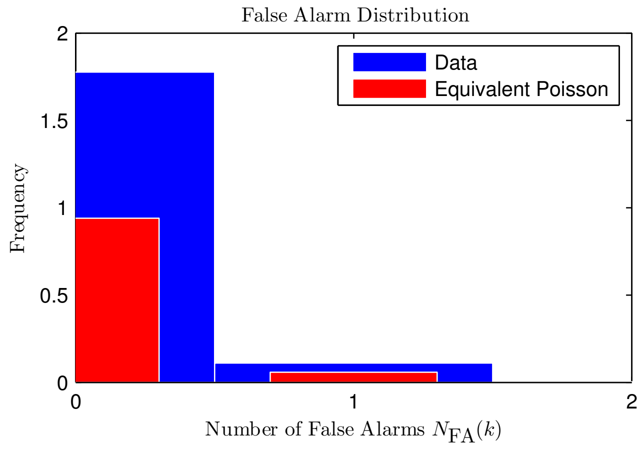


Figure 4.6: *Distribution of the number of false alarms per scan, from the simulation. The red graph shows the equivalent Poisson distribution.*

4.4.3 Estimating Feature Likelihoods $g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta})$ and $g_{FA}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta})$

In general, after choosing a suitable descriptor, data needs to be obtained in an environment with features identified by independent means in order to model the probability distributions $p_D(\boldsymbol{\theta})$ and $p_{\mathrm{FA}}(\boldsymbol{\theta})$.

As in [49], where the conditional distribution $p_D(a|d)$ was determined from a Rayleigh distribution, a descriptor which depends on the assumed sufficient statistic $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ is necessary here. A descriptor $\boldsymbol{\theta} = n_{\mathrm{d}}$ is therefore defined, corresponding to the number of range points used in the detection of a feature. As stated in section 4.3, n_{d} would be equivalent to a and $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ to d in [49]. Therefore, $g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta}) = p(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ needs to be determined. To model the distribution $p(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$, first the joint probability distribution $p(n_{\mathrm{d}}, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ is modelled (Fig. 4.7). For this purpose, n_{d} is plotted as a function of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, from a total of 10091 simulated laser scans which produced 27193 measurements, in Fig. 4.7. As can be seen in the figure there is a strong dependency of n_{d} on $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$.

Estimating $p(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ directly would be difficult given the sparsity of the data at some values of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ (see Fig. 4.7 at values of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ of 120 and higher). A contour of an equivalent Normal distribution is also shown in Fig. 4.7. Using the estimated mean μ and covariance Σ the conditional distribution can be obtained as

$$\begin{aligned} p(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \mathcal{N}\big(n_{\mathrm{d}}|\mu(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)),& \\ \Sigma(n_{\mathrm{d}}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))\big),& \qquad (4.21) \end{aligned}$$

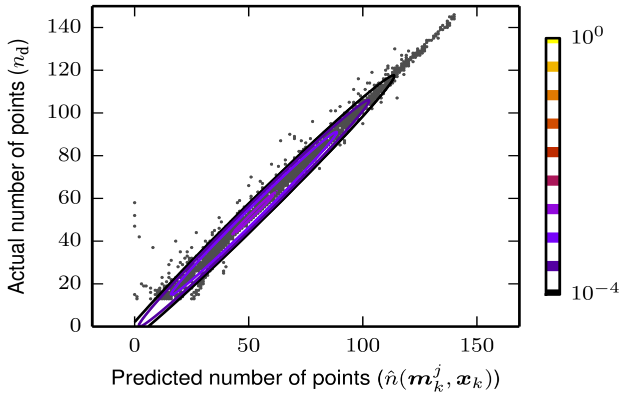


Figure 4.7: *Number of points n_d measured as a function of the predicted number of points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. The dependency of n_d on $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ can be observed in this figure. The contours of an equivalent bivariate Normal distribution are also shown.*

where

$$\mu(n_\mathrm{d}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \mu_1 + \frac{\Sigma_{12}}{\Sigma_{22}}(\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) - \mu_2) \ , \tag{4.22}$$

$$\Sigma(n_\mathrm{d}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \Sigma_{11} - \frac{\Sigma_{12}\Sigma_{21}}{\Sigma_{22}} \ , \tag{4.23}$$

where Σ_{lm} is the element of the matrix Σ in row l and column m, and μ_l is the l^{th} element of vector μ.

Finally, by using the fact that the RANSAC circle detector cannot generate detections with less points than a threshold N_min (= 13 in this case), the Normal distribution should be rescaled. This final conditional distribution is

$$g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_\mathrm{vol}}(\boldsymbol{\theta}) = \begin{cases} \frac{p(n_\mathrm{d}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))}{\int_{N_\mathrm{min}}^{\infty} p(n_\mathrm{d}|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) dn_\mathrm{d}} & \text{if } n_\mathrm{d} \geq N_\mathrm{min} \\ 0 & \text{if } n_\mathrm{d} < N_\mathrm{min} \end{cases} , \tag{4.24}$$

and is shown in Fig. 4.8.

For the case of false alarms the distribution of the number of points $g_{FA}^{\boldsymbol{\theta}_\mathrm{vol}}(\boldsymbol{\theta})$ has to be modelled using known false alarms from the same simulated dataset. Fig. 4.9 shows the histogram for the number of false alarms, along with an equivalent exponential distribution

$$g_{FA}^{\boldsymbol{\theta}_\mathrm{vol}}(\boldsymbol{\theta}) = \lambda_{n_\mathrm{d}} \exp\left(-\lambda_{n_\mathrm{d}}(n_\mathrm{d} - N_\mathrm{min})\right), \tag{4.25}$$

where, from Fig. 4.9, $\lambda_{n_\mathrm{d}} = 0.041$ and $N_\mathrm{min} = 13$. The results of applying this descriptor in simulated SLAM trials, will be shown in Section 4.5.

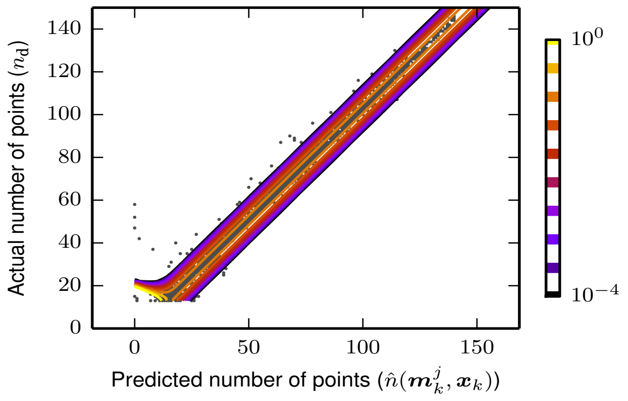


Figure 4.8: *The conditional distribution of n_d given $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$, calculated from the Normal distribution of Fig. 4.7.*

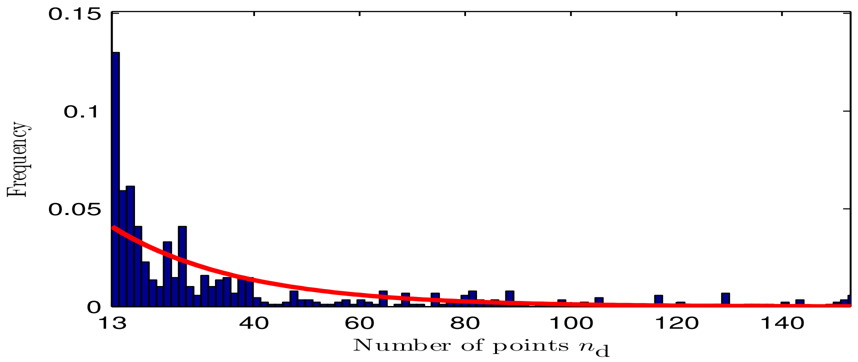
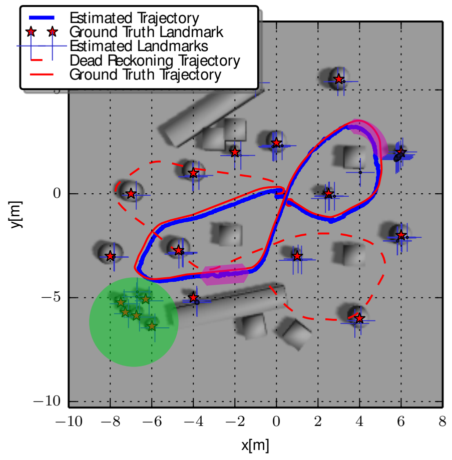


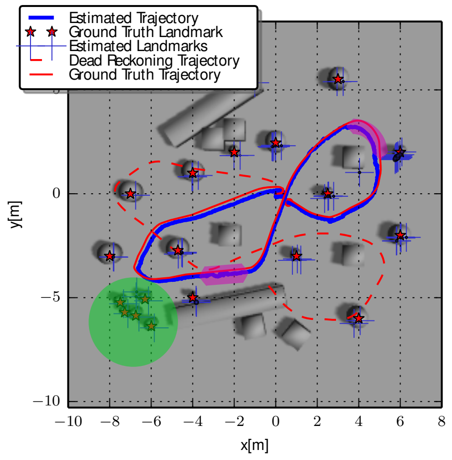
Figure 4.9: *Descriptor distribution for false alarms, which resembles an exponential distribution.*

4.5 Simulated SLAM Results

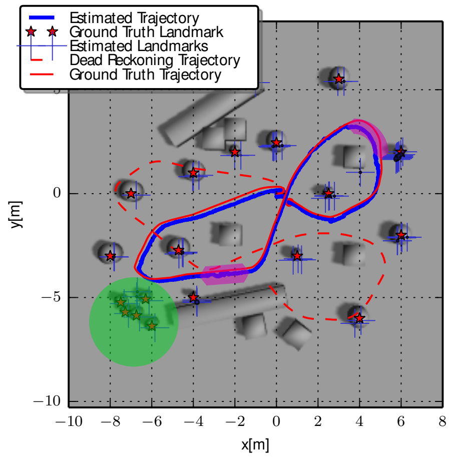
Table 4.1 shows the parameters used in the simulations. Figure 4.10 presents the results of running the algorithms with and without the proposed detection statistics. The ground truth map features are displayed using red stars and the estimated map features using blue crosses. The ground truth trajectory is shown with a red line and the estimated trajectory with a continous blue line. The red dashed line shows the odometry input (dead reckoning). A satisfactory SLAM solution should estimate both the map and trajectory accurately, hence ideally the red stars and blue crosses and the red and blue lines should coincide. The background of each graph is a top-down view of the simulated environment. Figs. 4.10a and 4.10b show the results of using a constant probability of detection of 0.7 within the sensors FoV for RB-PHD-SLAM and MH-FastSLAM respectively. Figs. 4.10c and 4.10d show the results with the proposed detection statistics from (4.5) and Fig. 4.5 for RB-PHD-SLAM and MH-FastSLAM respectively, while Figs. 4.10e and 4.10f include both the proposed detection statistics as well as the



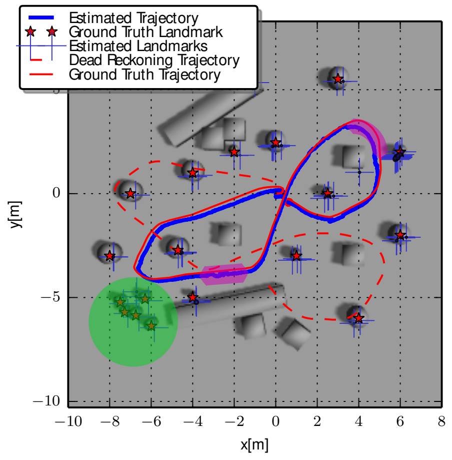
(a) *RB-PHD-SLAM with constant P_D.*



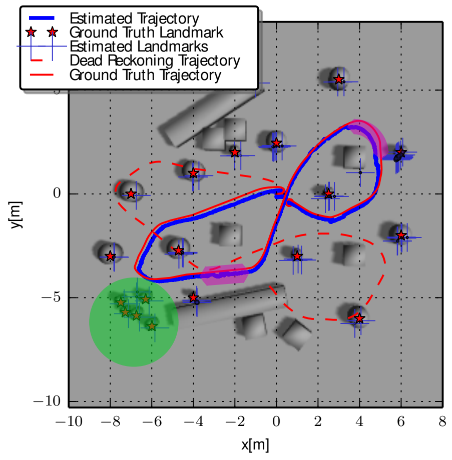
(b) *MH-FastSLAM with constant P_D.*



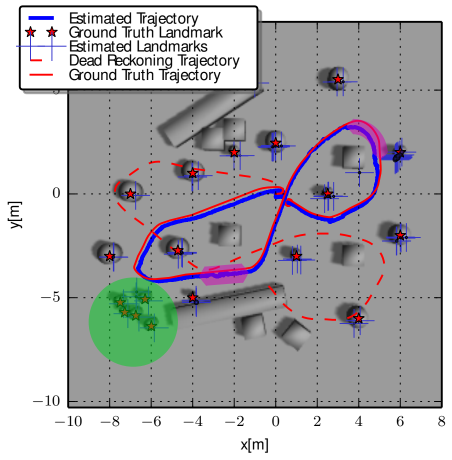
(c) *RB-PHD-SLAM with the learned probability of detection.*



(d) *MH-FastSLAM with the learned probability of detection.*



(e) *RB-PHD-SLAM with the learned probability of detection and descriptor information.*



(f) *MH-FastSLAM with the learned probability of detection and descriptor information.*

Figure 4.10: *SLAM results in a simulated environment. Highlighted in green: area where the detector performance significantly differs. Highlighted in pink: sections of trajectory where only a single feature was in the FoV of the robot's sensor.*

Parameter	Value	Equation
$P_{\mathrm{D}}(\boldsymbol{m}_k^j \mid \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$	See Fig. 4.5	(4.5)
λ	0.06	(4.20)
$P_{\mathrm{FA}}(\boldsymbol{z}_k^l \mid \boldsymbol{x}_k)$	2.34%	(4.6)
μ	$[52.25\ 52.98]^\top$	(4.22)
Σ	$\begin{bmatrix} 602.10 & 628.97 \\ 628.97 & 664.65 \end{bmatrix}$	(4.23)
$\lambda_{n_{\mathrm{d}}}$	0.041	(4.25)

Table 4.1: *The parameters used in the SLAM simulations.*

proposed descriptor likelihoods from (4.24) and (4.25) for RB-PHD-SLAM and MH-FastSLAM respectively.

Including the proposed detection statistics improves the SLAM estimates for RB-PHD-SLAM and in the initial part of the trajectory for MH-FastSLAM (Figs. 4.10c and 4.10d). The incorporation of the descriptor information (Figs. 4.10e and 4.10f) has its largest impact on the MH-FastSLAM results. An area where the detector produced false measurements is highlighted in green in the figures. It can be seen that both algorithms, when based on a constant probability of detection, fail to map the area correctly, with RB-PHD-SLAM missing two features, and MH-FastSLAM including many false alarms. Only by adding both the descriptor and detection statistics is MH-FastSLAM able to map the area correctly. Sections of the robot's trajectory where only a single feature was visible are highlighted as a pink shaded area. In these sections, both algorithms provide improved trajectory estimation when using the descriptor information. From Figs. 4.10a and 4.10b it can be seen that MH-FastSLAM has more trouble at the initial sharp "U" turn (close to the origin). This can be explained by the fact that the PHD filter is quick to initialize new estimates, allowing it to deal with the initial sharp turn. Conversely, when a feature's probability of detection is non-zero, the PHD filter can remove features too quickly, if not sensed. This explains why two of the features in the green area are never correctly mapped at the end of each run.

Figures 4.11 and 4.12 show the trajectory and mapping errors, respectively. The errors show similar results as can be observed in figure 4.10. In particular, the inclusion of the detection statistics improves the results in the first part of the trajectory (up to approximately 1300$[s]$). After this time, the inclusion of the detection statistics without using the descriptor in MH-FastSLAM causes the algorithm to diverge more quickly than the algorithm using a constant probability of detection. Importantly both algorithms have a significantly improved trajectory when including both the descriptor and the variable probability of detection.

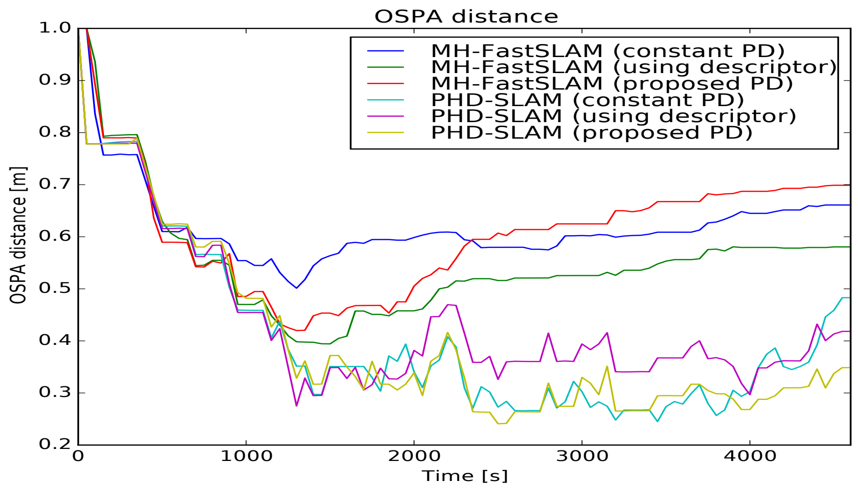


Figure 4.11: *Average OSPA errors between the ground truth map and the map estimates. Errors where averaged over 5 different runs.*

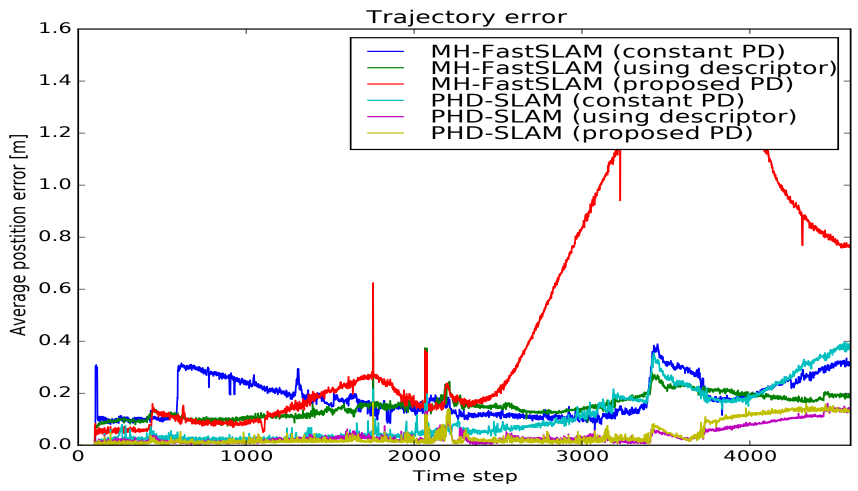


Figure 4.12: *Average position error over time. Errors where averaged over 5 different runs.*

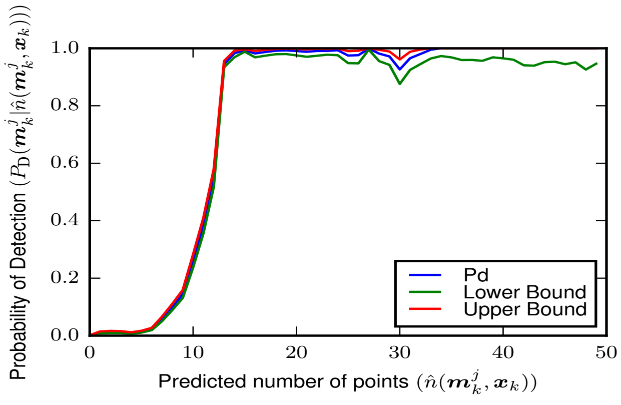


Figure 4.13: *The probability of detection as a function of the number of unoccluded points* $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. *Based on the number of points available in the test data set, a 99% confidence interval is calculated, the bounds of which are shown.*

4.5.1 Using a different training dataset

To test the generality of using the proposed descriptor, a second training dataset has been used to produce different detection statistics. Figure 4.4 shows the environments used. These two environments differ in terms of the radii of the circles and their spacing. The spacing of the cylinders is important since, for example, closely located cylinders can cause the detector to generate less circles than are actually present, with false centers and radii. For example, the closely spaced cylinders in the lower left corner of Fig. 4.4a result in the false alarms within the green circles in Figs. 4.10b and 4.10d.

Fig. 4.13 shows the probability of detection obtained from the training dataset of Fig. 4.4b. As can be seen from the figure, the transition from close to zero probability of detection to a high probability of detection occurs at approximately the same value $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) = 13$, but the values of the probability of detection are significantly different. The SLAM algorithms were then rerun based on the detection statistics shown in Fig. 4.13.

As can be seen in Figure 4.14 the mapping errors are similar to the statistics calculated from the same environment. MH-FastSLAM does not benefit from using only the detection statistics, but it does benefit slightly from adding both the detection statistics and the descriptor information. On the other hand PHD-SLAM benefits mainly from having accurate detection statistics and the descriptor only produces a small improvement.

Figure 4.15 shows the trajectory errors obtained using the new detection statistics. The trajectory error plots clearly display the issues PHD-SLAM has when lacking the detection statistics, while having similar results with and without the descriptor information. MH-FastSLAM has less benefits than when using the environment-specific detection statistics, but when using the both the descriptor and detection statistics the pose error is still lowest for most of the

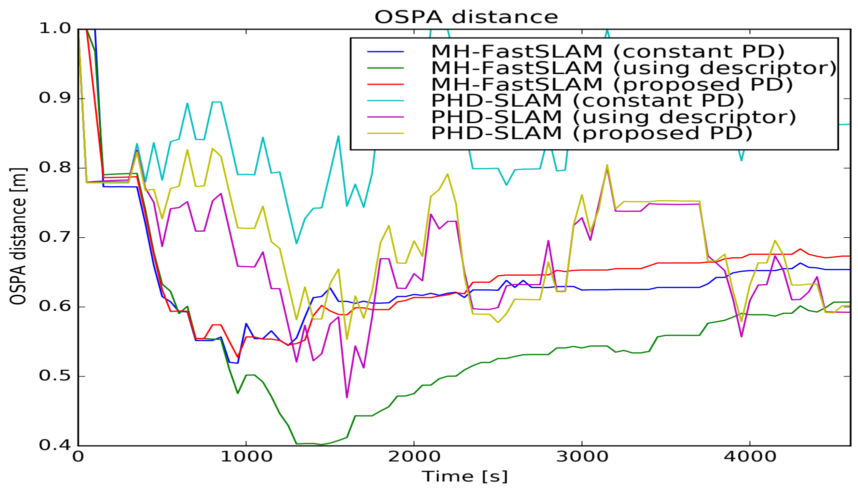


Figure 4.14: *Average OSPA errors between the ground truth map and the map estimates. Errors where averaged over 5 different runs.*

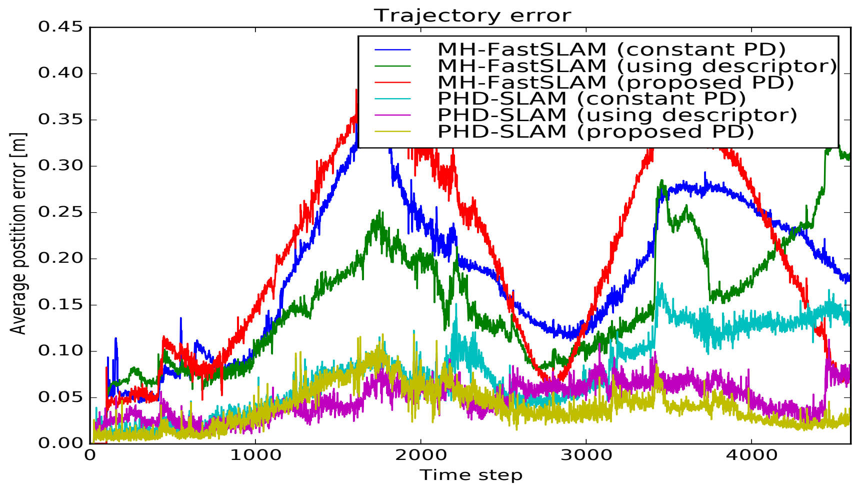


Figure 4.15: *Average position error over time. Errors where averaged over 5 different runs.*

trajectory. Annex B shows the same algorithms used to generate Fig. 4.10 based on the new training environment. A single SLAM result is shown for each algorithm out of five Monte Carlo runs.

4.6 Learning from a Park Environment

This section applies the concepts described in Section 4.4 to data obtained in a real park environment. The method explained is valid for any shape detector to which ray casting can be performed. A simple circular feature detector, based on the detector described by Guivant et. al. in [53], is applied here. This detector first segments the laser scan into clusters, using a simple distance threshold, and then fits a circle to each cluster, via mean squared error minimization. Further details of the detector can de found in [46]. The dataset used to learn $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ and $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ was recorded in a park near the Universidad de Chile. The data set consisted of 2397 scans taken from 19 known vehicle poses. The area covered by the dataset contained 174 tree trunks and lamp posts, the number and locations of which where verified by independent means to form ground truth. Naturally, the generality of such an environment is questionable. In general, any semantic feature detection statistics can be determined in a manner similar to methods given in the following sub-sections but should be based on data sets from environments known to contain a significant number of the type of semantic feature sought.

4.6.1 Estimating Detection Probability $P_{\mathbf{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$

Fig. 4.16 shows the results of applying (4.5) to the data set, where the probability of detecting a circular object (in this case a tree) is highly dependant on the number of unoccluded points[4]. The figure also shows a 99% confidence interval, based on the number of data points available for each value of $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. It should be noted that there is less data corresponding to features that have a high number of unoccluded points. Therefore, for such features, the uncertainty in the estimated value of $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ is higher, as shown by the confidence interval. Additionally, it can be seen that the probablity of detection goes down when the predicted number of points increases past 6. This may be due to the fact that many of the observed real trees are not perfectly circular, and with an increased number of points the circle detector may be rejecting detections due to the real trees not being circular enough.

4.6.2 Estimating $P_{\mathbf{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k)$ and $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$

Similarly to the explanation in Section 4.4.2, for the case of the park dataset, the histogram of the number of false alarms per scan is shown in Fig. 4.17, along with its equivalent, theoretical Poisson distribution (with parameter $\lambda = 9.3$) plotted in red. $\kappa(\boldsymbol{z}_k^i|\boldsymbol{x}_k)$ was again determined from (4.20), while $P_{\mathrm{FA}}(\boldsymbol{z}_k^l|\boldsymbol{x}_k)$ was determined from (4.6) to be 47.19%.

4.6.3 Estimating Feature Likelihoods $g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta})$ and $g_{FA}^{\boldsymbol{\theta}_{\mathrm{vol}}}(\boldsymbol{\theta})$

As in Section 4.4.3, to model the distribution of the number of range values used to generate a detection, given $p(n|\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$, first the joint probability distribution $p(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$ is

[4]Note that the values of the probability in Fig. 4.16 differ significantly from those in Fig. 4.5. This is because the feature detector used in the park can detect features using less range points than the RANSAC detector, which needs a relatively large number of inliers to differentiate detections from any other randomly generated circle. Further, the RANSAC detector was used in a simulated environment with a lower clutter rate.

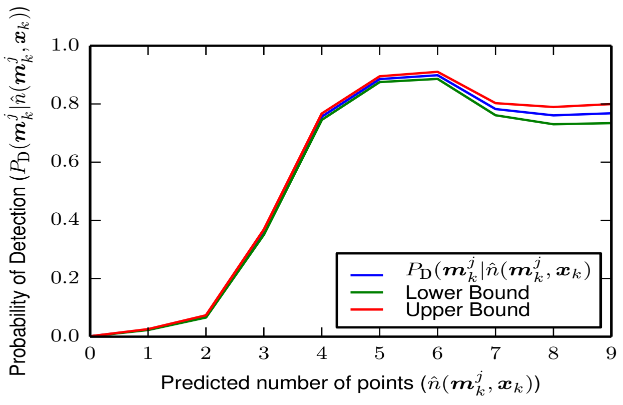


Figure 4.16: *The probability of detection as a function of the number of unoccluded points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. Based on the number of points available in the test data set, a 99% confidence interval is calculated, the bounds of which are shown.*

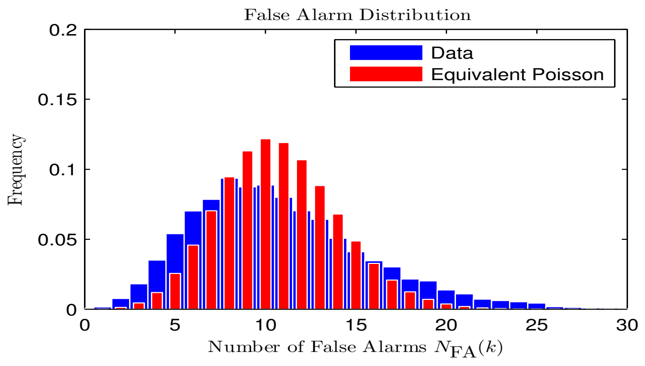


Figure 4.17: *A distribution of the number of false alarms per scan, obtained from the test dataset.*

modelled, shown in Fig. 4.18. In contrast to the simulated range data analysed in Section 4.4.3,

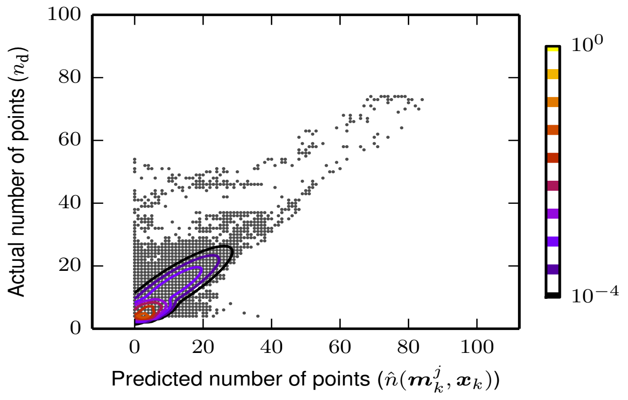


Figure 4.18: *Number of measured points n_d as a function of the predicted number of points $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. The contours of an equivalent Gaussian mixture (comprised of 4 Gaussian components) are also shown.*

a single Gaussian distribution would not accurately model the data in Fig. 4.18, therefore a

Gaussian mixture

$$p(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \sum_i w_i \mathcal{N}(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) | \mu_i, \Sigma_i) \tag{4.26}$$

is used. To estimate the Gaussian mixture parameters, the Expectation Maximization (EM) [54] algorithm was then applied, based on a predetermined (in this case selected to be 4 to avoid overfitting to the data) number of Gaussian components. A contour plot of the result is also shown in Fig. 4.18. Using the estimated Gaussian mixture the conditional distribution can be obtained as

$$p(n | \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) = \frac{p(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))}{p(\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))} \tag{4.27}$$

$$= \frac{p(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))}{\int p(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) dn} . \tag{4.28}$$

By substituting (4.26) into (4.27) the conditional distribution becomes

$$g_{\boldsymbol{\theta}}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta}) = p(n | \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)) \tag{4.29}$$

$$= \frac{\sum_i w_i \mathcal{N}(n, \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) | \mu_i, \Sigma_i)}{\sum_i w_i \mathcal{N}(\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k) | \mu_i(2), \Sigma_i(2,2))} ,$$

where $\mu_i(2)$ and $\Sigma_i(2,2)$ are the elements of the mean and covariance matrix respectively, which correspond to $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$. Fig. 4.19 shows a contour plot of the conditional distribution calculated using (4.29) corresponding to the Gaussian mixture of Fig. 4.18.

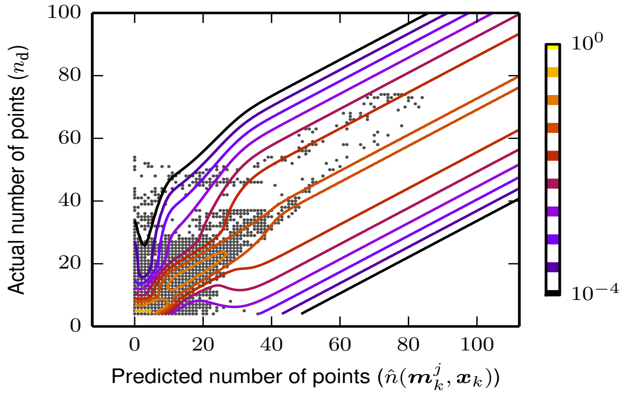


Figure 4.19: *The conditional distribution of n_d given $\hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k)$ calculated from the Gaussian mixture of Fig. 4.18.*

For the case of false alarms, the distribution of n_d, based on known false alarms from the park environment and shown in blue in Fig. 4.20, was modelled with an exponential distribution

$$g_{FA}^{\boldsymbol{\theta}_{\text{vol}}}(\boldsymbol{\theta}) = \lambda_{n_{\text{d}}} \exp\left(-\lambda_{n_{\text{d}}}(n_{\text{d}} - N_{\min})\right), \tag{4.30}$$

with parameters $\lambda_{n_{\text{d}}} = 0.5124$ and $N_{\min} = 4$ (red curve). The results of applying this descriptor

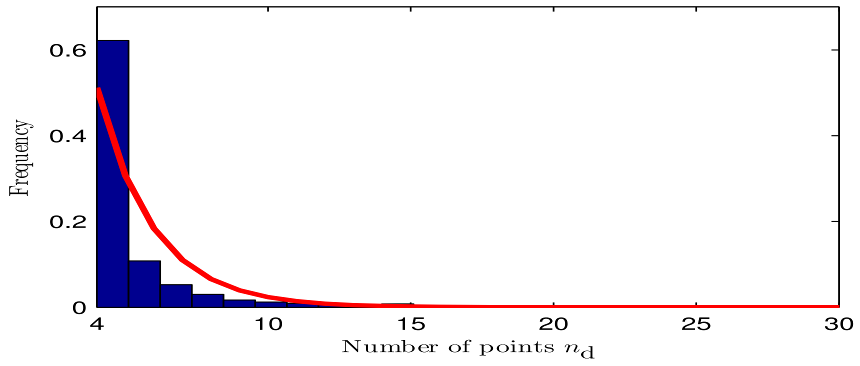


Figure 4.20: *Descriptor distribution for false alarms (blue), and its equivalent exponential distribution (red).*

in SLAM will be shown in Section 4.7.

4.7 Experimental SLAM Results

The performance of utilizing the probability of detection and descriptor information in Parque O'Higgins, Santiago and with the publicly available Victoria Park dataset [53], are evaluated in Sections 4.7.1 and 4.7.2 respectively. Table 4.2 shows the parameters used, which are common to both datasets.

4.7.1 Experiments in Parque O'Higgins, Santiago

The robotic platform for collecting the experimental dataset was a Clearpath Husky A-200 robot equipped with a Sick LD-LRS-1000 laser range finder, with a reported maximum range of 80m at 10% reflectivity. The Husky's wheel encoders provided odometry measurements, $\boldsymbol{u}_k$, used in the motion model in (1.1). The experiments were conducted in the same environment where the detection statistics were determined, albeit with a different dataset. Although not a very general result, this provides the best estimate of the real detection statistics that will be encountered in the experiments.

The robot ended its trajectory at approximately the same position as it started. The results for the constant probability of detection versions of RB-PHD-SLAM and MH-FastSLAM

Parameter	Value	Equation
$P_{\mathrm{D}}(\boldsymbol{m}_k^j \mid \hat{n}(\boldsymbol{m}_k^j, \boldsymbol{x}_k))$	See Fig. 4.16	(4.5)
λ	9.3	(4.20)
$P_{\mathrm{FA}}(\boldsymbol{z}_k^l \mid \boldsymbol{x}_k)$	47.19%	(4.6)
ω_1	0.41	(4.29)
μ_1	$[2.75\ 4.32]^\top$	(4.29)
Σ_1	$\begin{bmatrix} 2.31 & 0.24 \\ 0.24 & 0.22 \end{bmatrix}$	(4.29)
ω_2	0.18	(4.29)
μ_2	$[10.96\ 13.38]^\top$	(4.29)
Σ_2	$\begin{bmatrix} 52.82 & 33.44 \\ 33.44 & 27.93 \end{bmatrix}$	(4.29)
ω_3	0.03	(4.29)
μ_3	$[16.01\ 22.42]^\top$	(4.29)
Σ_1	$\begin{bmatrix} 195.57 & 112.94 \\ 112.94 & 177.9 \end{bmatrix}$	(4.29)
ω_4	0.38	(4.29)
μ_4	$[3.89\ 6.23]^\top$	(4.29)
Σ_4	$\begin{bmatrix} 5.96 & 1.12 \\ 1.12 & 1.48 \end{bmatrix}$	(4.29)
$\lambda_{n_{\mathrm{d}}}$	0.5124	(4.25)

Table 4.2: *The parameters used in the SLAM experiments.*

	Average Final position error	
	PHD-SLAM	MH-FastSLAM
Constant $P_{\mathrm{D}}(\boldsymbol{m}_k^j \mid \boldsymbol{x}_k)$	1.36m	1.07m
Proposed $P_{\mathrm{D}}(\boldsymbol{m}_k^j \mid \boldsymbol{x}_k)$	1.33m	0.68m
Using descriptor	0.53m	0.67m

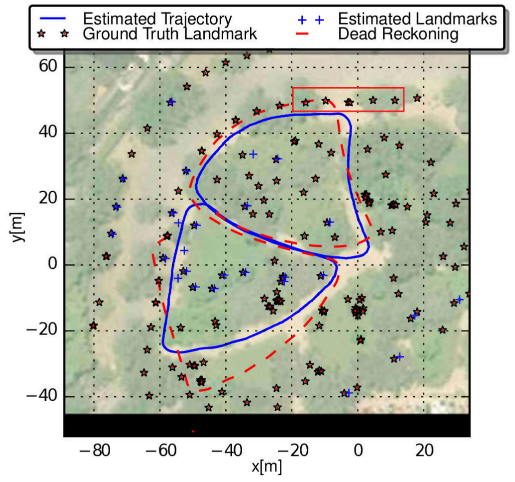
Table 4.3: *Average error in the final position (averaged over 5 monte carlo runs).*

are shown in Figs. 4.21a and 4.21b, in which optimal performance in each case occurred for $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k) = 0.7$ and 0.5 respectively.

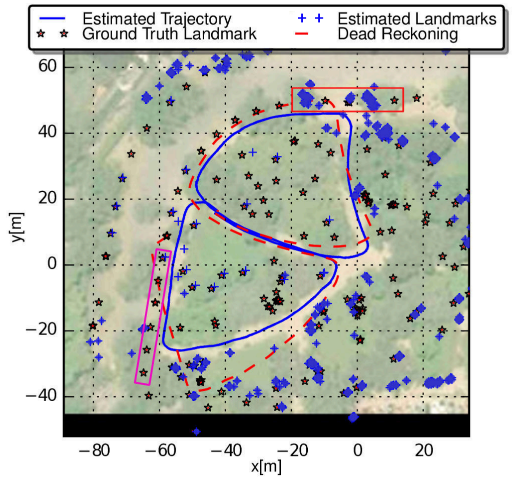
The variable probability of detection versions of RB-PHD-SLAM and MH-FastSLAM, which use (4.5) and Fig. 4.16 are shown in Figs 4.21c and 4.21d respectively, and the modified versions, which include feature descriptor information from (4.29) and (4.30), are shown in Figs. 4.22a and 4.22b, respectively.

In the case of RB-PHD-SLAM, visually, the map estimates of both the proposed algorithms appear to be better than the algorithm using a constant probability of detection, which removes most features from the map as soon as they stop being detected. This is because of the probable mismatch between the assumed constant probability of detection and the actual probability of detection. This effect can be observed particularly in the section of the maps highlighted with a red rectangle. In this rectangle, the algorithm which assumes a constant probability of detection, completely removes the estimates from its map. This occurs because the Gaussians that represent the estimates are multiplied many times by a factor of $(1 - P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k))$, with a falsely high probability of detection (see the first term in the RHS of (3.15)). However, by using the proposed detection statistics the filter is able to maintain the estimates. The trajectory estimate of the constant probability RB-PHD-SLAM filter (Fig. 4.21a) is also worse than its variable $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ counterpart (Fig. 4.21c), as the pose estimate does not exactly return to its starting position, as shown in Table 4.3. The difference between the variable probability RB-PHD-SLAM and the version modified to include descriptor information is more subtle, but can still be appreciated in Figs. 4.21c and 4.22a. Map estimates of the latter algorithm are closer to the ground truth. This can be observed mostly in the lower half of the figures. Additionally, as can be seen in Table 4.3, the final estimated robot position is closer to the starting point when the the descriptor information was used.

