SIMULTANEOUS LOCALIZATION AND MAPPING USING SINGLE CLUSTER PROBABILITY HYPOTHESIS DENSITY FILTERS

Chee Sing LEE

Dipòsit legal: Gi. 1991-2015
http://hdl.handle.net/10803/323637

DOCTORAL THESIS

Simultaneous Localization and Mapping Using Single Cluster Probability Hypothesis Density Filters

Chee Sing Lee

2014

DOCTORAL THESIS

Simultaneous Localization and Mapping Using Single Cluster Probability Hypothesis Density Filters

Chee Sing Lee

2014

Doctoral Program in Technology

Supervised By:

Prof. Joaquim Salvi (Universitat de Girona)

Prof. Daniel Clark (Heriot-Watt University)

Work submitted to the University of Girona in fulfillment of the requirements for the degree of Doctor of Philosophy

Acknowledgements

Throughout this doctoral thesis, I have been fortunate to have the support of my family, friends, and colleagues. In particluar I would like to thank my supervisors Daniel and Quim, who took unprecedented interest in my development as a professional and a person. My lovely wife Chrissy has sat by my side this entire time keeping me on track. I am grateful to all of the wonderful people in the labs (you know who you are) at both Girona and Edinburgh for making me feel welcome as a stranger in strange lands. Among the countless people I need to thank, here are but a few who stand out: Josep – you got me out of the lab and showed me the beauty of the Catalan countryside, Sharad – for late night editing sessions and somehow still having the energy to laugh over a cup of coffee, Kos – thank you for walking the PhD road together and being a trusted friend and confidante, Sebas – for an in-depth tour of Catalan culture and cattle, and Daniel – thanks again for helping me believe in myself when I thought I couldn't keep going.

List of Publications

The work in this thesis encompasses a number of collaborative efforts, which are listed here:

Journal Articles

Lee, C.S., Nagappa, S., Palomeras, N., Clark, D., Salvi, J. *in press. SLAM with SC-PHD filters: an underwater vehicle application.* IEEE Robotics and Automation Magazine: Special Issue on Emerging Applications of Stochastic Geometry in Autonomous Robotics.

Lee, C.S., Clark, D., Salvi, J. 2013. *SLAM with Dynamic Targets via Single-Cluster PHD Filtering.* IEEE Journal of Selected Topics in Signal Processing. vol.7, no.3, pp.543-552.

Conference Participations

First International Summer School on Finite Set Statistics, Edinburgh, UK, August 2013.

Nagappa, S., Palomeras, N., **Lee, C.S.**, Gracias, N., Clark, D., Salvi, J. 2013. *Single Cluster PHD SLAM: Application to Autonomous Underwater Vehicles using Stereo Vision.* OCEANS'13 MTS/IEEE Conference, Bergen, Norway, June 2013.

Lee, C.S., Clark, D., Salvi, J. 2012. *SLAM with Single Cluster PHD Filters.* 2012 IEEE International Conference on Robotics and Automation (ICRA), St. Paul, USA, May, 2012.

Programming and Tuning Massively Parallel Systems(PUMPS) Summer School, Barcelona, Spain, July 2011 .

Publications in Preparation

Franco Monsalve, J., **Lee, C.S.**., Houssineau, J., Clark, D. *in preparation. Accelerating the Single Cluster Probability Hypothesis Density Filter with a GPU Implementation*

List of Acronyms and Initialisms

In alphabetical order:

AHRS attitude and heading reference system

AUV autonomous underwater vehicle

DVL Doppler velocity log

ESDF Exactly Sparse Delayed-State Filter

EKF Extended Kalman Filter

FoV field of view

GM-PHD Gaussian Mixture PHD

GPS global positioning system

i.i.d. independent and identically distributed

IMU inertial measurement unit

JCBB joint compatibility branch and bound

OSPA optimal sub-pattern assignment

PDF probability density function

p.g.fl. probability generating functional

PHD probability hypothesis density

RB-PHD Rao-Blackwellized PHD

RFS random finite set

SC-PHD single cluster PHD

SLAM simultaneous localization and mapping

SURF speeded up robust features

UKF Unscented Kalman Filter

usbl]USBL]Ultra-short baseline

List of Figures

List of Tables

Contents

Abstract

This thesis focuses on the problem of SLAM, in particular feature-based SLAM, where a mobile autonomous vehicle explores an unknown environment, navigating by observing landmarks. As the environment is previously unexplored, SLAM algorithms must estimate the location of the landmarks in addition to the location of the vehicle. The majority of research in feature-based SLAM builds on the legacy of foundational work using the Extended Kalman Filter (EKF), a single-object estimation technique. Because feature-based SLAM is an inherently multi-object problem, this has led to a number of suboptimalities in popular solutions. We hypothesize that a feature based SLAM algorithm derived from multi-object estimation techniques can achieve superior estimation performance compared to conventional algorithms, especially in conditions where the aforementioned weaknesses are most prominent. In this work, we develop such an algorithm, using the SC-PHD filter, a multi-object estimator modeled on cluster processes. This algorithm hosts capabilities not typically seen with feature-base SLAM solutions such as principled handling of false alarm measurements and missed detections, and navigation with a mixture of stationary and moving landmarks. We present experiments with the SC-PHD SLAM algorithm on both synthetic and real datasets using an autonomous underwater vehicle. We also compare our method to the RB-PHD SLAM, showing that it requires fewer approximations in its derivation and thus achieves superior performance.

Keywords: simultaneous localization and mapping, mobile autonmous vehicles, underwater autonmous vehicles, probability hypothesis density filters, multi-object estimation, finite set statistics.

Resumen

Esta tesis se centra en el problema de la localización y el mapeo simultáneo (SLAM), en particular en el SLAM basado en características, donde un vehículo móvil autónomo explora un entorno desconocido navegando mientras observa dichas características. Dado que inicialmente se desconoce el entorno, los algoritmos SLAM deben estimar la localización de las características del entorno además de la localización del propio vehículo. La mayor parte de la investigación en SLAM basado en características se nutre del legado fundamental en filtros de Kalman extendido (EKF), técnica de estimación uni-objeto. Dado que el SLAM basado en características es intrínsecamente un problema multi- objeto, ello ha popularizado un número de soluciones subóptimas. En esta tesis se hace la hipótesis de que un SLAM basado en características derivado de una técnica de estimación multi-objeto puede conseguir un mayor grado de rendimiento en la estimación comparado con los algoritmos convencionales, especialmente en situaciones donde las debilidades antes mencionadas son más presentes. En esta tesis se desarrolla este algoritmo a partir de un filtro PHD con un único grupo (SC-PHD), una técnica de estimación multi-objeto basado en procesos de agrupación. Este algoritmo tiene unas capacidades que normalmente no se ven en algoritmos de SLAM basados en características, ya que es capaz de tratar falsas características así como características no detectadas por los sensores del vehículo, además de navegar en un entorno con la presencia de características estáticas y móviles de forma simultánea. Se presentan resultados experimentales del algoritmo SC-PHD en entornos reales y simulados utilizando un vehículo autónomo submarino. Los resultados son comparados con el algoritmo de SLAM Rao-Blackwellized PHD (RB-PHD), demostrando que se requiere menos aproximaciones en su derivación y por consiguiente se obtiene un rendimiento superior.

Resum

Aquesta tesis es centra en el problema de la localització y la construcció de mapes de forma simultània (SLAM), en particular en el SLAM basat en característiques, on un vehicle mòbil autònom explora un entorn desconegut navegant mentre observa aquestes característiques. Donat que inicialment l'entorn es desconegut, els algoritmes SLAM han d'estimar la localització d'aquestes característiques de l'entorn al mateix temps que estima la localització del propi vehicle. La major part de la recerca en SLAM basat en característiques s'alimenta del llegat fonamental en filtres de Kalman estès (EKF), tècnica d'estimació uni-objecte. Al ser el SLAM basat en característiques intrínsecament un problema multi-objecte s'han popularitzat un nombre de solucions sub-òptimas. En aquesta tesis en fa la hipòtesi de que un SLAM basat en característiques derivat d'una tècnica d'estimació multi-objecte pot aconseguir un major grau de rendiment en l'estimació comparat amb els algoritmes convencionals, especialment en aquelles situacions on les debilitats anomenades anteriorment són més presents. En aquesta tesis es desenvolupa aquest algoritme a partir d'un filtre PHD amb un únic grup (SC-PHD), una tècnica d'estimació multi-objecte basat en processos d'agrupació. Aquest algoritme té unes capacitats que normalment no es veuen en els algoritmes de SLAM basats en característiques, ja que és capaç de tractar falses característiques, així com característiques no detectades pels sensors del vehicle, a més de navegar en un entorn amb la presència de característiques estàtiques i característiques en moviment de forma simultània. Es presenten els resultats experimentals de l'algoritme SC-PHD en entorns reals i simulats utilitzant un vehicle autònom submarí. Els resultats són comparats amb l'algoritme de SLAM Rao- Blackwellized PHD (RB-PHD), demostrant que es requereixen menys aproximacions en la seva derivació i en conseqüència s'obté un rendiment superior.

Chapter 1

Introduction

This chapter introduces the problem of simultaneous localization and mapping (SLAM), in particular feature-based SLAM. We discuss the current shortcomings of feature-based SLAM which provide motivation for this work.

1.1 Simultaneous Localization and Mapping

SLAM is a key component in mobile autonomous systems. It describes the ability of a vehicle, once placed in an unknown environment, to explore and map that environment, while at the same time estimating its own position in the environment, using only its onboard sensing capabilities. Several difficulties are encountered while performing this task, including but not limited to:

- Inaccurate sensor models – Sensors can rarely be fully characterized forl all operational conditions. Despite careful calibration efforts, some systemic bias is likely to present in received measurements.
- Unreliable sensing of the environment – This includes not only a lack of precision in sensor returns, but also false-positive sensor returns and failures in detecting environmental features.
- Inaccurate motion models – SLAM algorithms will often incorporate a vehicle motion model, but these models rarely describe the motion of

the vehicle with complete fidelity.

- Unreliable sensing of vehicle motion and position – In some applications, feedback sensors are employed to provide additional information on vehicle motion. Like the environmental sensors, these readings are both imprecise and inaccurate. For example, encoder-based odometry can be confounded by wheel slippage. Relying solely on odometry to estimate position (dead reckoning) is an untenable strategy for extended operations as odometry error accumulates over time. GPS measurements can mitigate this problem, but in many environments where autonomous mobile robots are used (e.g. underwater, subterranean, extraterrestrial), GPS is intermittent or completely unavailable.

- Unanticipated changes in the environment or vehicle state – Portions of the environment may change with time. Abrupt terrain changes may impart motion on the vehicle which is drastically different from model parameters.

The result of these adverse conditions is that nothing regarding the vehicle's environment or position can be known with complete certainty. Solving the SLAM problem, therefore, requires the use of probabilistic methods which give the best estimate of the vehicle and environment, given the sensor data received.

1.2 Motivation and Objectives

The objective of this work this to apply the principles and methods of multi-object estimation to the problem of feature-based SLAM.

1.2.1 SLAM for Underwater Vehicles

The subsea industry is increasingly interested in the use of autonomous underwater vehicles (AUVs) to perform inspection, maintenance and light intervention tasks at submarine facilities. Of particular interest is the ability to have vehicles operating unattended for extended periods of time which is key

to reducing operating costs. A critical issue in this is the vehicle's awareness of its environment as well as its place within that environment, i.e., SLAM. In order to perform pathfinding operations, surveys or intervention, the vehicle must have a map of its surroundings as well as a firm knowledge of its own position relative to the local area. It is for these reasons that SLAM is widely described as a fundamental problem in mobile robotics.

Localization of AUVs is a particularly challenging area of research. The use of global positioning systems (GPSs) is limited to the surface of the water due to high levels of attenuation. An attitude and heading reference system (AHRS) is used to estimate orientation of the vehicle and position is estimated using a dead reckoning algorithm with measurements from a pressure sensor and Doppler velocity log (DVL). The DVL measures velocity with respect to the water or the sea bottom. It is possible to perform corrections due to drift in the sensors or noisy measurements by incorporating this estimation into a SLAM framework.

1.2.2 Feature-Based SLAM

One possible taxonomy for SLAM solutions is based on their choice of environment modeling. In this work, we will approach the SLAM problem using a feature-based algorithm. Feature-based SLAM algorithms consider the environment map to be composed of a number of distinct landmarks, or features. At a minimum, these features are stored as coordinates in 2D or 3D space, but higher dimension representations are possible, augmenting the spatial coordinates with descriptive parameters (e.g. appearance, size, shape). As a vehicle moves, repeated observation of landmarks will enable an estimate of the vehicle's motion relative to those landmarks and within the environment as a whole.

Since a feature-based map is essentially only a collection of points in Euclidean space, this type of representation is more tractable and memory-efficient compared to other SLAM paradigms which store the environment as a dense point cloud, or as a collection of images. However, alternate map representations are not without merit. For example, dense point clouds fa-

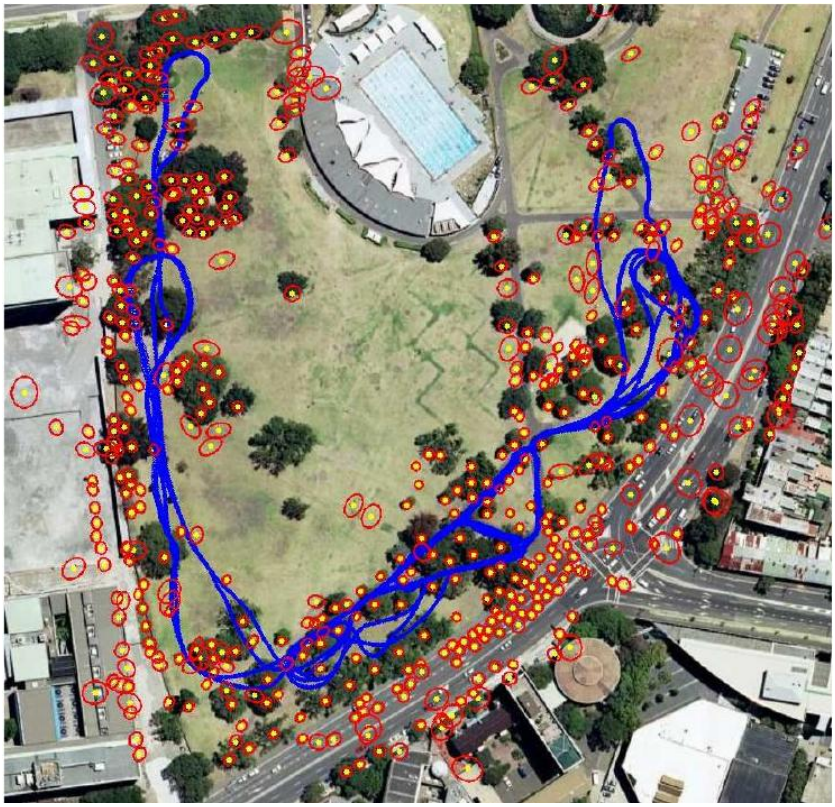


Figure 1.1: Example of feature-based SLAM. Red ellipses represent landmarks which have been observed by the vehicle, as it moves along the blue trajectory. Image from [4]

.

cilitate obstacle detection, and images are well suited for object recognition and semantics. While these isues fall outside the scope of this work, they are nevertheless important topics in mobile robotics.

1.2.3 Multi-object Estimation

In all but the most trivial feature-based SLAM scenarios, the map is composed of multiple features. It is therefore important that feature-based SLAM solutions adopt a probabilistic framework which encompasses this fact. The map should be represented as a probabilistic entity which admits not only uncertainty in the description of individual features, but also uncertainty in the number of features. However, much of the early and continuing work

in feature-based SLAM has not taken this into account. Rather, they have adapted well-established single-object estimation techniques such as the Extended Kalman Filter (EKF) to suit their needs. A multi-object state is coerced into a single object by concatenating the objects together into a single vector, resulting in the following shortcomings:

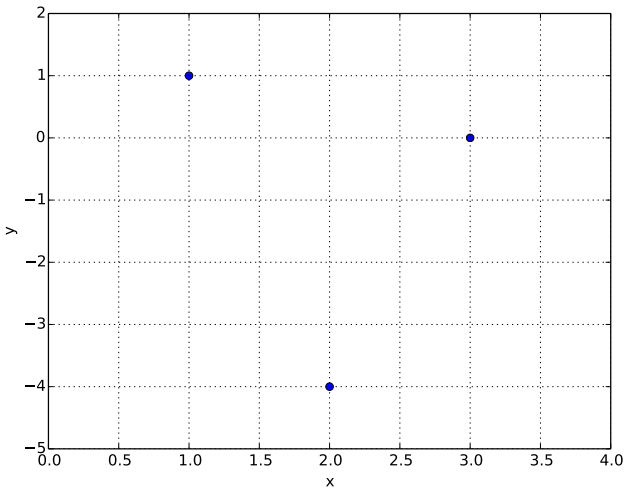
- Using a single vector to represent a collection of multiple objects introduces undue importance on ordering of the objects. As shown in Figure 1.2, the same collection of features can be represented by several mathematically unequal vectors.