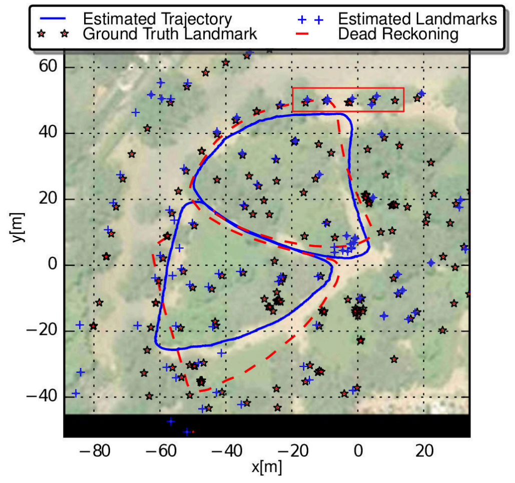
Fig. 4.21d, when compared with Fig. 4.21b, shows that with MH-FastSLAM, the proposed probability of detection also improves the map estimates, by allowing the filter to maintain the estimates of features when they are occluded. This is particularly noticeable in the row of trees located at approximately $x = -60m$ (pink rectangle), where the map estimates corresponding to these trees are maintained, while MH-FastSLAM based on a constant probability of detection



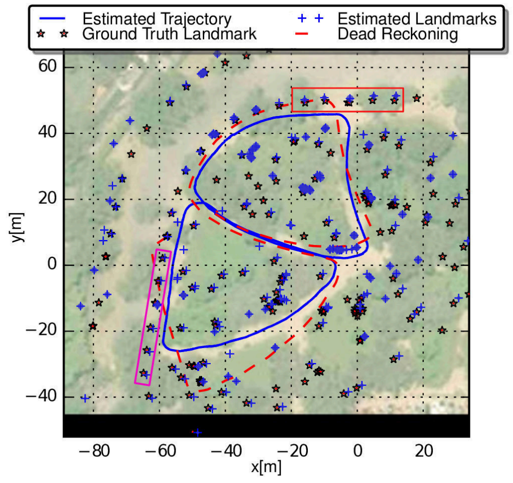
(a) *RB-PHD-SLAM with constant P_D.*



(b) *MH-FastSLAM with constant P_D.*

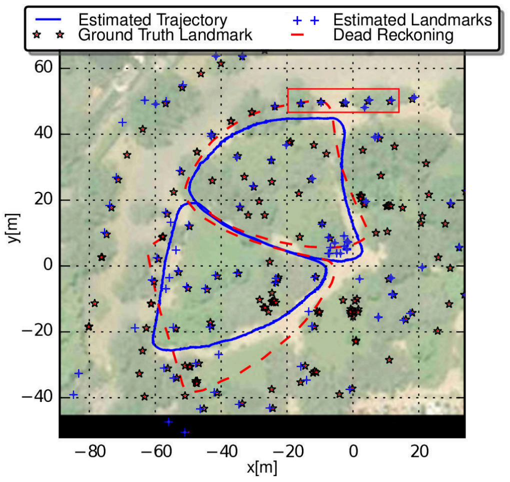


(c) *RB-PHD-SLAM with the learned $P_D(\cdot)$.*

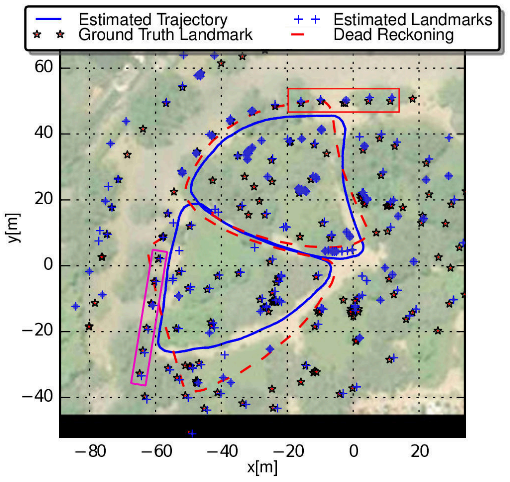


(d) *MH-FastSLAM with the learned $P_D(\cdot)$.*

Figure 4.21: *SLAM results in Parque O'Higgins, Santiago, Chile. Red rectangle: Area of the map that shows most improvement in RB-PHD-SLAM and MH-FastSLAM by including detection statistics. Pink rectangle: Area of the map that shows most improvement in MH-FastSLAM by using the detection statistics.*



(a) *RB-PHD-SLAM with the learned probability of detection and descriptor information.*



(b) *MH-FastSLAM with the learned probability of detection and descriptor information.*

Figure 4.22: *SLAM results in Parque O'Higgins, Santiago, Chile. Red rectangle: Area of the map that shows most improvement in RB-PHD-SLAM and MH-FastSLAM by including detection statistics. Pink rectangle: Area of the map that shows most improvement in MH-FastSLAM by using the detection statistics.*

failed to do so (Fig. 4.21b). This could be explained by the fact that these estimates are occluded by other trees for a long time before they exit the theoretical FoV. The effect of the variable $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ on the trajectory is less severe, but as can be seen in Table 4.3 the final point of the trajectory is closer to the starting point. Including descriptor information into MH-FastSLAM (Fig. 4.22b), removes most of the clusters of false features and improves the trajectory estimate.

To quantify the mapping errors in each case, figure 4.23 shows the OSPA errors between the estimated and ground truth maps, giving greater clarity of the mapping performance of each algorithm. In the case of both RB-PHD-SLAM and MH-FastSLAM, the metric confirms the higher map quality produced by employing the variable probability of detection and the further improved performance when using feature descriptor information[5].

4.7.2 Victoria Park dataset

To confirm the results obtained using the dataset gathered in Parque O'Higgins, the Victoria Park benchmark dataset was used [53]. The Victoria park dataset is similar to the Parque

[5]It should be noted, that the focus here is to demonstrate the improvement in both RB-PHD-SLAM and MH-FastSLAM when feature descriptor information is used, and not to compare RB-PHD-SLAM and MH-FastSLAM. Although under the relatively low clutter conditions used in the SLAM experiments here, MH-FastSLAM out performs RB-PHD-SLAM, the converse is true under higher clutter conditions [51].

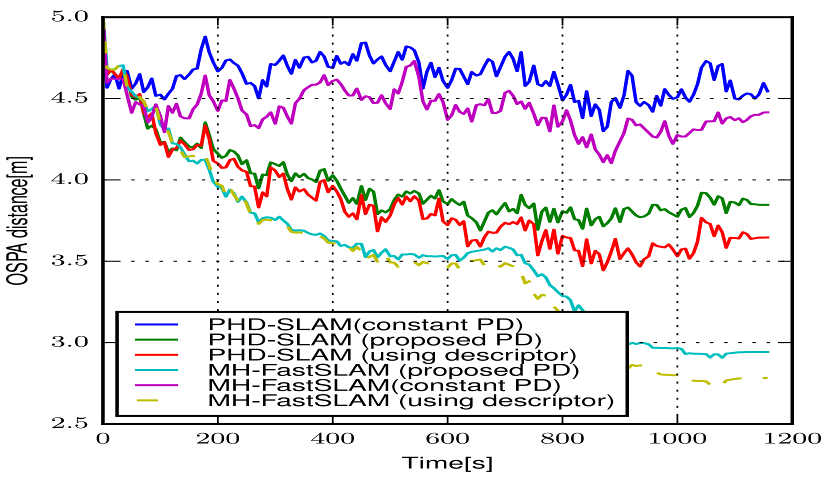


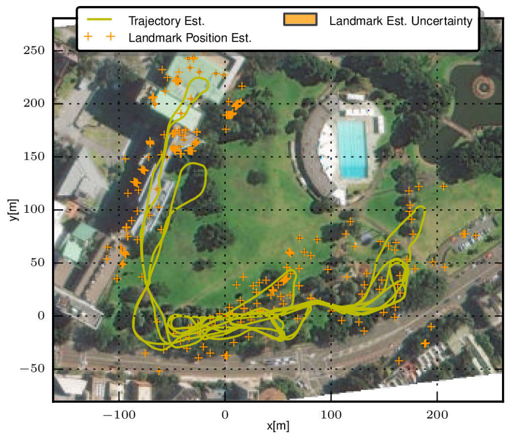
Figure 4.23: *Average OSPA errors between the ground truth map and the map estimates. Errors where averaged over 5 different runs.*

O'Higgins dataset, in the sense that trees are the natural features of the environment. This dataset provides the opportunity to test the generality of the detection statistics and descriptor likelihoods learned based on data from Parque O'Higgins with a dataset taken by different researchers and in a different location. Additionally, these statistics will be tested on the detector provided with the data set [53], which was only modified to provide the number of points used in each detection. This detector, although it also detects circular cross-sectioned objects, is different from the detector used to gather the statistics in Parque O'Higgins.

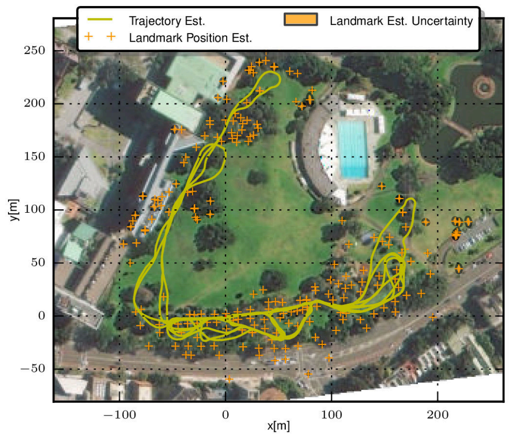
Figs. 4.24a and 4.24b show the results of the standard RB-PHD-SLAM and MH-FastSLAM filters, with constant values $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k) = 0.4$ and 0.7 respectively, determined by trial and error, to optimize filter performance in each case. The "forgetful" nature of RB-PHD-SLAM is evident, due to the probable mismatch in the estimated and actual probabilities of detection, causing it to diverge (Fig. 4.24a).

Figs. 4.24c and 4.24d show the results of RB-PHD-SLAM and MH-FastSLAM respectively, using the values of $P_{\mathrm{D}}(\boldsymbol{m}_k^j|\boldsymbol{x}_k)$ learned in Parque O'Higgins, and Figs. 4.24e and 4.24f show the effect of using the additional descriptor information. The solutions shown in Figs. 4.24c and 4.24e both converge to trajectories similar to other published SLAM solutions [6, 55].

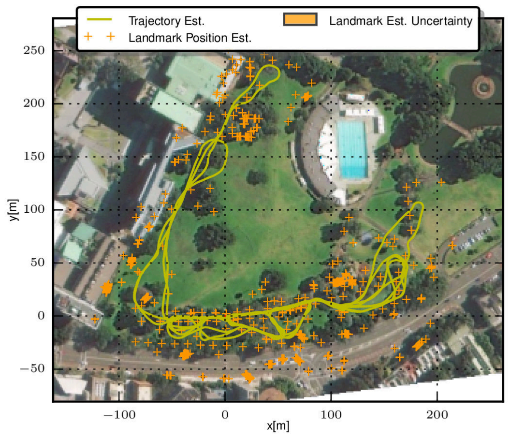
In the case of MH-FastSLAM all three versions converge to approximately the same trajectory, although using the learned detection statistics produced a map with many more features, while adding the descriptor removed a few of the added features. Due to the lack of map ground truth information and the partial absence of trajectory GPS information, it is difficult to judge the performance of the proposed modifications in this dataset. It should be noted however, that in the satellite image, some of the added features correspond to real trees, specifically in the lower right corner of the image, across the street from the park location.



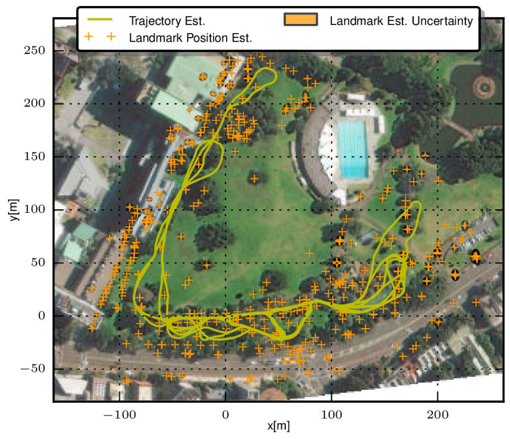
(a) *RB-PHD-SLAM with constant probability of detection.*



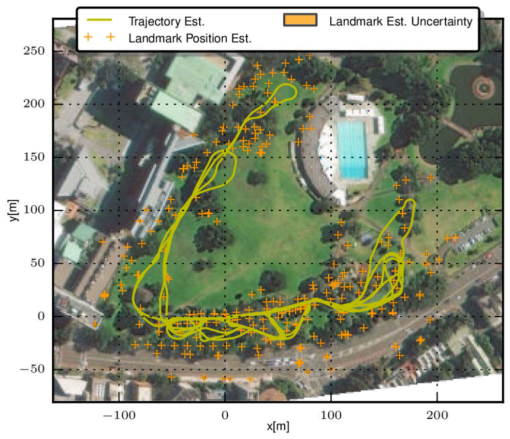
(b) *MH-FastSLAM with constant probability of detection.*



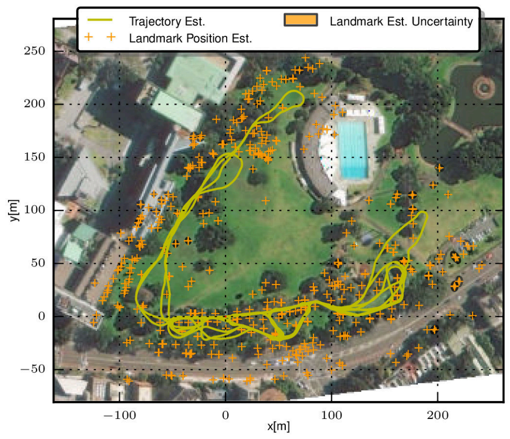
(c) *RB-PHD-SLAM with the probability of detection learned in Parque O'Higgins.*



(d) *MH-FastSLAM with the probability of detection learned in Parque O'Higgins.*



(e) *RB-PHD-SLAM with the learned probability of detection and descriptor information.*



(f) *MH-FastSLAM with the learned probability of detection and descriptor information.*

Figure 4.24: *SLAM results in Victoria Park, Sydney, Australia.*

4.8 Summary

The importance of detection, as well as the usually considered spatial, statistics upon SLAM performance was demonstrated in this chapter. Insights were derived from the target tracking community, in which the amplitude of radar data has been used as a target descriptor to provide detection statistics and likelihoods, to accompany radar range values. This concept was extended to give a generalized robotic map feature descriptor for range measuring sensors, based on the state dependent expected number of unoccluded points, available via ray casting. The difference between the actual and predicted number of range points received from a feature, proved to be a useful approximation of a sufficient statistic to aid feature-based SLAM, while accounting for map feature occlusions. SLAM simulations, with known map ground truth as well as outdoor SLAM experiments verified the importance of including detection statistics into SLAM algorithms. It was particularly encouraging, that the generation of detection statistics for SLAM maps based on tree locations, learned in one park location (Santiago, Chile), were able to provide improvements to SLAM algorithms using a data set and a different detector in another location (Sydney, Australia). The SLAM algorithms tested were vector-based MH-FastSLAM in conjunction with a binary Bayes filter, which utilized the derived detection statistics, and RFS-based glsrb-GM-PHD-SLAM, which incorporates the statistics directly into its set state-based Bayesian recursion.

Chapter 5

PSO-RFS-SLAM

5.1 Introduction

The facts presented in chapter 3 provide a compelling reason to derive a maximum likelihood RFS-based SLAM method, which should result in a superior performance both for large datasets and under high clutter and detection uncertainty. In this chapter an initial version of such a method will be presented. By maximizing the RFS-based measurement likelihood, this chapter demonstrates that Maximum Likelihood (ML) SLAM is possible without the need for external data association algorithms. Fundamentally, it will be shown that RFS-based ML-SLAM converges to an equivalent solution to its traditional vector-based counterpart, but without the need for correct data association information. The validity of this approach will be shown through the solution of 1D and 2D simulated datasets.

In Section 5.2 the PSO algorithm is adapted to the RFS-SLAM problem. Results of 1D and 2D simulations are shown in Section 5.3, and Section 5.4 summarizes the chapter.

5.2 Applying PSO to SLAM

To maximize the likelihood from RFS-based equation (3.25) using a PSO approach, a modified version of the SPSO-2007 algorithm is used. First, for numerical stability, the log-likelihood (3.28) is used. A set of particles is created, each with a state consisting of a trajectory and a map:

$$\boldsymbol{x}^i_j = (\mathbf{x}^i_{0:k}, \mathcal{M}^i). \tag{5.1}$$

5.2.1 Particle initialization

Given that the search space is not constrained as in the SPSO-2007 definition, to initialize the particle set, the robot motion model is used to generate a random dead reckoning trajectory. Once a particle has a trajectory for each of the measurements obtained, a new landmark is created with probability P_{init}. Landmarks are created with the inverse measurement model and the already initialized robot pose, and sampling noise from the landmark uncertainty generated from the inverse measurement model.

5.2.2 Particle motion

In most PSO definitions, the state is a vector. This means that equation (2.59) can be directly applied to the robot trajectory $\mathbf{x}_{0:k}$, however, the map $\mathcal{M}$ is a set, thus (2.59) needs to be adapted.

We therefore define the velocity of a set by attaching a velocity component to each vector element of the set. Then, to modify the velocity of a set we use a method inspired by the OSPA metric [56]. Map elements are associated by linear assignment as if calculating the OSPA metric, and then the velocity of the associated elements can be calculated directly, using equation (2.59). For the unassociated measurements a new parameter ϕ_{card} is added. Then, if a map element in $\boldsymbol{x}_j^i$ is not associated it will be eliminated with probability equal to ϕ_{card}, and unassociated map elements in $\boldsymbol{l}_j^i$ and $\boldsymbol{g}_j^i$ will be added to $\boldsymbol{x}_j^i$ with probability ϕ_{card}. This parameter is given a small value (e.g. $\phi_{card} \simeq 0.15$). As an added modification, even associated features can be removed with a smaller probability ($\simeq 0.01$). This small random move is designed to stop the optimizer converging to a local optima.

5.2.3 Gradient-based optimization step

Given the high correlation between states, which characterizes the SLAM problem, when the particle motion is applied to the trajectory, the measurement likelihood usually decreases. To compensate for this and to improve the convergence properties of PSO, a gradient-based optimization step is performed for every particle every K_{opt} steps ($K_{\text{opt}} \in [10, 100]$).

To maximize the likelihood from equation (3.25) the log-likelihood (3.28) is used. The summation on the left of (3.28) is a traditional non-linear least squares optimization problem (assuming Gaussian noise in $\boldsymbol{g}(\cdot)$), however, in the summation on the right, the $\log(\cdot)$ does not cancel with the exponentials. For the left summation, the gradient can be calculated in a straightforward manner, but for the summation on the right we obtain

$$\frac{\partial}{\partial \boldsymbol{x}_k} \log\left(\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)\right) = \frac{\frac{\partial}{\partial \boldsymbol{x}_k} \sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)}. \tag{5.2}$$

If the single feature measurement model is Gaussian and the clutter rate and probability of detection are assumed constant then the gradient $\frac{\partial}{\partial \boldsymbol{x}_k} \log(\cdot)$, defined in (5.2), would be

$$\frac{\partial}{\partial \boldsymbol{x}_k} \log\left(\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)\right) = \frac{1}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)} \sum_{\theta} \frac{\partial}{\partial \boldsymbol{x}_k} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right) \tag{5.3}$$

$$= \frac{1}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)} \sum_{\theta} \left[\frac{\partial}{\partial \boldsymbol{x}_k} \prod_{\boldsymbol{z}_i^j \in \mathcal{Z}_i^{A_\theta}} \mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right)\right], \tag{5.4}$$

$$\tag{5.5}$$

where $\Omega_{\boldsymbol{z}^j}$ is the square root of the information matrix of measurement $\boldsymbol{z}^j$. The information matrix of a distribution is defined as the inverse of its covariance matrix. Applying the product rule gives

$$= \frac{1}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)} \sum_{\theta} \left[\prod_{\boldsymbol{z}_i^j \in \mathcal{Z}_i^{A_\theta}} \mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right) \frac{\frac{\partial}{\partial \boldsymbol{x}_k} \mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right)}{\mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right)}\right] \tag{5.6}$$

$$= \frac{1}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)} \sum_{\theta} \left[p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right) \frac{\frac{\partial}{\partial \boldsymbol{x}_k} \mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right)}{\mathcal{N}\left(\boldsymbol{z}_i^j; \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)), \Omega_{\boldsymbol{z}^j}\right)}\right] \tag{5.7}$$

$$= \frac{1}{\sum_{\theta} p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right)} \sum_{\theta} \left[p\left(\mathcal{Z}_i \middle| \boldsymbol{x}_k, \mathcal{M}, \theta\right) \sum_{\boldsymbol{z}_i^j \in \mathcal{Z}_i^{A_\theta}} \Omega_{\boldsymbol{z}^j}(\boldsymbol{z}_i^j - \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)) \frac{\partial}{\partial \boldsymbol{x}_k} \hat{\boldsymbol{z}}_i^j(\boldsymbol{m}^{\theta(j)}, \boldsymbol{x}_k)\right]. \tag{5.8}$$

Therefore, we obtain a weighted average of the traditional non-linear least squares, weighted by the RFS measurement likelihoods. However, the non-log likelihoods still need to be evaluated, which can cause numerical instabilities, especially if the initialization point is too far away from the real values of the map and trajectory (as occurs most of the time with random initialization). Therefore to increase the numerical stability, the log-sum-exp trick is used.

The log-sum-exp trick

Adding a log-exp pair to the right-hand side term in equation (3.28) gives

$$
\begin{aligned}
l\left(\mathcal{Z}_{0:k}\middle|\boldsymbol{x}_{0:k},\boldsymbol{m},\boldsymbol{u}_{0:k-1}\right) &= \sum_{i=1}^{k}\log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1})\right) \\
&+\sum_{i=0}^{k}\log\left(\sum_{\theta}\exp(\log(p\left(\mathcal{Z}_i\middle|\boldsymbol{x}_i,\mathcal{M},\theta\right)))\right) \ . \qquad (5.9)
\end{aligned}
$$

Expanding $p\left(\mathcal{Z}_i\middle|\boldsymbol{x}_i,\mathcal{M},\theta\right)$ on the right hand side of (5.9):

$$
\begin{aligned}
\text{rhs}(5.9) &= \sum_{i=1}^{k}\log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1})\right)+\sum_{i=0}^{k}\log\Bigg(\sum_{\theta}\exp\Bigg[\\
&\log\Bigg(\frac{\prod_{\boldsymbol{z}_i^j\in\mathcal{Z}_i}\kappa(\boldsymbol{z}_i^j)}{\exp(\int\kappa(\boldsymbol{z})d\boldsymbol{z})}\prod_{\boldsymbol{m}^j\in\mathcal{M}}(1-P_{\mathrm{D}}(\boldsymbol{m}^j|\boldsymbol{x}_i)) \\
&\prod_{\boldsymbol{z}_i^j\in\mathcal{Z}^{A^\theta}}\frac{P_{\mathrm{D}}(\boldsymbol{m}^{\theta(j)}|\boldsymbol{x}_i)}{(1-P_{\mathrm{D}}(\boldsymbol{m}^{\theta(j)}|\boldsymbol{x}_i))\kappa(\boldsymbol{z}_i^j)}p\left(\boldsymbol{z}_i^j\middle|\boldsymbol{x}_i,\boldsymbol{m}^{\theta(j)}\right)\Bigg)\Bigg]\Bigg) \ . \qquad (5.10)
\end{aligned}
$$

Turning the log of the products into sums gives:

$$
\begin{aligned}
l\left(\mathcal{Z}_{0:k}\middle|\boldsymbol{x}_{0:k},\boldsymbol{m},\boldsymbol{u}_{0:k-1}\right) &= \sum_{i=1}^{k}\log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1})\right) \\
&+\sum_{i=0}^{k}\log\Bigg(\sum_{\theta}\exp\Bigg[\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}_i}\log(\kappa(\boldsymbol{z}_i^j))-\int\kappa(\boldsymbol{z})d\boldsymbol{z} \\
&+\sum_{\boldsymbol{m}^j\in\mathcal{M}^{A^\theta}}\log(1-P_{\mathrm{D}}(\boldsymbol{m}^j|\boldsymbol{x}_i))+\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}^{A^\theta}}\operatorname{logit}(P_{\mathrm{D}}(\boldsymbol{m}^{\theta(j)}|\boldsymbol{x}_i)) \\
&-\log(\kappa(\boldsymbol{z}_i^j))+l\left(\boldsymbol{z}_i^j\middle|\boldsymbol{x}_i,\boldsymbol{m}^{\theta(j)}\right)\Bigg]\Bigg) \ , \qquad (5.11)
\end{aligned}
$$

where $l\left(\cdot|\cdot\right)$ is the single-feature log-likelihood, and the logit function is defined as $\operatorname{logit}(x)=\frac{\log(x)}{1-\log(x)}$. Then the most likely data association can be taken out of the log function, by subtracting its value in every exponential giving:

$$
\begin{aligned}
& l\left(\mathcal{Z}_{0:k}\middle|\boldsymbol{x}_{0:k},\boldsymbol{m},\boldsymbol{u}_{0:k-1}\right)=\sum_{i=1}^{k}\log\left(\boldsymbol{g}(\boldsymbol{x}_i|\boldsymbol{x}_{i-1},\boldsymbol{u}_{i-1})\right)\\
& +\sum_{i=0}^{k}\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}_i}\Bigg[\log(\kappa(\boldsymbol{z}_i^j))+\int\kappa(\boldsymbol{z})d\boldsymbol{z}\\
& +\sum_{\boldsymbol{m}^j\in\mathcal{M}}\log(1-P_{\mathrm{D}}(\boldsymbol{m}^j|\boldsymbol{x}_i))+\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}^{A^{\theta}_{\max}}}\operatorname{logit}(P_{\mathrm{D}}(\boldsymbol{m}^{\theta_{\max}(j)}|\boldsymbol{x}_i))\\
& -\log(\kappa(\boldsymbol{z}_i^j))+l\left(\boldsymbol{z}_i^j\middle|\boldsymbol{x}_i,\boldsymbol{m}^{\theta_{\max}(j)}\right)\Bigg]\\
& +\log\Bigg(\sum_{\theta}\exp\Bigg[\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}^{A^{\theta}}}\operatorname{logit}(P_{\mathrm{D}}(\boldsymbol{m}^{\theta(j)}|\boldsymbol{x}_i))-\log(\kappa(\boldsymbol{z}_i^j))\\
& +l\left(\boldsymbol{z}_i^j\middle|\boldsymbol{x}_i,\boldsymbol{m}^{\theta(j)}\right)-\sum_{\boldsymbol{z}_i^j\in\mathcal{Z}^{A^{\theta}_{\max}}}\operatorname{logit}(P_{\mathrm{D}}(\boldsymbol{m}^{\theta_{\max}(j)}|\boldsymbol{x}_i))\\
& -\log(\kappa(\boldsymbol{z}_i^j))+l\left(\boldsymbol{z}_i^j\middle|\boldsymbol{x}_i,\boldsymbol{m}^{\theta_{\max}(j)}\right)\Bigg]\Bigg]\Bigg)\ .
\end{aligned}
\tag{5.12}
$$

This objective function can now be interpreted as the log-likelihood using the most likely data association $\theta_{\max}$, which is calculated using the Hungarian method, plus a correction factor, which accounts for all other association possibilities. This correction factor approaches zero whenever only one data association is likely. Conversely, the correction factor will be nonzero whenever multiple data associations are likely. Note therefore, that the Hungarian method is only necessary in order to rewrite equation (5.11) in terms of differences between the log-likelihoods of the associations and the most likely association, resulting in a more numerically stable form. This avoids the calculation of exponentials of large negative values. This function is then optimized using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. Note that only the spatial position of the landmarks and poses of the robot are optimized, meaning that during this step the map size does not change. However, the estimate of the map size is optimized in the PSO particle motion step.

5.3 Simulated SLAM Results

5.3.1 Proof of concept for RFS maximum likelihood SLAM in 1D

To show that using an RFS approach to maximum likelihood SLAM is feasible, a 1D simulation is carried out and compared with a traditional method using the correct data association. In this simulation, a 1D robot moves through a 1D map, observing landmarks and its own odometry. Both of these measurements are corrupted by zero mean Gaussian noise. To be able to use a traditional least squares random vector method, measurements are simulated with a label representing the landmark that generated it. These labels are then used by the traditional method for measurement detection to landmark association. On the contrary, the RFS formulation ignores these labels. A constant probability of detection of 0.9 and a clutter intensity of 0.01 was used to generate missed detections and Poisson distributed false alarms in the robot's field of view. Measurements were generated with a spatial variance of $10^{-4}[\mathrm{m}^2]$, while odometry readings used a variance of $10^{-2}[\mathrm{m}^2]$.

Figures 5.1 and 5.2 show an example solution to this problem, solved by maximizing the traditional measurement likelihood and the RFS-based measurement likelihood from (3.25), respectively. As can be seen from the figures, both solutions converge to the ground truth trajectory and map. It is important to note that, in addition to not having the data association, the RFS-based solution is also solving for the map size, making the optimization problem partly integer, which is significantly more difficult than solving only for the positions of the map elements. Figure 5.3 shows the robot position error of both algorithms, averaged over 5 Monte Carlo runs. Both errors are of the same order of magnitude.

5.3.2 RFS maximum likelihood in 2D

To demonstrate the validity of RFS ML SLAM further, a 2D simulation was carried out and compared with the traditional vector-based approach. Measurements were simulated with a 0.9 probability of detection and a clutter intensity of 0.001 in the sensor's field of view. Measurements were generated with a range variance of $10^{-4}[\mathrm{m}^2]$ and a bearing variance of $10^{-4}[\mathrm{rad}^2]$, while odometry readings used a variance of $2 \times 10^{-4}[\mathrm{m}^2]$, both in the x and y directions, and an angular variance of $2 \times 10^{-4}[\mathrm{rad}^2]$. As can be seen from figures 5.4 and 5.5, both trajectories and maps converge to a solution very close to ground truth. Further, both solutions are similar to each other, presenting a small translational error in the same direction. This error may be because of the random nature of the measurements, meaning that the most likely trajectory is not necessarily the correct one. Figure 5.6 shows the x and y positional errors (averaged over 5 Monte Carlo runs) between the estimates and the simulated ground truth. As can be seen

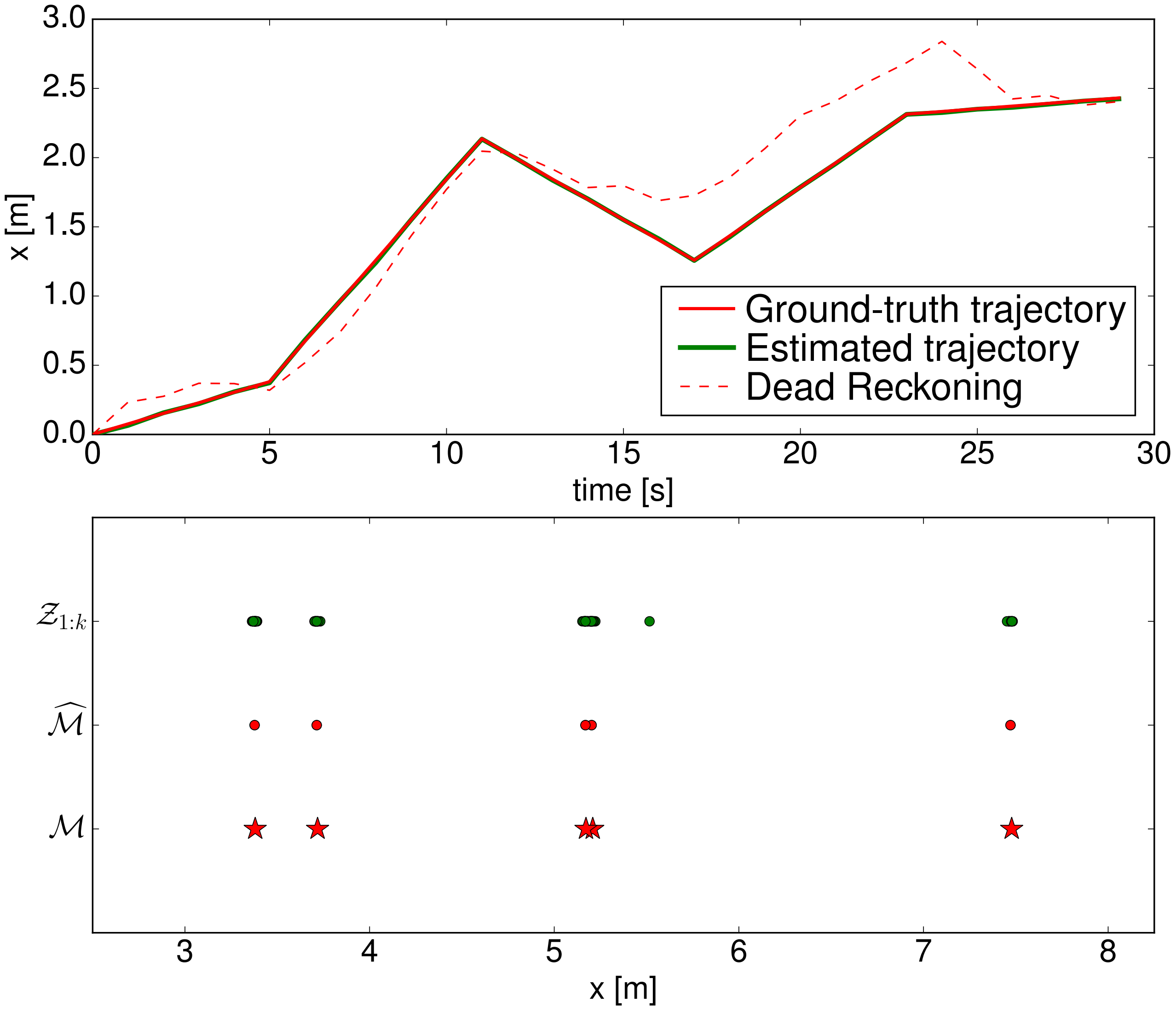


Figure 5.1: The solution to a 1D SLAM problem using the traditional least squares approach. The ground truth, estimated, and dead reckoning trajectories are plotted as red, green, and dashed red lines, respectively (top). In the lower graph, the ground truth map is shown as red stars, the estimated map as red points, and all the measurements are shown as green points.

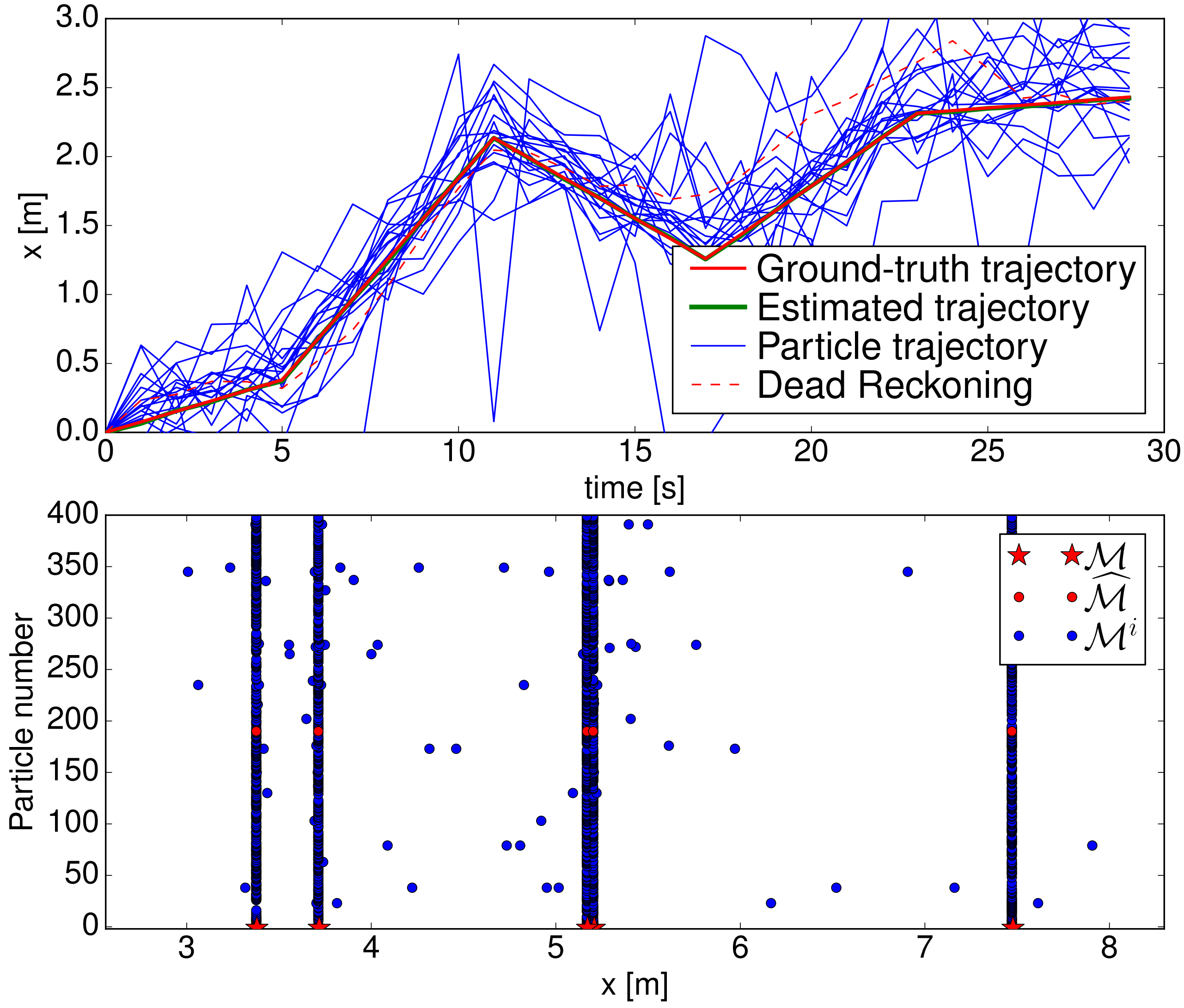


Figure 5.2: The solution to a 1D SLAM problem by maximizing the RFS measurement likelihood using an adapted PSO method. The particle, ground truth, estimated, and dead reckoning trajectories are plotted as blue, red, green, and dashed red lines, respectively (top). Note that most particles have converged to the ground truth so they appear almost superimposed. The noisy trajectories that can be seen are a small fraction of the particles. In the lower graph, the ground truth map is shown as red stars (at $y = -1$), the estimated map is shown as red points (at $y =$ index of the most likely particle) and the estimated map of each particle is shown as blue points plotted at $y =$ index of each particle.

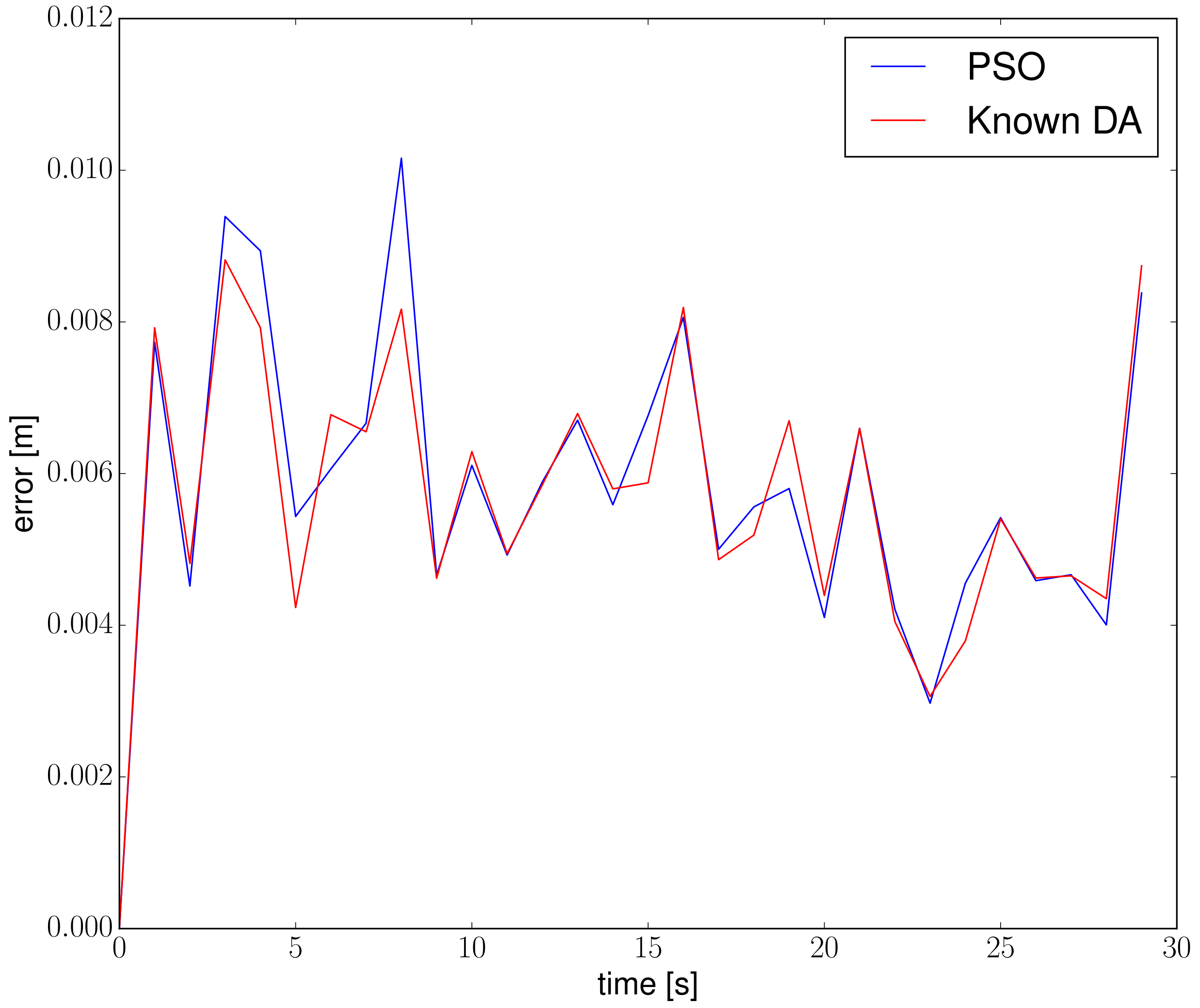


Figure 5.3: The absolute robot position error of the maximum likelihood solution obtained through the traditional vector-based, least squares, approach plotted in red, and the RFS-based maximum likelihood solution plotted in blue. Errors were averaged over 5 Monte Carlo runs. As can be seen, both errors are of the same order of magnitude.

from the figure, again both algorithms yield very similar error curves, to the point that they are closer to each other than to the ground truth.

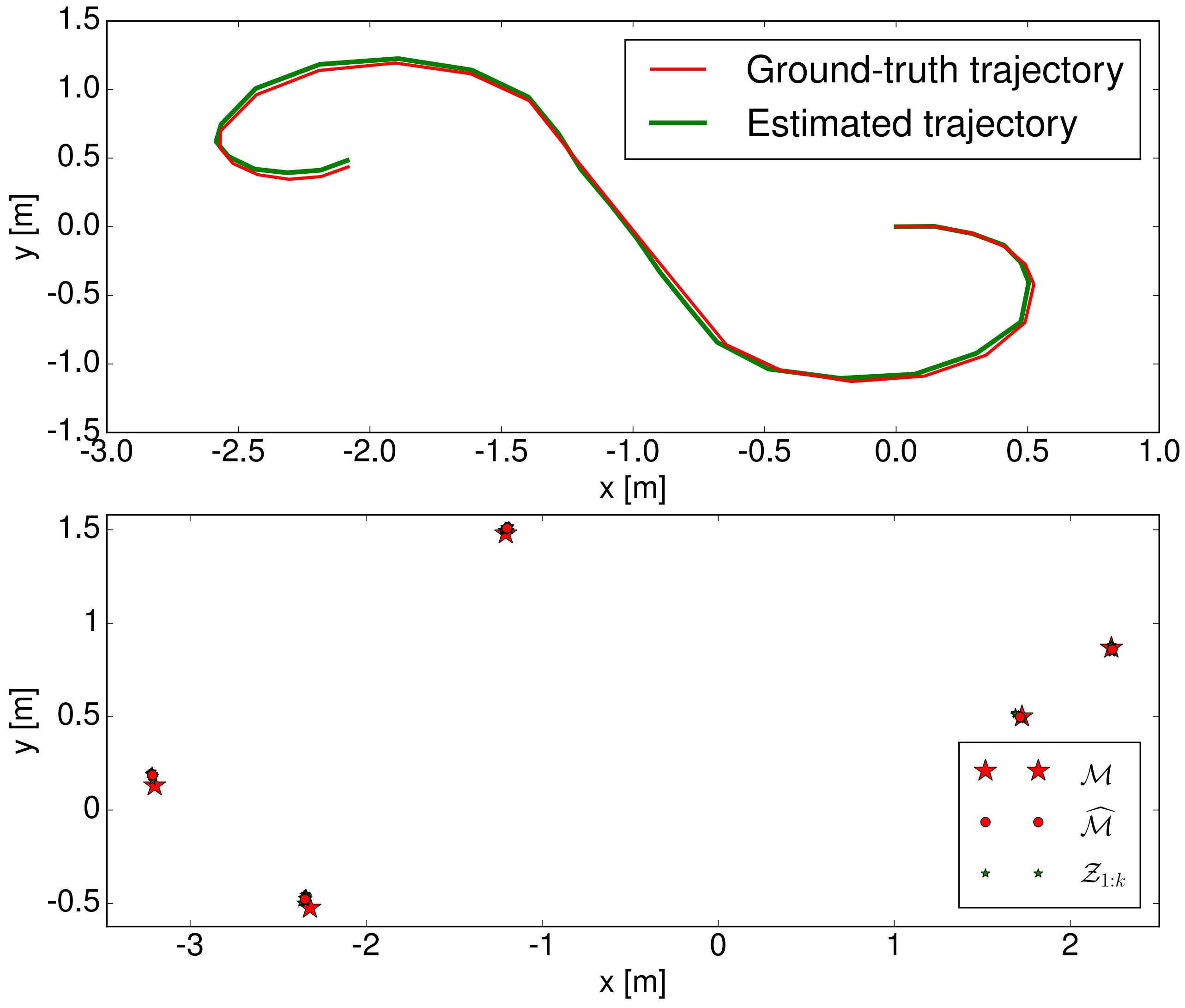


Figure 5.4: The solution to a 2D SLAM problem using the traditional least squares method. The ground truth and estimated trajectories are plotted as red and green lines, respectively (top). In the lower graph, the ground truth map is shown as red stars, the estimated map is shown as red points and all the measurements are shown as small green stars (mostly being occluded by the markers for the estimated landmarks).

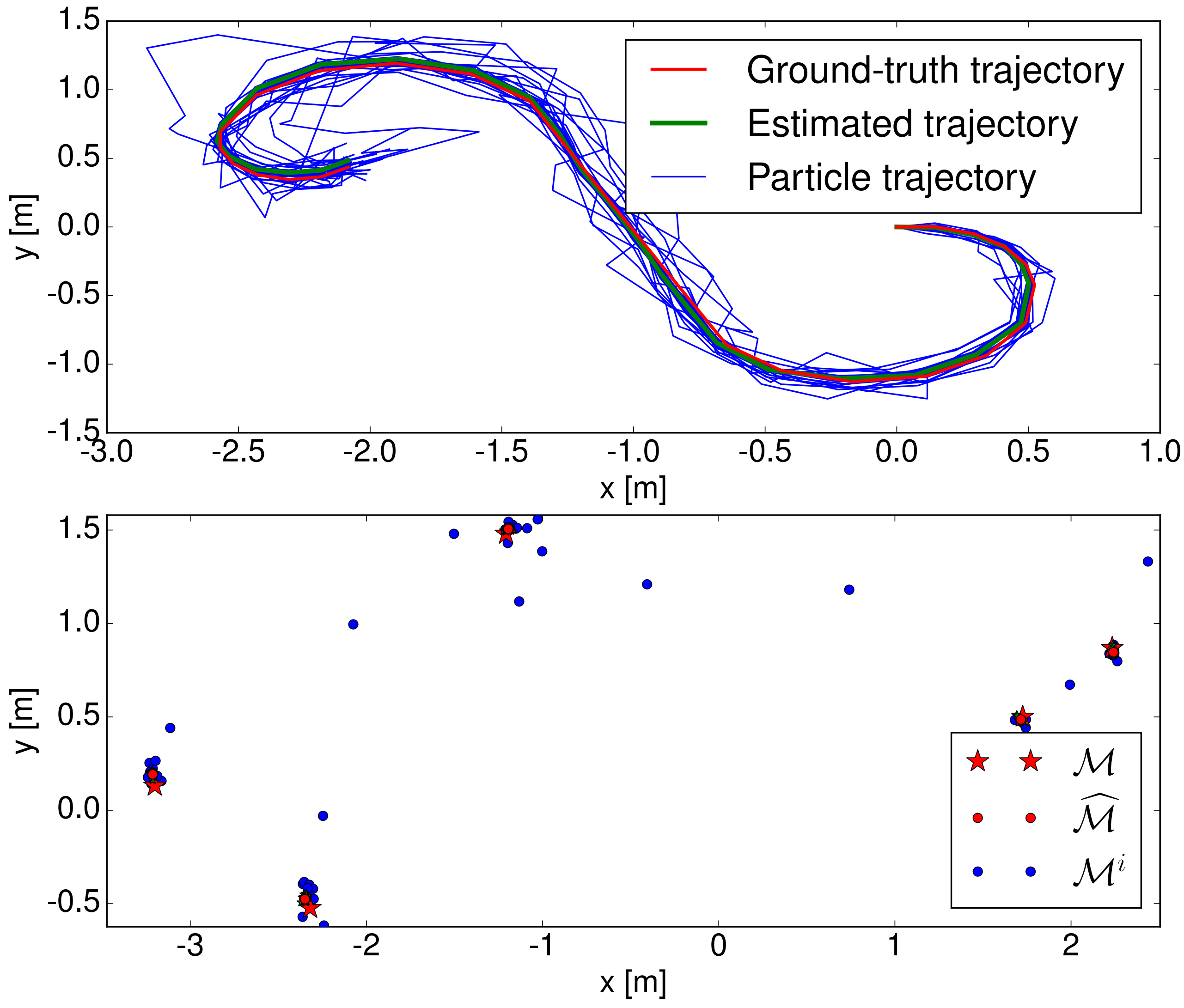


Figure 5.5: The solution to a 2D SLAM problem by maximizing the RFS measurement likelihood using an adapted PSO method. The particle, ground truth and estimated trajectories are plotted as blue, red, and green lines, respectively (top). Note that most of the particles have converged to the ground truth so they are shown superimposed. In the lower graph, the ground truth map is shown as red stars, the estimated map is shown as red points and the estimated map of each particle is shown as blue points.

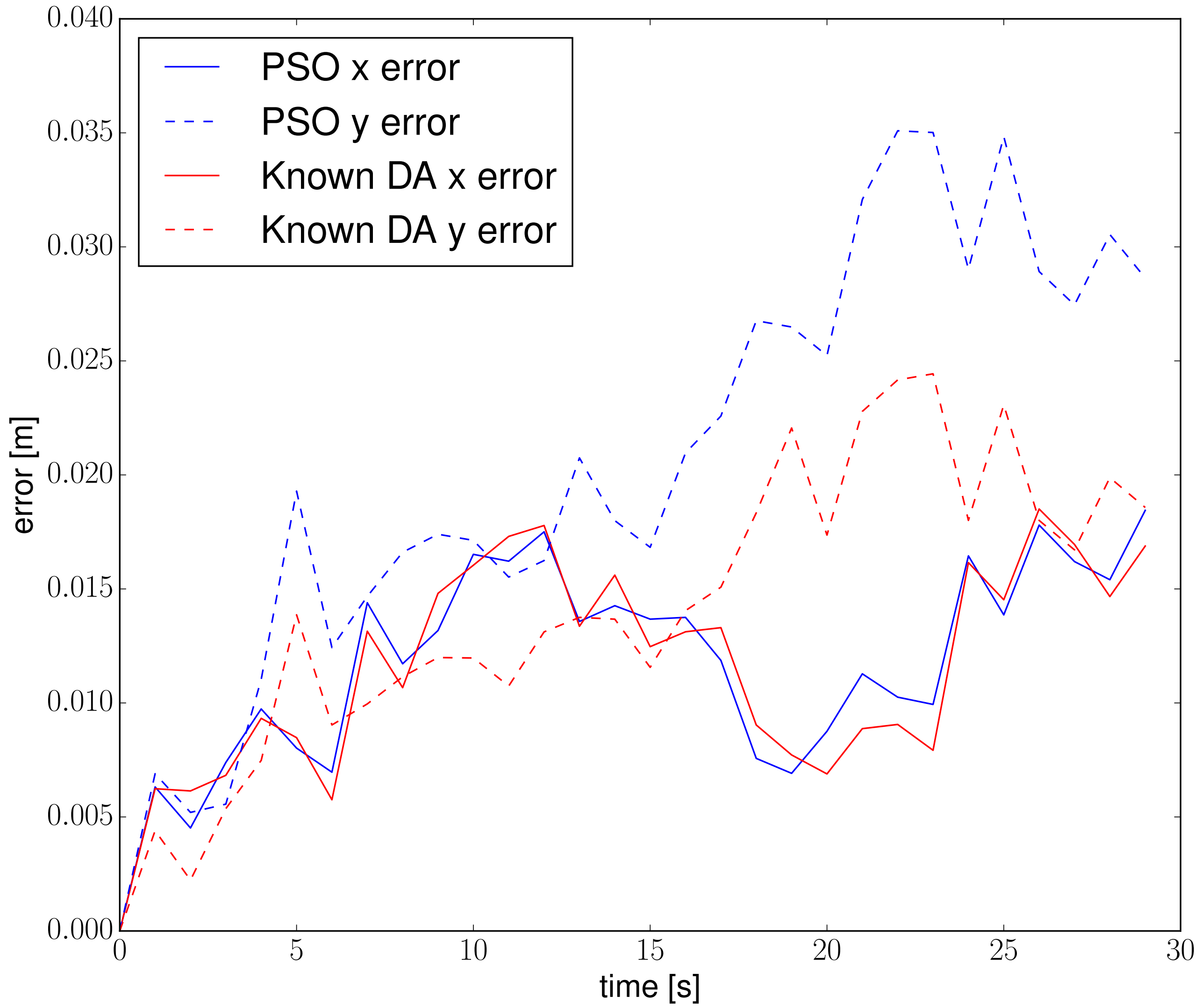


Figure 5.6: The absolute robot position error of the maximum likelihood solution obtained through the traditional vector-based, least squares, approach plotted in red, and the RFS-based maximum likelihood solution plotted in blue. The errors in the x variable are plotted as continuous lines and the errors in the y variable are plotted as dashed lines. Both error curves are similar.

5.4 Summary

In this Chapter, an RFS-based maximum likelihood solution to SLAM has been introduced. The likelihood function to maximize was presented using RFS theory and the PSO algorithm was adapted to this problem and used together with gradient-based methods. The optimization of this function was compared to the use of non-linear least squares solvers on the traditional likelihood function which assumes known data association and no clutter, using 1D and 2D simulated SLAM datasets. It was shown that the optimization of this new function can converge to the solution obtained with known data association, even though no such information was provided to the ML-RFS-SLAM algorithm. The ability to perform robust SLAM, without the necessity of fragile data association decisions, opens avenues for significant research. Algorithms that maximize the RFS measurement likelihood and simultaneously offer computationally tractable solutions could vastly increase the robustness of SLAM.

In chapter 6 an improved solution, with a maximum a posteriori approach, will address the computational complexity of the optimization method so that the RFS maximum likelihood SLAM concept can be successfully applied to larger real-world datasets.

Chapter 6

RFS-based Non-linear Least Squares Optimization Approach

6.1 Introduction

The previous exploration of the batch RFS approach has shown the limitations of general optimization methods, such as PSO when compared to the specialized algorithms that are used in the SLAM graph optimization approach (e.g. Levenberg Marquardt). With the objective of using these algorithms to exploit the correlation between the states being estimated, and its sparse nature (i.e. the sparse information matrix) this section will therefore introduce an RFS-based formulation which includes these correlations, so that state of the art optimization approaches can be used as a part of the proposed algorithm.

We know from the robotics literature that, given the correct data association, the SLAM problem can be solved by the use of a non-linear least squares solver. This can be modified to be robust to a limited number of outliers by changing the function being minimized from the squared error function to an outlier tolerating error function, such as the Huber function [57].

Solving the SLAM problem in this way allows the linearisation point of the non-linear likelihood function to be updated when the estimated state changes significantly. In contrast to filtering approaches, in which the likelihood function at time k is linearised using the information up to time k (usually by linearising at the predicted state), batch approaches can update this linearisation point using information from measurements taken at times beyond k. It also enables the use of sparse matrix methods to exploit the sparsely connected measurement graph typical in the SLAM formulation.

Therefore in this section, the SLAM state is modeled as a joint distribution, representing the vector-valued trajectory and the set-valued map, referred to as the Vector-Generalized

Labeled Multi Bernoulli (V-GLMB) distribution. This distribution has components that can be calculated using state-of-the-art methods such as the g2o solver [21]. Using this model, an optimization approach can still be used to maximize the likelihood from (3.28). The V-GLMB distribution will be introduced and then used to generate a SLAM formulation in the following sections, together with results comparing V-GLMB-SLAM to state-of-the-art solutions.

6.2 A Joint Vector-Bernoulli distribution

To provide a SLAM solution which has the advantages of both batch and RFS-based approaches, we first consider the simplest possible set model, in which the map is a random set with at most one element. We define a Joint Vector-Bernoulli distribution as a distribution on the tuple $\chi_K = (\mathbf{x}_{0:K}, \mathcal{M})$. The distribution is defined as $\pi(\chi_K)$,

$$\pi(\chi_K) = \begin{cases} (1-r)\pi_0(\mathbf{x}_{0:K}) & \chi_K = (\mathbf{x}_{0:K}, \varnothing) \\ r\pi_1(\mathbf{x}_{0:K}, m) & \chi_K = (\mathbf{x}_{0:K}, \{m\}) \\ 0 & \text{else} \end{cases} \tag{6.1}$$

where r is the probability of existence of feature m, $\pi_0(\mathbf{x}_{0:K})$ is a vector distribution on the vehicle trajectory, given that the map is empty, $\pi_1(\mathbf{x}_{0:K}, m)$ is a joint vector distribution on the vehicle trajectory (vector) and map element m (set) tuple, given that the map contains only a single element m. Note that the vector state $\mathbf{x}_{0:K}$ and element of the set m can, in general, be correlated either linearly or otherwise.

It can be shown that this distribution integrates to 1 over the joint vector-set statespace by stating the joint integral as follows.

$$\begin{aligned}
\int_{\chi_K} \pi(\chi_K) &= \int_{\mathbf{x}_{0:K}} \int_{\mathcal{M}} \pi(\chi_K) \\
&= \int_{\mathbf{x}_{0:K}} \int_{\mathcal{M}} \begin{cases} (1-r)\pi_0(\mathbf{x}_{0:K}) & \chi_K = (\mathbf{x}_{0:K}, \varnothing) \\ r\pi_1(\mathbf{x}_{0:K}, m) & \chi_K = (\mathbf{x}_{0:K}, \{m\}) \\ 0 & \text{else} \end{cases} \\
&= \int_{\mathcal{M}} \begin{cases} (1-r) \int_{\mathbf{x}_{0:K}} \pi_0(\mathbf{x}_{0:K}) & \chi_K = (\mathbf{x}_{0:K}, \varnothing) \\ r \int_{\mathbf{x}_{0:K}} \pi_1(\mathbf{x}_{0:K}, m) & \chi_K = (\mathbf{x}_{0:K}, \{m\}) \\ 0 & \text{else} \end{cases} \\
&= \frac{1}{0!}(1-r) \int_{\mathbf{x}_{0:K}} \pi_0(\mathbf{x}_{0:K}) + \frac{1}{1!} r \int_m \int_{\mathbf{x}_{0:K}} \pi_1(\mathbf{x}_{0:K}, m) \\
&= (1-r) + r = 1
\end{aligned} \tag{6.2}$$

were we have used the fact that $\pi_0(\mathbf{x}_{0:K})$ and $\pi_1(\mathbf{x}_{0:K}, m)$ are vector distributions and therefore integrate to 1.

6.3 A Joint Vector-GLMB distribution

We can generalize the previous distribution to arbitrary set sizes by following the work of Vo and Vo [12] to extend the joint vector-set idea to a GLMB set distribution. We write a V-GLMB distribution as

$$\pi(\chi_K) = \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \ , \tag{6.3}$$

where $\Delta(\mathcal{M})$ is the distinct label indicator, defined as

$$\Delta(\mathcal{M}) = \delta_{|\mathcal{M}|}(|\mathcal{L}(\mathcal{M})|) = \begin{cases} 1 & \text{for no repeated labels in } \mathcal{M} \\ 0 & \text{else} \ , \end{cases} \tag{6.4}$$

which will be zero if there are any repeated labels in the set $\mathcal{M}$, and unity otherwise.

$$\vec{v}(\mathcal{M}) = \boldsymbol{M} \tag{6.5}$$

is a function, that we call "vectorize", which takes a labeled RFS $\mathcal{M}$, and returns a vector version of the labeled RFS using the order of the labels (this could be the elements of the set in that same label order, or it could be any fixed permutation of that order[1]). To make sure that this distribution is normalized , i.e. it integrates to 1, the component weights must comply with with

$$\sum_{L \subseteq \mathbb{L}} \sum_{c \in \mathcal{C}} w^c(L) \equiv 1. \tag{6.6}$$

A short proof of this fact is shown in Annex C.

In the case of feature-based SLAM, "labels" correspond to distinct feature/landmark identities. $\mathcal{C}$ is a finite index set. $\pi^c(\cdot)$ is a vector distribution, possibly including correlation between target/features in $\mathcal{M}$ and the vector state $\mathbf{x}_{0:k}$. In the implementation of the GLMB filter [12], i.e. the application of Bayes theorem, the index c corresponds to a single data association hypothesis between a subset of the available measurements and the elements m of the map $\mathcal{M}$, the remaining measurements being hypothesized to be false alarms.

Equation (6.3) can be expressed in δ-GLMB form as:

$$w^c(\mathcal{L}(\mathcal{M})) = \sum_{I \in \mathcal{F}(\mathbb{L})} w^{(c,I)} \delta_I(\mathcal{L}(\mathcal{M})) \ , \tag{6.7}$$

where $\delta_I(\mathcal{L}(\mathcal{M}))$ is the set-valued Kronecker delta which discards all sets $\mathcal{M}$ whose labels are not equal to I, $w^{(c,I)}$ is a real-valued weight, and $\mathcal{F}(\mathbb{L})$ is the set of all possible subsets of the label space $\mathbb{L}$. Substituting (6.7) into (6.3) yields

$$\pi(\chi_k) = \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} \sum_{I \in \mathcal{F}(\mathbb{L})} w^{(c,I)} \delta_I(\mathcal{L}(\mathcal{M})) \pi^c(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) \ . \tag{6.8}$$

However, since for each c there will only be a finite number of non-zero weights $w^{(c,I)}$, we can simplify the notation by using the index set $\mathcal{C}_k$ to index the pair (c_k, I). This means the set of existing labels I depends on the index c and is now referred to as I^c. The δ-V-GLMB distribution can then be restated as:

$$\pi(\chi_k) = \Delta(\mathcal{M}) \sum_{c_k \in \mathcal{C}_k} w^{c_k} \delta_{I^{c_k}}(\mathcal{L}(\mathcal{M})) \pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) \ . \tag{6.9}$$

Table 6.1 shows an example of a simple V-GLMB distribution with two components, a trajectory of length 1, and an index set $\mathcal{C}_k = \{0, 1\}$. This example is in 1D space and therefore both the trajectory and landmarks are 1D variables. The distribution is given by (6.9), with the parameters given by the table. This means that $k = 0$ and the trajectory is composed only

[1]Defining the "vectorize" function to use some other order (like the inverse), would change the definition of the vectorized set. However, this would not change any other aspect of the posterior probability distribution, or the resulting algorithm to approximate it

c_k	w^{c_k}	I^{c_k}	$\pi^c(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}))$	$\vec{v}(\mathcal{M})$	μ^{c_k}	Σ^{c_k}
0	0.5	$\{0\}$	$\mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^0, \Sigma^0\right)$	$\begin{bmatrix}\mathbf{x}_0\\ \mathbf{m}^1\end{bmatrix}$	$\begin{bmatrix}0.0\\ 1.0\end{bmatrix}$	$\begin{bmatrix}0.3 & 0.01\\ 0.01 & 0.2\end{bmatrix}$
1	0.5	$\{0,1\}$	$\mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^1, \Sigma^1\right)$	$\begin{bmatrix}\mathbf{x}_0\\ \mathbf{m}^1\\ \mathbf{m}^2\end{bmatrix}$	$\begin{bmatrix}0.0\\ 1.0\\ 3.0\end{bmatrix}$	$\begin{bmatrix}0.2 & 0.01 & 0.01\\ 0.01 & 0.2 & 0.02\\ 0.01 & 0.2 & 0.2\end{bmatrix}$

Table 6.1: Example of a simple V-GLMB distribution with 2 components.

of one pose $\mathbf{x}_0$, usually set to $\mathbf{x}_0 = 0.0$. Note that the length of the trajectory, being a vector, does not change throughout all the components. This distribution has just 2 components and it describes two landmarks, where landmark 1 is hypothesized to exist in every component, and landmark 2 is hypothesized to exist only in component $c_k = 1$. In this example, the distributions are Normally distributed, with correlations between both the trajectory and the landmarks.

To formulate the SLAM solution, it is possible to apply Bayes theorem to the Vector-GLMB distribution (6.9) to get the full posterior. The prediction step is

$$p\left(\chi_{k+1}\middle|\mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k+1}\right) = p\left(\chi_k\middle|\mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) \quad , \tag{6.10}$$

and the corresponding update step is

$$p\left(\chi_{k+1}\middle|\mathcal{Z}_{0:k+1}, \boldsymbol{u}_{1:k+1}\right) = \frac{p\left(\chi_k\middle|\mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) p\left(\mathcal{Z}_{k+1}\middle|\mathbf{x}_{k+1}, \mathcal{M}\right)}{p\left(\mathcal{Z}_{k+1}\right)} \quad , \tag{6.11}$$

where $p\left(\mathcal{Z}_{k+1}\right)$ is a normalization constant and does not need to be explicitly calculated. The measurement likelihood can be stated as in (3.26), rewritten here as

$$\begin{aligned} p\left(\mathcal{Z}_k\middle|\mathbf{x}_k, \mathcal{M}\right) &= p\left(\mathcal{Z}_k\middle|\mathbf{x}_k, \left\{\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^{|\mathcal{M}|}\right\}\right) \\ =& \sum_{\theta_k}\left[p_\kappa\left(\mathcal{Z}_k^{\bar{A}_{\theta_k}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_k}}} \left(1 - P_\text{D}\left(\mathbf{m}^j|\mathbf{x}_k\right)\right) \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} P_\text{D}\left(\mathbf{m}^{a_{\theta_k}(i)}|\mathbf{x}_k\right) p\left(\mathbf{z}_k^i\middle|\mathbf{m}^j, \mathbf{x}_k\right)\right] \quad , \end{aligned} \tag{6.12}$$

where the association function $a_{\theta_k}(i)$ is the inverse of the $a_{\theta_k}(j)$ that was used in (3.26)[2]. Substituting the measurement likelihood (6.12) and (6.9) into (6.11) and noting that $\pi(\chi_k) =$

[2]A single association θ_k can either be expressed as a function relating the landmark to measurement index (as in (3.26)) or relating the measurement index to the associated landmark index, as in (6.12). Since every term in the summation in (6.12) has the same set of measurements and since this set is usually much smaller than the set of landmarks, it is more convenient to use the latter formulation.

$p\left(\chi_k \middle| \mathcal{Z}_{0:k}, \boldsymbol{u}_{1:k}\right)$ yields

$$p\left(\chi_{k+1} \middle| \mathcal{Z}_{0:k+1}, \boldsymbol{u}_{1:k+1}\right) \sim \Delta(\mathcal{M}) \sum_{c_k \in \mathcal{C}_k} w^{c_k} \delta_{I^{c_k}}(\mathcal{L}(\mathcal{M})) \pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \boldsymbol{u}_{k+1}\right)$$
$$\sum_{\theta_{k+1}} \Bigg[p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j | \mathbf{x}_{k+1}\right)\right)$$
$$\prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)} | \mathbf{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_{k+1}\right) \Bigg] . \tag{6.13}$$

Moving the summation over associations θ_{k+1} yields

$$p\left(\chi_{k+1} \middle| \mathcal{Z}_{1:k+1}, \boldsymbol{u}_{1:k+1}\right) \sim \Delta(\mathcal{M}) \sum_{c_k \in \mathcal{C}_k} \sum_{\theta_{k+1}} \Bigg[w^{c_k} \delta_{I^{c_k}}(\mathcal{L}(\mathcal{M})) \pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}))$$
$$p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \boldsymbol{u}_{k+1}\right) p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j | \mathbf{x}_{k+1}\right)\right)$$
$$\prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)} | \mathbf{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i \middle| \mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) \Bigg] , \tag{6.14}$$

which is a new Joint Vector-δ-GLMB distribution of the form:

$$\pi(\chi_{k+1}) = \Delta(\mathcal{M}) \sum_{c_{k+1} \in \mathcal{C}_{k+1}} w^{c_{k+1}} \delta_{I^{c_{k+1}}}(\mathcal{L}(\mathcal{M})) \pi^{c_{k+1}}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) , \tag{6.15}$$

with the new index c_{k+1} being extended by every possible data association θ_{k+1}. The new set of indices $\mathcal{C}_{k+1}$ will be the product of the existing indices $\mathcal{C}_k$ with the set of all possible data associations Θ_{k+1}. Therefore

$$\mathcal{C}_{k+1} = \mathcal{C}_k \times \Theta_{k+1} , \tag{6.16a}$$
$$c_{k+1} = (c_k, \theta_{k+1}) , \tag{6.16b}$$
$$w^{c_{k+1}} = w^{c_k} \int \pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \boldsymbol{u}_{k+1}\right) p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \tag{6.16c}$$
$$\prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j | \mathbf{x}_{k+1}\right)\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)} | \mathbf{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i \middle| \mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) d\chi_{k+1} ,$$

where,

$$\pi^{c_{k+1}}(\mathbf{x}_{0:k+1}, \vec{v}(\mathcal{M})) = \frac{w^{c_k}}{w^{c_{k+1}}} \pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) p\left(\mathbf{x}_{k+1} \middle| \mathbf{x}_k, \boldsymbol{u}_k\right) p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_k}}\right) \ , \tag{6.16d}$$
$$\prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_\mathrm{D}\left(\mathbf{m}^j | \boldsymbol{x}_{k+1}\right)\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_\mathrm{D}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)} | \boldsymbol{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i \middle| \mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) \ .$$

With this definition, it is possible to start with an V-GLMB-SLAM prior and recursively calculate the full posterior of the V-GLMB-SLAM distribution

6.3.1 Normal distribution approximation

To apply the V-GLMB model we take equations (6.16a) to (6.16d), and model $\pi^{c_k}(\cdot)$ with a Normal distribution.

$$\pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) = \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) \ , \tag{6.17}$$

where μ^{c_k} and Σ^{c_k} are the mean and covariance of the normal distribution of component c_k, respectively. This can be updated with linearized motion and measurement models through equations (6.16a) to (6.16d) by the use of a(n) (extended) Kalman Filter for each new component c_{k+1} in $\mathcal{C}_{k+1}$. This means we would end up with an EKF-SLAM solution for each possible data association. However, as stated previously, graph-based optimization approaches have been shown to provide solutions superior to filter-based approaches for the non-linear yet sparse nature of SLAM. Therefore a graph-based approximation, yet still Gaussian, follows.

Substituting the normal distribution approximation into (6.3) yields

$$\pi(\chi_k) = \Delta(\mathcal{M}) \sum_{c_k \in \mathcal{C}_k} w^{c_k} \delta_{I^{c_k}}(\mathcal{L}(\mathcal{M})) \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) \ . \tag{6.18}$$

The resulting posterior requires the update of $\mathcal{C}_{k+1}$, c_{k+1}, $w^{c_{k+1}}$ and $\pi^{c_{k+1}}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}))$, which are given by replacing the spatial distribution $\pi^{c_k}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}))$ with the normal distribution

$\mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right)$ in (6.16):

$$\mathcal{C}_{k+1} = \mathcal{C}_k \times \Theta_{k+1} \ , \tag{6.19a}$$

$$c_{k+1} = (c_k, \theta_{k+1}) \ , \tag{6.19b}$$

$$w^{c_{k+1}} = w^{c_k} \int \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j|\mathbf{x}_{k+1}\right)\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)}|\mathbf{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i\middle|\mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) d\chi_{k+1} \ , \tag{6.19c}$$

$$\pi^{c_{k+1}}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) = \frac{w^{c_k}}{w^{c_{k+1}}} \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}\left(\mathbf{m}^j|\mathbf{x}_{k+1}\right)\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}\left(\mathbf{m}^{a_{\theta_{k+1}}(i)}|\mathbf{x}_{k+1}\right) p\left(\mathbf{z}_{k+1}^i\middle|\mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) \ . \tag{6.19d}$$

By assuming that the probability of detection $P_{\mathrm{D}}\left(\mathbf{m}^j|\mathbf{x}_{k+1}\right)$ varies slowly with both the robot pose $\mathbf{x}_{k+1}$ and the landmark location $\mathbf{m}^j$[3], we can approximate the posterior by using the value of $P_{\mathrm{D}}\left(\mathbf{m}^j|\mathbf{x}_{k+1}\right)$ at the mean μ^{c_k} of the normal distribution and assuming it is constant for the sake of integration. It will therefore be referred to as $P_{\mathrm{D}}^{(j,k+1)}$ from here on. Then:

$$\mathcal{C}_{k+1} = \mathcal{C}_k \times \Theta_k \ , \tag{6.20a}$$

$$c_{k+1} = (c_k, \theta_{k+1}) \ , \tag{6.20b}$$

$$w^{c_{k+1}} = w^{c_k} p_\kappa\left(\mathcal{Z}_{k+1}^{\bar{A}_{\theta_{k+1}}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_{k+1}}}} \left(1 - P_{\mathrm{D}}^{(j,k+1)}\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} P_{\mathrm{D}}^{(a_{\theta_{k+1}}(i),k+1)} \int \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} p\left(\mathbf{z}_{k+1}^i\middle|\mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) d\chi_{k+1} \ , \tag{6.20c}$$

$$\pi^{c_{k+1}}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M})) \sim \mathcal{N}\left(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}); \mu^{c_k}, \Sigma^{c_k}\right) p\left(\mathbf{x}_{k+1}\middle|\mathbf{x}_k, \boldsymbol{u}_{k+1}\right) \prod_{\mathbf{z}_{k+1}^i \in \mathcal{Z}_{k+1}^{A_{\theta_{k+1}}}} p\left(\mathbf{z}_{k+1}^i\middle|\mathbf{m}^{a_{\theta_{k+1}}(i)}, \mathbf{x}_{k+1}\right) \ , \tag{6.20d}$$

where (6.20d) has been stated as a similarity (without writing the normalization constant) for

[3]This approximation may not be accurate at the edge of the field of view (FoV) of the sensor, accounting for this could be an avenue for future research

brevity. Since it is a vector probability distribution, it simply needs to be normalized. Finally equations (6.20) can be used to solve the SLAM problem using any choice of the existing vector-based SLAM methods. We can use equations (6.20a) to (6.20d) to state the posterior after K updates:

$$\mathcal{C}_K = \mathcal{C}_\varnothing \times \Theta_0 \times \Theta_1 \times \cdots \times \Theta_K \ , \tag{6.21a}$$

$$c_K = (c_\varnothing, \theta_0, \cdots, \theta_K) \ , \tag{6.21b}$$

$$w^{c_K} = w^{c_{K-1}} p_\kappa\left(\mathcal{Z}_K^{\bar{A}_{\theta_K}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_K}}} \left(1 - P_{\mathrm{D}}^{(j,K)}\right) \prod_{\mathbf{z}_K^i \in \mathcal{Z}_K^{A_{\theta_K}}} P_{\mathrm{D}}^{(a_{\theta_K}(i),K)}$$
$$\int \pi^{c_{K-1}}(\mathbf{x}_{0:K-1}, \vec{v}(\mathcal{M})) p\left(\mathbf{x}_K \middle| \mathbf{x}_{K-1}, \boldsymbol{u}_K\right) \prod_{\mathbf{z}_K^i \in \mathcal{Z}_K^{A_{\theta_K}}} p\left(\mathbf{z}_K^i \middle| \mathbf{m}^{a_{\theta_K}(i)}, \mathbf{x}_K\right) d\chi_K \ , \tag{6.21c}$$

$$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \sim \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) \ . \tag{6.21d}$$

Note that the component weight w^{c_K} is still stated as a function of the previous weight $w^{c_{K-1}}$ and spatial distribution $\pi^{c_{K-1}}(\mathbf{x}_{0:K-1}, \vec{v}(\mathcal{M}))$. We can solve for the weight by replacing $\pi^{c_K}(\mathbf{x}_{0:K-1}, \vec{v}(\mathcal{M}))$ in (6.21c) using the normalized version of (6.21d):

$$w^{c_K} = w^{c_{K-1}} p_\kappa\left(\mathcal{Z}_K^{\bar{A}_{\theta_K}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_K}}} \left(1 - P_{\mathrm{D}}^{(j,K)}\right) \prod_{\mathbf{z}_K^i \in \mathcal{Z}_K^{A_{\theta_K}}} P_{\mathrm{D}}^{(a_{\theta_K}(i),K)}$$
$$\int \frac{\mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K-1} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K-1} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right)}{\int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K-1} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K-1} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right)}$$
$$p\left(\mathbf{x}_K \middle| \mathbf{x}_{K-1}, \boldsymbol{u}_K\right) \prod_{\mathbf{z}_K^i \in \mathcal{Z}_K^{A_{\theta_K}}} p\left(\mathbf{z}_K^i \middle| \mathbf{m}^{a_{\theta_K}(i)}, \mathbf{x}_K\right) d\chi_K \tag{6.22}$$

which can be restated as

$$w^{c_K} = w^{c_{K-1}} p_\kappa\left(\mathcal{Z}_K^{\bar{A}_{\theta_k}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_K}}} \left(1 - P_{\mathrm{D}}^{(j,k)}\right) \prod_{\mathbf{z}_K^i \in \mathcal{Z}_K^{A_{\theta_K}}} P_{\mathrm{D}}^{(a_{\theta_K}(i),K)}$$
$$\frac{\int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_{k-1}\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) d\chi_K}{\int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K-1} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K-1} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) d\chi_{K-1}} . \tag{6.23}$$

This recursive equation can be expanded as

$$w^{c_K} = w^{c_\varnothing} \prod_{k=0}^{K} p_\kappa\left(\mathcal{Z}_k^{\bar{A}_{\theta_k}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_k}}} \left(1 - P_{\mathrm{D}}^{(j,i)}\right) \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} P_{\mathrm{D}}^{(a_{\theta_k}(i),k)}$$
$$\int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c}, \Sigma^{c}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) d\chi_K . \tag{6.24}$$

Therefore we can state the full vector-GLMB posterior as

$$\mathcal{C}_K = \mathcal{C}_\varnothing \times \Theta_0 \times \Theta_1 \times \cdots \times \Theta_K , \tag{6.25a}$$

$$c_K = (c_\varnothing, \theta_0, \cdots, \theta_K) , \tag{6.25b}$$

$$w^{c_K} = w^{c_\varnothing} \prod_{k=0}^{K} p_\kappa\left(\mathcal{Z}_k^{\bar{A}_{\theta_k}}\right) \prod_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_k}}} \left(1 - P_{\mathrm{D}}^{(j,i)}\right) \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} P_{\mathrm{D}}^{(a_{\theta_k}(i),k)}$$
$$\int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) d\chi_K , \tag{6.25c}$$

$$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \sim \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) . \tag{6.25d}$$

6.4 Calculating the component weight

Equations (6.25a) to (6.25d) provide the full SLAM posterior. We know that when using a non-linear least squares approach, we can efficiently solve for the spatial posterior $\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$. We still need an efficient method for calculating the component weight w^{c_K} for each data association hypothesis. The posterior (6.25d) in its full form is:

$$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) = \frac{\mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c\varnothing}, \Sigma^{c\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{i=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right)}{q(c_k, \mathcal{Z}_{1:K}, \boldsymbol{u}_{1:K})} \tag{6.26}$$

where

$$q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K}) = \int \mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c\varnothing}, \Sigma^{c\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) d\chi_K \tag{6.27}$$

is the normalization factor.