- Initializing new features and deleting spurious ones involves adding and removing entries to the state vector. Thus, the domain of the state vector is constantly fluctuating throughout the estimation process, with no probabilistic framework to model this fluctuation.

- To correctly update the state vector with feature measurements, individual features must be properly matched to individual measurements. This is commonly referred to as the *data association* problem. Data association algorithms will tend to fail if the measurements contain a high number of false alarms, and/or the detection rate for features is low.

- We wish to consider map which contains a combination of indistinguishable static and moving features. At each iteration of the estimation process, we must predict each individual feature as either moving or remaining stationary. In a vector-based estimation framework, it is cumbersome to develop a state transition function which would accomplish this, while remaining agnostic to the order of features in the state vector.

The aim of this work is to construct a SLAM solution based on the principles of multi-object estimation and thus avoid the problems listed above.



$$\begin{bmatrix} 1 \\ 1 \\ 2 \\ -4 \\ 3 \\ 5 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 1 \\ 3 \\ 5 \\ 2 \\ -4 \end{bmatrix} \neq \begin{bmatrix} 2 \\ -4 \\ 3 \\ 5 \\ 1 \\ 1 \end{bmatrix} \neq \cdots$$

Figure 1.2: A feature map consisting of three 2D features located at coordinates $(1, 1)$, $(2, -4)$, and $(3, 5)$. A vector representation of this map would be a concatenation of the features, e.g. $[1, 1, 2, -4, 3, 5]^T$. There are $3! = 6$ different permutations for constructing such a vector. While they all represent the same three features, none of them are mathematically equivalent.

1.3 Context of Work

The majority of the time spent on this work was split between the Computer Vision and Robotics (VICOROB) group at the University of Girona, and the Institute of Sensors, Signals, and Systems (ISSS) at Heriot-Watt University.

The VICOROB group hosts the Underwater Robotics Research Center, which has made several important contributions to the field of autonomous underwater vehicles (AUVs). This includes the development of several AUVs, the most recent being the Girona 500 and Sparus II; winning several European-level AUV competitions; and performing numerous real-world underwater survey and inspection missions. In addition to autonomous underwater systems, the VICOROB group has also made contributions to robotic navigation using catadioptric camera systems.

Daniel Clark's group at the Heriot-Watt University specializes in statistical signal processing and multi-object estimation. Professor Clark is regarded to be a leading figure in the field of finite set statistics. Major contributions include variance-based metrics for performance assessment in multi-object filtering, PHD filter solutions for multi-sensor and distributed sensor systems, and a generalized chain rule for functional derivatives. The group maintains collaborative relationships with the University Defense Research Council (UDRC) and the Defense Science and Technlogy Organisation (DSTO) in Australia.

This work has received funding from the following Spanish and European project grants:

- FP7-ICT-2011-7 project PANDORA—Persistent Autonomy through Learning, Adaptation, Observation and Re-planning (Ref 288273) funded by the European Commission

- RAIMON – Autonomous Underwater Robot for Marine Fish Farms Inspection and Monitoring (Ref CTM2011-29691-C02-02) funded by the Ministry of Economy and Competitiveness of the Spanish Government.

1.4 Organization

This thesis is organized into several chapters:

1. Chapter 2 outlines the history of developments in the field of feature-based SLAM. The past research work is classified according to the various approaches to solving the SLAM problem, and key publications in each category are cited. The chapter also explores prior work in Random Finite Set methods, and highlights the developments that led to their application in SLAM.

2. Chapter 3 discusses finite set statistics and multi-object calculus, introducing concepts such as set integrals, functional derivatives, probability generating functionals (p.g.fl.s), and point processes. This provides a mathematical foundation necessary for the discussion of algorithms in following chapters.

3. Chapter 4 explains the methodology used to derive first-order moment filters for multi-object estimation. The probability hypothesis density (PHD) filter will be derived first, and then generalized to the single cluster PHD (SC-PHD) filter which is used for the solving the SLAM problem.

4. Chapter 5 illustrates the application of the SC-PHD filter to the SLAM problem. A concrete numerical implementation of the filter is presented which uses a hybrid particle/Gaussian mixture approach. Simulation results on synthetic datasets are shown, and comparisons are drawn between our work and Rao-Blackwellized PHD (RB-PHD) SLAM.

5. Chapter 6 Explains the strategies used to create a parallel implementation of the SC-PHD filter on a GPU with the CUDA software framework.

6. Chapter 7 presents an application of SC-PHD SLAM on a real world dataset collected by the Girona500 underwater vehicle at the Underwater Robotics Research Center in Girona, Spain. We discuss the chal-

lenges of implementing the SLAM algorithm on a real vehicle, and show that our method compares favorably with other SLAM solutions.

7. Chapter 8 summarizes the work done in this thesis and comments on possible avenues for future research.

In addition, a pseudo-code listing is included as an appendix to aid readers who wish to implement our method and reproduce these results.

Chapter 2

State of the Art

In this chapter, a history of developments in the field of SLAM is outlined. The past research work is classified according to the various approaches to solving the SLAM problem, and key publications in each category are cited. We also explore prior work in Random Finite Set methods, and highlight the developments that led to their application in SLAM.

2.1 SLAM algorithms

Distilled to its core, SLAM is a state estimation problem: given a sequence of noisy sensor readings and odometry inputs, what is the most likely configuration of the robot's trajectory and environment? Early breakthroughs in SLAM were provided by utilising the Kalman filter and EKF to address this as a joint state estimation problem, coupling the positions of the vehicle and landmarks [85, 52].

2.1.1 Gaussian Filter Methods

The Kalman filter is a Bayesian filter framework in which a Gaussian random variable with mean $\mathbf{x}$ and covariance matrix $\mathbf{P}$ is estimated through periodic measurements. The filter is typically comprised of two main steps: prediction and update. These are repeated over a series of time steps, delineated by the arrival measurements.

The prediction step propagates the state estimate through time. The Kalman filter uses a linear dynamic model, characterized by a linear operator $\mathbf{F}$, and is subject to zero-mean Gaussian-distributed noise, with covariance matrix $\mathbf{Q}$. The prediced mean and covariance is therefore:

$$\mathbf{x}_{k|k-1} = \mathbf{F}\mathbf{x}_{k-1} \tag{2.1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k|k-1}\mathbf{F}^T + \mathbf{Q} \tag{2.2}$$

The update step uses the measurement $\mathbf{z}$ to update the state estimate. The Kalman filter uses a linear measurement model, characterized by a linear operator $\mathbf{H}$, and is subject to zero-mean Gaussian-distributed noise, with covariance matrix $\mathbf{R}$. Using a quantity called the *optimal Kalman gain*, $\mathbf{K}$, the Kalman filter update yields the minimum mean squared error estimate, given the measurement:

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_{k|k-1}) \tag{2.3}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{k|k-1} \tag{2.4}$$

$$\mathbf{K} = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}^{-1} \tag{2.5}$$

$$\mathbf{S} = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \tag{2.6}$$

This method was validated by Dissanayake et. al. [26], who provided a proof that as the number of observations approach infinity,

1. The covariance of any feature in the map converges toward the initial vehicle covariance.

2. The feature locations become fully correlated with each other.

This implies that for linear dynamic and measurement system models, it is possible to construct perfect relative map using the Kalman filter approach, and therefore solve the SLAM problem.
The Extended Kalman Filter (EKF) is a generalization of the Kalman filter to non-linear models. Let $\mathbf{x}_k = f(\mathbf{x}_{k-1})$ and $\mathbf{z}_k = h(\mathbf{x}_k)$ be non-linear functions describing the dynamic and measurement models respectively. The EKF

prediction and update equations are identical to the Kalman filter except that the predicted updated means are:

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}) \tag{2.7}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}(\mathbf{z}_k - h(\mathbf{x}_{k|k-1})) \tag{2.8}$$

In addition the matrices $\mathbf{F}$ and $\mathbf{H}$ are the linearizations of the functions f and h with respect to the state vector. As such, the EKF is an approximate estimator, and the quality of its estimate will depend on how closely the linearized models fit the originals. The performance of the EKF suffers when the original models are highly non-linear.

The Unscented Kalman Filter (UKF) is another adaptation of the Kalman filter to non-linear models, offering reduced linearization error compared to the EKF at the cost of increased computational expense. The *unscented transform* generates a number of deterministic *sigma points* from a Gaussian distribution, which capture the higher order moment properties of the underlying distribution [46]. Rather than using linear approximations of the dynamic and measurement models, the sigma points are propagated through the exact models. The predicted and corrected distributions can be recovered as a weighted linear regression of the sigma points. A SLAM solution based on the UKF was proposed in [58].