Approximating the motion and measurement models, $p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right)$ and $p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right)$, by non-linear models with additive Gaussian noise

$$p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) = \mathcal{N}\left(\mathbf{x}_k; f_u(\mathbf{x}_{k-1}), \Sigma^{odo}\right) \tag{6.28}$$

$$p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) = \mathcal{N}\left(\mathbf{z}_k^i; f_z(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k), \Sigma^{z}\right) \ , \tag{6.29}$$

where $f_u(\mathbf{x}_{k-1})$ and $f_z(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k)$ are non-linear functions producing the mean of the motion and measurement models respectively. Σ^{odo} and Σ^{z} are the covariances of the motion and measurement models. Using (6.28) and (6.29), it is possible to approximate the spatial posterior with a Gaussian distribution:

$$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) = \mathcal{N}\left(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}); \mu^{MAP}, \Sigma^{MAP}\right) \tag{6.30}$$

with parameters determined by the least squares solvers. Then:

$$q(c_K, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K}) = \frac{\mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c\varnothing}, \Sigma^{c\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right) \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right)}{\mathcal{N}\left(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}); \mu^{MAP}, \Sigma^{MAP}\right)} \tag{6.31}$$

where $\mu^{c\varnothing}$ and $\Sigma^{c\varnothing}$ represent the mean and covariance of the initial spatial estimate of the prior distribution. μ^{MAP} and Σ^{MAP} correspond to the mean and covariance of the posterior spatial estimate given by the least squares solver. Taking logs of (6.31) gives

$$\log(q(c_K, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) = \log\left(\mathcal{N}\left(\mathbf{x}_0, \vec{v}(\mathcal{M}); \mu^{c\varnothing}, \Sigma^{c\varnothing}\right)\right) + \sum_{k=1}^{K} l\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right)$$
$$\sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} l\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) - \log\left(\mathcal{N}\left(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}); \mu^{MAP}, \Sigma^{MAP}\right)\right) \ . \tag{6.32}$$

Writing out the Normal distributions we get

$$\log(q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) = -\frac{1}{2}\Bigg[|\chi_0| \log(2\pi) + (\chi_0 - \mu^{c\varnothing})^\top (\Sigma^{c\varnothing})^{-1} (\chi_0 - \mu^{c\varnothing}) + \log(\det(\Sigma^{c\varnothing}))$$
$$+ \sum_{k=1}^{K} |\mathbf{x}_k| \log(2\pi) + (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k))^\top (\Sigma^{odo})^{-1} (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k)) + \log(\det(\Sigma^{odo}))$$
$$+ \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k))^\top (\Sigma^z)^{-1} (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k)) + \log(\det(\Sigma^z))$$
$$- \left(|\chi_K| \log(2\pi) + \log(\det(\Sigma_{MAP})) + (\chi_K - \mu^{MAP})^\top \Sigma_{MAP}^{-1} (\chi_K - \mu^{MAP})\right)\Bigg] \ . \tag{6.33}$$

By noting that $|\chi_K| = |\chi_0| + K|\mathbf{x}_k|$ and substituting this into (6.33) various terms cancel as follows

$$\log(q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) = -\frac{1}{2}\Bigg[\cancel{|\chi_0| \log(2\pi)} + (\chi_0 - \mu^{c\varnothing})^\top (\Sigma^{c\varnothing})^{-1} (\chi_0 - \mu^{c\varnothing}) + \log(\det(\Sigma^{c\varnothing}))$$
$$+ \sum_{k=1}^{K} \cancel{|\mathbf{x}_k| \log(2\pi)} + (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k))^\top (\Sigma^{odo})^{-1} (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k)) + \log(\det(\Sigma^{odo}))$$
$$+ \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k))^\top (\Sigma^z)^{-1} (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k)) + \log(\det(\Sigma^z))$$
$$- \left(\cancel{|\chi_K| \log(2\pi)} + \log(\det(\Sigma_{MAP})) + (\chi_K - \mu^{MAP})^\top \Sigma_{MAP}^{-1} (\chi_K - \mu^{MAP})\right)\Bigg] \ . \tag{6.34}$$

We can also use the approximation that near the SLAM solution, the non-linear error functions can be substituted with a truncated Taylor series to second order, as follows

$$\begin{aligned}
&(\chi_0 - \mu^{c\varnothing})^\top (\Sigma^{c\varnothing})^{-1}(\chi_0 - \mu^{c\varnothing}) + \sum_{k=1}^{K} (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k))^\top (\Sigma^{odo})^{-1} (\mathbf{x}_k - f(\mathbf{x}_{k-1}, \boldsymbol{u}_k)) \\
&+ \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k))^\top (\Sigma^z)^{-1} (\mathbf{z}_k^i - h(\mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k)) \qquad (6.35) \\
&\approx \sum_r e_r^\top \Omega_r e_r + \Delta\chi_K{}^\intercal J_r^\intercal \Omega_r J_r \Delta\chi_K \ ,
\end{aligned}$$

with $e_r^\top \Omega_r e_r$ being the error of each constraint evaluated at the spatial estimate μ^{MAP}. Then the weight can be calculated as

$$\begin{aligned}
&\log(q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) \approx -\frac{1}{2}\Bigg[\sum_{k=1}^{K} \log(\det(\Sigma^{odo})) + \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + \log(\det(\Sigma^z)) \\
&+ \sum_r e_r^\top \Omega_r e_r + \Delta\chi_K{}^\intercal J_r^\intercal \Omega_r J_r \Delta\chi_K - \Big(\log(\det(\Sigma_{MAP})) + (\chi_K - \mu^{MAP})^\top \Sigma_{MAP}^{-1} (\chi_K - \mu^{MAP}) \Big) \Bigg] \\
&= -\frac{1}{2}\Bigg[\sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + \sum_r e_r^\top \Omega_r e_r + \Delta\chi_K{}^\intercal J_r^\intercal \Omega_r J_r \Delta\chi_K - \log(\det(\Omega_r)) \\
&- \Big(\log(\det(\Sigma_{MAP})) + (\chi_K - \mu^{MAP})^\top \Sigma_{MAP}^{-1} (\chi_K - \mu^{MAP}) \Big) \Bigg] \ . \qquad (6.36)
\end{aligned}$$

It should also be noted that $\Sigma_{MAP}^{-1} = \sum_r J_r^\intercal \Omega_r J_r$ which cancels out the term $\Delta\chi_K{}^\intercal J_r^\intercal \Omega_r J_r \Delta\chi_K$ as follows

$$\begin{aligned}
&\log(q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) \approx -\frac{1}{2}\Bigg[\sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + \sum_r e_r^\top \Omega_r e_r + \cancel{\Delta\chi_K{}^\intercal J_r^\intercal \Omega_r J_r \Delta\chi_K} \\
&- \log(\det(\Omega_r)) - \Big(\log(\det(\Sigma_{MAP})) + \cancel{(\chi_K - \mu^{MAP})^\top \Sigma_{MAP}^{-1} (\chi_K - \mu^{MAP})} \Big) \Bigg] \\
&\log(q(c_k, \mathcal{Z}_{0:K}, \boldsymbol{u}_{1:K})) \approx -\frac{1}{2}\Bigg[\sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi) + \sum_r e_r^\top \Omega_r e_r - \log(\det(\Omega_r)) \\
&- \Big(\log(\det(\Sigma_{MAP})) \Big) \Bigg] \qquad (6.37)
\end{aligned}$$

giving

$$\log(q(c_K, \mathcal{Z}_{1:K}, \boldsymbol{u}_{0:K})) \approx -\frac{1}{2}\Bigg[\sum_r e_r^\top \Omega_r e_r + \log(\det(\sum_r J_r^\intercal \Omega_r J_r)) - \sum_r \log(\det(\Omega_r)) \\ + \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi)\Bigg] . \tag{6.38}$$

This has linear complexity in the number of landmarks and measurements, except for the $\log(\det(\sum_r J_r^\top \Omega_r J_r))$ term. However, some optimization algorithms (among which are iSam and the g2o solver) conveniently calculate the Cholesky decomposition of matrix $\sum_r J_r^\top \Omega_r J_r$. Then we can calculate its determinant simply by multiplying all the diagonal elements in the Cholesky lower triangular matrix L, keeping the complexity linear in the number of measurements. i.e. if $\sum_r \Omega_r = LL^*$, and L is a lower triangular matrix, L^* denoting the conjugate transpose of L, then

$$\log(\det(\sum_r J_r^\top \Omega_r J_r)) = 2\sum_i \log(L_{i,i}) . \tag{6.39}$$

Taking logs of both sides of (6.25c) and substituting (6.39) and (6.38), gives

$$\log(w^{c_K}) = \log(w^{c_\varnothing}) + \sum_{k=0}^{K} \log(p_\kappa\left(\mathcal{Z}_k^{\bar{A}_{\theta_k}}\right)) + \sum_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_i}}} \log\left(1 - P_\mathrm{D}^{(j,i)}\right) + \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} \log(P_\mathrm{D}^{(a_{\theta_k}(i),k)}) \\ - \frac{1}{2}\Bigg[\sum_r e_r^\top \Omega_r e_r + 2\sum_i \log(L_{i,i}) - \sum_r \log(\det(\Omega_r)) + \sum_{i=0}^{k} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi)\Bigg] , \tag{6.40}$$

which can be used in the implementation of this SLAM method. Finally, all that remains is to include a reasonable prior:

$$\{w^{c_\varnothing}, \pi^{c_\varnothing}(\mathbf{x}_0, \vec{v}(\mathcal{M}))\}_{c_\varnothing \in \mathcal{C}_\varnothing} . \tag{6.41}$$

We will model the prior as a known initial position $\mathbf{x}_0^*$, for which a Dirac delta is used, i.e. $p(\mathbf{x}_0) = \delta(\mathbf{x}_0 - \mathbf{x}_0^*)$, and a Multi-Bernoulli distribution can be used to model the map $\mathcal{M}$, with a predefined set of possible landmarks $\mathcal{M}_0$.

Then we have a distribution

$$\pi^{c_\varnothing}(\mathbf{x}_0, \mathcal{M}) = \delta(\mathbf{x}_0 - \mathbf{x}_0^*)\mathcal{N}\left(\vec{v}(\mathcal{M}); \mu^{c_\varnothing}, \Sigma^{c_\varnothing}\right) \tag{6.42}$$

$$w^{c_\varnothing} = \prod_{\mathbf{m}^j \in \mathcal{M}} \frac{P_E(j)}{1 - P_E(j)} \prod_{\mathbf{m}^j \in \mathcal{M}_0} 1 - P_E(j) \tag{6.43}$$

$$\mu^{c_\varnothing} = \begin{bmatrix} \mu_0 \\ \cdots \\ \mu_j \\ \cdots \\ \mu_{|\mathcal{M}|} \end{bmatrix}_{\mathbf{m}^j \in \mathcal{M}} \tag{6.44}$$

$$\Omega^{c_\varnothing} = \begin{bmatrix} \Omega_0^{c_\varnothing} & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & \Omega_j^{c_\varnothing} & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \Omega_{|\mathcal{M}|}^{c_\varnothing} \end{bmatrix}_{\mathbf{m}^j \in \mathcal{M}} , \tag{6.45}$$

where each element j of $\mathcal{M}_0$ can exist with probability $P_E(j)$ (set to a constant value according to the expected density of landmarks) and if it exists, it has a Normal spatial distribution with mean μ_j and information matrix $\Omega_j^{c_\varnothing}$. Applying this as a prior, the posterior becomes.

$$\mathcal{C}_K = \mathcal{C}_\varnothing \times \Theta_0 \times \Theta_1 \times \cdots \times \Theta_K \ , \tag{6.46a}$$

$$c_K = (c_\varnothing, \theta_0, \cdots, \theta_K) \ , \tag{6.46b}$$

$$\begin{aligned}\log(w^{c_K}) = & \sum_{\mathbf{m}^j \in \mathcal{M}} \log(P_E(j)) - \log(1 - P_E(j)) + \sum_{\mathbf{m}^j \in \mathcal{M}_0} \log(1 - P_E(j)) \\ & + \sum_{k=0}^{K} \log(p_\kappa\left(\mathcal{Z}_k^{\bar{A}_{\theta_k}}\right)) + \sum_{\mathbf{m}^j \in \mathcal{M}^{\bar{A}_{\theta_k}}} \log(\left(1 - P_{\mathrm{D}}^{(j,k)}\right)) + \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} \log(P_{\mathrm{D}}^{(a_{\theta_k}(i),k)}) \\ & - \frac{1}{2}\left[\sum_r e_r^\top \Omega_r e_r + 2\sum_i \log(L_{i,i}) - \sum_r \log(\det(\Omega_r)) + \sum_{k=0}^{K} \sum_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} |\mathbf{z}_k^i| \log(2\pi)\right] ,\end{aligned} \tag{6.46c}$$

and

$$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \sim \delta(\mathbf{x}_0 - \mathbf{x}_0^*)\mathcal{N}\left(\vec{v}(\mathcal{M}); \mu^{c\varnothing}, \Sigma^{c\varnothing}\right) \prod_{k=1}^{K} p\left(\mathbf{x}_k \middle| \mathbf{x}_{k-1}, \boldsymbol{u}_k\right)$$

$$\times \prod_{k=0}^{K} \prod_{\mathbf{z}_k^i \in \mathcal{Z}_k^{A_{\theta_k}}} p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) \ , \tag{6.46d}$$

with

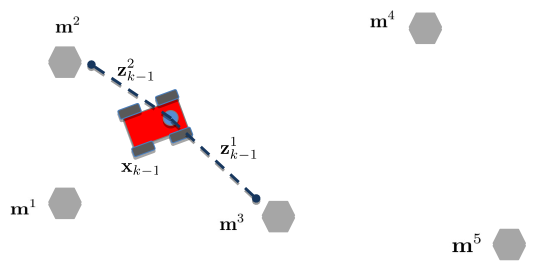
$$\mu^{c\varnothing} = \begin{bmatrix} \mu_0 \\ \ldots \\ \mu_j \\ \ldots \\ \mu_{|\mathcal{M}|} \end{bmatrix}_{\mathbf{m}^j \in \mathcal{M}} \tag{6.46e}$$

$$\Omega^{c\varnothing} = \begin{bmatrix} \Omega_0^{c\varnothing} & 0 & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & \Omega_j^{c\varnothing} & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \Omega_{|\mathcal{M}|}^{c\varnothing} \end{bmatrix}_{\mathbf{m}^j \in \mathcal{M}} \ . \tag{6.46f}$$

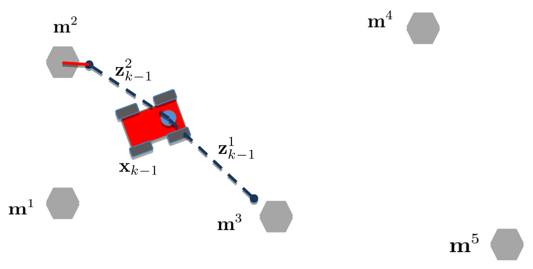
6.5 A visual demonstration of V-GLMB-SLAM

In this section, a graphical representation of an example of V-GLMB-SLAM is explained. In figure 6.1 several data association hypotheses at time $k-1$ are shown. These, along with their component weights and the spatial location of the trajectory and landmark positions, given by solving the graph optimization problem (using g2o, iSAM or some other vector-based SLAM method), constitute the SLAM posterior distribution. In this figure, the hypothesized landmarks, from the prior V-GLMB distribution, are shown as grey hexagons. Landmark measurements are shown as blue dashed lines. Solid red lines show associations. As an example, in the first component of the V-GLMB solution, shown in figure 6.1a, no measurement is associated. Therefore in this component, no landmarks are hypothesized to exist. Given the absence of previous measurements to indicate that the landmarks should exist this would be the most likely possibility and would have the highest component weight, given typical parameters[4]. Figure 6.1b shows a single association hypothesis between measurement $\mathbf{z}_{k-1}^2$ and landmark $\mathbf{m}^2$.

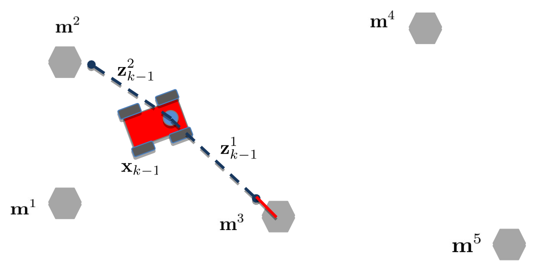
[4]One could have a prior in which initial landmarks have higher probability of existence, e.g. when SLAM is being initialized with a map estimated by some other method. However, typically SLAM is initialized with an unknown map so each initial landmark's probability of existence should be low.



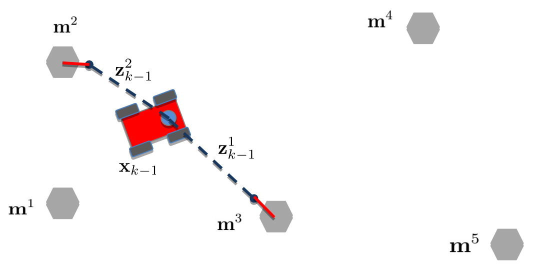
(a) No measurements are associated and therefore no landmarks exist.



(b) Measurement $\mathbf{z}^2_{k-1}$ is associated with landmark $\mathbf{m}^2$.



(c) Measurement $\mathbf{z}^1_{k-1}$ is associated with landmark $\mathbf{m}^1$.

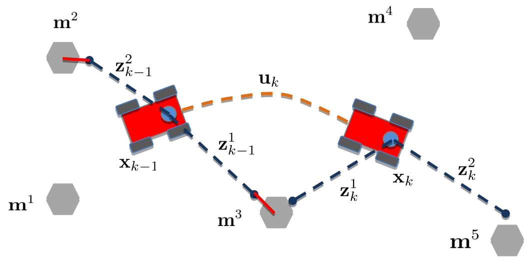


(d) Both measurements are associated.

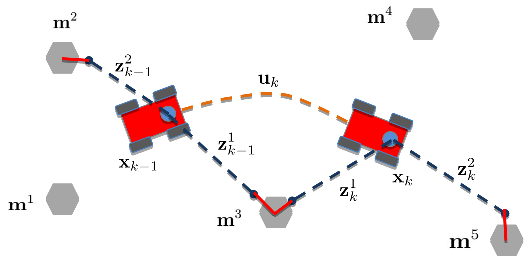
Figure 6.1: A visual representation of an V-GLMB distribution at time $k-1$, composed of 4 components. Note that not all possible data associations are included, thus the number of components in this distribution has been truncated.

Therefore only $\mathbf{m}^2$ is deemed to exist. Similarly figures 6.1c and 6.1d show another two data association hypotheses. In figure 6.1c, landmark $\mathbf{m}^3$ is hypothesized to be associated with $\mathbf{z}^1_{k-1}$, with $\mathbf{m}^3$ hypothesized to exist. Figure 6.1d shows a both $\mathbf{m}^2$ and $\mathbf{m}^3$ being hypothesized to exist and being associated with $\mathbf{z}^2_{k-1}$ and $\mathbf{z}^3_{k-1}$, respectively. One could expect more components to exist, representing every possible data association. Were the V-GLMB algorithm to be applied without limiting the number of components, all of these hypotheses would be in the estimate. However, due to computational constraints, the number of components has to be limited and in this example, it has been limited to 4.

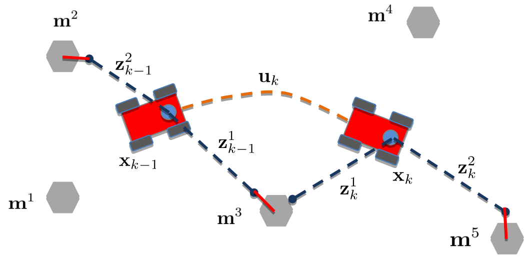
In figure 6.2 the components of the V-GLMB distribution at time k are shown. In this Figure, the orange dashed lines correspond to odometry (dead-reckoning) measurements/inputs. In our proposed algorithm we would start with no measurements associated at time k, as shown in



(a) No measurements at time k are associated.

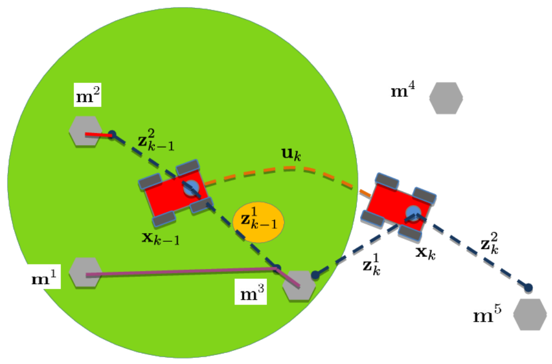


(b) Measurement $\mathbf{z}_k^1$ is associated with landmark $\mathbf{m}^3$, Measurement $\mathbf{z}_k^2$ is associated with landmark $\mathbf{m}^5$.

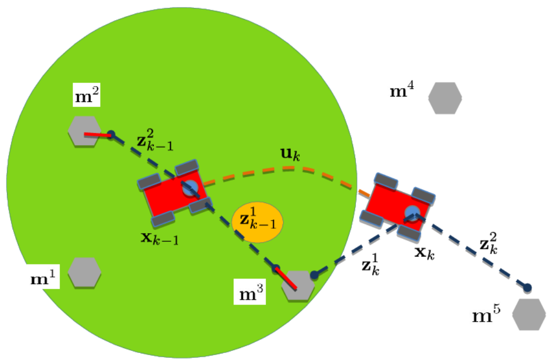


(c) Measurement $\mathbf{z}_k^1$ is associated with landmark $\mathbf{m}^3$.

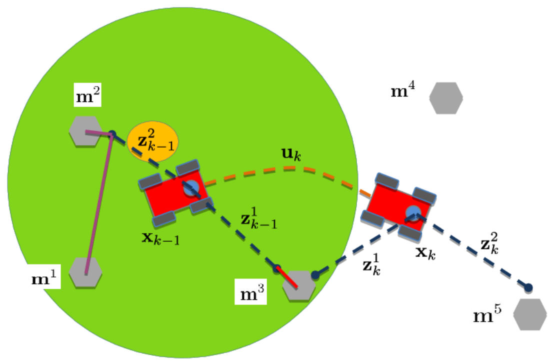
Figure 6.2: A representation of four components of an V-GLMB at time k



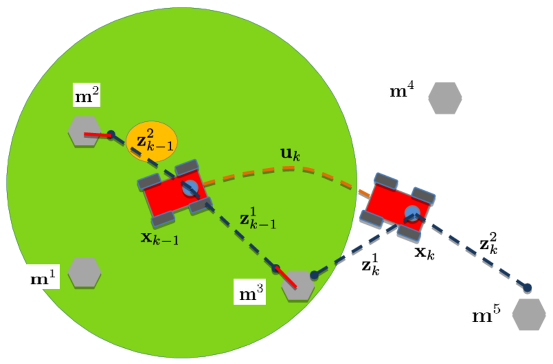
(a) Gibbs sampling step 1: The association hypothesis of $\mathbf{z}^1_{k-1}$ is being sampled, and possible associations are shown in brown.



(b) Gibbs sampling step 1 outcome: The association hypothesis of $\mathbf{z}^1_{k-1}$ was sampled, $\mathbf{m}^3$ was selected.



(c) Gibbs sampling step 2: The association hypothesis of $\mathbf{z}^2_{k-1}$ is being sampled, and possible associations are shown in brown.

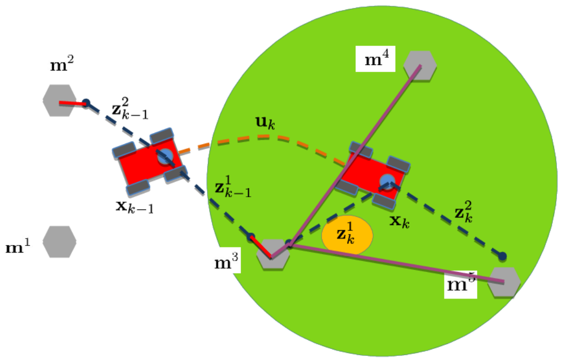


(d) Gibbs sampling step 2 outcome: The association hypothesis of $\mathbf{z}^2_{k-1}$ was sampled, $\mathbf{m}^2$ was selected.

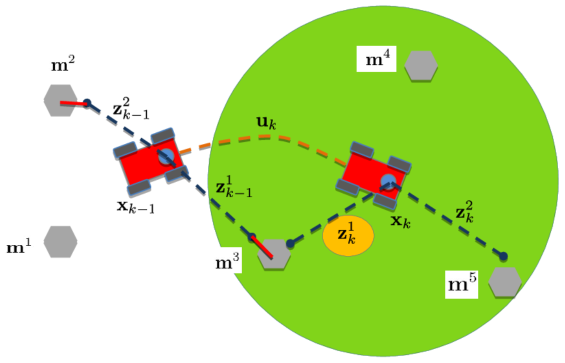
Figure 6.3: A representation of the Gibbs sampling process. Possible associations are shown as brown lines, selected associations hypotheses are shown as red lines, and the field of view of the sensor is shown as a green circle.

figure 6.1a. Then the Gibbs sampler iterates over the other possible data association hypotheses, two of which are shown on Figures 6.2b and 6.2c

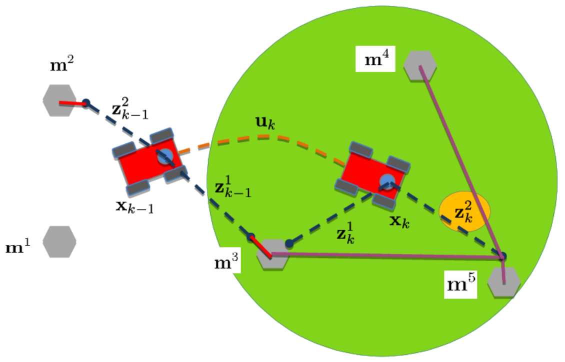
Figures 6.3 and 6.4 show the Gibbs sampler iterating through each data association hypothesis corresponding to the component shown in Figure 6.2a, and the resulting sample, shown in Figure 6.4d, corresponds to the component shown in 6.2c. The field of view of the robot's sensor is shown as a green circle. The Gibbs sampling process iterates through each measurement in ascending order (i.e. $\mathbf{z}^1_{k-1}, \mathbf{z}^2_{k-1}, \mathbf{z}^1_k, \mathbf{z}^2_k,$). Figure 6.3a shows the first step, the initial association corresponds to the one shown in Figure 6.2a. In this step, the association hypothesis of $\mathbf{z}^1_{k-1}$ is being sampled, which is highlighted with a yellow ellipse. Possible association hypotheses are shown as brown lines, all starting from $\mathbf{z}^1_{k-1}$ and going to each possible landmark. As can be



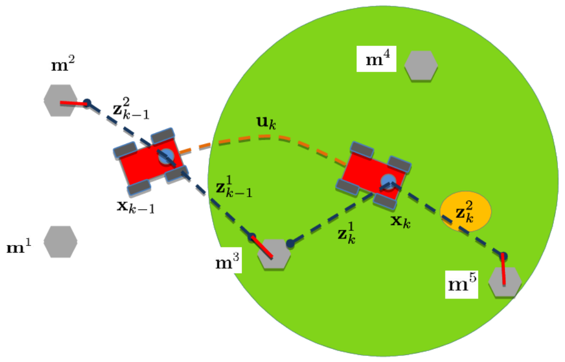
(a) Gibbs sampling step 3: The association of $\mathbf{z}_k^1$ is being sampled, possible associations are shown in brown.



(b) Gibbs sampling step 3 outcome: The association of $\mathbf{z}_k^1$ is was sampled, $\mathbf{z}_k^1$ is hypothesized to be a false alarm.



(c) Gibbs sampling step 4: The association of $\mathbf{z}_k^2$ is being sampled, possible associations are shown in brown.



(d) Gibbs sampling step 4 outcome: The association of $\mathbf{z}_k^2$ is was sampled, $\mathbf{m}^5$ was selected .

Figure 6.4: A visual representation of the Gibbs sampling process

seen, only landmarks inside the field of view and that are not already associated to another measurement at time $k-1$ are possible association hypotheses. Figure 6.3b shows the outcome of the first step. As the most likely candidate, landmark $\mathbf{m}^3$ is selected. Figure 6.3c shows the second step, now the association of measurement $\mathbf{z}_{k-1}^2$ is being sampled. Similarly to before, the most likely outcome is selected, shown in figure 6.3d, with $\mathbf{m}^2$ being selected as the current hypothesis. Figure 6.4a shows the third step corresponding to measurement $\mathbf{z}_k^1$, with $\mathbf{m}^3$, $\mathbf{m}^4$ and $\mathbf{m}^5$ all being possible associations. This time the most likely association is not selected, due to randomness, measurement $\mathbf{z}_k^1$ is hypothesized to be a false alarm, as shown in figure 6.4b. In the last step, shown in figures 6.4c and 6.4d, $\mathbf{z}_k^2$'s association hypothesis is being sampled, this time $\mathbf{m}^5$ is hypothesised to be associated with $\mathbf{z}_k^2$. This results in the sampled data association hypothesis of the new component, as shown in figure 6.2c.

6.6 V-GLMB-SLAM implementation

As expected, the full Bayes posterior (6.46a) to (6.46f) will include all possible measurements to map feature state data associations. However, calculating every one will be computationally infeasible for all but the most basic examples. We, therefore, need to approximate the posterior with a smaller number of possible data association hypotheses. In the filtering approach, the spatial estimates remain the same when selecting different data association hypotheses. On the contrary, in this approach, when selecting a new data association, an optimization has to be carried out to adjust the spatial estimates before calculating the new measurement likelihood. Therefore there is no way to select an association by weight before running the optimization problem defined by the association hypothesis and there is no way to iterate through the associations in decreasing weight order. However, every additional component added to the estimate will decrease the error between the estimate and the true Bayesian posterior (6.46) which contains all possible components at time K in $\mathcal{C}_K$. This was proven for the GLMB distribution using the L1 distance in [4]. Therefore it is not necessary to add the components to the approximated posterior in strictly decreasing order by weight, even if it is desirable to decrease the error faster. Therefore, even a two-component V-GLMB estimate will have lower error when compared with the true posterior V-GLMB distribution than an estimate from a single data association using the data association from either component.

It is in this step that the proposed algorithm allows for external information to explore the data association hypotheses. This can be seen as a problem, given that inefficient exploration of data association hypotheses can make the algorithm have less than adequate performance. On the other hand, being able to include additional possible data association hypotheses for example given by an external "loop closure" algorithm, can be an advantage, because these

```
**Data:** $\mathcal{Z}_{0:K}$ , $\boldsymbol{u}_{0:K-1}$
**Result:** $\{c_K, w^{c_K}\}_{c_K \in \mathcal{C}_K}$, $c_{Kmax}$, $w^{c_{Kmax}}$, $\pi^{c_{Kmax}}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$
$i \leftarrow 0$;
$c_K \leftarrow$ initialAssociationGuess();
$\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \leftarrow$ optimize($c_K, \mathcal{Z}_{0:K}$ , $\boldsymbol{u}_{0:K}$);
$w^{c_K} \leftarrow$ calculateWeight($c_K, \pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$);
$w^{c_{Kmax}} \leftarrow w^{c_K}$;
$c_{Kmax} \leftarrow c_K$;
$\pi^{c_{Kmax}}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \leftarrow \pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$;
**while** $i <$ *max iterations* **do**
    $c_K \leftarrow$ guessAssociation( $\{c_K, w^{c_K}\}_{c_K \in \mathcal{C}_K}$);
    **if** $c_K \in \mathcal{C}_K$ **then**
        go to next iteration;
    **else**
        $\mathcal{C}_K \leftarrow \mathcal{C}_K \cup c_K$;
        $\pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \leftarrow$ optimize($c_K, \mathcal{Z}_{0:K}$ , $\boldsymbol{u}_{0:K}$);
        $w^{c_K} \leftarrow$ calculateWeight($c_K, \pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$);
        **if** $w^{c_K} > w^{c_{Kmax}}$ **then**
            $w^{c_{Kmax}} \leftarrow w^{c_K}$;
            $c_{Kmax} \leftarrow c_K$;
            $\pi^{c_{Kmax}}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \leftarrow \pi^{c_K}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$;
        **end**
    **end**
**end**
```

Algorithm 1: V-GLMB-SLAM: Overview

solutions will be naturally included in the Bayes posterior.

The pseudo-code of the proposed algorithm is shown in Algorithm 1. As can be seen, the output of the algorithm will be the set of examined data association hypotheses and their weights $\{c_K, w^{c_K}\}_{c_K \in \mathcal{C}_K}$, and the trajectory estimate with the highest weight and its associated landmarks estimate $\pi^{c_{Kmax}}(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$. The algorithm proceeds by successively "guessing" the association hypothesis, ideally producing the higher weighted components c_K of the posterior distribution.

The algorithm can then use either g2o, isam2, or a similar graph optimization algorithm to produce spatial estimates, and a principled way of generating new data association possibilities based on the already explored data associations and their weights $\{c_K, w^{c_K}\}_{c_K \in \mathcal{C}_K}$.

We propose to use a Gibbs sampler, which will sample the data association hypotheses based on their conditional likelihood (conditioned on the current state). The data association hypotheses are "multi-scan", in a manner similar to [58], i.e. we iterate over the association hypotheses at every time $k \in \{0 : K\}$ and not just at the latest time K. However, if we are dealing with a static map, the implementation is much simpler than in [58]. i.e. we don't have to deal with target birth, death or movement.

In summary, we need to sample from the highest weight components from the posterior V-GLMB (6.25). However, the weight cannot be calculated before running an expensive optimization. This is because the normal distribution will change it's mean and covariance based on the new data association hypothesis.

To simplify the implementation, the components c_k, whose data association hypothesis has landmarks existing without any detections hypothesized to be associated with them, can be discarded. This is because even though the components c_k in which landmarks that are never detected exist have non-zero weight, they have low weight and are not useful. These landmark hypotheses come from the prior map V-GLMB distribution. This is particularly true since the part of the map which has never been in the field of view of the sensor(s) will have many such possibly existing landmark hypotheses. Discarding the previously defined components, allows us to define, within a single component, that a landmark is hypothesized to exist if and only if it has one or more measurements associated with it. Without this assumption, each component hypothesis would have to store which landmarks exist, alongside the data association hypothesis which would only associate landmarks that exist according to the hypothesis.

A data association hypothesis θ can be represented by extending the already defined association function $a_{\theta_k}(i)$ to return -1 if $\boldsymbol{z}_k^i$ is hypothesized to be a false alarm. Then a data association θ will be completely determined by the data association function $a_{\theta_k}(i)$ with k ranging from 0 to K. Given an initial component weight[5] w^{c_K}, and distribution $\pi^{c_K}(\mathbf{x}_{0:k}, \vec{v}(\mathcal{M}))$, which has a data association hypothesis $a_{\theta_k}(i)$, we will sample a new data association hypothesis $\theta^\star$ with the Gibbs sampler. i.e. for each measurement $\boldsymbol{z}_k^i$ in $\mathcal{Z}_{0:k}$ we would like to sample the association of $\boldsymbol{z}_k^i$ given all the other data associations, proportional to the weights of the components. For this, we need to approximate the weights of the new components that will result when changing $a_{\theta_k}(i)$ to $a_{\theta_k^\star}(i)$. First, the landmarks that are in the field of view at time k are calculated as $\mathcal{M}_k$. Then the probability of associating measurement $\boldsymbol{z}_k^i$ to landmark $\mathbf{m}^j$ in the field of view will be proportional to the component weight $\log(w^{c_K})$ (6.46c). Therefore, the log of the probability of setting $a_{\theta_k^\star}(i) = j$ is

$$\log(p(a_{\theta_k^\star}(i) = j)) = \log(w^{c_K}) + \text{constant} \tag{6.47}$$

$$\begin{aligned}\log(p(a_{\theta_k^\star}(i) = j)) \approx{}& \log(P_{\mathrm{D}}^{(j,k)}) - \log(1 - P_{\mathrm{D}}^{(j,k)}) \\ &- \frac{1}{2}\left[e_r^\top \Omega_r e_r + 2\sum_i \log(L_{i,i}) + |\mathbf{z}_k^i| \log(2\pi) \right],\end{aligned} \tag{6.48}$$

[5]Stored as $\log(w^{c_K})$ for numerical stability

if $\mathbf{m}^j$ exists (i.e. has at least one other measurement associated), and

$$\log(p(a_{\theta^\star_k}(i) = j)) \approx \log(P_{\mathrm{D}}^{(j,k)}) - \log(1 - P_{\mathrm{D}}^{(j,k)}) + \log(P_E(j)) - \log(1 - P_E(j)) + \sum_{k|\mathbf{m}^j \in \mathcal{M}_k} \log(1 - P_{\mathrm{D}}^{(j,k)}) - \frac{1}{2}\left[e_r^\top \Omega_r e_r + 2\sum_i \log(L_{i,i}) + |\mathbf{z}_k^i| \log(2\pi) \right] \tag{6.49}$$

if $\mathbf{m}^j$ does not exist. It should be noted that

$$\log(p(a_{\theta^\star_k}(i) = -1)) \approx \log(p_\kappa\left(\mathbf{z}_k^i\right)) \ , \tag{6.50}$$

where, for the approximation, we have assumed that the spatial estimate (the position of the landmarks and robot) will not change. To approximate the change in the determinant of the Hessian matrix[6], we will calculate the change in the determinant of the sub-matrix related to the current measurement (i.e. the sub-matrix involving landmark j and pose k). i.e., we approximate

$$\frac{\det(\Omega^{c_\star})}{\det(\Omega^c)} = \frac{\det(J_z(\mathbf{m}^j, \mathbf{x}_k)^\top \Omega_z J_z(\mathbf{m}^j, \mathbf{x}_k) + \Omega^c)}{\det(\Omega^c)} \approx \frac{\det(J_z(\mathbf{m}^j, \mathbf{x}_k)^\top \Omega_z J_z(\mathbf{m}^j, \mathbf{x}_k) + \Omega^c_{\mathbf{m}^j,\mathbf{x}_k})}{\det(\Omega^c_{\mathbf{m}^j,\mathbf{x}_k})} \ , \tag{6.51}$$

where Ω^c is the information matrix of the spatial estimate resulting from the existing data association θ, $\Omega^{c_\star}$ is the information matrix of the spatial estimate resulting from the new data association $\theta^\star$, and $\Omega^c_{\mathbf{m}^j,\mathbf{x}_k}$ is the submatrix of Ω^c, corresponding to landmark $\mathbf{m}^j$ and robot pose $\mathbf{x}_k$.

By sampling new data association hypotheses $a_{\theta^\star_k}(i)$ for each measurement i according to these weights, we can define a Gibbs "move" function that can sample from possible data association hypotheses approximately in proportion to the component weight w^{c_K}.

6.6.1 A landmark birth move

When examining the probabilities used by the Gibbs sampler, as well as the preliminary performance of the sampling algorithm, we note that, because the Gibbs sampler changes associations one at a time, moving from a hypothesis of a landmark not existing to it existing, will necessarily mean that the landmark has only a single detection. Therefore the probability of making that change (6.49) will be dominated by the missed detections and the Gibbs sampler will very rarely create new landmarks. We, therefore, add a reversible jump Markov step to create new landmarks.

[6]The Hessian matrix approximates the information matrix of the component

During Gibbs sampling, when a measurement association hypothesis is set to false alarm, the most likely data association (among nonexisting landmarks) for it is found. A counter $n_{max}(\mathbf{m}^j)$ for the most likely landmark is then incremented. This is equivalent to, for each landmark that is hypothesized not to exist, the number of measurements that are currently hypothesized to be false alarms and are closest to that landmark is calculated as $n_{max}(\mathbf{m}^j)$. After sampling using the Gibbs sampler, each landmark hypothesized not to exist is created with a probability equal to

$$p_{birth}(\mathbf{m}^j) = \min\left(\frac{n_{max}(\mathbf{m}^j)}{\sum_{k|\mathbf{m}^j \in \mathcal{M}_k} P_{\mathrm{D}}^{(j,k)}}, 1\right) \ , \tag{6.52}$$

where the denominator corresponds to the expected number of measurements for that specific landmark. This new landmark hypothesis is associated with all the measurements that were used to calculate $n_{max}(\mathbf{m}^j)$.

6.6.2 A landmark death move

Intuitively, once a birth move has been added, the next heuristic to add would be landmark death. Landmarks can be randomly removed with probability proportional to the number of missed detections and inversely proportional to the number of detections.

6.6.3 Sampling the associations

With the Gibbs, birth, and death "moves" algorithm 1 is defined, which determines guessAssociation($\{c_K, w^{c_K}\}_{c_K \in \mathcal{C}_K}$) to be applying either the Gibbs sampler with some probability p_{Gibbs} or birth-death move with probability $1 - p_{\mathrm{Gibbs}}$. Additionally, to speed up the convergence, the estimate is initialized by using the nearest neighbor data association, thus determining initialAssociationGuess().

$$\mathcal{C}_K^0 = \{(c_K^0, \log(w^{c_K^0}))\} \tag{6.53}$$

$$\tag{6.54}$$

In order to save memory, the association and weight are stored in an ordered (by weight) container (e.g. an ordered tree, or some other data structure), and the trajectory and spatial locations of landmarks are discarded. Then the new estimate of the V-GLMB distribution will be.

$$\mathcal{C}_K^{i+1} = \mathcal{C}_K^i \cup \left(c_K^{i+1}, \log(w^{c_K^{i+1}})\right) \ , \tag{6.55}$$

where the new included component $\left(c_K^{i+1}, \log(w^{c_K^{i+1}})\right)$ will be calculated in a manner similar to simulated annealing, by first sampling n components from $\mathcal{C}_K^i$ in proportion to $\exp(\log(w^{c_K})/T(i))$. Where $T(i)$ is a parameter called temperature, that starts high and is lowered according to a cooling schedule at each iteration. Then, each sampled component first uses the birth move to randomly create landmarks from false alarm measurements, and the death move to remove landmarks with few detections and many missed detections. The algorithm then proceeds to sample the data association hypothesis according to the Gibbs sampler n_{Gibbs} times. With the sampled data association hypothesis, each component uses a non-linear least squares optimizer (in our case g2o), to optimize the spatial estimate and produce a Hessian/information matrix, which is used to calculate the new weight, $\log(w^{c_K^{i+1}})$. Finally, the new component is added to the V-GLMB distribution.

6.6.4 A note on computational complexity

The computational complexity of the resulting V-GLMB-SLAM algorithm in $O(\cdot)$ notation would be

$$O(\text{V-GLMB-SLAM}) = O\left(I\left(O(\text{Gibbs sampler}) + O(\text{Optimizer}) + O(\text{CalculateWeight})\right)\right) \tag{6.56}$$

$$O(\text{V-GLMB-SLAM}) = O\left(I\left(O(\text{CalcDAProbs} + N_{\text{Gibbs}}\,|\mathcal{Z}_{0:k}|) + O(\text{Optimizer}) + |\mathcal{Z}_{0:k}| + K\right)\right) \tag{6.57}$$

$$O(\text{V-GLMB-SLAM}) = I\left(|\mathcal{Z}_{0:k}|\,|\mathcal{M}| + N_{\text{Gibbs}}\,|\mathcal{Z}_{0:k}| + O(\text{Optimizer}) + |\mathcal{Z}_{0:k}| + K\right) \tag{6.58}$$

$$O(\text{V-GLMB-SLAM}) = I\left(|\mathcal{Z}_{0:k}|\left(|\mathcal{M}| + N_{\text{Gibbs}}\right) + O(\text{Optimizer}) + |\mathcal{Z}_{0:k}| + K\right) \tag{6.59}$$

where I is the number of V-GLMB iterations, N_{Gibbs} is the number of Gibbs sampler iterations. Importantly the value of I that is required to generate good results, depends on the sampling method that generates candidate association hypotheses. Therefore I could be reduced by improving or replacing the current Gibbs sampler. It can be observed that the batch optimizer must be run I times, resulting in a higher computational complexity than traditional algorithms that just use a single heuristically determined data association. The complexity could be reduced by improving the sampling algorithm, therefore sampling less unlikely associations thus sampling the more likely associations faster.

6.7 Simulated Results

The V-GLMB-SLAM algorithm was applied to 2D simulated data, from the simulator included with the g2o software [21], which simulates a random trajectory and measurements with given data associations. This simulation allows for missed detections with a constant probability of missed detection. False alarms were added to the simulated detections, in order to evaluate the algorithm's tolerance to them. V-GLMB-SLAM is then compared with the solution provided by the g2o solver directly using the correct data association, which will be considered "ground truth". In addition to using the ground truth data association, g2o is also run using the nearest neighbor data association calculated at each time k, using the robot pose determined by solving g2o up to time $k-1$ with the already determined data associations. Maximum likelihood data association is also determined using the Hungarian algorithm based on the likelihoods given by the RFS model of the measurements, as RB-GLMB-SLAM would do for its first component (if using Murty's algorithm and not the Gibbs sampler to select associations).

These three algorithms were tested, and compared with "ground truth" using multiple values for the odometry covariance matrix (Ω_u^{-1}) as well as by varying the average number of added false alarms (λ) and the probability of detection (P_{D}). The odometry information matrix Ω_u is

$$\Omega_u = \begin{bmatrix} \omega_u & 0 & 0 \\ 0 & \omega_u & 0 \\ 0 & 0 & 10\omega_u \end{bmatrix}, \tag{6.60}$$

where the information for the orientation part of the pose is higher. All three algorithms were tested until the settings generated unsatisfactory trajectory estimates. At each pair $((\omega_u, \lambda))$ of odometry noise covariance and false alarm rate, the simulated measurements (both landmark and odometry) are generated 20 times with the same ground truth trajectory and map. All three algorithms are executed on the 20 simulated datasets. These Monte Carlo simulations all have the same trajectory but with different instantiations (seeds) for the noise added to odometry and landmark measurements.

The Relative Pose Error metric is calculated using the python package [59] for each estimated trajectory and the value is averaged over the 20 Monte Carlo simulations at each pair of values (ω_u, λ). Figures 6.5 and 6.6 show the relative pose error and absolute pose error calculated for $0, 0.01, 0.03, 0.1, 0.5, 1$ added false alarms per scan, respectively. As can be seen in figure 6.5, when no false alarms are added the relative pose error of the V-GLMB solution is approximately equal to the maximum likelihood and nearest neighbor solution for low odometry noise covariance ($\omega_u >= 300$) and is better than both these algorithms for more challenging odometry covariance values ($70 < \omega_u \leq 100$) and all algorithms fail when the odometry is so

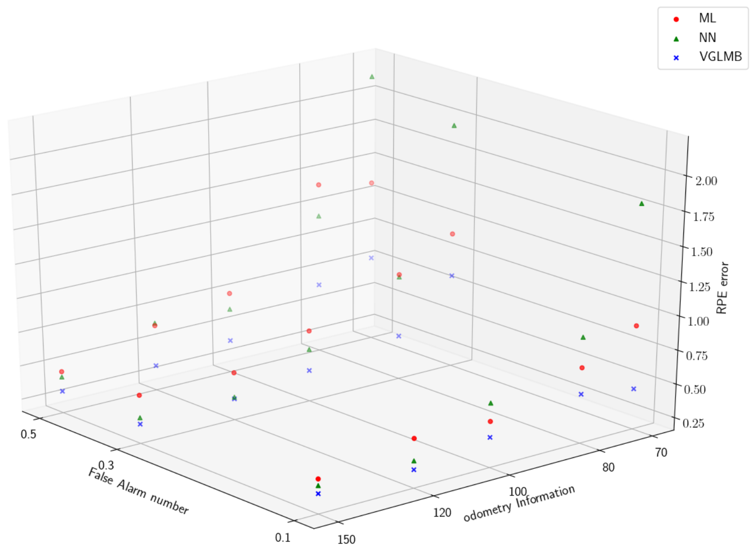


Figure 6.5: Median Relative Position Error, varying odometry noise and number of false alarms.

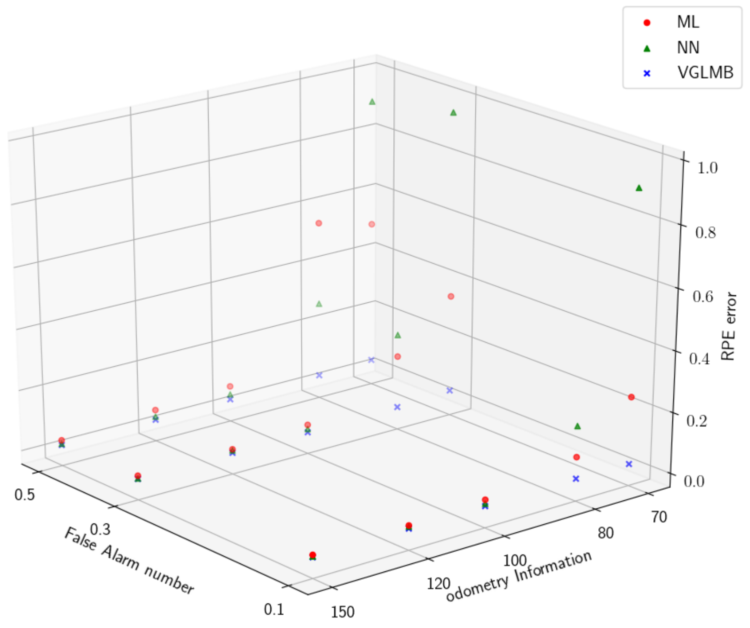


Figure 6.6: Median Absolute Position Error, varying odometry noise and number of false alarms.

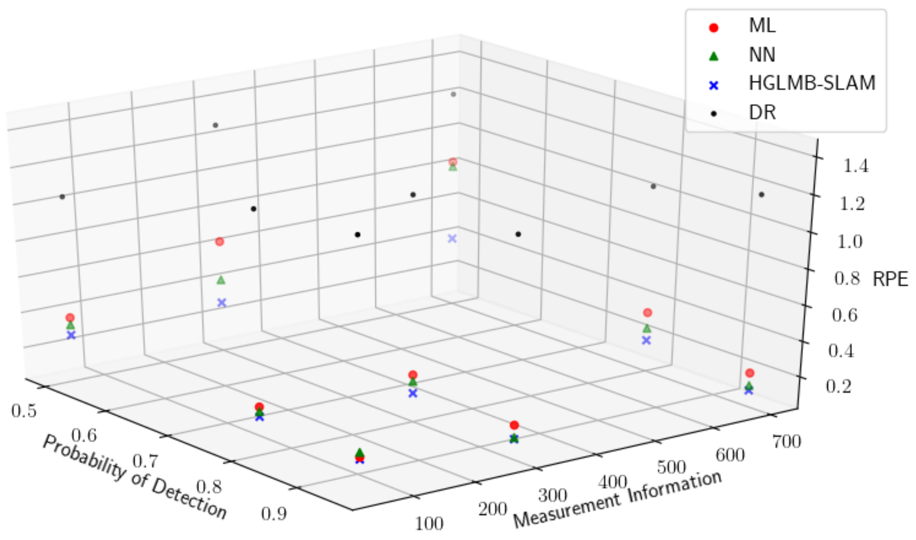


Figure 6.7: Median Relative Position Error, varying P_{D} and the measurement noise.

bad that associating the landmarks is too difficult ($\omega_u = 70$). Additionally, the measurement noise (Ω_z), where

$$\Omega_z = \begin{bmatrix} \omega_z & 0 \\ 0 & \omega_z \end{bmatrix} , \tag{6.61}$$

and the probability of detection P_{D} were also varied. This was carried out by fixing the odometry information to $\omega_u = 100$ and varying P_{D} to be 0.5, 0.8, or 0.95 and varying ω_z to be 50, 300, or 700.

Figures 6.7 and 6.8 show the median value for the relative position error and absolute position error,avergaed over 20 Monte Carlo runs, respectively. As can be seen in both figures V-GLMB-SLAM has the best performance for all the values of $(\omega_z, P_{\mathrm{D}})$ tested. Note that as P_{D} is lowered, the difference between V-GLMB-SLAM and maximum likelihood or nearest neighbor data association increases, which is to be expected. Interestingly the difference also increases when the measurement information increases, which is unexpected since less noisy measurments should convey more information. However, if the motion noise is much higher than the measurement noise, it is possible that some measurements are erroneously determined to be separate landmarks, while V-GLMB-SLAM is able to find the correct association.

Figures 6.9, 6.10 and 6.11 show the solutions obtained by using the g2o optimization library with nearest neighbor data association, maximum likelihood data association, and our proposed algorithm, respectively. The g2o solution using the ground truth data association is shown as

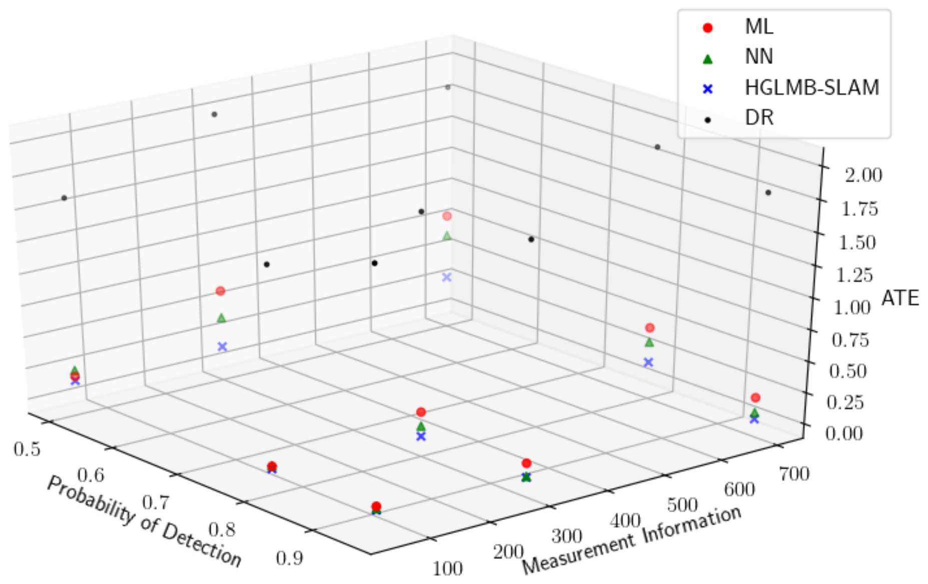


Figure 6.8: Median Absoute Position Error, varying P_{D} and the measurement noise.

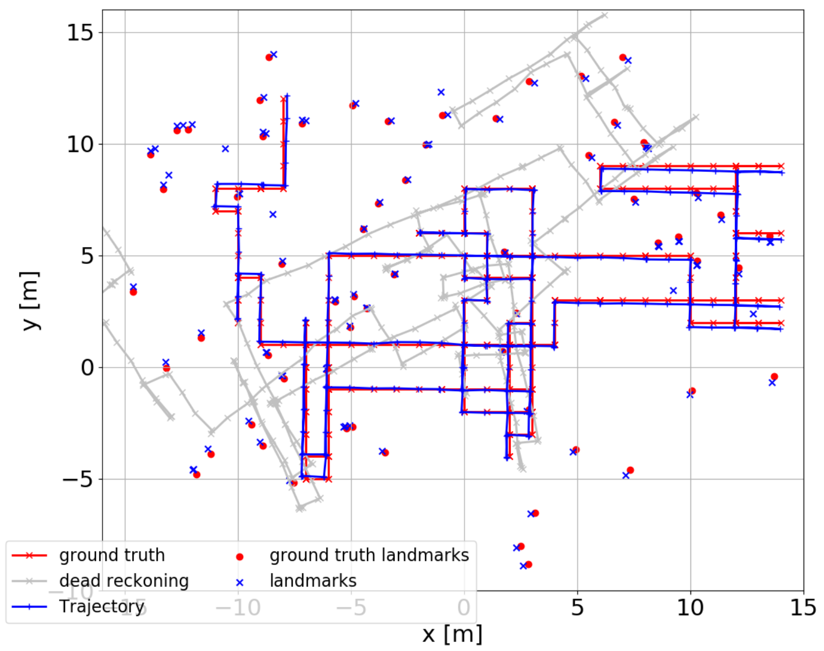


Figure 6.9: Nearest neighbor data association. Low odometry noise.

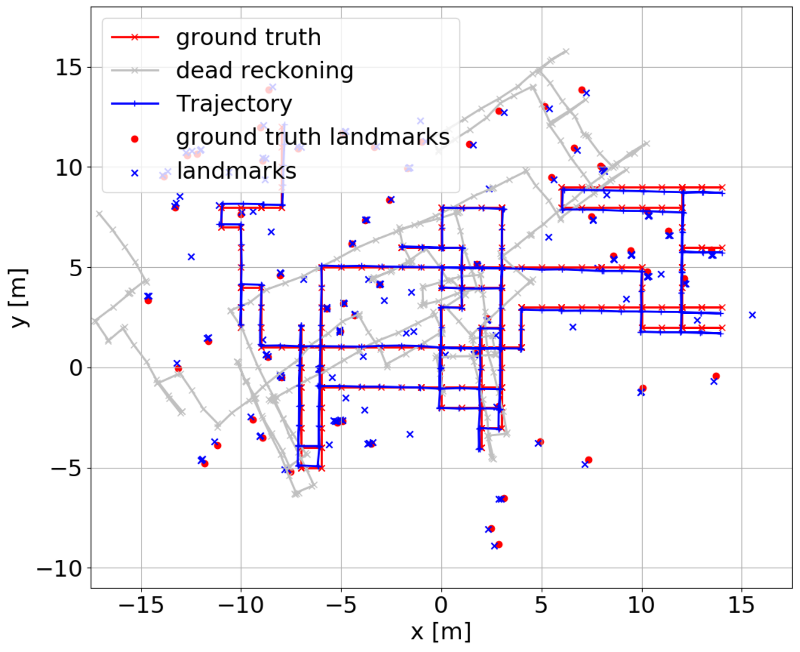


Figure 6.10: Maximum likelihood data association. Low odometry noise.

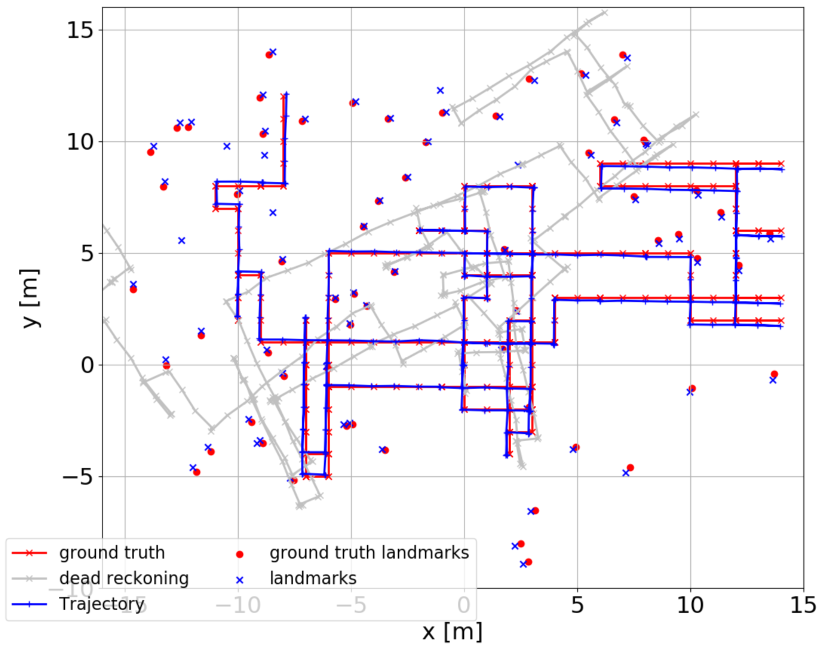


Figure 6.11: V-GLMB-SLAM solution. Low odometry noise.

ground truth in red in all three figures. In this case, the information matrix corresponding to the landmark measurements is

$$\Omega_z = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} , \tag{6.62}$$

where the matrix elements correspond to a standard deviation in the x and y directions of $0.03[\text{m}] \approx \sqrt{\frac{1}{1000}}$. The odometry information matrix is

$$\Omega_u = \begin{bmatrix} 150 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 1500 \end{bmatrix} , \tag{6.63}$$

where the matrix elements correspond to a standard deviation of $0.08[\text{m}] \approx \sqrt{\frac{1}{150}}$ in the x and y directions and $0.026[\text{rad}] \approx \sqrt{\frac{1}{3000}}[\text{rad}]$ in the angular direction. The probability of detection was $P_D = 0.95$ in a circular FoV with radius $r = 5[\text{m}]$ and zero outside. With these settings, it can be seen that all three algorithms perform adequately, with the estimated trajectories being much closer to the ground truth trajectory than the initial “dead-reckoning” trajectory. With the odometry noise being relatively low, the association between measurements and landmarks is adequately solved by all three algorithms. In the case of V-GLMB the correct association is not the only one in the posterior, however, it is the most likely one.

Figures 6.12, 6.13 and 6.14 show the solutions obtained by using the g2o optimization library with nearest neighbor data association, maximum likelihood data association, and our proposed algorithm, respectively. The settings are the same as the previous example except the odometry information was reduced to

$$\Omega_u = \begin{bmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 800 \end{bmatrix} , \tag{6.64}$$

where the matrix elements correspond to a standard deviation of $0.11[\text{m}] \approx \sqrt{\frac{1}{80}}$ in the x and y directions and $0.035[\text{rad}] \approx \sqrt{\frac{1}{800}}$ in the angular direction. With this decreased odometry information (increased odometry noise) the maximum likelihood and nearest neighbor data association heuristics are unable to find the correct association and the iterative method proposed here finds a data association that is closer to the ground truth and provides a superior estimated trajectory.

Figures 6.15, 6.16 and 6.17 show the solutions obtained by using the g2o optimization library with nearest neighbor data association, maximum likelihood data association, and our proposed

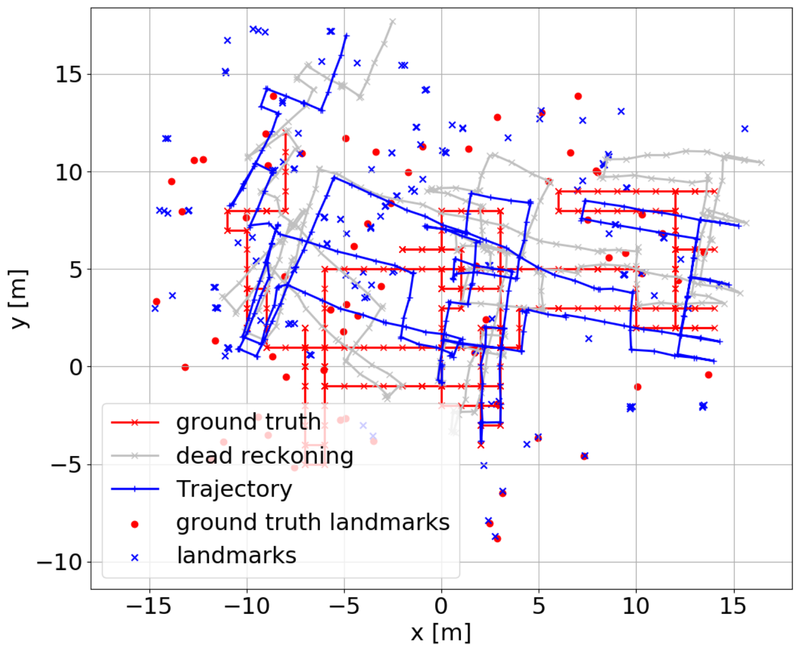


Figure 6.12: Nearest neighbor data association. Medium odometry noise.

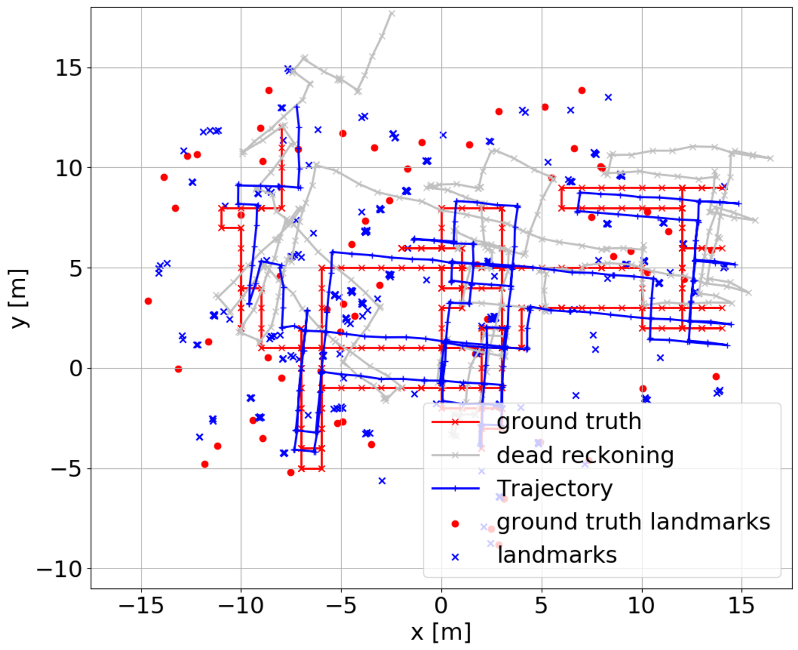


Figure 6.13: Maximum likelihood data association. Medium odometry noise.

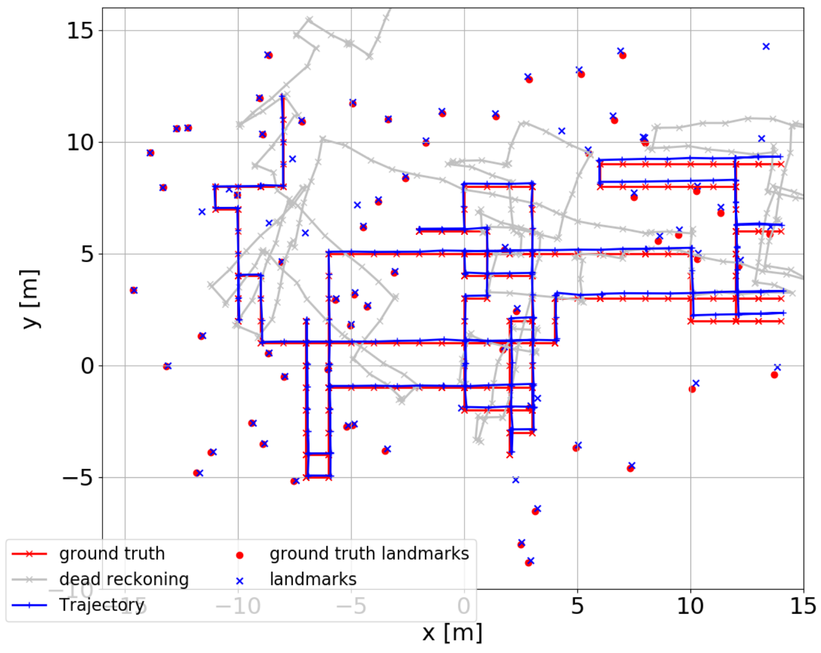


Figure 6.14: V-GLMB-SLAM solution. Medium odometry noise.

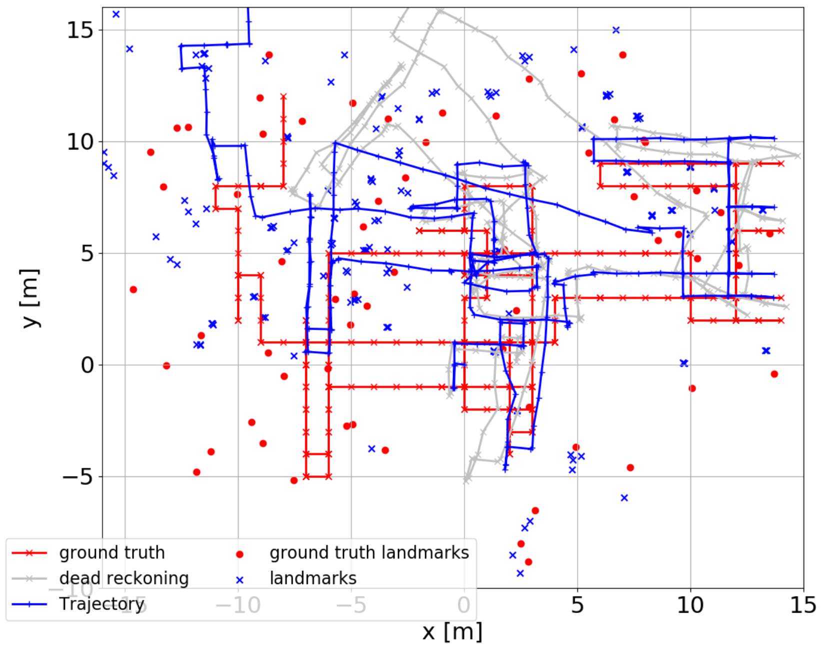


Figure 6.15: Nearest neighbor data association. High odometry noise.

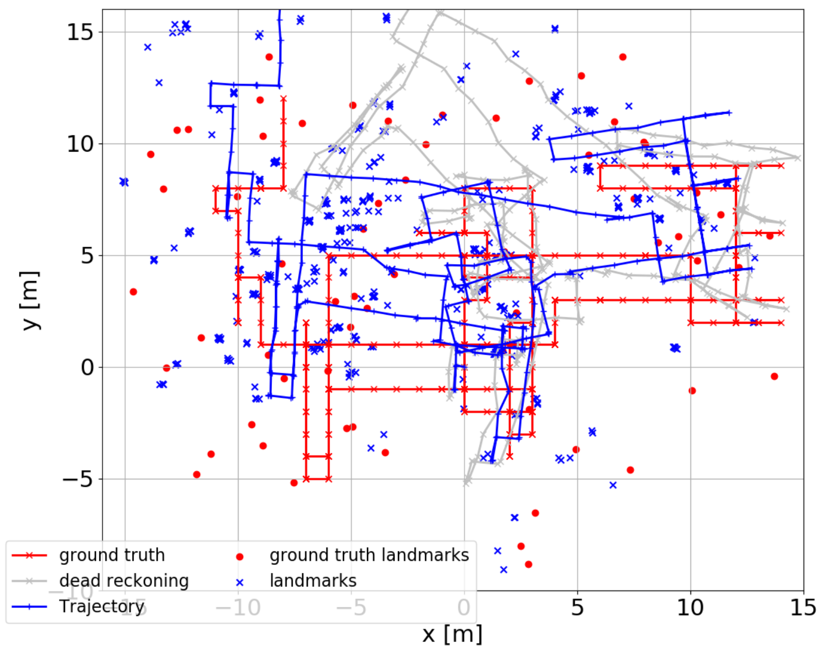


Figure 6.16: Maximum likelihood data association. High odometry noise.

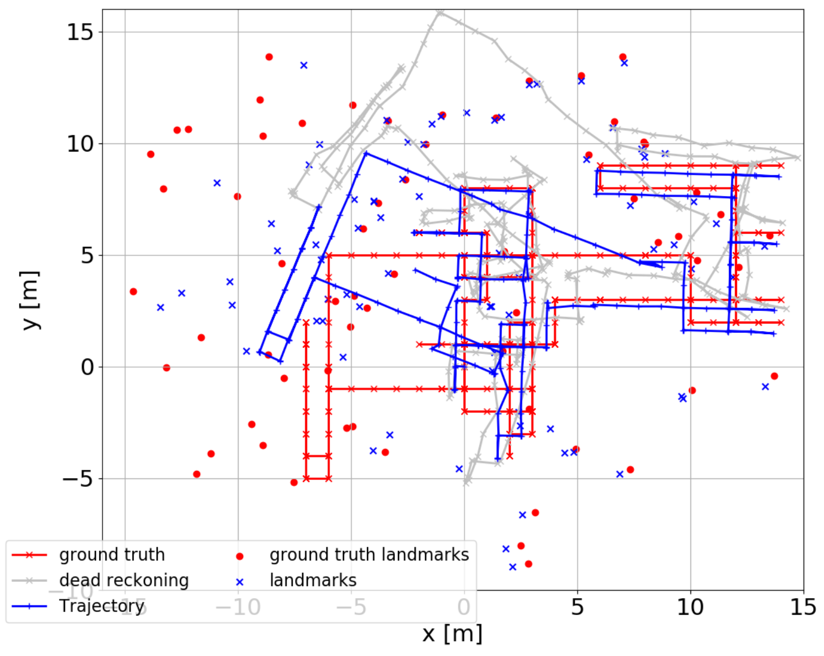


Figure 6.17: V-GLMB-SLAM solution. High odometry noise.

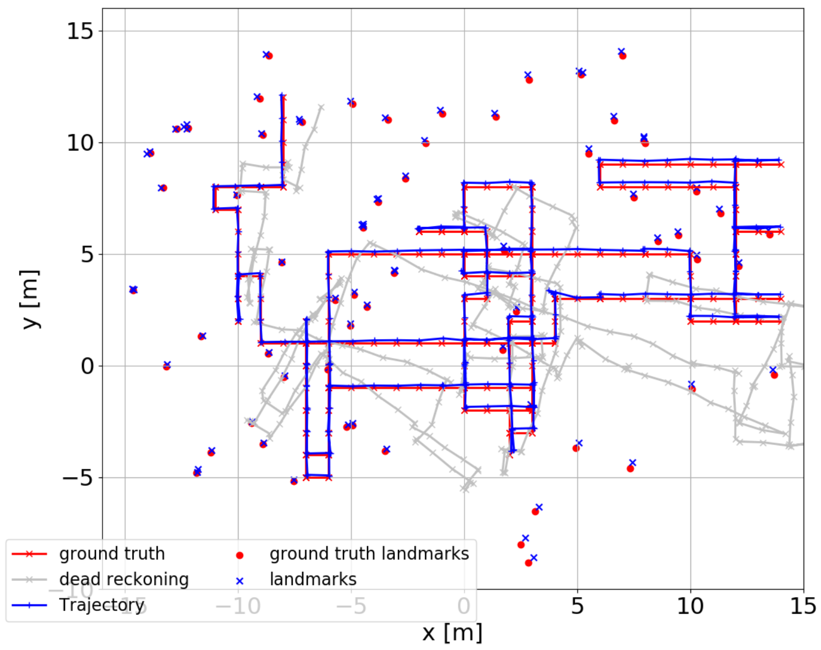


Figure 6.18: Nearest neighbor data association. High P_D.

algorithm, respectively. The odometry information has been further reduced to

$$\Omega_u = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 500 \end{bmatrix}, \tag{6.65}$$

where the matrix elements correspond to a standard deviation of 0.14[m] $\approx \sqrt{\frac{1}{50}}$ in the x and y directions and 0.044[rad] $\approx \sqrt{\frac{1}{500}}$ in the angular direction. Under these more difficult conditions no algorithm is able to correctly estimate the trajectory. As can be seen in figure 6.17, in this case the V-GLMB-SLAM algorithm, after creating a “bad” estimate for a while, it stops producing new data associations. Therefore the trajectory estimate is incomplete, this can be easily detected and reported, allowing an autonomous system to fail gracefully instead of proceeding with a bad trajectory estimate.

Figures 6.18, 6.19 and 6.20 show the solutions obtained by using the g2o optimization library with nearest neighbor data association, maximum likelihood data association, and our proposed

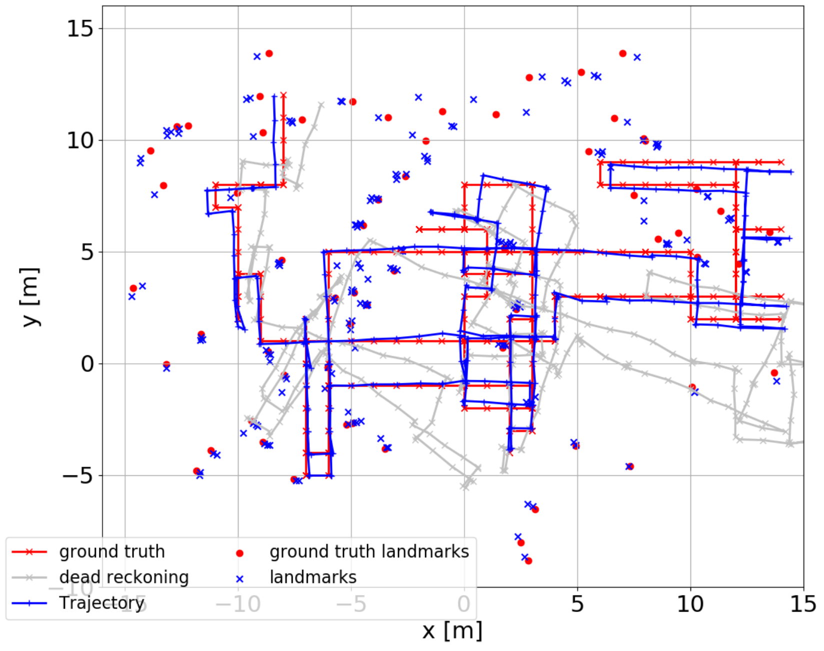


Figure 6.19: Maximum likelihood data association. High P_{D}.

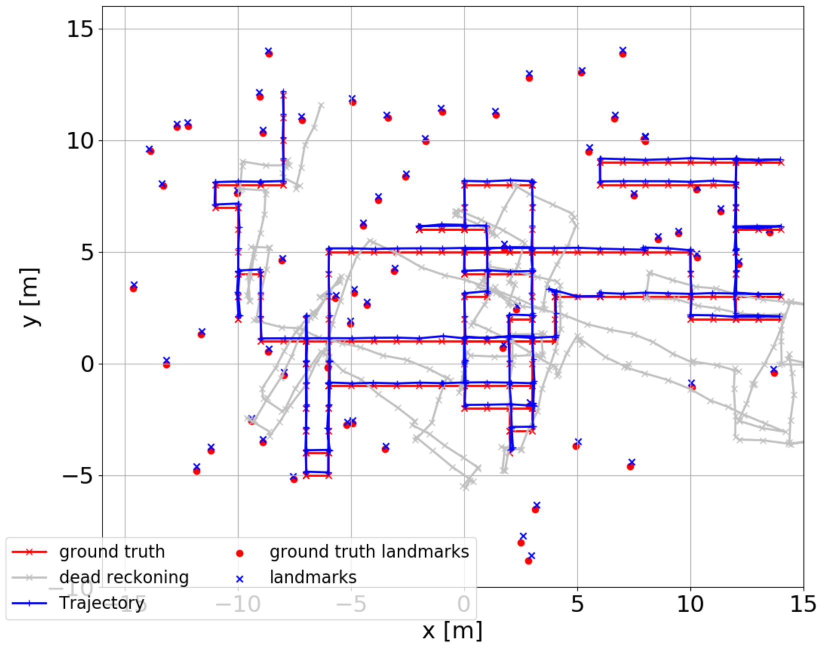


Figure 6.20: V-GLMB-SLAM solution. High P_{D}.