A substantial amount of research has been directed toward improving the computational performance of EKF-based SLAM methods in large-scale scenarios. *Submapping* algorithms achieve this by dividing the state into smaller subsets of landmarks, each estimated by a separate EKF [49, 41, 90]. Members of this family of algorithms vary by the hierarchy that relates the separate submaps [10, 36, 76], or the strategy that controls propagation of information between submaps [30, 75, 77]. Submapping SLAM has been demonstrated in real-world experiments with underwater vehicles [100, 2, 3].

Information Filter Methods

The majority of the computational cost involved with the EKF is in the update step, which involves the inversion of the full covariance matrix. This

is an operation of complexity $O(n^2)$ with respect to the map size n. The Extended Information Filter (EIF) is an alternate formulation of the EKF designed reduce the complexity of SLAM the update phase [91]. Whereas the EKF represents the SLAM state in the *moment form*, the EIF employs an *information form* representation which reduces the update stage to a simple addition operation. The tradeoff is that the mean state and covariance are more difficult to obtain. In order to do so, the information matrix must be inverted, which in a naïve approach, is an equally intensive operation as the EKF update. In general, the information matrix is an approximately-sparse matrix. The smaller values in the matrix represent the weaker correlations between map features, and thus an exactly sparse approximation can be made by zeroing then out. In this case, the state information can be recovered with iterative methods which are linear in the number of features [92]. Eustice et. al. observed that the dense nature of the information matrix in SLAM was due to the marginalization of the previous vehicle pose during the prediction step. By preserving the pose history in the state vector, the information matrix retains an exactly sparse structure. The result is the Exactly Sparse Delayed-State Filter (ESDF), which features the computational advantages of a sparse information filter, without requiring any sparsification approximations [33]. One notable application of the ESDF was the visual mapping of the wreck of the RMS Titanic using an underwater vehicle [32].

2.1.2 Particle Filter Methods

Filter-based methods such as the EKF and EIF make two fundamental assumptions: that the SLAM posterior distribution is Gaussian, and that the process and measurement functions may be reasonably approximated with a first order Taylor expansion. If one or both of these assumptions do not hold, then the algorithms will fail to provide a good solution. SLAM techniques based on particle filters are not constrained by either of those assumptions. Rather than using a parameterized representation for the SLAM posterior, they use a finite number of samples, or particles. Particle filtering has been successfully applied to robot localization, with a known map [91, 24, 37].

However, the SLAM posterior distribution not only contains an estimate of vehicle position, but also describes the locations of map features. Sampling over such a high dimensional space would require an enormous number of particles. Murphy proposed the Rao-Blackwellized particle filter [66] which notes that only the vehicle trajectory needs to be sampled because the map may be computed from it analytically, this key insight forms the basis of particle filter-based SLAM. In FastSLAM [59], each particle represents a distinct hypothesis of the vehicle trajectory and has its own EKF associated with it to estimate the map. The particles are dispersed according to the motion model, to create a proposal distribution. The particles are then weighted and resampled according to the measurement model. FastSLAM 2.0 [60] refines the algorithm by modifying the proposal distribution to improve the richness of particles in the most likely regions of the posterior distribution. DP-SLAM [27][28] is an algorithm analogous to FastSLAM, where evidence grids are used rather than EKFs to represent map information. Particle filter SLAM techniques have seen applications such as exploration of underwater sinkholes [34] and bathymetry surveys [7]. Particle filter algorithms are robust to data association errors. When faced with ambiguous decisions, particles conforming each of the competing hypotheses will be preserved, and the incorrect ones will be eliminated after future measurements and resamplings. Due to the large number of particles, careful memory management is needed for efficient implementation. Moreover, the particle resampling process can cause the distribution to collapse to a single hypothesis if steps are not taken to preserve particle diversity. Methods such as progressive correction and particle regularization [73, 67] can be used to avoid particle degeneracy.

2.1.3 Posegraph Methods

Instead of approaching SLAM as a recursive filtering problem, posegraph methods cast it as an optimization problem. In the foundational work in this area, Lu and Milios consider a collection vehicle poses $\mathbf{X} = [X_0, \cdots, X_n]^T$ is considered and the differences D_{ij} between them. These differences are observed via odometry and sensor inputs as normally-distributed measure-

ments $\bar{D}_{ij}$. with covariance C_{ij}. The maximum likelihood estimate of poses can then be obtained by minimizing the Mahalanobis distance:

$$W = \sum_{0 \leq i \leq j \ leqn} (X_i - X_j - \bar{D}_{ij})^T C_{ij} (X_i - X_j - \bar{D}_{ij})$$

or in matrix form:

$$W = (\bar{\mathbf{D}} - \mathbf{HX})^T \mathbf{C}^{-1} (\bar{D} - \mathbf{HX})$$

where $\mathbf{D}$ is the concatenation of all differences D_{ij} and $\mathbf{H}$ is the matrix operator which implements the difference operation between poses X_i and X_j. In general, the measurement and motion models are non-linear, so non-linear least squares methods such as Gauss Newton or Leverberg-Marquardt are employed [53, 40]. Square root SAM [25] incorporates improvements from matrix factorization techniques, and is further refined in iSAM [48] with the use of Givens rotations. The Thin Junction Tree Filter [74] and Treemap [38] algorithms perform this optimization in tree-like topologies. Posegraph SLAM seeks to jointly optimize the entire history of vehicle poses, under the constraints given by odometry and measurements. By considering the entire trajectory in ensemble more consistent estimates can be obtained compared to filtering methods, which only maintain estimates of the most recent state. To achieve comparable results with filtering, forward-backwards smoothing techniques must be applied. As optimization methods are highly data-driven, they are well suited for applications with rich sensor data such as dense point cloud scans of camera images. However, with increasing amounts of measurement data comes increased risk of data association failures, and thus introduction of spurious constraints into the optimization problem. Outlier rejection algorithms which dynamically alter the topology of the pose graph have been proposed to improve the robustness of the posegraph optimization back-end [86].

2.1.4 Data Association

The SLAM algorithms discussed thus far rely on the ability to determine the correspondence between newly-arriving measurements and previously seen portions of the environment. In EKF SLAM, for example, computing innovation $z_k - h(x_{k|k-1})$, requires that the predicted measurement $h(x)$ be computed from the specific sub-vector in x which matches the environment feature that generated z_k. In posegraph SLAM, constraints between poses are generated by determining their locations relative to commonly observed environment features. In practice, this *data association* is a problem which must be solved separately before the predicted behavior from a SLAM algorithm can be obtained. In feature-based SLAM, the most straightforward approach is to match each measurement to its nearest neighbor, or create a new feature when no existing ones are found within a certain distance threshold [26]. A potential pitfall of this method is that multiple measurements may be matched to a single feature. A *joint compatibility* constraint can be employed to ensure that only one-to-one matchings are considered. The joint compatibility branch and bound (JCBB) algorithm uses a depth-first search of the hypothesis space to produce the maximal jointly-compatible association hypothesis [70]. The 1-Point RANSAC algorithm couples the data association closely to the SLAM algorithm. A number single-point associations are generated at random, and used to perform a partial EKF update. The one which produces the fewest outliers in the full measurement set after the partial update is selected and used to perform the full update [13]. Probabilistic Multiple Hypothesis Tracking (PMHT), an iterative expectation maximization technique, has also been applied to data association in SLAM [23]. Under real-world conditions, a number of measurements will be spurious, or be generated by non-persistent features including these in the SLAM state can potentially degrade the overall algorithm performance. The Feature Stability Histogram is a technique used to constrain the SLAM state to the most reliable environment features [5].

2.1.5 Random Finite Sets

Working from the perspective of target tracking, Mahler proposed framing the problem of tracking multiple targets as estimating a random finite set of multiple vectors, rather than single state vector used in Kalman filtering [54]. A recursive filter is used to estimate a random finite set through its first-order moment, known as the probability hypothesis density (PHD). The random finite set approach is advantageous because the estimation process innately handles the possiblity of spurious measurements. In contrast, traditional vector-based approaches require post-processing to detect and remove spurious measurements. The so-called PHD-Filter methodology was applied to SLAM by Mullane et. al. [65] and has been shown to perform favorably against FastSLAM 2.0 in simulations with high levels of measurement clutter. The PHD filter has also been used to track targets in forward-scan sonar data [18].