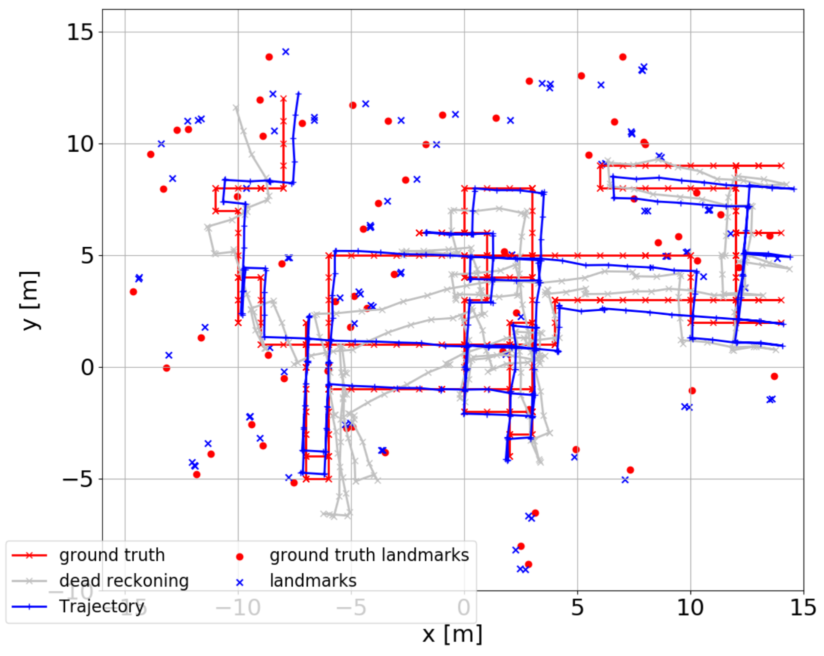


Figure 6.21: Nearest neighbor data association. Medium P_{D}.

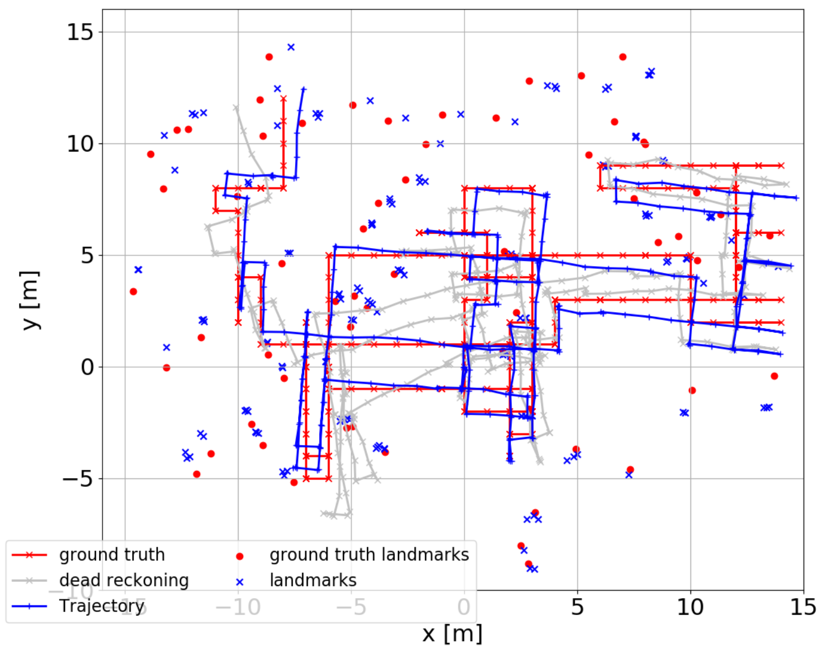


Figure 6.22: Maximum likelihood data association. Medium P_{D}.

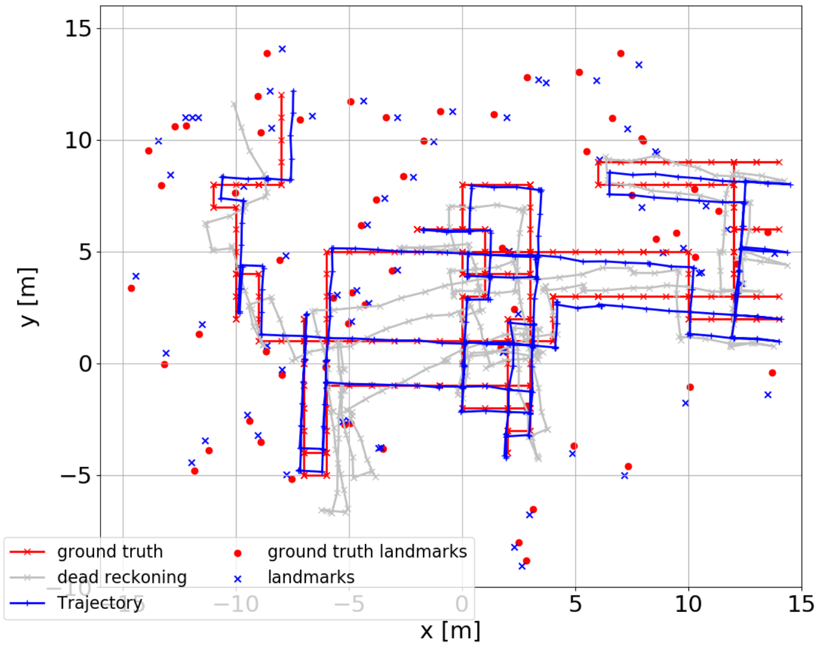


Figure 6.23: V-GLMB-SLAM solution. Medium P_{D}.

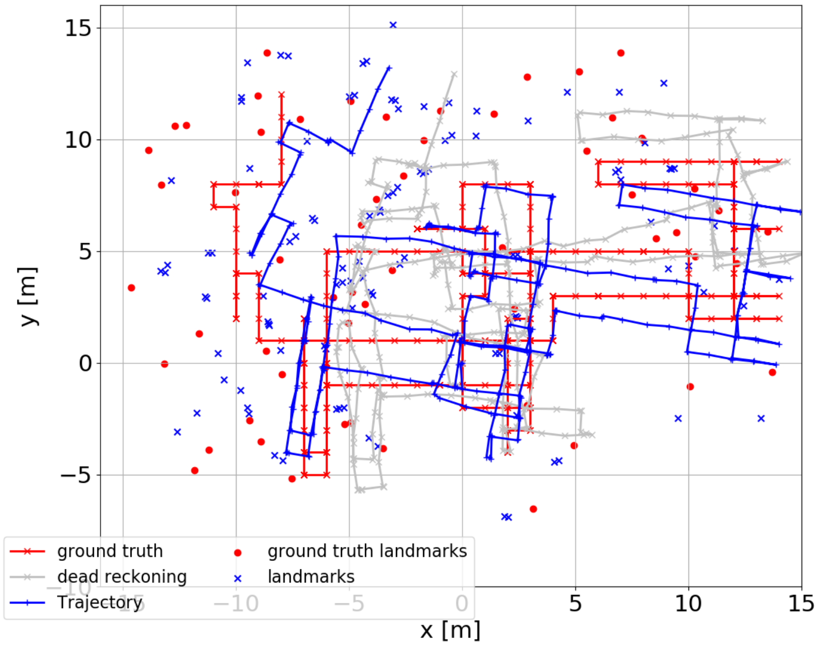


Figure 6.24: Nearest neighbor data association. Low P_{D}.

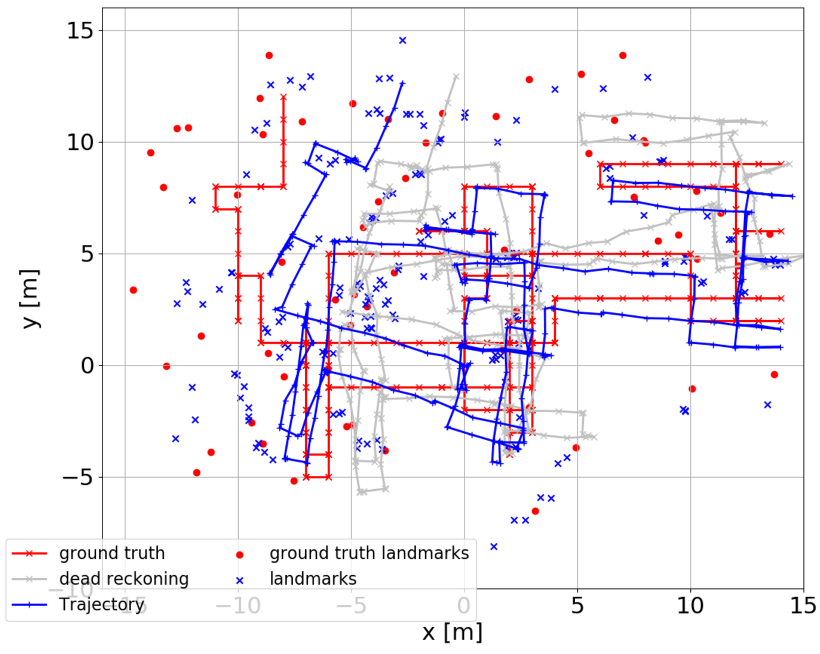


Figure 6.25: Maximum likelihood data association. Low P_{D}.

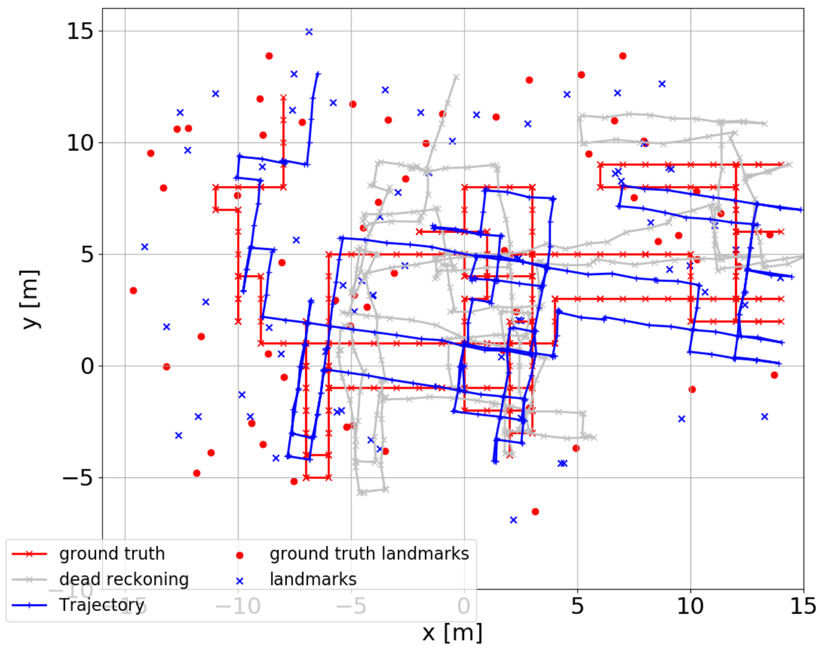


Figure 6.26: V-GLMB-SLAM solution. Low P_{D}.

algorithm, respectively. In these examples the odometry noise was fixed at

$$\Omega_u = \begin{bmatrix} 70 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 700 \end{bmatrix} , \tag{6.66}$$

while the measurement information was

$$\Omega_z = \begin{bmatrix} 300 & 0 \\ 0 & 300 \end{bmatrix} , \tag{6.67}$$

and the probability of detection was set to a high value $P_\mathrm{D} = 0.95$. As can be seen from these figures all three algorithms perform adequately. Then in figures 6.21, 6.22 and 6.23 the probability of detection was lowered to $P_\mathrm{D} = 0.8$. In these figures it can be appreciated that both maximum likelihood and nearest neighbor fail have higher error in their estimated trajectories. This can be explained due to the fact that they overestimate the number of landmarks. it can be seen in several spots that both algorithms have multiple estimated landmarks where they should have only a single landmark. V-GLMB-SLAM however is able to produce adequate estimates of both the trajectory and landmarks. When lowering the probability of detection even more to $P_\mathrm{D} = 0.5$, it can be seen from figures 6.24, 6.25 and 6.26 that the same effect are even more pronounced, and while the performance of V-GLMB-SLAM degrades, according to the lower quality data, the trajectory and map are still acceptable with only a global rotation.

Figures 6.27 to 6.35 show the effects of varying the measurement noise/information. In these examples the odometry noise was fixed at

$$\Omega_u = \begin{bmatrix} 70 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 700 \end{bmatrix} , \tag{6.68}$$

while the probility of detection was fixed to $P_\mathrm{D} = 0.8$. Figures 6.27, 6.28 and 6.29 show the solutions obtained by using the g2o optimization library with nearest neighbor data association, maximum likelihood data association, and our proposed algorithm, respectively, all with a low measurement information of

$$\Omega_z = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix} . \tag{6.69}$$

Interestingly under higher measurement noise all three algorithms perform adequately. Then

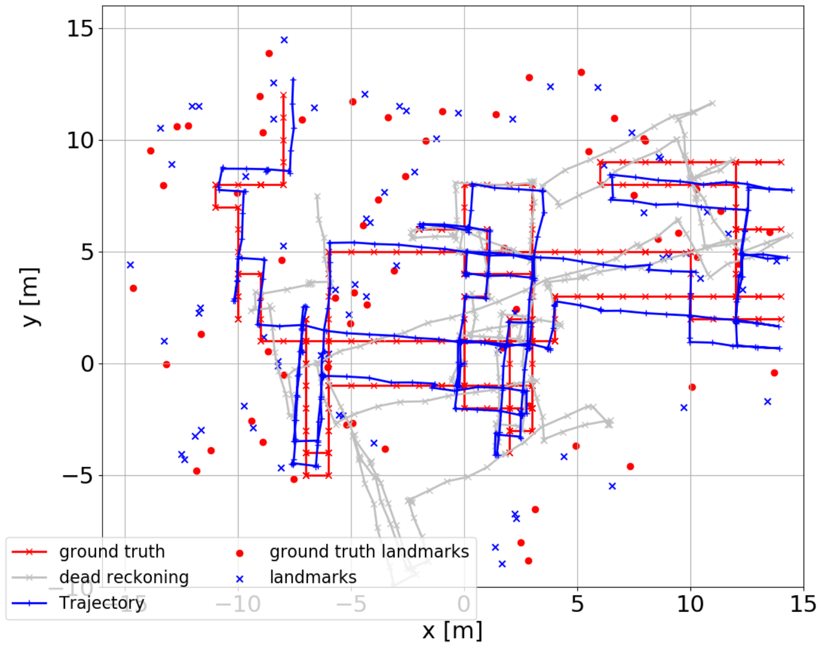


Figure 6.27: Nearest neighbor data association. Low measurement Information.

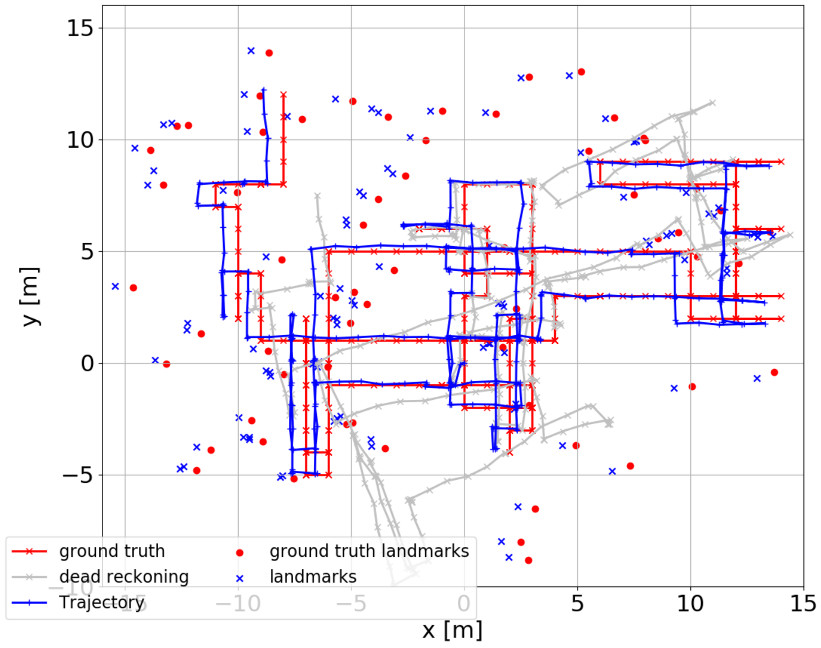


Figure 6.28: Maximum likelihood data association. Low measurement Information.

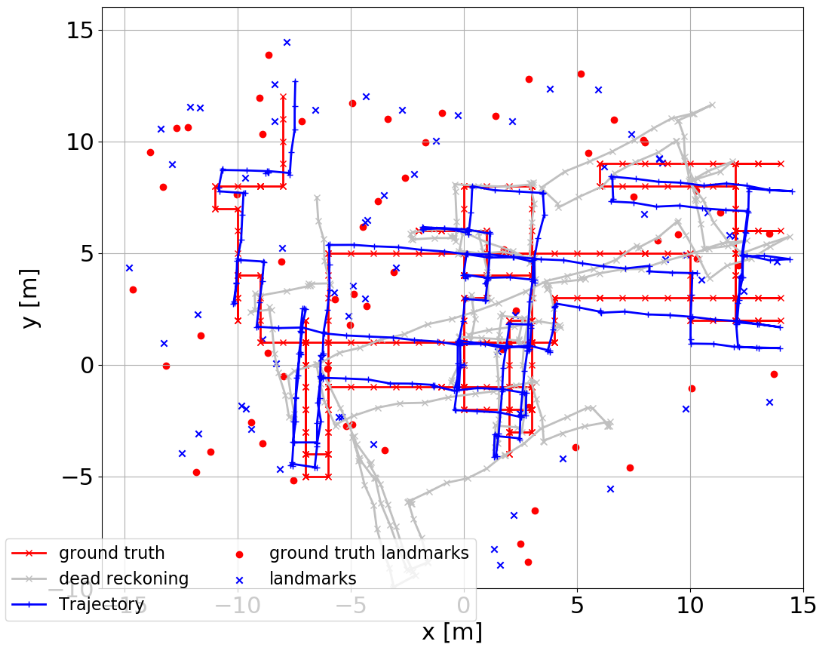


Figure 6.29: V-GLMB-SLAM solution. Low measurement Information.

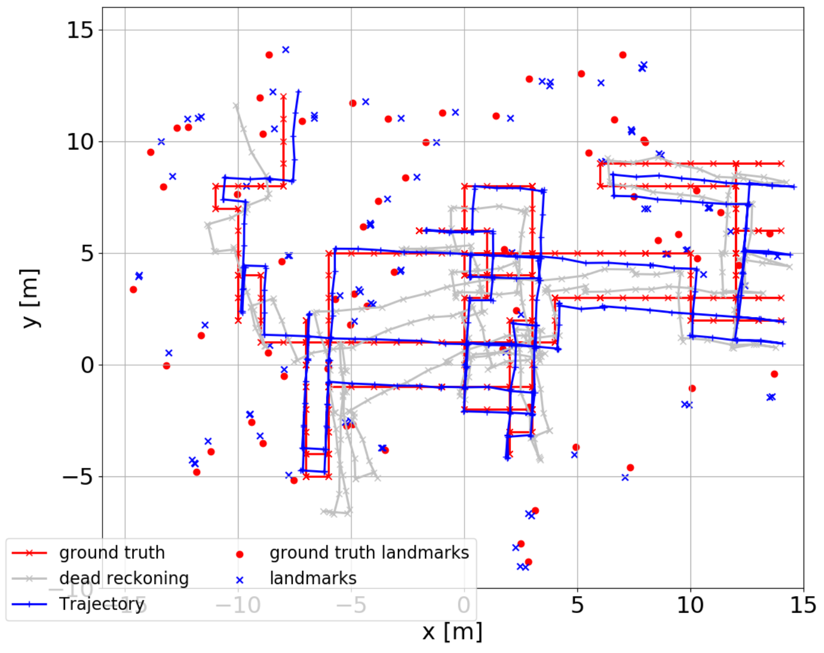


Figure 6.30: Nearest neighbor data association. Medium measurement Information.

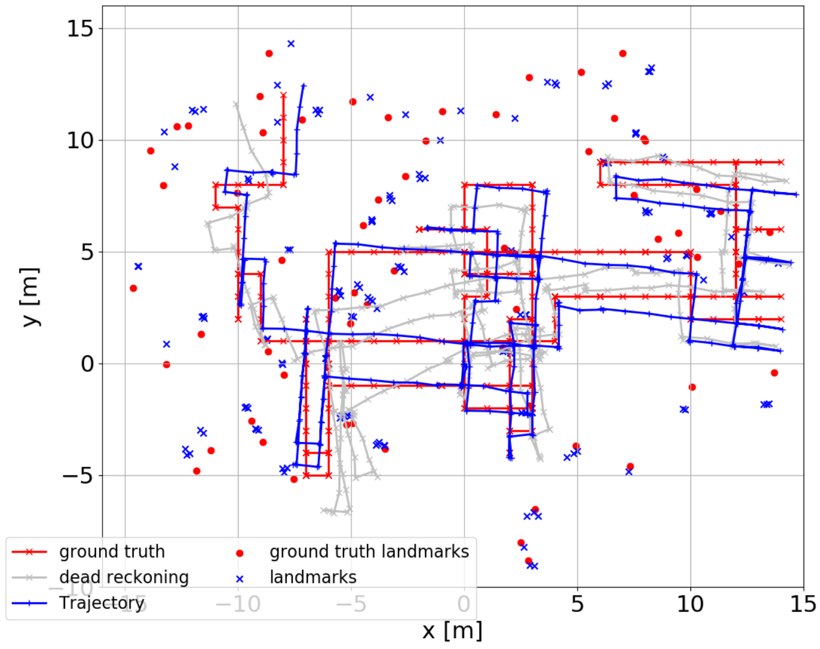


Figure 6.31: Maximum likelihood data association. Medium measurement Information.

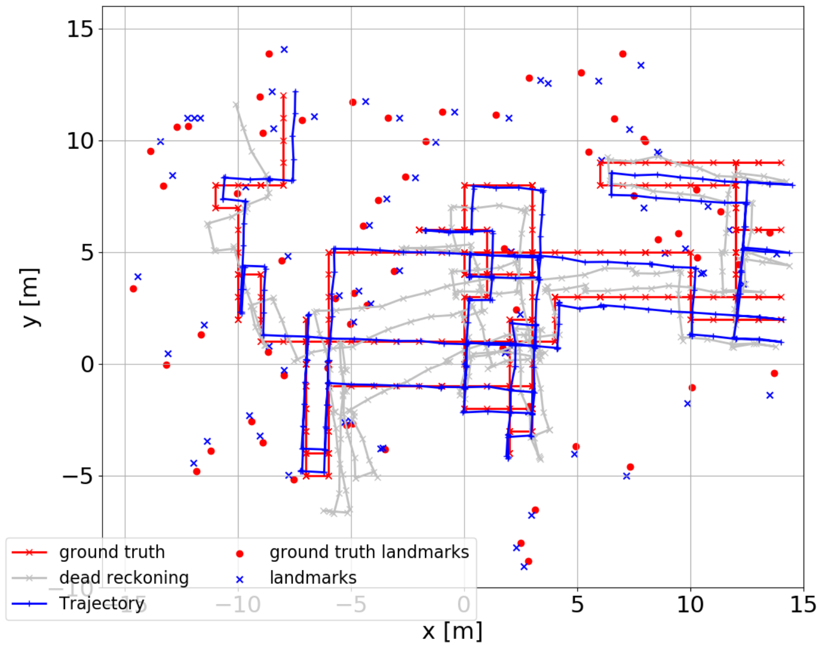


Figure 6.32: V-GLMB-SLAM solution. Medium measurement Information.

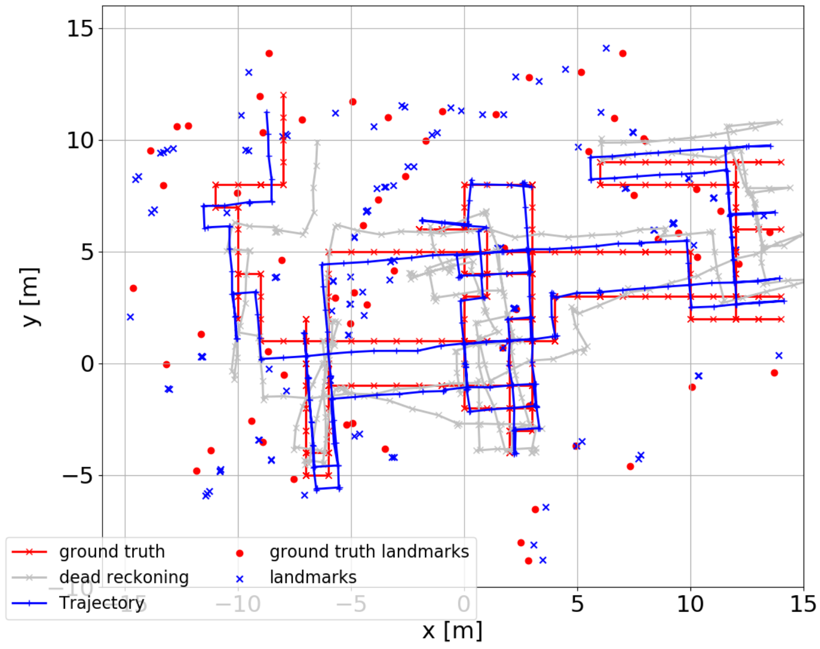


Figure 6.33: Nearest neighbor data association. High measurement Information.

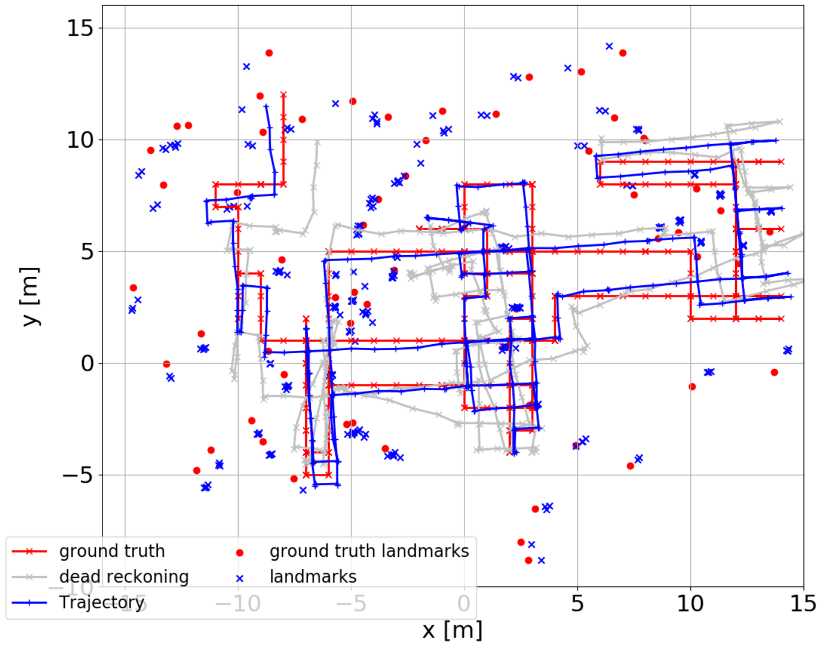


Figure 6.34: Maximum likelihood data association. High measurement Information.

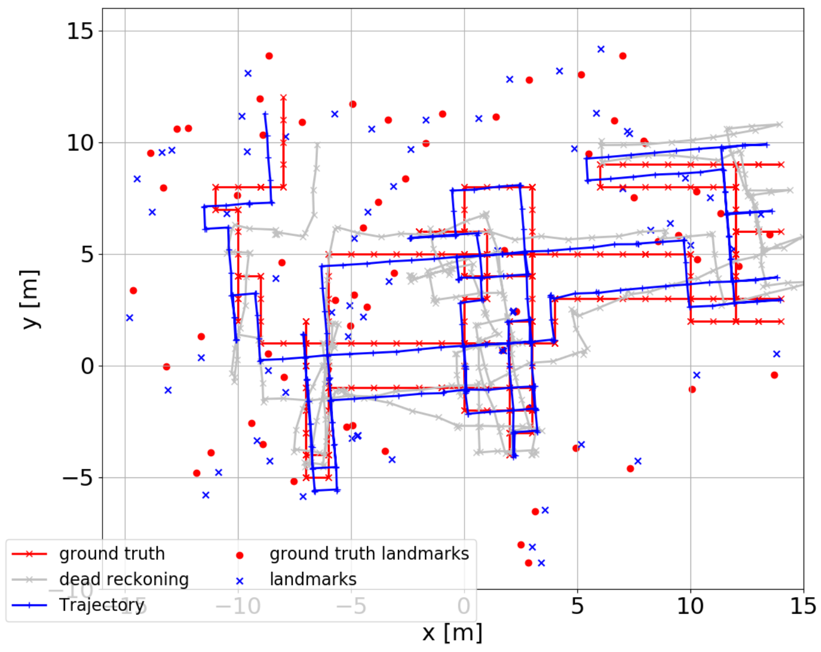


Figure 6.35: V-GLMB-SLAM solution. High measurement Information.

when decreasing the measurement noise, by setting

$$\Omega_z = \begin{bmatrix} 300 & 0 \\ 0 & 300 \end{bmatrix} , \tag{6.70}$$

as can be seen in figures 6.30, 6.31 and 6.32, the robustness of V-GLMB-SLAM starts to show. Again both maximum likelihood and nearest neighbor data association make the same mistake of splitting landmarks into multiple estimates. This effect only increases when further increasing the measurement information to

$$\Omega_z = \begin{bmatrix} 700 & 0 \\ 0 & 700 \end{bmatrix} \tag{6.71}$$

in figures 6.33, 6.34 and 6.35.

6.8 Stereo Visual SLAM results

The algorithm proposed in the simulations was adapted to work with visual stereo data, specifically from the Euroc dataset. Visual data was chosen over 3D LIDAR data because of the availability of keypoint detector descriptor pairs. 3D LIDAR SLAM is typically performed either with scan-matching approaches or with landmark detectors that can produce multiple detections for a single landmark, which violates the Measurement Model used in the current formulation. Extended target RFS detection models could be used along with this kind of data in future research.

Therefore a state-of-the-art visual SLAM algorithm was chosen as inspiration for the development of a V-GLMB-based SLAM algorithm. In particular the ORBSLAM3 algorithm [25] was chosen due to its performance, robustness and open source codebase. As in all three ORB-SLAM algorithms, the ORB keypoint detector and descriptor are used by V-GLMB-SLAM as its measurements.

ORB is a visual keypoint detector and descriptor. The detection part of ORB is based on the FAST keypoint detector [60], with an additional orientation component. Each detection is then enhanced with a Binary Robust Independent Elementary Features (BRIEF) descriptor [61]. This descriptor is a binary string computed by a comparing the pixel values of predefined pairs in a vicinity of the keypoint, these pairs are rotated by the keypoint orientation.

6.8.1 Measurement Model

Stereo measurements are modeled as

$$\mathbf{z}_k^i = [u_l \ , \ v_l \ , \ u_r \ , \ \text{orb}_z]^\intercal \ , \tag{6.72}$$

where $\mathbf{z}_k^i$ is a single element of the measurement set $\mathcal{Z}_k$, u_l and v_l are the left image pixel coordinates of the detection, u_r is the pixel x coordinate of the stereo detection in the right image, and orb_z is the ORB descriptor of measurement $\mathbf{z}_k^i$. v_r, the y coordinate of the pixel in the right image, is not used because its value is determined by the stereo epipolar constraint, therefore its value does not contribute information. The map elements of $\mathcal{M}$ are modeled as:

$$\mathbf{m}^j = [x \ , \ y \ , \ z \ , \ \text{orb}_m]^\intercal \ , \tag{6.73}$$

where x, y, and z are the 3D coordinates of the keypoint and orb_m is the ORB descriptor of the keypoint.

Using these definitions, the single feature measurement model is determined as

$$p\left(\mathbf{z}_k^i \middle| \mathbf{m}^{a_{\theta_k}(i)}, \mathbf{x}_k\right) = \mathcal{N}(u_l, v_l, u_r | \mu(\mathbf{m}^j, \mathbf{x}_k), \Sigma_{\mathbf{stereo}})\lambda_{\text{orb}} \exp(-\lambda_{\text{orb}} d_h(\text{orb}_z, \text{orb}_m)) \ , \tag{6.74}$$

where $d_h(\cdot)$ is the Hamming distance between the ORB descriptors of the map and measurement, λ_{orb} is a parameter of the distribution on this Hamming distance[7], $\mathbf{x}_k$ is the robot/sensor pose

[7]This parameter determines how well the ORB descriptor matches between detections

at time k, and $\Sigma_{\mathbf{stereo}}$ is the covariance of the resulting pixel coordinates given by

$$\Sigma_{\mathbf{stereo}} = \begin{bmatrix} \sigma_{\text{pix}}^2 & 0 & 0 \\ 0 & \sigma_{\text{pix}}^2 & 0 \\ 0 & 0 & \sigma_{\text{pix}}^2 \end{bmatrix} . \tag{6.75}$$

Another important component of the overall measurement model is the probability of detection model, which is determined based on the implementation of the ORBSLAM algorithms, as follows:

$$P_{\text{D}}^{(j,k)} = P_{\text{D}}(\mathbf{m}^j, \mathbf{x}_k) = P_{\text{D}} \cdot \text{isInFrustum}(\mathbf{m}^j, \mathbf{x}_k) \cdot \text{depthCheck}(\mathbf{m}^j, \mathbf{x}_k) \cdot \text{angleCheck}(\mathbf{m}^j, \mathbf{x}_k) \tag{6.76}$$

where P_{D} is a constant value, which is set as a parameter of the algorithm, isInFrustum$(\mathbf{m}^j, \mathbf{x}_k)$ is a function that checks that the keypoint $\mathbf{m}^j$ is within the frustum of both cameras, i.e. their predicted pixel values fall within both cameras' image sensors:

$$\text{isInFrustum}(\mathbf{m}^j, \mathbf{x}_k) = \begin{cases} 1 & \begin{aligned} &\text{if } \min_u \leq \hat{u}_l, \hat{u}_r \leq \max_u\,, \\ &\text{and } \min_v \leq \hat{v}_l \leq \max_v\,, \\ &\text{and } \hat{z} > 0\,, \end{aligned} \\ 0 & \text{else} \quad , \end{cases} \tag{6.77}$$

where $\hat{u}_l, \hat{u}_r, \hat{v}_l$ are the predicted pixel coordinates based on the robot pose $\mathbf{x}_k$ and keypoint estimate $\mathbf{m}^j$, $\min_u, \max_u, \min_v, \max_v$ are the minimum and maximum pixel coordinates, and $\hat{z}$ is the z coordinate of the keypoint in the camera frame, confirming that the keypoint is not behind the camera.

depthCheck$(\mathbf{m}^j, \mathbf{x}_k)$ is a function that checks that the predicted depth (the distance between the camera pose $\mathbf{x}_k$ and the keypoint $\mathbf{m}^j$) is within the minimum and maximum depth values, mindistance$(\mathbf{m}^j)$ and maxdistance$(\mathbf{m}^j)$, respectively:

$$\text{depthCheck}(\mathbf{m}^j, \mathbf{x}_k) = \begin{cases} 1 & \text{if mindistance}(\mathbf{m}^j) \leq \|\mathbf{m}^j - \mathbf{x}_k\| \leq \text{maxdistance}(\mathbf{m}^j)\,, \\ 0 & \text{else}\,. \end{cases} \tag{6.78}$$

These minimum and maximum values are calculated, in a manner similar to those used in ORB-SLAM, by using the detections already in the detection hypothesis of $\mathbf{m}^j$. For each detection, the predicted maximum depth is calculated as 1.8scaleFactor(level), where scaleFactor(level) is the scale factor of the level at which the ORB detection was detected. maxdistance$(\mathbf{m}^j)$ is then the average of the max distances calculated from each detection. The minimum depth is calculated as $\text{mindistance}(\mathbf{m}^j) = \frac{0.4\text{maxdistance}(\mathbf{m}^j)}{\text{scaleFactor}(\text{maxlevel})}$.

$\text{angleCheck}(\mathbf{m}^j, \mathbf{x}_k)$ is a function that checks the viewing angle is consistent with previous detections,

$$\text{angleCheck}(\mathbf{m}^j, \mathbf{x}_k) = \begin{cases} 1 & \text{if } \left\langle \hat{N}(\mathbf{m}^j), \frac{\mathbf{m}^j - \mathbf{x}_k}{\|\mathbf{m}^j - \mathbf{x}_k\|} \right\rangle < \text{viewingCosLimit}\,, \\ 0 & \text{else}\,, \end{cases} \tag{6.79}$$

where viewingCosLimit is a parameter that limits the maximum possible viewing angle for a keypoint to be in the Field of view. To determine this, the average viewing direction $\hat{N}(\mathbf{m}^j)$ is computed using all current associations by averaging the normalized difference between the robot position $\mathbf{x}_k$ and the keypoint $\mathbf{m}^j$.

$$\hat{N}(\mathbf{m}^j) = \frac{\sum_{k \in \text{isdetected}(\mathbf{m}^j)} \frac{\mathbf{m}^j - \mathbf{x}_k}{\|\mathbf{m}^j - \mathbf{x}_k\|}}{\sum_{k \in \text{isdetected}(\mathbf{m}^j)} 1}\ . \tag{6.80}$$

6.8.2 Motion Model

The drone's motion is modeled with pure Brownian motion, which is the simplest type of motion model. This is implemented using the GTSAM "BetweenFactor", which simply evaluates it's error as a function of the difference between two poses. Since this factor does not involve any set states[8], then changing the factor to a more advanced model, such as one including IMU measurements, does not change the theoretical formulation.

6.8.3 Keypose logic

As more and more robot poses are continually being added to the trajectory being optimized, the computational cost of running the optimization grows without bounds. A common way to reduce the speed at which this cost increases is to only optimize a subset of the trajectory's poses. These poses can be determined using heuristic logic, as is done in ORBSLAM [25]. In that approach, the number of keypoints that are detected in the current pose/frame are also detected in other already existing keyposes/keyframes are counted. If most of the keypoints detected in the current frame are detected in multiple other keyframes, then the current frame is determined not to be a keyframe, else the current frame is determined to be a keyframe. However, the joint formulation being explored in this thesis has different data associations for

[8]The motion model factor only includes two robot poses, $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, both of which are in the vector portion of the V-GLMB model.

Algorithm	V201	V202	V203
ORBSLAM2	0.037	0.035	–
ORBSLAM3	0.041	0.028	0.521
Vins-Fusion	0.230	0.200	–
SVO	0.050	0.090	0.790
V-GLMB-SLAM	0.038	0.046	–

Table 6.2: Absolute Trajectory Error on the Euroc Vicon Room 2 dataset.

every component in the current estimate. This implies that, were the keypose logic to depend on the associations, as in ORBSLAM, then the set of keyposes being optimized would vary from component to component in the V-GLMB formulation, rendering the components incomparable with each other with respect to their weight. For this reason, the keypose logic here is limited to using a predetermined set of poses. Having an adaptable logic for keyposes, while keeping the functionality of the Bayesian inference, is left as a possible avenue of future research.

For the stereo vision implementation, the keypose logic was determined with two integer parameters numPosesToOptimize and keyposeSkip. The most recent numPosesToOptimize are all included in the optimization, while one out of every keyposeSkip of the older poses are included in the optimization as keyposes.

6.8.4 Euroc Dataset results

With all these steps defined, the completed algorithm can be executed using the ORB stereo detections on the Euroc Vicon Room 2 (V02) dataset. This dataset consists of three separate trajectories V201, V202, and V203. These trajectories varied in the motion of the drone, which had increasing speeds and accelerations and were labeled as “easy”, “medium”, and “difficult” for the V201, V202, and V203 trajectories, respectively. The results with this particular dataset can be compared with some of the state-of-the-art SLAM methods by calculating the ATE metric. Table 6.2 shows the RMSE ATE metric for V-GLMB-SLAM and 4 other state-of-the-art algorithms, namely ORBSLAM2, ORBSLAM3, Vins-Fusion, and SVO. The row corresponding to V-GLMB-SLAM was filled using the evo python package [59] to calculate the RMSE ATE. The rest of the rows were filled using the values from [25]. It can be seen from the table that even though V-GLMB-SLAM is not the best in each group, it is highly competitive while solving the data association problem in a purely Bayesian framework, removing the need for the data association heuristics, necessary with the other algorithms. Note that the entries in Table 6.2 containing a “-” mean that the corresponding SLAM algorithm was unable to provide an estimate for the entire trajectory.