2.2 Random Finite Set methods

Random Finite Set methods are based in point process theory, which is concerned with characterizing the occurrence of random events over measureable spaces [22]. Engineering applications arose with the proposal of first-order moment filters for Poisson point processes (PHD filter) [54] and independent and identically distributed (i.i.d.) point processes (CPHD filter) [55]. An alternate derivation of these filters has been demonstrated, showing them to be the limit of bin-occupancy filters [29]. Development of a general chain rule for functional derivatives [14] has provided a mathematical framework for deriving more general moment filters [16]. Practical implementations for these filters were developed using particle [97] and Gaussian Mixtures [96, 98] representations for the moment densities. Early applications of these filters were for multiple object tracking in sonar imagery, [19, 17], tracking of multiple extended objects or groups of objects [39, 89], and target tracking fusing measurements from multiple sensors [93, 68]. Recent work has seen the application area for PHD filters broadened to traffic monitoring [12] and

microscopic video processing [101].
The first work applying PHD filters to SLAM used a Rao-Blackwellized particle filter akin to FastSLAM, replacing the per-particle Kalman filter with a PHD filter [64]. Subsequent work expanded on the single-cluster PHD filter used for group tracking [88], casting SLAM as an analogous problem to group tracking, with the vehicle in lieu of the group, and the environment map as the group components [50], this framework also proposed a method for navigating within environments containing a mixture of static and moving landmarks [51]. Already, these algorithms are gaining traction with researchers in the area of SLAM due to their ability to implicitly model the data association of landmarks and deal with high rates of false positives/negatives. Recently, the IEEE Conference on Robotics and Automation (ICRA) included an entire workshop dedicated to the use of random finite set (RFS) methods in SLAM [63, 61, 15].

2.2.1 Summary

Table 2.1 summarizes the different categories of SLAM solutions that have been discussed. There are several important distinguishing characteristics between the different categories, but they all share the common property of considering uncertainty in vehicle motion and environment sensing. However, with the exception of RFS methods, the algorithms reviewed here are reliant on data association. A complete SLAM solution would typically consist of the algorithm itself with data association "bolted on". RFS methods, in addition to considering uncertainty in sensor motion and landmark state, take into account additional complexities present in environment measurements such as fluctuating landmark cardinality, missed detections, and clutter measurements. In this way, data association is rendered into a non-problem. This attractive property of RFS methods motivates our investigation into how they might be exploited for solving the SLAM problem.

Table 2.1: Classification of SLAM methods

Category	Notable Attributes
Extended Kalman Filter (EKF)	Classical SLAM solution. Assumes Gaussianity of SLAM posterior, complexity of update is quadratic with map size
Extended Information Filter variants	Canonical form dual of the EKF. Update is addtive, but state recovery is quadratic. Constant time execution can be achieved by sparse approximation of information matrix
Rao-Blackwellized Particle Filters	Suitable for non-linear or non-parametric vehicle motion models. Innate multi-hypothesis data association. Requires careful memory management and prone to sample impoverishment
Posegraphs	Employ linear algebra methods to solve the "Full SLAM" problem. Data association is necessary for constructing constraints.
Random Finite Sets	Estimation of a set of random vectors, rather than a single state vector. Innate handling of measurement clutter. In early stages of development.

Chapter 3

Mathematical Prerequisites

The purpose of this chapter is to provide the reader with a mathematical foundation necessary to fully appreciate the work in the remainder of this thesis. The bulk of the material here will relate to finite set statistics and multi-object calculus. Readers who are already familiar with the subject may safely skip this chapter, but are encouraged to stay and review.

3.1 Finite Set Statistics

Multi-object estimation requires a distinct set of mathematical tools compared to vector-based single-object estimation. Collectively, this set of tools is referred to as Finite Set Statistics (FISST) [57]. The key concept in FISST is that of the random finite set (RFS).

3.1.1 Random Finite Sets

The individual objects in our multi-object state are random vector-valued quantities $x_i \in \mathcal{X}$. In many cases, these vectors are contained in some subset of continuous Euclidean space

$$\mathcal{X} \subseteq \mathbb{R}^d$$

Let the entire multi-object state be expressed as an unordered set of indi-

vidual object state vectors. The multi-object state is then itself a random variable on a space we will denote $\mathcal{F}(\mathcal{X})$. This space should contain every possible combination of single object states. That is, it should contain all singleton sets of one object:

$$\mathcal{F}(\mathcal{X}) \supset \mathcal{X}_1 = \{x_1\} \forall\, x_1 \in \mathcal{X}$$

All sets containing two distinct objects:

$$\mathcal{F}(\mathcal{X}) \supset \mathcal{X}_2 = \{x_1, x_2\} \forall\, x_1, x_2 \in \mathcal{X}, x_1 \neq x_2$$

As well as all sets containing three, four, five distinct objects, and so on. Lastly, it should contain the state representing zero objects:

$$\mathcal{F}(\mathcal{X}) \supset \mathcal{X}_0 = \emptyset$$

We thus call $\mathcal{F}(\mathcal{X})$ *the set of all finite sets on* $\mathcal{X}$, and a random variable which takes on values in $\mathcal{F}(\mathcal{X})$ is called a random finite set on $\mathcal{X}$.

3.1.2 Multi-object PDFs and Set Integrals

Like vector-valued random variables, RFSs can be characterized by a probability density function (PDF) which describes the relative likelihood of the RFS to take on certain values in $\mathcal{F}(\mathcal{X})$. We define the multi-object PDF $f_{\mathbf{X}} : \mathcal{F}(\mathcal{X}) \mapsto \mathbb{R}^+$ as:

$$f_{\mathbf{X}}(\mathbf{Y}) = \Pr(\mathbf{X} = \{y_1, \ldots, y_n\}) \tag{3.1}$$

Like their single-object counterparts, multi-object PDFs are real-valued, non-negative functions. Because $\mathbf{X}$ and $\mathbf{Y}$ are unordered sets, the function $f_{\mathbf{X}}(\mathbf{Y})$ is considered to be symmetric in its arguments, that is, for any permutation of the elements $\{y_1, \ldots, y_n\}$, $f_{\mathbf{X}}(\mathbf{Y})$ produces the same value. Such functions are also referred to as *Janossy densities* [22, pg. 125].

In order to develop multi-object filters, we need to develop the concept of integration with multi-object PDFs. Let $f(\mathbf{X})$ be a function whose argument

is an RFS on $\mathcal{X}$. The *set integral* over a region $S \in \mathcal{X}$ is:

$$\int_S f(\mathbf{X})\ \mathrm{d}\mathbf{X} = \sum_{n=0}^{\infty} \frac{1}{n!} \int_{S^n} f(x_1, \ldots, x_n)\ \mathrm{d}x_1 \cdots \mathrm{d}x_n \tag{3.2}$$

$$= f(\emptyset) + \int_S f(x_1)\ \mathrm{d}x_1 + \frac{1}{2} \int_{S \times S} f(x_1, x_2)\ \mathrm{d}x_1\, \mathrm{d}x_2 + \cdots \tag{3.3}$$

The presence of the factor $\frac{1}{n!}$ reflects the fact that $f(\mathbf{X})$ is symmetric in its arguments. The set integral allows us to define some properties of a multi-object PDF.

We can evaluate the probability that $\mathbf{X}$ is contained within a region $S \in \mathcal{X}$ by taking the set integral of $f_{\mathbf{X}}$ over S:

$$\int_S f_{\mathbf{X}}(\mathbf{Y})\ \mathrm{d}\mathbf{Y} = \Pr(\mathbf{X} \subseteq S) \tag{3.4}$$

From the multi-object PDF, we can derive an auxiliary distribution function called the *cardinality distribution*, defined as follows:

$$p_{\mathbf{X}}(n) = \int_{|\mathbf{X}|=n} f_{\mathbf{X}}(\mathbf{Y})\ \mathrm{d}\mathbf{Y} = \Pr(|\mathbf{X}| = n) \tag{3.5}$$

$$= \frac{1}{n!} \int f_{\mathbf{X}}(\{x_1, \ldots, x_n\})\, \mathrm{d}y_1 \cdots \mathrm{d}y_n \tag{3.6}$$

The cardinality distribution is a non-negative function over the natural numbers (zero included), where $p_{\mathbf{X}}(n)$ is the probability that $\mathbf{X}$ contains n elements.

3.1.3 Functional Derivatives

Recall the definition of the classical derivative for a scalar function $f(\mathbf{x})$, along a vector $\mathbf{w}$:

$$\delta f(\mathbf{x}; \mathbf{w}) = \lim_{\epsilon \to 0} \frac{f(\mathbf{x} + \epsilon \mathbf{w}) - f(\mathbf{x})}{\epsilon} \tag{3.7}$$

The derivative is difference in the value of $f(\mathbf{x})$ caused by an infinitesimally small perturbation in the value of the argument, in the direction of $\mathbf{w}$, divided by the size of that perturbation. This derivative is in fact a specialization for scalar functions of the *Gâteaux derivative*:

$$\delta f(x; \eta) = \lim_{\epsilon \to 0} \frac{f(x + \epsilon \eta) - f(x)}{\epsilon} \tag{3.8}$$

The Gâteaux derivative is defined for functions which map between locally convex topological vector spaces. Among other things, these vector spaces encompass coordinate spaces such as $\mathbb{R}^n$ (in which case Equation (3.7) may be applied), systems of linear equations, and function spaces. For our purposes, we are interested in the latter case. To develop the PHD filter equations, it is necessary to take derivatives of functions whose arguments are functions (functionals). Thus, we need to develop further rules for applying the Gâteaux derivative.

Higher order Functional Derivative

The function $f(x)$ may be iteratively differentiated along multiple directions $\eta_1, \ldots, \eta_n$. This nth-order derivative is defined as:

$$\delta^n f(x; \eta_1, \ldots, \eta_n) = \delta(\delta^{n-1} f(x; \eta_1, \ldots, \eta_{n-1}); \eta_n) \tag{3.9}$$

Chain rule for higher order functional derivatives

In order to derive the equations for the PHD filter, we will need to perform higher-order derivatives of compounded functionals (functionals of the form $F[G[h]]$). Given functions f and g mapping between locally convex topolog-

ical vector spaces, the nth-order derivative of the compounded function $f \circ g$ in the directions $\eta_1, \ldots, \eta_n$ is defined as [21]:

$$\delta^n(f \circ g)(x; \eta_1, \ldots, \eta_n) = \sum_{\pi \in \Pi(\eta_{1:n})} \delta^{|\pi|} f(g(x); \xi_{\pi_1}(x), \ldots, \xi_{\pi_{|\pi|}}(x)) \tag{3.10}$$

The derivative is a sum over $\Pi(\eta_{1:n})$: the set of partitions of the set $\{\eta_1, \ldots, \eta_n\}$. Each term in the summation is a $|\pi|$th-order derivative of f, where $|\pi|$ is the cardinality of the partition π. The directions for this derivative are defined as

$$\xi_\omega(x) = \delta^{|\omega|} g(x; \omega_1, \ldots, \omega_{|\omega|}) \tag{3.11}$$

where $\{\omega_1, \ldots, \omega_{|\omega|}\} \subseteq \{\eta_1, \ldots, \eta_n\}$ is a single block of the partition π. It should be noted that this higher order chain rule is applied to *chain differentials*. These are a slightly restricted class of Gâteaux derivatives, incorporating certain continuity assumptions [9], which for our purpose, are appropriate in all but the most exotic modeling scenarios.

3.1.4 Probability Generating Functionals

p.g.fl. are an alternative yet equivalent expression of an RFS's probabilistic behavior. In some cases, they are a more concise representation than the multi-object PDF. As will be seen in the next chapter, they are also an important tool in developing the equations for multi-object moment filters [54, 62]. We first define a *test function* $h(y)$, which is a real-valued function of $y \in \mathcal{X}$, whose values lie in the range $0 \leq h(y) \leq 1$. The p.g.fl. for an RFS with PDF $f_{\mathbf{X}}$ is defined as the set integral:

$$F[h] = \int h^{\mathbf{Y}} f_{\mathbf{X}}(\mathbf{Y}) \, \mathrm{d}\mathbf{Y} \tag{3.12}$$

where the expression $h^{\mathbf{Y}}$ is defined as follows:

$$h^{\mathbf{Y}} = \begin{cases} 1 & \text{if } \mathbf{Y} = \emptyset \\ \prod_{i=1}^{n} h(y_n) & \text{if } \mathbf{Y} = \{y_1, \ldots, y_n\} \end{cases} \tag{3.13}$$

Certain values for the test function h give rise to some interesting cases:

- $F[0] = f_{\mathbf{X}}(\emptyset)$
- $F[1] = \int f_{\mathbf{X}}(\mathbf{Y})\ \mathrm{d}\mathbf{Y} = 1$
- if h is a constant value n, then:

$$\begin{aligned} F[n] &= \int n^{\mathbf{Y}} f_{\mathbf{X}}(\mathbf{Y})\ \mathrm{d}\mathbf{Y} \\ &= f_{\mathbf{X}}(\emptyset) + n \int f_{\mathbf{X}}(\{y_1\})\ \mathrm{d}y_1 + \frac{n^2}{2} \int f_{\mathbf{X}}(y_1, y_2)\ \mathrm{d}y_1\, \mathrm{d}y_2 + \cdots \\ &= \Pr(|\mathbf{X}| = 0) + n\Pr(|\mathbf{X}| = 1) + \frac{n^2}{2}\Pr(|\mathbf{X}| = 2) + \cdots \end{aligned}$$

This is the probability generating function (p.g.f.) of $p_{|\mathbf{X}|}(n)$, the cardinality distribution of $\mathbf{X}$. To recover the distribution from its p.g.f., we perform higher-order derivatives:

$$\Pr(|\mathbf{X}| = n) = \frac{\mathrm{d}^n F}{\mathrm{d}^n a}[a]$$

Just as the single-object distribution $p_{|\mathbf{X}|}(n)$ can be recovered by taking derivatives of its p.g.f., so can a multi-object probability density be recovered by taking derivatives of its p.g.fl.

$$f_{\mathbf{X}}(\mathbf{Y}) = \delta F(0; \mathbf{Y}) \tag{3.14}$$

Finally, consider the union of two statistically independent RFSs, $\mathbf{X}, \mathbf{Y} \in \mathcal{F}(\mathcal{X})$, whose p.g.fl.s are $G_{\mathbf{X}}$ and $G_{\mathbf{Y}}$ respectively. The p.g.fl. of the resulting union is equal to:

$$G_{\mathbf{X} \cup \mathbf{Y}}[h] = G_{\mathbf{X}}[h] \cdot G_{\mathbf{Y}}[h] \tag{3.15}$$

3.1.5 I.I.D. Processes and Poisson Processes

In single object estimation, we generally work with random variables that are described by some parameterized distribution, e.g. Gaussian random vari-

ables or binomial random variables. It should come to no surprise, therefore, that there also exist parameterized distributions for RFSs. We arrive at these distributions by considering the RFS as a realization of a particular type of point process. Here, we discuss two such processess: the i.i.d. process, and the Poisson process.

The probability that an RFS $\mathbf{X} \in \mathcal{F}(\mathcal{X})$ generated by an i.i.d. process takes on a value $\mathbf{Y} = \{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$ is given by the following multi-object probability density function:

$$f_{\mathbf{X}}(\mathbf{Y}) = n!p(n)f(\mathbf{y_1}) \cdots f(\mathbf{y_n}) \tag{3.16}$$

Here, $p(n)$ is the cardinality distribution of $\mathbf{X}$, and $f(y)$ is a probability density function on $\mathcal{X}$. If $f(y)$ is a uniform distribution over $\mathcal{X}$, we call the process *homogeneous*, otherwise, it is *inhomogeneous*.

Poisson processes are a subset of i.i.d. processes, where the cardinality distribution is a Poisson distribution, parameterized by λ, the mean number of points.

$$p(n) = \frac{1}{n!}\lambda^n \exp(-\lambda) \tag{3.17}$$

$$f_{\mathbf{X}}(\mathbf{Y}) = \lambda^n \exp(-\lambda) f(\mathbf{y_1}) \cdots f(\mathbf{y_n}) \tag{3.18}$$

Given these parameterized multi-object probability densities, we can derive the corresponding p.g.fl.s. Recall the definition of the p.g.fl.:

$$F[h] = \int h^{\mathbf{Y}} f_{\mathbf{X}}(\mathbf{Y}) \, \mathrm{d}\mathbf{Y} \tag{3.19}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \int h(\mathbf{y_1}) \cdots h(\mathbf{y_n}) f_{\mathbf{X}}(\mathbf{y_1}, \ldots, \mathbf{y_n}) \, \mathrm{d}\mathbf{y_1} \cdots \mathrm{d}\mathbf{y_n} \tag{3.20}$$