Figures 6.36 to 6.50 show the detailed trajectories estimated by V-GLMB-SLAM for each of the three trajectories. The ground truth trajectory is shown as a grey dashed line, while the estimated trajectory is shown as a colored solid line. The start of each trajectory is marked with a black circle. The estimated line color shows the instantaneous position error in meters. As can be seen from Figure 6.36, in the "easy" dataset the whole trajectory is estimated very accurately, with its highest error being at the end of the trajectory. This can be easily explained by the fact that when landing the drone, some of its highest accelerations occured, while simultaneously the stereo camera points at the floor which has low texture and does not produce ORB detections. Figures 6.37 and 6.38 show top-down and side view 2D perspectives, respectively. Figure 6.39 show the trajectory obtained from running the latest version of the publicly available ORBSLAM3 code[9]. Importantly, the performance is lower than the values that Table 6.2 indicates, which may have been run with a different version of the orbslam3 code, or run with a more powerful computer. Figure 6.40 shows a plot of the instantaneous position error for V-GLMB-SLAM in blue and ORBSLAM3 in green, used to calculate the ATE metric. From this plot it can be seen that V-GLMB-SLAM instantaneous error is lower most of the time, while at the end of the trajectory ORBSLAM3 does not provide an estimate, this is because the ORBSLAM3 algorithm is in its lost state for the same reasons that the V-GLMB-SLAM has its highest error in that part of the trajectory. Arguably, ORBSLAM3 could detect and close a loop were the trajectory to continue further. Figure 6.41 shows a violin plot [62] of the error for both V-GLMB-SLAM in blue and ORBSLAM3 in green. As can be seen from the figure, in this dataset the error of V-GLMB-SLAM is slightly lower than the one obtained with the currently available version of ORBSLAM3.

A similar performance resulted for trajectory V202 as shown in Figures 6.42, 6.43, and 6.44. This time however we can see some artifacts, or single poses with quite high instantaneous error, as is particularly evident in Figure 6.43 near $(x, y) = (2.0, 2.5)$. Figure 6.45 shows the trajectory obtained from running the publicly available ORBSLAM3 code. Figure 6.46 shows the instantaneous position error of V-GLMB-SLAM, as a blue line, and ORBSLAM3, as a green line. From this plot it can be seen that the error of V-GLMB-SLAM is lower than ORBSLAM3, while it has some peaks of instantanously high error. Figure 6.47 shows a violin plot [62] of the error for both V-GLMB-SLAM in blue and ORBSLAM3 in green. Interestingly, as can be seen in figure 6.47, with this version of the ORBSLAM3 code, the performance is significantly lower than V-GLMB-SLAM error. However, it should be noted that ORBSLAM3 is a real-time algorithm while V-GLMB-SLAM currently runs somewhere between 50 to 100 times slower than real-time.

In the "difficult" dataset, which has the highest speeds and accelerations, the V-GLMB-

[9]Taken from the git repository at `https://github.com/UZ-SLAMLab/ORB_SLAM3` with the latest version available, i.e. commit 4452a3c4ab75b1cde34e5505a36ec3f9edcdc4c4 .

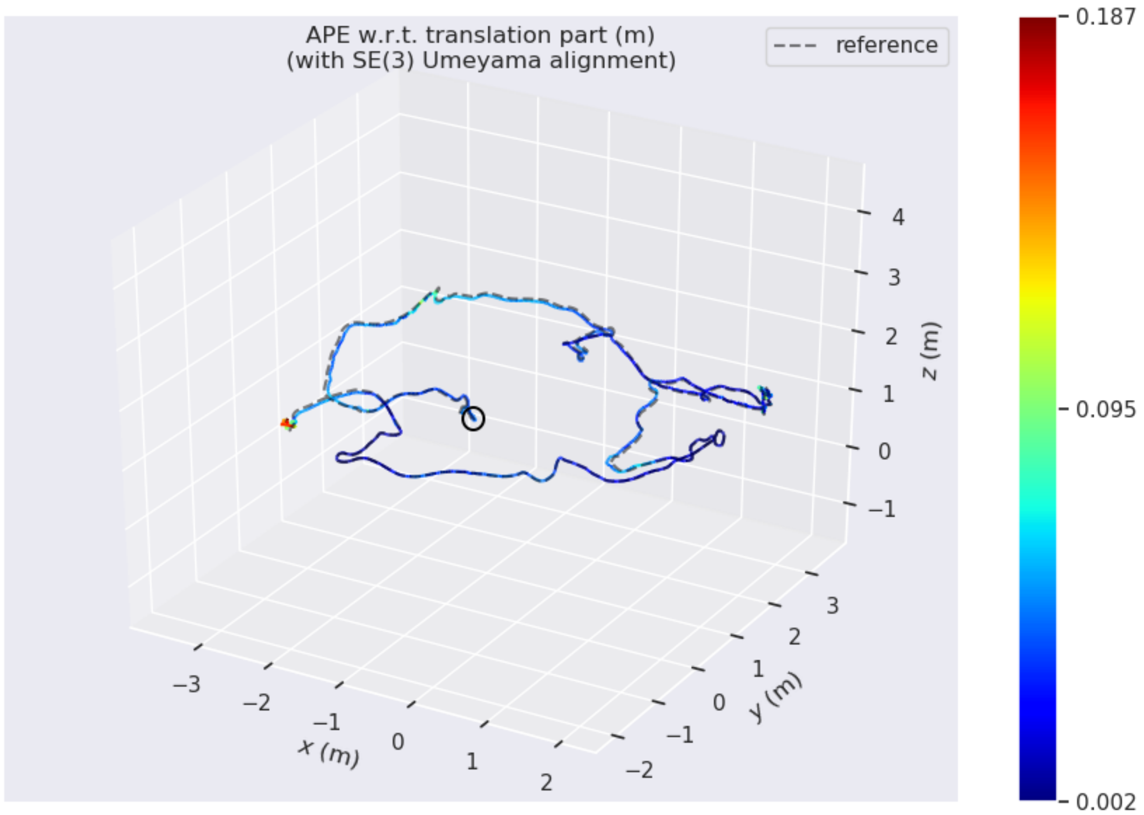


Figure 6.36: V-GLMB-SLAM results with the euroc V201 “easy” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

SLAM algorithm fails to produce a complete estimate of the trajectory. This happens when the Gibbs sampler stops producing data associations that produce V-GLMB components with higher weights. Figures 6.48, 6.49, and 6.50 show the incomplete trajectory estimate. Figure 6.51 shows the trajectory obtained from running the publicly available ORBSLAM3 code. Figure 6.52 shows a plot of the instantaneous position error, of the estimated part of the trajectory, shown for V-GLMB-SLAM as a blue line and for ORBSLAM3 as a green line. We can see in these figures that the trajectory error for V-GLMB-SLAM is at its highest just before the V-GLMB-SLAM algorithm stops producing more and more likely data associations. It can also be seen that with this version of ORBSLAM3, the algorithm gets lost multiple times and although it recovers partially from some of them is not able to recover from its final lost state and therefore it does not provide an estimate of the full trajectory. The histogram (violin plot) for this trajectory is not plotted because with both algorithms estimating different parts of the trajectory it would not be meaningful to compare them.

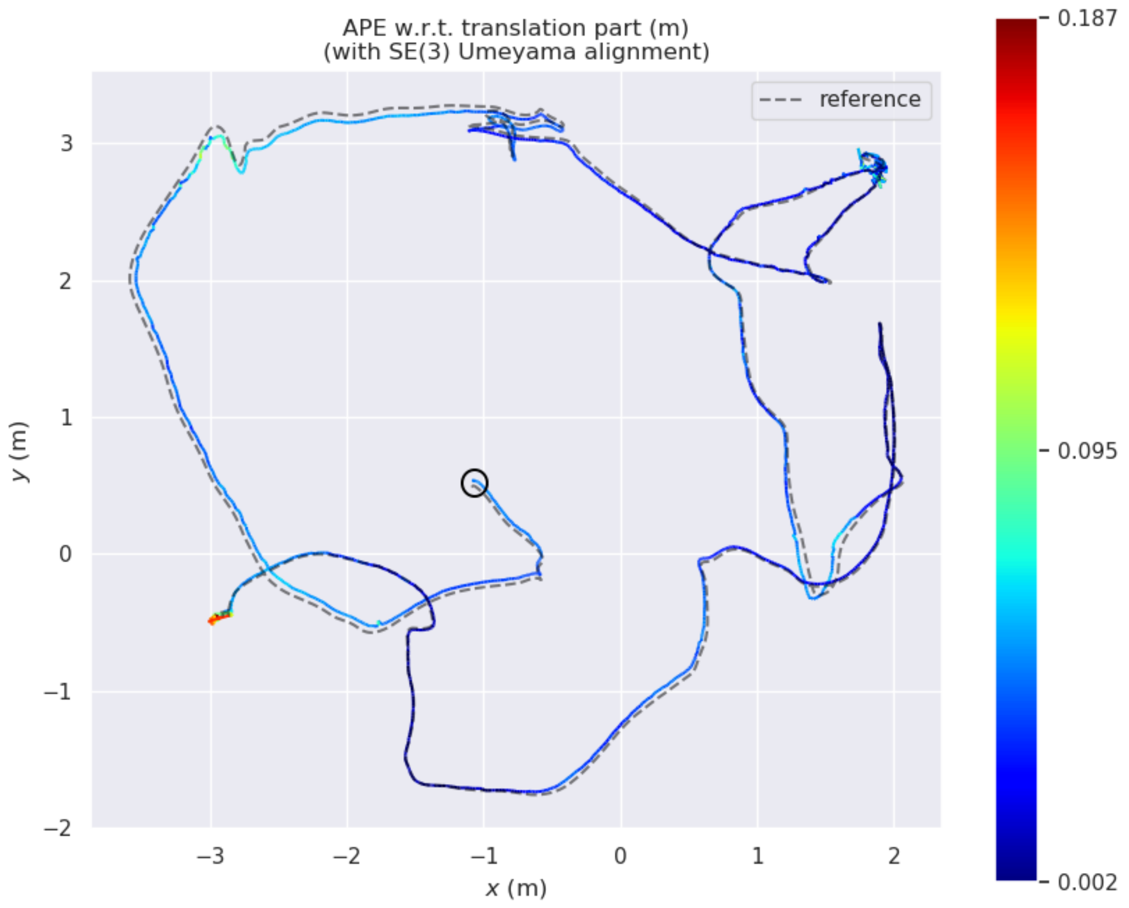


Figure 6.37: V-GLMB-SLAM results with the euroc V201 “easy” dataset - Top-down view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

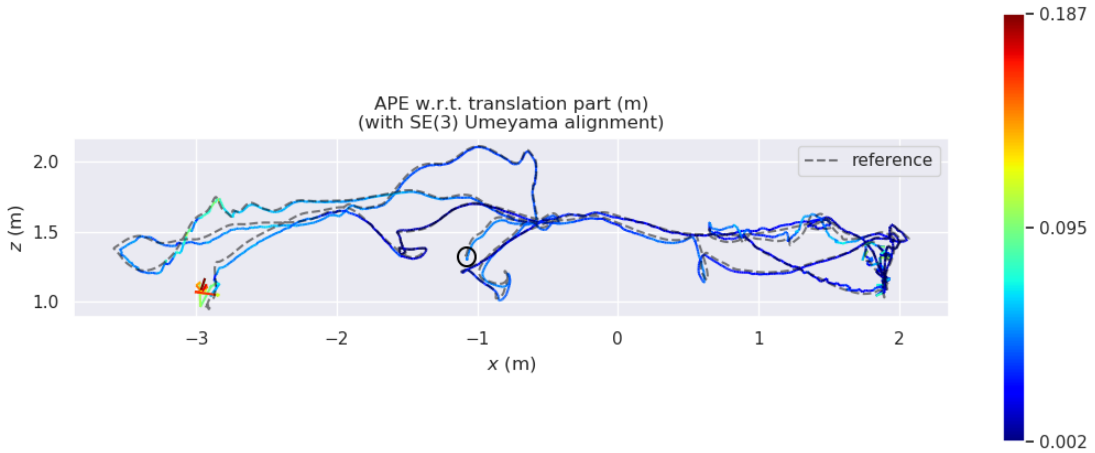


Figure 6.38: V-GLMB-SLAM results with the euroc V201 "easy" dataset - Side view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

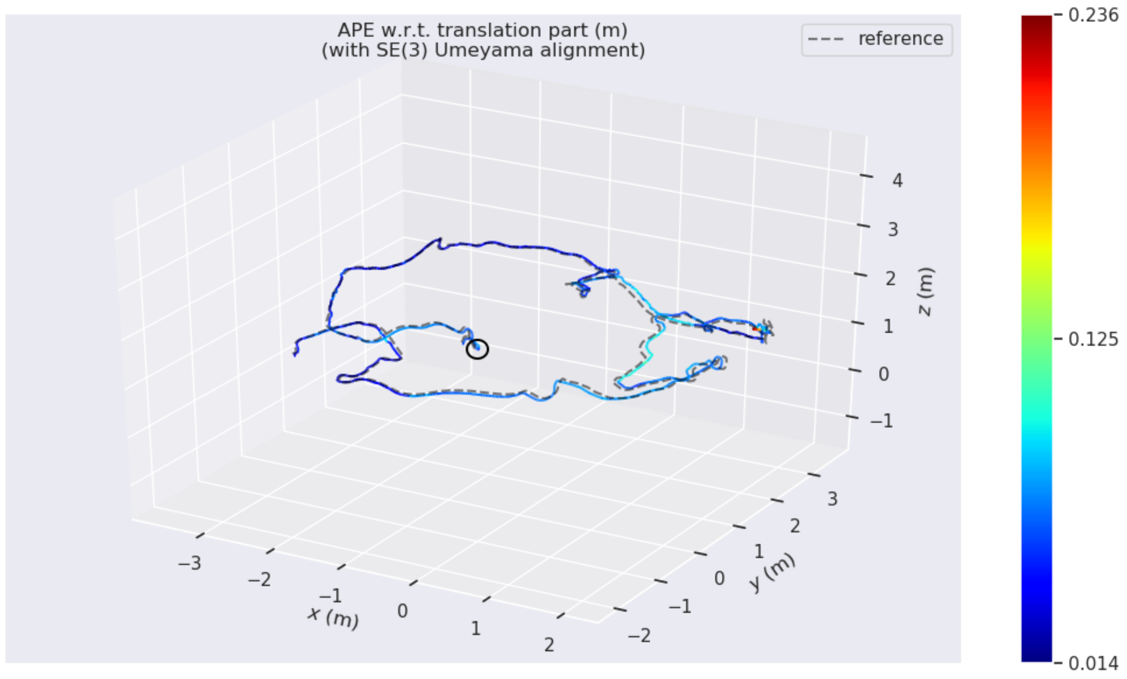


Figure 6.39: ORBSLAM3 results on the euroc V201 "easy" dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

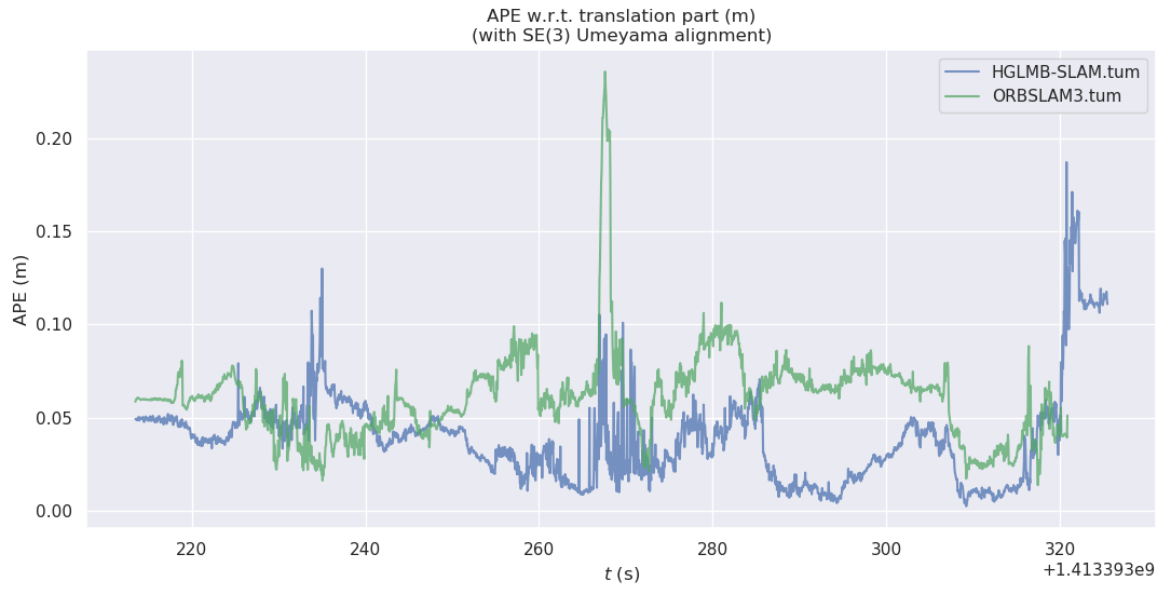


Figure 6.40: Position error over time of V-GLMB-SLAM, shown as a blue line , and ORB-SLAM3, shown as a green line, with the euroc V201 “easy” dataset. The x axis shows time and the y axis is the instantaneous position error in meters.

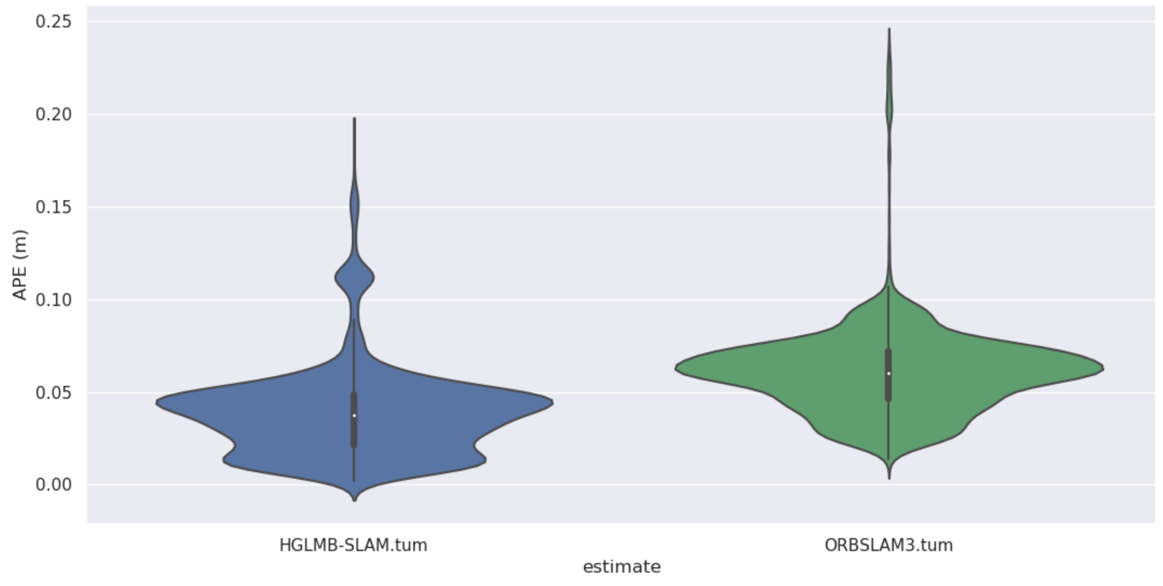


Figure 6.41: Histogram of the position error of V-GLMB-SLAM and ORBSLAM3 with the euroc V201 “easy” dataset. The y axis shows position error in meters and the width of each plot shows the frequency of that error.

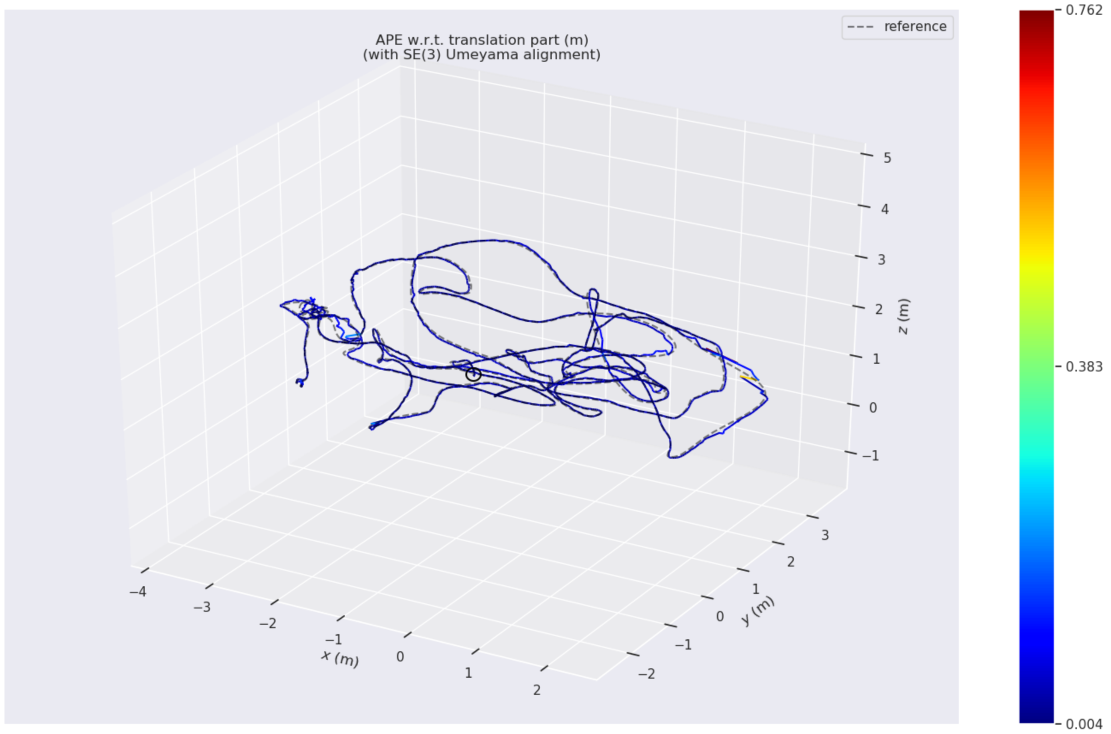


Figure 6.42: V-GLMB-SLAM results with the euroc V202 “medium” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

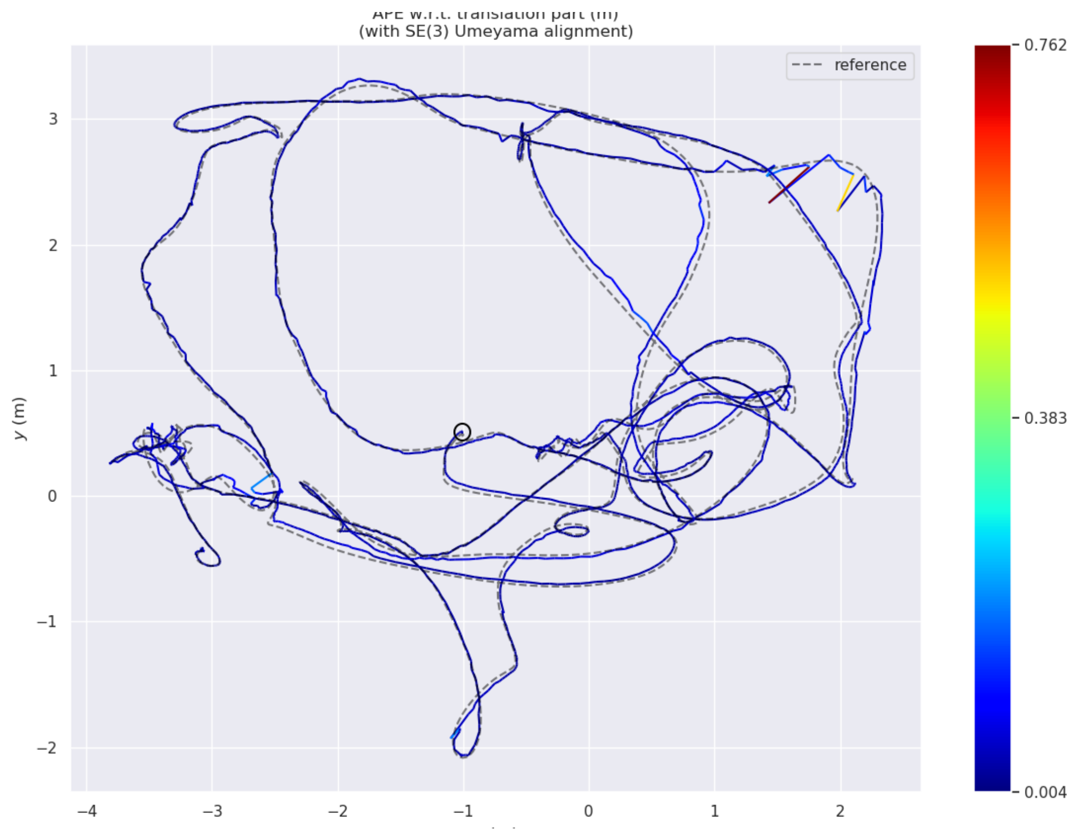


Figure 6.43: V-GLMB-SLAM results with the euroc V202 "medium" dataset - Top-down view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

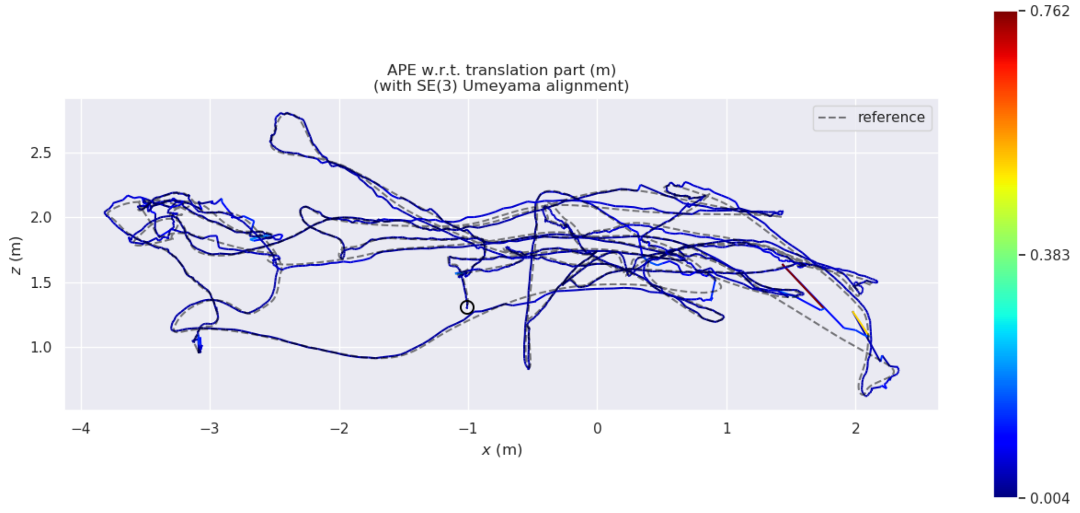


Figure 6.44: V-GLMB-SLAM results with the euroc V202 "medium" dataset - Side view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters.

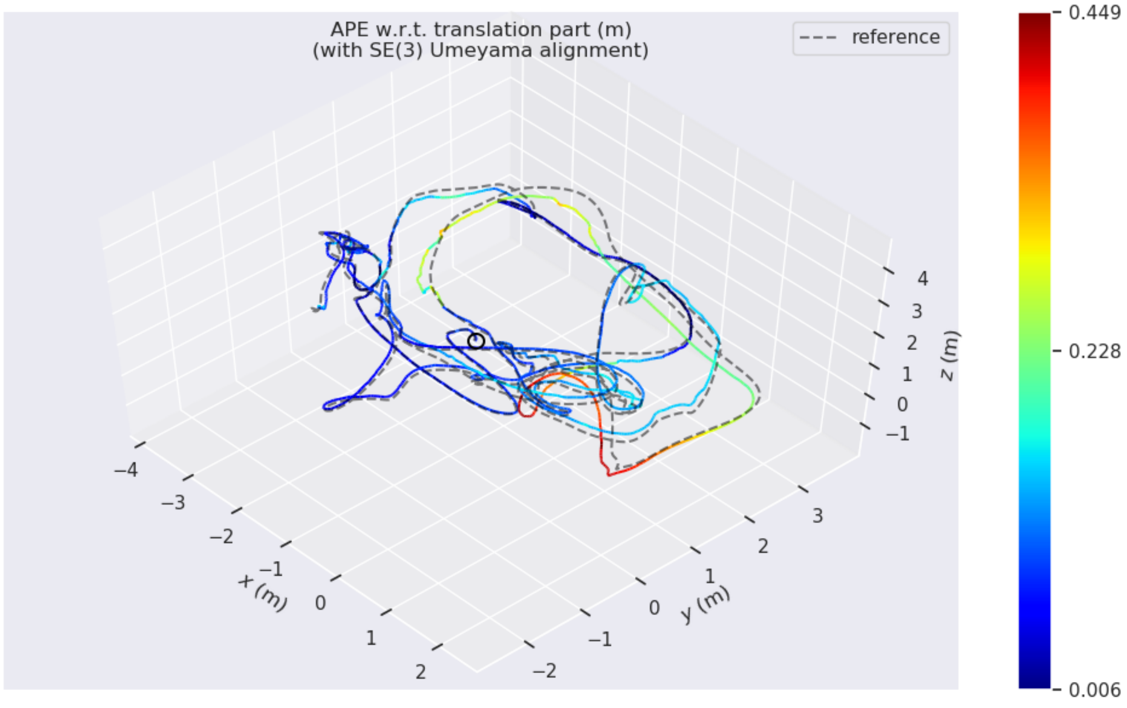


Figure 6.45: ORBSLAM3 results on the euroc V202 "medium" dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

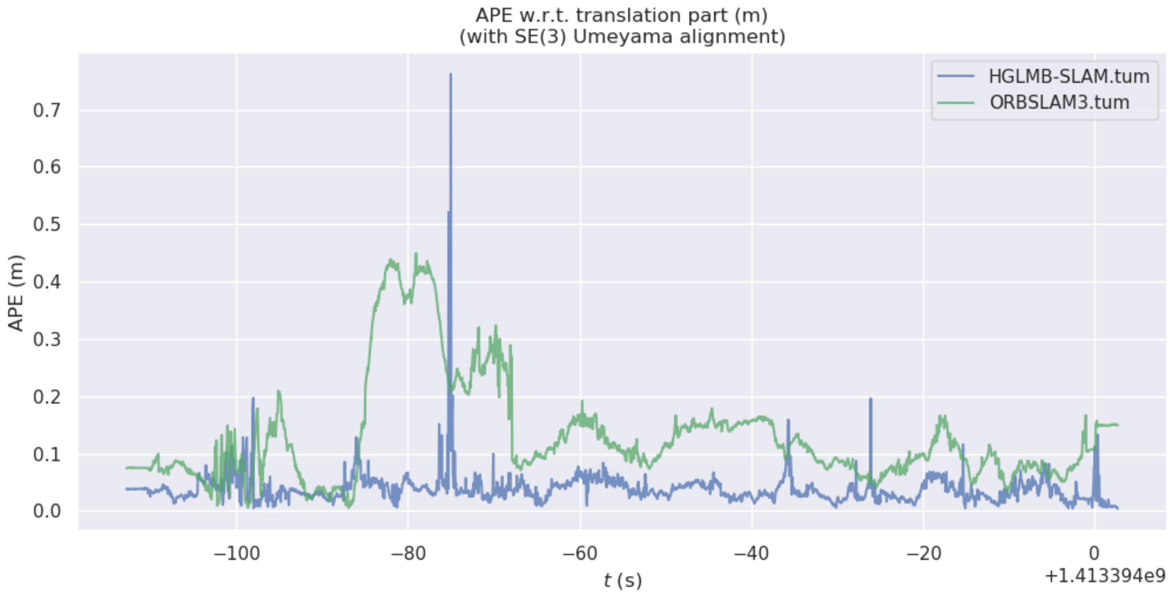


Figure 6.46: Position error over time of V-GLMB-SLAM results with the euroc V202 "medium" dataset. The x axis shows time and the y axis is the instantaneous position error in meters.

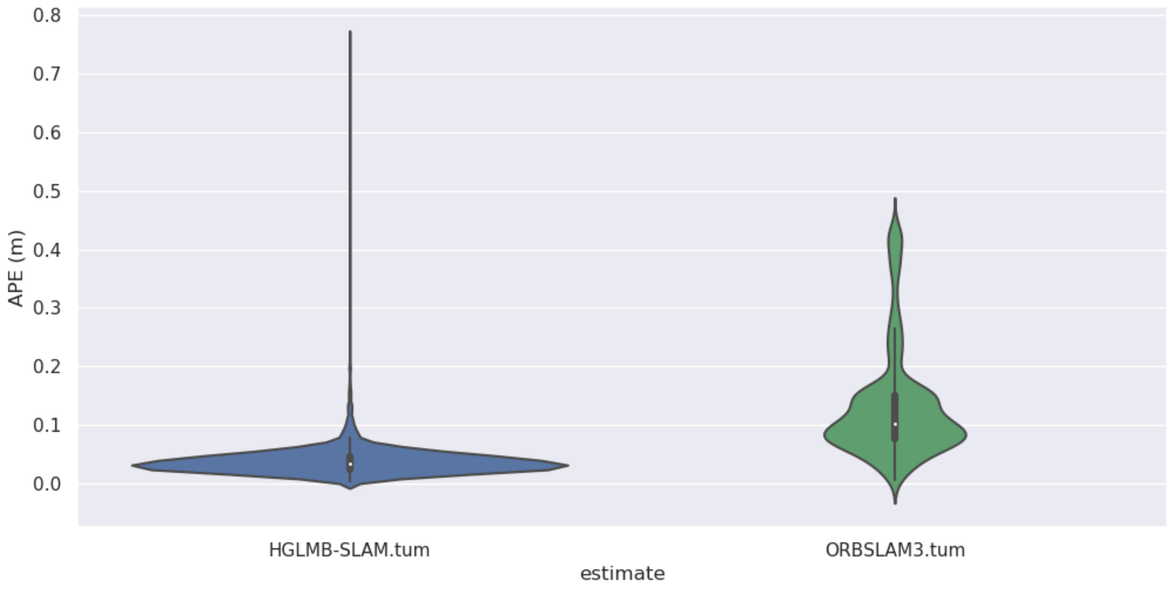


Figure 6.47: Position error over time of ORBSLAM3 results on the euroc V202 “medium” dataset. The x axis shows time and the y axis is the instantaneous position error in meters.

Figures 6.53 and 6.54 show interesting intermediate results on the V202 dataset. These results occur just before and just after the stereo camera turns and observes keypoints that were first detected close to the start of the trajectory, thus closing a “loop”. As can be seen from Figure 6.54, the error is considerably lowered and the trajectory is significantly corrected backward by more than 100 poses. This type of “loop closure” corrects too many poses for a particle filter-based SLAM algorithm to be able to correct, since the number of particles needed to represent the uncertainty over a trajectory this long would be prohibitively high.