Substituting the value of $f_{\mathbf{X}}(\mathbf{Y})$ for the Poisson process (3.18):

$$F[h] = \exp(-\lambda) + \lambda \exp(-\lambda) \int h(\mathbf{y_1}) f(\mathbf{y_1}) \, \mathrm{d}\mathbf{y_1} \tag{3.21}$$

$$+ \frac{1}{2}\lambda^2 \exp(-\lambda) \int h(\mathbf{y_1}) h(\mathbf{y_2}) f(\mathbf{y_1}) f(\mathbf{y_2}) \, \mathrm{d}\mathbf{y_1} \, \mathrm{d}\mathbf{y_2} + \cdots \tag{3.22}$$

$$= \exp(-\lambda) \sum_{n=0}^{\infty} \frac{(\lambda \int h(\mathbf{y}) f(\mathbf{y}) \, \mathrm{d}\mathbf{y})^n}{n!} \tag{3.23}$$

$$= \exp(\lambda f[h] - \lambda) \tag{3.24}$$

where

$$f[h] = \int h(\mathbf{y}) f(\mathbf{y}) \, d\mathbf{y}$$

Through a similar reasoning, we can derive the p.g.fl. of an i.i.d. process to be:

$$F[h] = \sum_{n=0}^{\infty} p(n) (\int h(\mathbf{y}) f(\mathbf{y}) \, d\mathbf{y})^n \tag{3.25}$$

$$= G(f[h]) \tag{3.26}$$

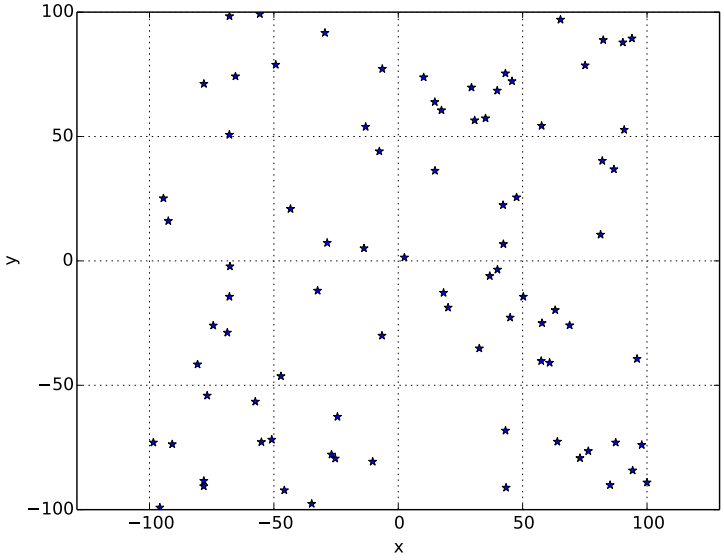
Where $G(y) = \sum_{n=0}^{\infty} p(n) y^n$ is the probability generating function of the cardinality distribution $p(n)$. These results, combined with the functional derivatives in Subsection 3.1.3, will be the tools necessary to derive the multi-object filters in the following chapter.

Simulating Point Processes

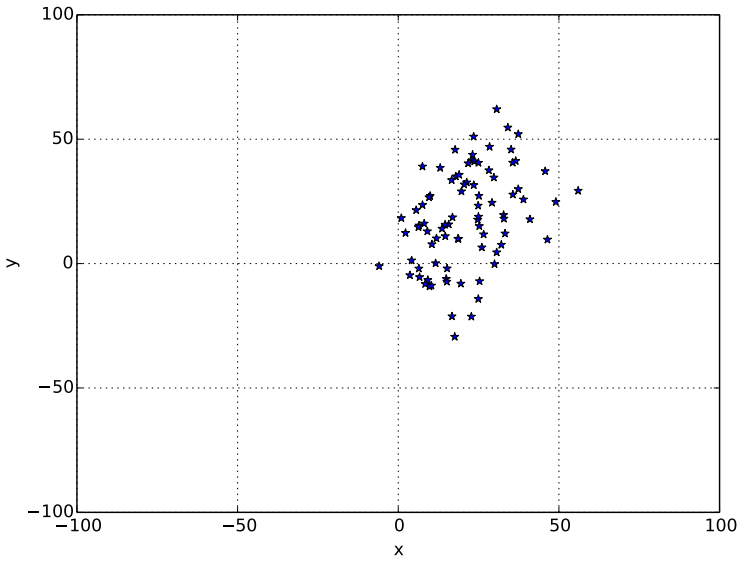
To simulate an i.i.d. process, we first sample $n \sim p(n)$ to get the number of objects, and then we draw n independent samples from $f(y)$. This process is outlined in procedure simulateiidprocess. Example results are shown in Figure 3.1.

Procedure simulateiidprocess(p,s)

Input: cardinality distribution p, spatial distribution s
$n \sim p(n)$
for $i = 1 \ldots n$ **do**
  $x^i \sim s(x)$
end
return $\{x^1, \ldots, x^n\}$



(a) Homogeneous



(b) Inhomogeneous

Figure 3.1: Two simulated Poisson point processes over the region $\mathcal{X} = [x, y] \in [-100, 100]$. In both cases, the cardinality is drawn from a Poisson distribution with $\lambda = 100$. The homogeneous process samples from a uniform distribution over $\mathcal{X}$. The second process uses a spatial distribution which is a 2D normal with mean $[20, 20]$ and variance $\sigma_x^2 = 160$, $\sigma_y^2 = 360$ and $\sigma_{xy}^2 = 100$.

3.1.6 Cluster Processes

Cluster processes are hierarchical relationships between two point processes, in which the realization of a *daughter process* is conditioned on a *parent process*. This concept is best illustrated through examples.

Segment Cox Process

The Segment Cox Process is illustrated in Figure 3.2. The parent process is a Poisson process which uniformly samples the space of line segment locations and orientations. For each generated line segment, a daughter Poisson process samples points along its length.

Matérn Process

The Matérn Process is illustrated in Figure 3.3. The parent process is a Poisson process which uniformly samples the space of cluster center locations. For each generated cluster, a daughter Poisson process samples points within a fixed radius R.

From these examples, we see that only the realizations of the daughter process are observed as points. The realizations of the parent process are observed indirectly as a conditioning on the daughter process. In the given examples, the daughter processes are spatially centered about the realizations of the parent process. Although this serves well to illustrate the link between the parent and the daughter, such a spatial positioning is not necessary for cluster processes. Rather, the defining characteristic is only that the daughter be somehow conditioned on the parent process. This is best conveyed through the language of probability generating functionals. The p.g.fl. of any cluster process can be expressed as:

$$G_c[h] = G_p[G_d[h|\cdot]] \tag{3.27}$$

Where $G_p[h]$ is the p.g.fl. of the parent process, and $G_d[h|\cdot]$ is the p.g.fl. of the daughter process, conditioned on the parent process.

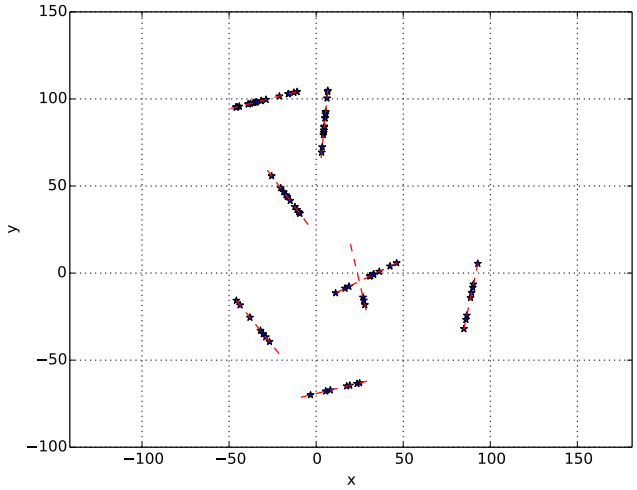


Figure 3.2: Segment Cox process. The parent Poisson process has a rate parameter $\lambda_1 = 5$ and the daughter Poisson process has a rate parameter $\lambda_2 = 10$. Line segments have a fixed length $L = 40$.

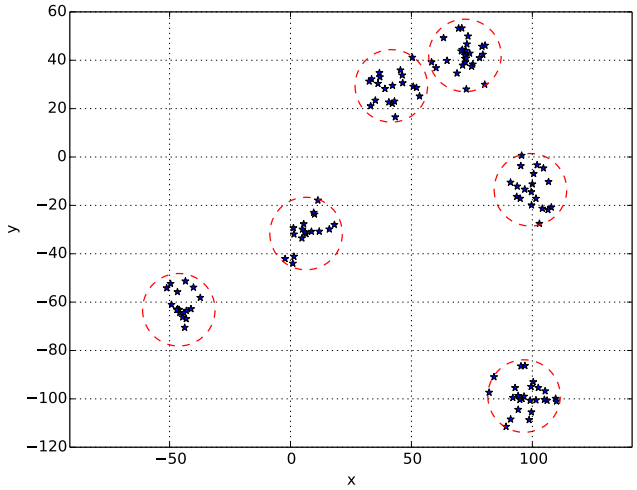


Figure 3.3: Matérn process. The parent Poisson process has a rate parameter $\lambda_1 = 5$ and the daughter Poisson process has a rate parameter $\lambda_2 = 20$. Clusters have a fixed radius $R = 20$.