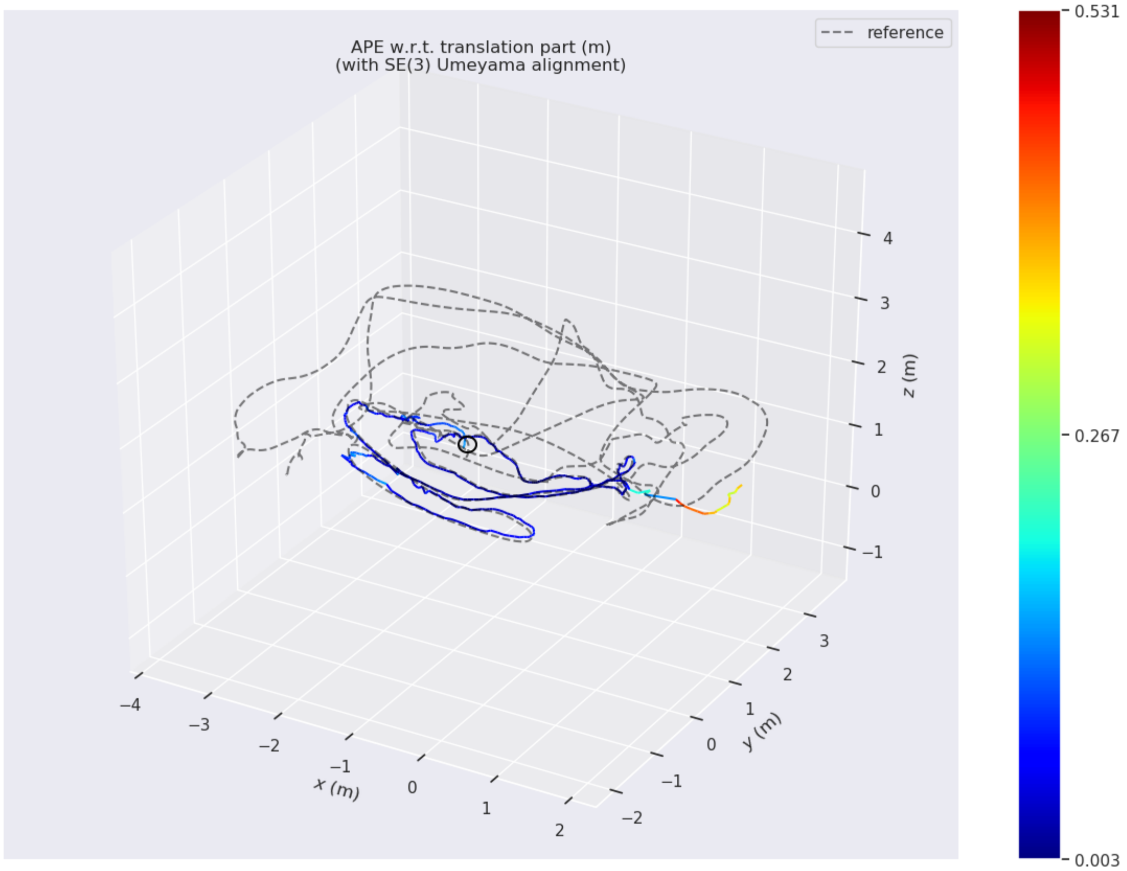


Figure 6.48: V-GLMB-SLAM results with the euroc V203 “difficult” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

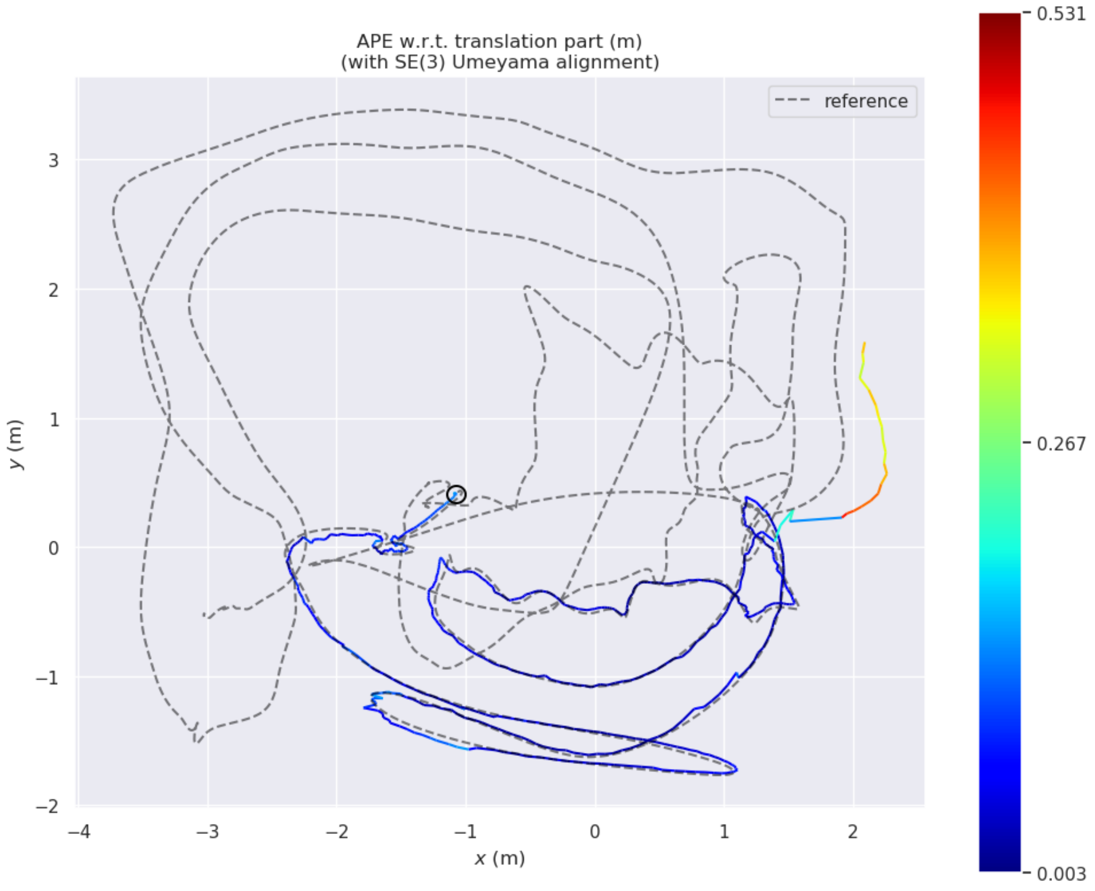


Figure 6.49: V-GLMB-SLAM results with the euroc V203 “difficult” dataset - Top-down view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

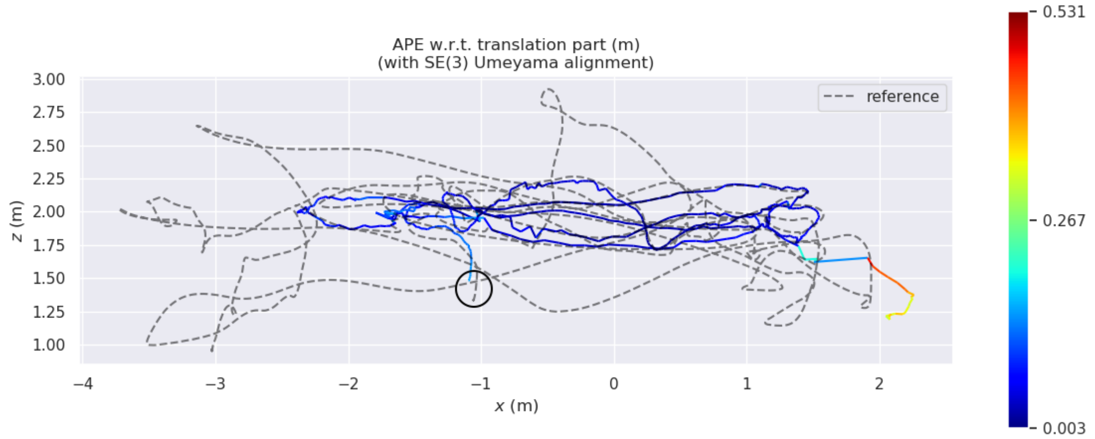


Figure 6.50: V-GLMB-SLAM results with the euroc V203 “difficult” dataset - Side view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

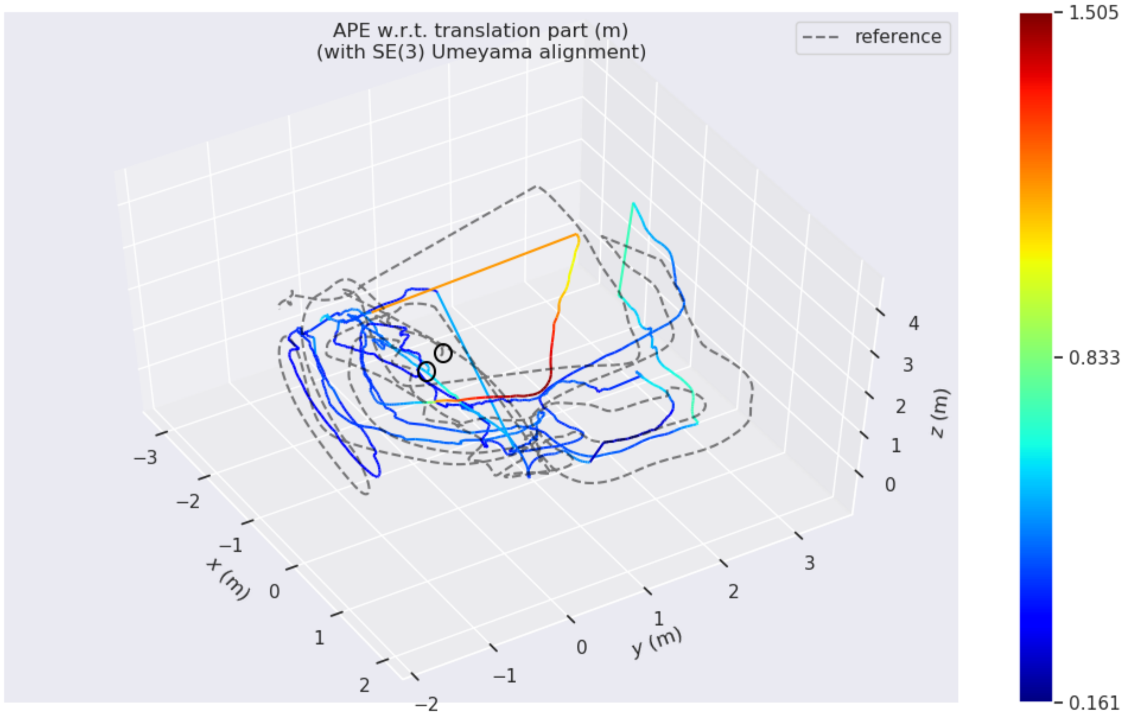


Figure 6.51: ORBSLAM3 results on the euroc V203 “difficult” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of both trajectories are marked with black circles.

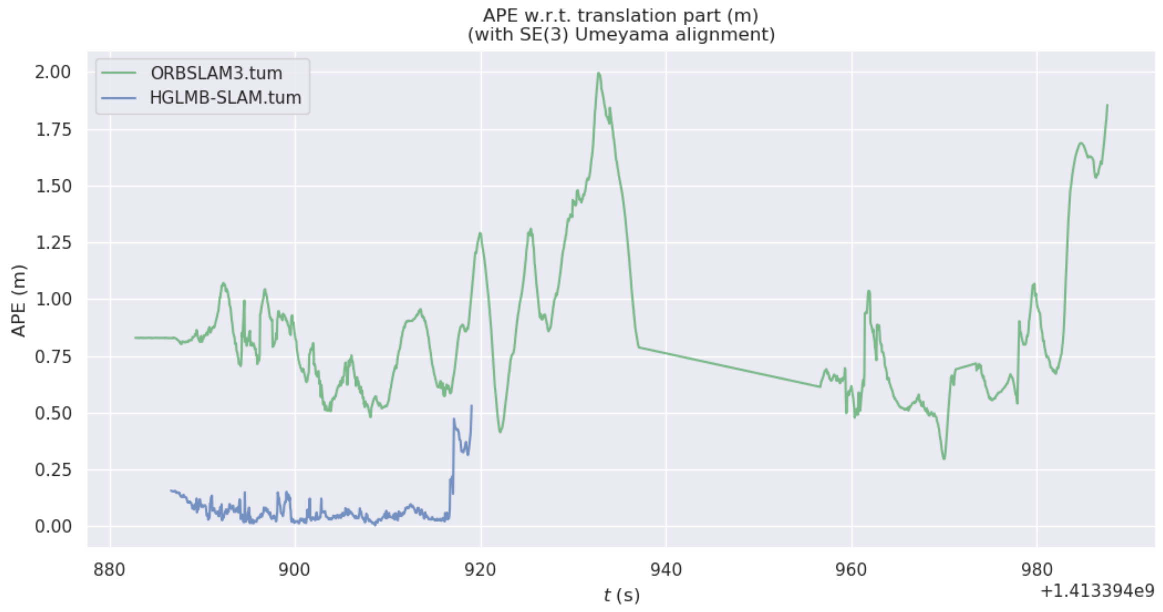


Figure 6.52: Position error over time of V-GLMB-SLAM results with the euroc V203 “difficult” dataset. The x axis shows time and the y axis is the instantaneous position error in meters.

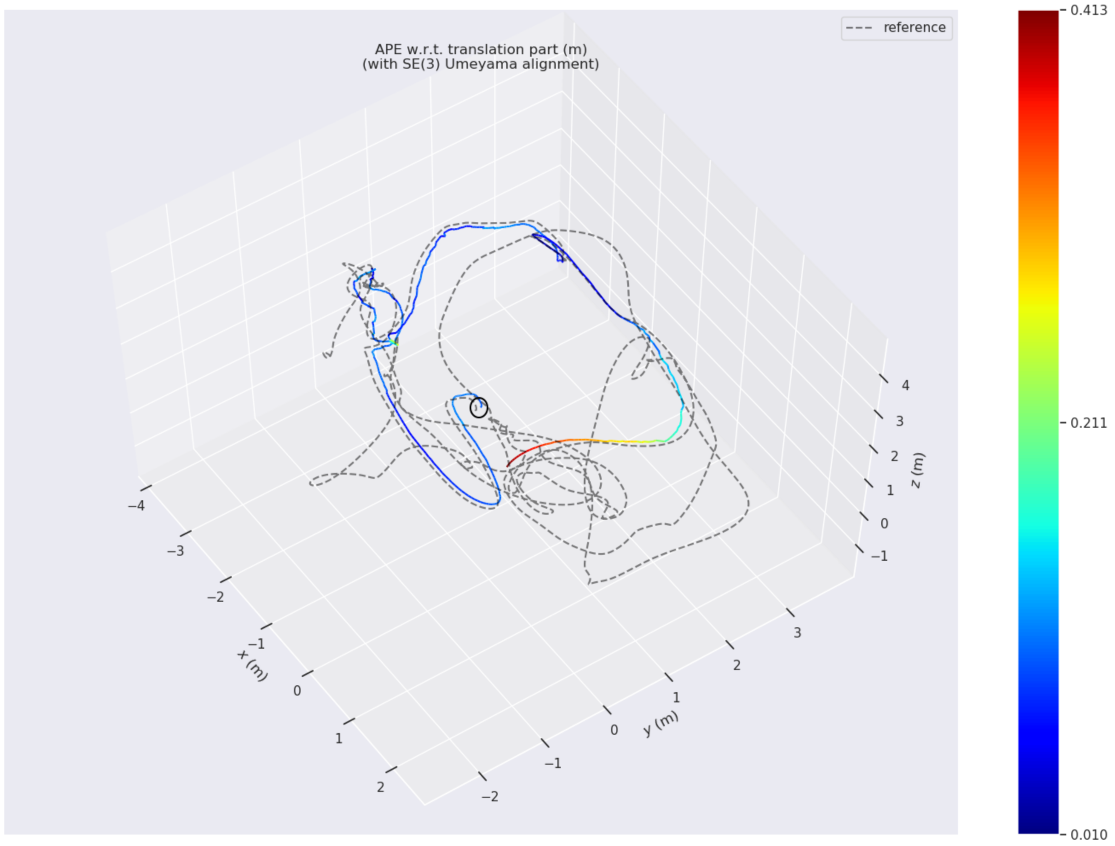


Figure 6.53: Intermediate V-GLMB-SLAM results with the euroc V202 “medium” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

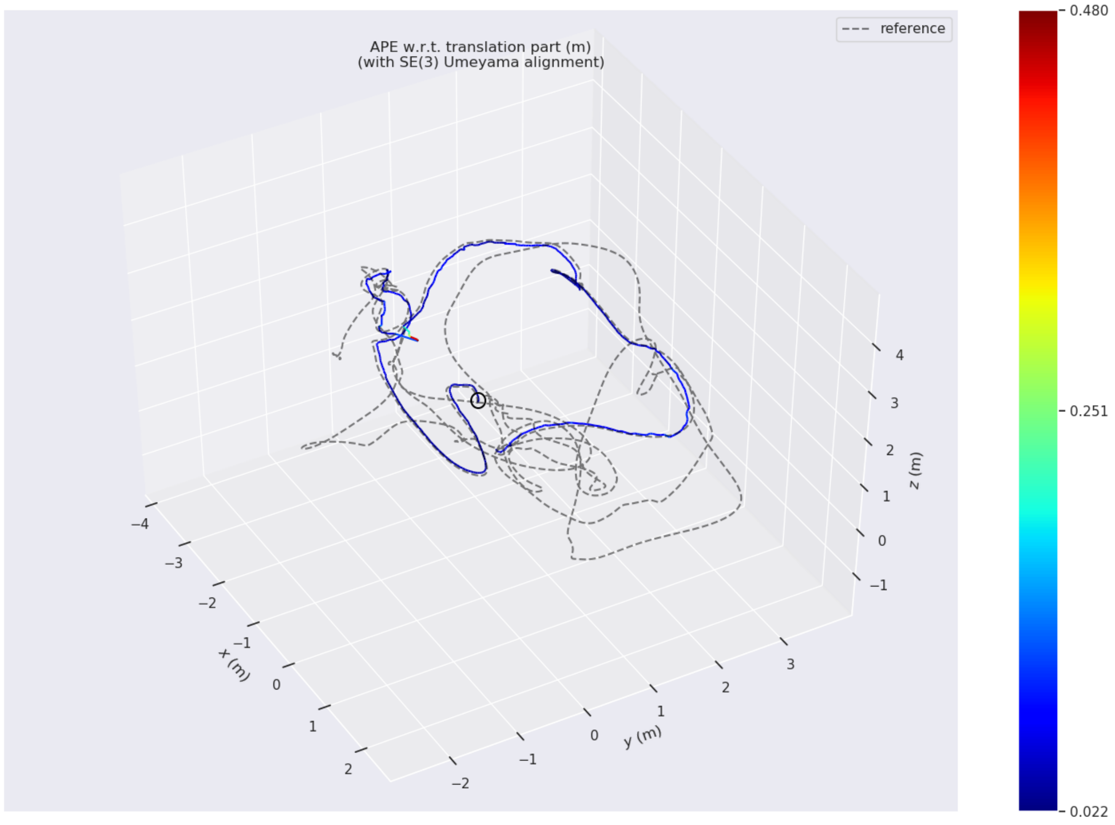


Figure 6.54: Intermediate V-GLMB-SLAM results with the euroc V202 “medium” dataset - 3D view. The ground truth trajectory is shown as a grey dashed line, and the estimated trajectory as a solid line, colored with the instantaneous position error in meters. The start of the trajectory is marked with a black circle.

6.9 Summary

In this chapter, a set-based SLAM posterior has been defined using a joint set-vector-based state. This formulation enables the proposed V-GLMB-SLAM algorithm to merge the SLAM back and front ends into a single Bayesian formulation while using sparse graph optimization methods, which have been shown to be well-suited to the SLAM problem. As with other RFS formulations of SLAM, this formulation uses a model of the probability of detection and descriptor likelihoods. Simulations have shown promising results, comparing the proposed algorithm with graph optimization methods using the theoretically correct data association, and Maximum Likelihood data association. Real stereo-vision SLAM results also show competitive results with state-of-the-art algorithms. This new formulation requires far fewer heuristics than competing state-of-the-art algorithms and solves the data association problem using purely Bayesian information. With the current implementation, this increased theoretical soundness comes at an increased computational cost.

Chapter 7

Conclusions and future work

In this thesis, we have explored the idea of using RFSs to include detection statistics into the SLAM formulation, as well as reduce the number of heuristics required for a complete SLAM solution. We have derived a model for the detection statistics of a 2D lidar tree/circle detector that includes occlusions along with a descriptor for that same detector. These models were used and shown to improve the performance of both MH-FastSLAM and RB-PHD-SLAM. These results have shown that incorporating detection statistics and descriptor information into SLAM is a worthwhile endeavor.

Second, a batch SLAM approach using an RFS formulation was attempted based on a general unconstrained optimization algorithm PSO. Results showed that this approach is severely limited by the high computational cost of running a general optimization method, especially when other more specialized methods are possible.

Third, a new distribution based on the RFS-based GLMB distribution was created, namely the V-GLMB distribution. This distribution was then used to take advantage of existing non-linear least squares optimization methods. The advantages of this algorithm were explored in simulation and a stereo-vision-based SLAM algorithm was developed using this algorithm and the ORB keypoint detector and descriptor pair, with results on par with the state-of-the-art ORBSLAM3 algorithm. Even though ground truth landmark positions cannot be obtained for ORB features, the probability of detection and descriptor likelihoods were successfully included in the Bayesian estimation by creating models for them inspired by the heuristics used in the ORBSLAM family of algorithms. With the current implementation, the additional theoretical soundness of this model comes at the cost of increased computational complexity.

7.1 Future avenues of research

Future research should focus on the association sampling step of the V-GLMB-SLAM algorithm. Since possible associations need to be chosen before performing the expensive calculation of the component weight w^{c_K}, which requires graph optimization, the order in which these association hypotheses are selected is critical for the resulting performance of the SLAM solution. Two possible approaches to improve the Gibbs sampler are proposed next:

7.1.1 Random sample consensus

Currently, the Gibbs sampler uses the previously estimated pose at time k, $\mathbf{x}_k$, to approximate the pose after the graph optimization step. This pose is used in calculating the partial association probabilities, from which association hypotheses are sampled, assuming the pose will not move during the optimization step.

A possible alternative to this approach would be to randomly sample the minimum number of associations to give an estimate of the pose, as is done in RANSAC. The pose $\mathbf{x}_k$ could be calculated directly given the rest of the trajectory, or the optimizer could be used to perform a less costly optimization limited to $\mathbf{x}_k$. This pose could then be used by the Gibbs sampler as it currently uses the previous estimate, which can then sample the new association hypothesis. This sampled hypothesis is then used by the full optimization method evaluated with the component weight. Arguably this method could improve the proposed associations by more quickly exploring data associations, which would have an unlikely probability of being correct based on the currently estimated data association but would have a high likelihood after performing the graph optimization.

7.1.2 Association matrix

A second particular possible improvement would be to maintain a matrix of the association likelihoods at each time k using the existing components c_K with their associations $(\theta_0, \cdots, \theta_K)$ and weights w^{c_K}. This matrix A_k would contain values proportional to the likelihood of associations being correct.

$$A_k(i,j) = \sum_{w^{c_K} | a_{\theta_k}(i)=j} w^{c_K} \ . \tag{7.1}$$

Once the estimated posterior has several components, sampling new possible associations from A_k may produce new likely associations. It should be noted however that in this s associated wit A_k does not model the correlations between the different associations.

Bibliography

[1] R. P. Mahler. *Statistical multisource-multitarget information fusion*, volume 685. Artech House Norwood, MA, USA, 2007.

[2] B.-T. Vo, B.-N. Vo, and A. Cantoni. The cardinality balanced multi-target multi-Bernoulli filter and its implementations. *Signal Processing, IEEE Transactions on*, 57(2):409–423, 2009.

[3] S. Reuter, B.-T. Vo, B.-N. Vo, and K. Dietmayer. The labeled multi-Bernoulli filter. *Signal Processing, IEEE Transactions on*, 62(12):3246–3260, June 2014.

[4] B.-N. Vo, B.-T. Vo, and D. Phung. Labeled random finite sets and the bayes multi-target tracking filter. *IEEE Transactions on Signal Processing*, 62(24):6554–6567, 2014. doi: 10.1109/TSP.2014.2364014.

[5] H. Deusch, S. Reuter, and K. Dietmayer. The labeled multi-Bernoulli SLAM filter. *Signal Processing Letters, IEEE*, 22(10):1561–1565, Oct 2015.

[6] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[7] H. Strasdat, J. Montiel, and A. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[8] S. Thrun and M. Montemerlo. The graphSLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6): 403–429, 2006.

[9] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[10] F. Inostroza. The estimation of detection statistics in simultaneous localization and mapping. Master's thesis, Universidad de Chile, 2015.

[11] S. Reuter, B.-T. Vo, B.-N. Vo, and K. Dietmayer. The labeled multi-bernoulli filter. *Signal Processing, IEEE Transactions on*, 62(12):3246–3260, 2014.

[12] B.-T. Vo and B.-N. Vo. Labeled random finite sets and multi-object conjugate priors. *Signal Processing, IEEE Transactions on*, 61(13):3460–3475, July 2013.

[13] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, Jun 2001. ISSN 1042-296X. doi: 10.1109/70.938381.

[14] K. Lee, W. Wijesoma, and J. Ibanez Guzman. On the observability and observability analysis of SLAM. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3569–3574. IEEE, 2006.

[15] Z. Wang and G. Dissanayake. Observability analysis of SLAM using Fisher information matrix. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 1242–1247. IEEE, 2008.

[16] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[17] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. National Conf. Artificial Intelligence*, pages 593–598, 2002.

[18] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[19] J. Nieto, J. Guivant, E. Nebot, and S. Thrun. Real time data association for FastSLAM. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 412–418. IEEE, 2003.

[20] S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6): 403–429, 2006.

[21] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. doi: 10.1109/ICRA.2011.5979949.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi: 10.1109/TRO.2015.2463671.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.

[24] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. doi: 10.1109/TRO.2017.2705103.

[25] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. doi: 10.1109/TRO.2021.3075644.

[26] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, 2014.

[27] T. Shan and B. Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018. doi: 10.1109/IROS.2018.8594299.

[28] J. Mullane, B.-N. Vo, and M. Adams. Rao-Blackwellised PHD SLAM. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5410–5416, 2010.

[29] K. Leung, F. Inostroza, and M. Adams. Generalizing random-vector SLAM with random finite sets. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4583–4588, 2015.

[30] D. Moratuwage, M. Adams, and F. Inostroza. δ-generalised labelled multi-bernoulli simultaneous localisation and mapping. In *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 175–182, 2018. doi: 10.1109/ICCAIS.2018.8570448.

[31] L. Gao, G. Battistelli, and L. Chisci. Phd-slam 2.0: Efficient slam in the presence of missdetections and clutter. *IEEE Transactions on Robotics*, 37(5):1834–1843, 2021. doi: 10.1109/TRO.2021.3052078.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012. doi: 10.1109/IROS.2012.6385773.

[33] D. Schuhmacher, B.-T. Vo, and B.-N. Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Trans. Signal Processing*, 56(8):3447–3457, 2008.

[34] B. K. Panigrahi, Y. Shi, and M.-H. Lim. *Handbook of swarm intelligence: concepts, principles and applications*, volume 8. Springer Science & Business Media, 2011.

[35] J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[36] F. Dellaert and Contributors. borglab/gtsam, May 2022. URL https://github.com/borglab/gtsam).

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

[38] K. Leung, F. Inostroza, and M. Adams. An improved weighting strategy for Rao-Blackwellized probability hypothesis density simultaneous localization and mapping. In *Proc. Int. Conf. Control, Automation, and Information Sciences*, 2013.

[39] J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo. A random-finite-set approach to Bayesian SLAM. *Robotics, IEEE Transactions on*, 27(2):268–282, 2011.

[40] J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo. Random finite sets for robot mapping and SLAM. *Springer Tracts in Advanced Robotics*, 72, 2011.

[41] M. Adams, B.-N. Vo, R. Mahler, and J. Mullane. New concepts in map estimation: SLAM gets a PHD. *IEEE Robotics and Automation Magazine*, 21(2):26–37, 2014.

[42] K. Granstrom and U. Orguner. A PHD filter for tracking multiple extended targets using random matrices. *IEEE Trans, Signal Processing*, 60(11):5657–5671, 2012.

[43] R. Mahler. A survey of PHD filter and CPHD filter implementations. In *Proc. SPIE Defense & Security Symposium of Signal Processing, Sensor Fusion and Target Recognition XII*, April 2007.

[44] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[45] M. Barkat. *Signal detection and estimation.* Artech House Boston, 2005.

[46] F. Inostroza, K. Leung, and M. Adams. Semantic feature detection statistics in set based simultaneous localization and mapping. In *IEEE 17th Int. Conf. Information Fusion*, 2014.

[47] F. Inostroza, K. Leung, and M. Adams. Incorporating estimated feature descriptor information into Rao-Blackwellized-PHD-SLAM. In *IEEE 18th Int. Conf. Information Fusion*, pages 1688–1695. IEEE, 2015.

[48] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395, June 1981.

[49] D. Clark, B. Ristic, B.-N. Vo, and B.-T. Vo. Bayesian multi-object filtering with amplitude feature likelihood for unknown object SNR. *IEEE Trans. Signal Processing*, 58(1):26–37, Jan 2010.

[50] M. Skolnik. Introduction to radar systems. *McGraw Hill*, 2001.

[51] K. Leung, F. Inostroza, and M. Adams. Evaluating set measurement likelihoods in random-finite-set SLAM. In *IEEE 17th Int. Conf. Information Fusion.* IEEE, 2014.

[52] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE Int. Conf. Intelligent Robots and Systems*, pages 2149–2154, 2004.

[53] J. Guivant and E. Nebot. Simultaneous localization and map building: Test case for outdoor applications. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002.

[54] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[55] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. 18th International Joint Conf. Artificial Intelligence*, IJCAI'03, pages 1151–1156, 2003.

[56] D. Schuhmacher, B.-T. Vo, and B.-N. Vo. A consistent metric for performance evaluation of multi-object filters. *Signal Processing, IEEE Transactions on*, 56(8):3447–3457, 2008.

[57] P. J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

[58] B.-N. Vo and B.-T. Vo. A multi-scan labeled random finite set model for multi-object state estimation. *IEEE Transactions on Signal Processing*, 67(19):4948–4963, Oct 2019. doi: 10.1109/TSP.2019.2928953.

[59] M. Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

[60] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (1):105–119, 2010. doi: 10.1109/TPAMI.2008.275.

[61] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15561-1.

[62] J. L. Hintze and R. D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998. doi: 10.1080/00031305.1998.10480559. URL https://www.tandfonline.com/doi/abs/10.1080/00031305.1998.10480559.

[63] F. Inostroza, M. Adams, and K. Leung. Modeling detection statistics in feature-based robotic navigation for range sensors. *NAVIGATION*, 65(3):297–318, 2018. doi: https://doi.org/10.1002/navi.261. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.261.

[64] K. Leung, D. Lühr, H. Houshiar, F. Inostroza, D. Borrmann, M. Adams, A. Nüchter, and J. Ruiz del Solar. Chilean underground mine dataset. *The International Journal of Robotics Research*, 36(1):16–23, 2017.

[65] K. Y. K. Leung, F. Inostroza, and M. Adams. Relating random vector and random finite set estimation in navigation, mapping, and tracking. *IEEE Transactions on Signal Processing*, 65(17):4609–4623, Sept 2017. ISSN 1053-587X. doi: 10.1109/TSP.2017.2701330.

[66] P. Barrios, M. Adams, K. Leung, F. Inostroza, G. Naqvi, and M. E. Orchard. Metrics for evaluating feature-based mapping performance. *IEEE Transactions on Robotics*, 33(1): 198–213, 2017. doi: 10.1109/TRO.2016.2627027.

[67] D. Moratuwage, M. Adams, and F. Inostroza. δ-generalized labeled multi-bernoulli simultaneous localization and mapping with an optimal kernel-based particle filtering approach. *Sensors*, 19(10), 2019. ISSN 1424-8220. doi: 10.3390/s19102290. URL https://www.mdpi.com/1424-8220/19/10/2290.

[68] K. Y. K. Leung, F. Inostroza, and M. Adams. Multifeature-based importance weighting for the phd slam filter. *IEEE Transactions on Aerospace and Electronic Systems*, 52(6): 2697–2714, 2016. doi: 10.1109/TAES.2016.150566.

[69] F. Inostroza and M. Adams. Addressing data association in maximum likelihood slam with random finite sets. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2047–2053, 2018. doi: 10.23919/ICIF.2018.8455273.

[70] P. Barrios, G. Naqvi, M. Adams, K. Leung, and F. Inostroza. The cardinalized optimal linear assignment (cola) metric for multi-object error evaluation. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 271–279, 2015.

Annexes

Annex A

List of Publications

A.1 Journal papers

- F. Inostroza, M. Adams, and K. Leung. Modeling detection statistics in feature-based robotic navigation for range sensors. *NAVIGATION*, 65(3):297–318, 2018. doi: https://doi.org/10.1002/navi.261. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.261

- K. Leung, D. Lühr, H. Houshiar, F. Inostroza, D. Borrmann, M. Adams, A. Nüchter, and J. Ruiz del Solar. Chilean underground mine dataset. *The International Journal of Robotics Research*, 36(1):16–23, 2017

- K. Y. K. Leung, F. Inostroza, and M. Adams. Relating random vector and random finite set estimation in navigation, mapping, and tracking. *IEEE Transactions on Signal Processing*, 65(17):4609–4623, Sept 2017. ISSN 1053-587X. doi: 10.1109/TSP.2017.2701330

- P. Barrios, M. Adams, K. Leung, F. Inostroza, G. Naqvi, and M. E. Orchard. Metrics for evaluating feature-based mapping performance. *IEEE Transactions on Robotics*, 33(1): 198–213, 2017. doi: 10.1109/TRO.2016.2627027

- D. Moratuwage, M. Adams, and F. Inostroza. δ-generalized labeled multi-bernoulli simultaneous localization and mapping with an optimal kernel-based particle filtering approach. *Sensors*, 19(10), 2019. ISSN 1424-8220. doi: 10.3390/s19102290. URL https://www.mdpi.com/1424-8220/19/10/2290

- K. Y. K. Leung, F. Inostroza, and M. Adams. Multifeature-based importance weighting for the phd slam filter. *IEEE Transactions on Aerospace and Electronic Systems*, 52(6): 2697–2714, 2016. doi: 10.1109/TAES.2016.150566

A.2 Conference papers

- K. Leung, F. Inostroza, and M. Adams. An improved weighting strategy for Rao-Blackwellized probability hypothesis density simultaneous localization and mapping. In *Proc. Int. Conf. Control, Automation, and Information Sciences*, 2013

- F. Inostroza, K. Leung, and M. Adams. Semantic feature detection statistics in set based simultaneous localization and mapping. In *IEEE 17th Int. Conf. Information Fusion*, 2014

- K. Leung, F. Inostroza, and M. Adams. Evaluating set measurement likelihoods in random-finite-set SLAM. In *IEEE 17th Int. Conf. Information Fusion*. IEEE, 2014

- F. Inostroza, K. Leung, and M. Adams. Incorporating estimated feature descriptor information into Rao-Blackwellized-PHD-SLAM. In *IEEE 18th Int. Conf. Information Fusion*, pages 1688–1695. IEEE, 2015

- K. Leung, F. Inostroza, and M. Adams. Generalizing random-vector SLAM with random finite sets. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 4583–4588, 2015

- D. Moratuwage, M. Adams, and F. Inostroza. δ-generalised labelled multi-bernoulli simultaneous localisation and mapping. In *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 175–182, 2018. doi: 10.1109/ICCAIS.2018.8570448

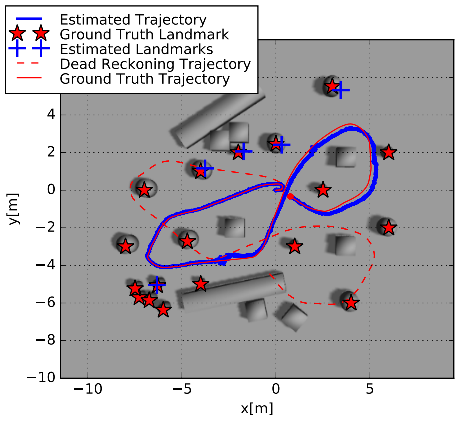
- F. Inostroza and M. Adams. Addressing data association in maximum likelihood slam with random finite sets. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2047–2053, 2018. doi: 10.23919/ICIF.2018.8455273

- P. Barrios, G. Naqvi, M. Adams, K. Leung, and F. Inostroza. The cardinalized optimal linear assignment (cola) metric for multi-object error evaluation. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 271–279, 2015

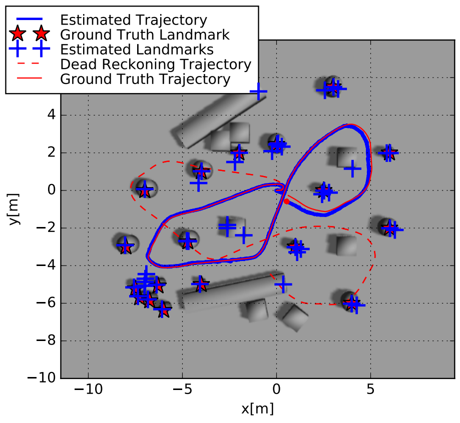
Annex B

Simulation results with other training dataset

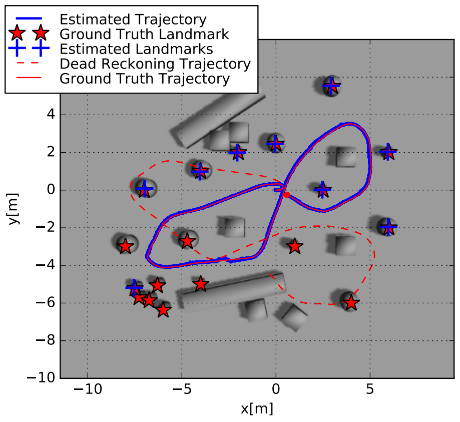
Figure B.1 shows the same algorithms used to generate Fig. 4.10 based on the training environment shown in Figure 4.4. A single SLAM result is shown for each algorithm out of five Monte Carlo runs.



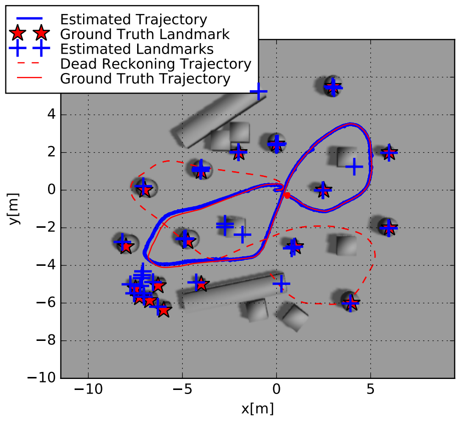
(a) *RB-PHD-SLAM with a constant probability of detection.*



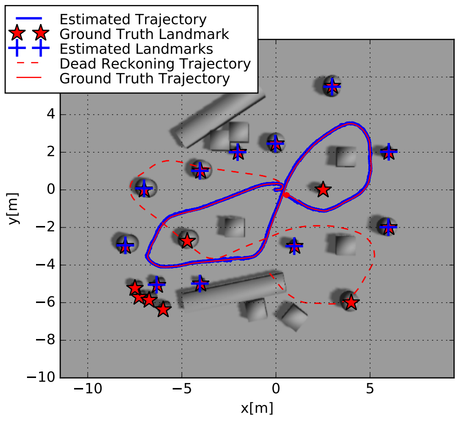
(b) *MH-FastSLAM with a constant probability of detection.*



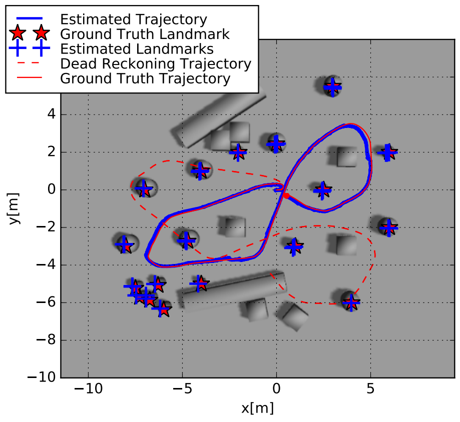
(c) *RB-PHD-SLAM with the learned probability of detection.*



(d) *MH-FastSLAM with the learned probability of detection.*



(e) *RB-PHD-SLAM with the learned probability of detection and descriptor information.*



(f) *MH-FastSLAM with the learned probability of detection and descriptor information.*

Figure B.1: *SLAM results in a simulated environment.*

Annex C

Proof that joint Vector-Set distribution integrates to 1

Repeating the definition of a joint Vector-Set distribution, from equation (6.3).

$$\pi(\chi_K) = \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \tag{C.1}$$

We write the joint integral over the vector trajectory $\mathbf{x}_{0:K}$ and set $\mathcal{M}$:

$$\int_{\chi_K} \pi(\chi_K) = \int_{\chi_K} \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \tag{C.2}$$

$$= \int_{\mathbf{x}_{0:K}} \int_{\mathcal{M}} \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \tag{C.3}$$

$$= \int_{\mathcal{M}} \Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M})) \int_{\mathbf{x}_{0:K}} \pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) \tag{C.4}$$

Using this form we note that $\int_{\mathbf{x}_{0:K}} \pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M}))$ will be the marginal spatial distribution of $\mathcal{M}$ in component c, since the trajectory is integrated out. We no relabel it as

$$\int_{\mathbf{x}_{0:K}} \pi^c(\mathbf{x}_{0:K}, \vec{v}(\mathcal{M})) = \pi^c_{\mathcal{M}}(\vec{v}(\mathcal{M})) \ . \tag{C.5}$$

The marginal distribution of $\mathcal{M}$ is then

$$\Delta(\mathcal{M}) \sum_{c \in \mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c_{\mathcal{M}}(\vec{v}(\mathcal{M})) \ , \tag{C.6}$$

which is very close to a GLMB distribution, with the significant caveat that in a GLMB component each element in the set is independently distributed, while the marginal distribution derived from the proposed joint distribution will inevitably include high correlations between the elements of the map.

Given this, the rest of the proof will be analogous to the proof that a GLMB distribution integrates to unity. Continuing with the proof

$$\int_{\chi_K} \pi(\chi_K) \tag{C.7}$$

$$= \int_{\mathcal{M}} \Delta(\mathcal{M}) \sum_{c\in\mathcal{C}} w^c(\mathcal{L}(\mathcal{M}))\pi^c_{\mathcal{M}}(\vec{v}(\mathcal{M})) \tag{C.8}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \int_{(L,M)\in(\mathbb{L},\mathbb{M})^n} \Delta(L) \sum_{c\in\mathcal{C}} w^c(L)\pi^c_{\mathcal{M}}(\vec{v}(L,M)) \tag{C.9}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{l_1\in\mathbb{L}} \cdots \sum_{l_n\in\mathbb{L}} \int_{m_1} \cdots \int_{m_n} \Delta((l_1 \cdots l_n)) \sum_{c\in\mathcal{C}} w^c((l_1 \cdots l_n))\pi^c_{\mathcal{M}}(\vec{v}((l_1,m_1)\cdots(l_n,m_n))) \tag{C.10}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{l_1\in\mathbb{L}} \cdots \sum_{l_n\in\mathbb{L}} \Delta((l_1 \cdots l_n)) \sum_{c\in\mathcal{C}} w^c((l_1 \cdots l_n)) \cancelto{1}{\int_{m_1} \cdots \int_{m_n} \pi^c_{\mathcal{M}}(\vec{v}((l_1,m_1)\cdots(l_n,m_n)))} \tag{C.11}$$

Noting that the remaining integrals integrate to one

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{l_1\in\mathbb{L}} \cdots \sum_{l_n\in\mathbb{L}} \Delta((l_1 \cdots l_n)) \sum_{c\in\mathcal{C}} w^c((l_1 \cdots l_n)) \ . \tag{C.12}$$

From here we note that every set of size n , $\{l_1 \cdots l_n\}$ will appear exactly $n!$ times and since the component weight $w^c(\cdot)$ does not depend on the order of the labels the integral becomes

$$= \sum_{n=0}^{\infty} \frac{1}{\cancel{n!}} \cancel{n!} \sum_{L\in F_n(\mathbb{L})} \sum_{c\in\mathcal{C}} w^c(L) \tag{C.13}$$

where $F_n(\mathbb{L})$ is the set of all sets of size n in label space $\mathbb{L}$.

$$= \sum_{n=0}^{\infty} \sum_{L \in F_n(\mathbb{L})} \sum_{c \in \mathcal{C}} w^c(L) \tag{C.14}$$

$$= \sum_{L \subseteq \mathbb{L}} \sum_{c \in \mathcal{C}} w^c(L) \equiv 1 \tag{C.15}$$

This is, by definition, unity.