3.2 Evaluating Multi-object State Estimates

When running simulations with a multi-object filter, it is useful to be able to perform comparisons between the filter state and the simulation ground truth, in order to evaluate its performance. For filters which use vector representations of the state variable, this is fairly straight-forward. Two vectors in $\mathcal{X}$ can easily be compared with the Euclidean distance. However multi-object states reside in the space $\mathcal{F}(\mathcal{X})$, where the Euclidean distance is not defined, and a different distance metric needs to be employed.
The Optimal Subpattern Assignment (OSPA) metric [84] is the distance measure routinely encountered in the multi-object estimation literature. For two multi-object states $X = \{x_1, \cdots, x_m\}, Y = \{y_1, \cdots, y_n\} \in \mathcal{F}(\mathcal{X})$, $n \geq m$ their OSPA distance is computed in the following fashion:

1. Specify two parameters:
   - the order p, with $1 \leq p \leq \infty$
   - the cutoff c, with $c > 0$
2. Choose a distance metric d on the single object space $\mathcal{X}$, for example the Euclidean distance. Modify the metric such that its maximum value is the cutoff parameter:
$$d^{(c)}(x, y) = min(d(x, y), c)$$
3. Use this distance to compute a cost matrix C containing the pairwise p-th order distances between every element of X and Y:
$$C_{ij} = d^{(c)}(x_i, y_j)^p$$
4. From this cost matrix, compute the optimal assignment $\pi \in \Pi_n$ which assigns elements from X to elements in Y such that the total cost is minimized. A modified Hungarian/Munkres algorithm [11] is used to perform this assignment.

5. If the two multi-object states have differing cardinalities, we will be left with unassigned features from the larger set. For each unassigned feature, assign a cost of c^p.

6. The OSPA distance is the p-th order average of the individual pairing costs.

$$d_p^{(c)}(X,Y) = (\frac{1}{n}(\min_{\pi\in\Pi_n}\sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)}^p + (n-m)c^p))^{1/p} \tag{3.28}$$

The resulting quantity is a true distance metric in the sense that if satisfies the following criteria:

1. Identity: $d_p^{(c)}(X,Y) = 0$, if and only if $X = Y$

2. Symmetry: $d_p^{(c)}(X,Y) = d_p^{(c)}(Y,X) \,\forall\, X, Y \in \mathcal{F}(\mathcal{X})$

3. Triangle Inequality: $d_p^{(c)}(X,Y) + d_p^{(c)}(Y,Z) \geq d_p^{(c)}(X,Z) \forall X, Y, Z \in \mathcal{F}(\mathcal{X})$

Moreover, Eq. (3.28) actually defines an entire family of metrics, determined by the choice of parameters p and c. The value of p affects how severely high inter-object distances are penalized when pairing objects in X and Y. For a fixed value of c, we have the following relationship:

$$d_{p_1}^{(c)}(X,Y) \leq d_{p_2}^{(c)}(X,Y) \quad 1 \leq p_1 \leq p_2 \leq \infty$$

The cutoff value c is the cost assigned to unmatched objects and thus determines the penalty for mismatched cardinalities. It also determines the maximum inter-object distance to consider when finding the optimal assignment.

An additional feature of the OSPA metric is that it may be decomposed into two components: one which represents the localization error, and another

which represents the cardinality error.

$$e_{loc} = (\frac{1}{n}\sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)}^p)^{1/p} \tag{3.29}$$

$$e_{card} = (\frac{(n-m)c^p}{n})^{1/p} \tag{3.30}$$

These components can provide additional insight on which factors contribute the total OSPA error.
The OSPA metrics reported in Chapters 5 and 7 use parameter values $p = 1$ and $c = 1$.

Chapter 4

Multi-object Moment Filters

This chapter illustrates how the mathematical tools presented in the first chapter may be used to derive multi-object moment filters that will be employed in the main body of this work. This chapter provides a tutorial on the single-cluster PHD filter proposed by Swain and Clark [87]. We will begin by discussing the exact Bayes' rule solution to multi-object filtering, and the reasons that it is impractical for most applications. We will continue by presenting a first moment approximation of the multi-object Bayes filter, namely the PHD filter. Finally, we will extend the PHD filter to single cluster point processes, to arrive at the SC-PHD which will be applied to our SLAM solution. Here, we will use rather general terminology to discuss these filters, while the following chapters will place them in the context of SLAM.

4.1 Multi-object Bayes' Filter

Consider an RFS-valued multi-object state, described by a multi-object probability density $p(\mathbf{X})$. Over successive time steps, k, demarcated by measurements $\mathbf{Z}_k$ of the multi-object state, we can maintain an estimate of the multi-object probability density with a Bayes' filter iteration.

In most estimation problems of interest, the multi-object state undergoes change through time. We model this change as a Markov transition density

$f(\mathbf{X}|\mathbf{X}')$. Between measurements we apply the following prediction equation:

$$p_{k|k-1}(\mathbf{X}) = \int f_k(\mathbf{X}|\mathbf{X}')p_{k-1}(\mathbf{X}')\,\mathrm{d}\mathbf{X}' \tag{4.1}$$

When the state is observed, producing a measurement set $\mathbf{Z}_k$, the probability density can be updated via Bayes' rule:

$$p_{k|k}(\mathbf{X}|\mathbf{Z}) = \frac{p_{k|k-1}(\mathbf{X})p_k(\mathbf{Z}|\mathbf{X})}{\int p_{k|k-1}(\mathbf{X})p_k(\mathbf{Z}|\mathbf{X})\,\mathrm{d}\mathbf{X}} \tag{4.2}$$

Where $p_k(\mathbf{Z}|\mathbf{X})$ is a multi-object measurement likelihood.

It is difficult, if not impractical, to implement the Bayes' filter exactly. The integrals in the prediction and denominator of the update do not always exist in closed form. In single-object filtering, this can be mitigated by approximating the prior distribution. Applying a Gaussian approximation yields the Kalman filter, while a particle representation can be used for non-parameterized priors, resulting in the particle filter. However, in multi-object filtering, the intractability of the integral is compounded by the fact that it is now the set integral defined in the previous chapter. As already seen, this integral quickly becomes cumbersome for higher numbers of objects. Finally, full multi-object probability densities are difficult to manipulate, even in parameterized form, due to the necessity of accounting for spatial distributions for every possible cardinality. For these reasons, we choose to use a filter which propagates the first moment approximation of the multi-object density. The following section will demonstrate the derivation of this filter.

4.2 PHD Filter

For single-object probability densities, the first order moment corresponds to the expected value:

$$E[\mathbf{x}] = \int \mathbf{x} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{4.3}$$

One might expect that the multi-object analog would be simple substitution of the vector-valued variable $\mathbf{x}$ for its RFS-valued counterpart $\mathbf{X}$, yielding:

$$E[\mathbf{X}] = \int \mathbf{X} f(\mathbf{X}) \, \mathrm{d}\mathbf{X} \tag{4.4}$$

The factor of $\mathbf{X}$ within the integrand is problematic because addition between random finite sets is not defined. To obtain a well-behaved integral, we apply a mapping between the RFS $\mathbf{X}$ and its underlying vector space $\mathcal{X}$:

$$T_{\mathbf{X}} = \begin{cases} 0 & \text{if } \mathbf{X} = \emptyset \\ \sum_{\mathbf{x} \in \mathbf{X}} \delta(\mathbf{x}) & \text{otherwise} \end{cases} \tag{4.5}$$

We now define the first order moment to be the expected value of $T_{\mathbf{X}}$. This moment is called the *Probability Hypothesis Density*, or PHD. The PHD is also referred to as the *intensity* of a multi-object distribution.

$$D_{\mathbf{X}}(x) = \int T_{\mathbf{X}} f(\mathbf{X}) \, \mathrm{d}\mathbf{X} \in \mathcal{X} \tag{4.6}$$

What does a PHD tell us about its corresponding probability distribution? As in the case of the single object first moment, the PHD can be considered to be a sort of expected value. Specifically, the integral of the PHD over a particular region $B \in \mathcal{X}$ yields the expected number of objects falling within B

$$\int_{B \in \mathcal{X}} D_{\mathbf{X}}(x) \, \mathrm{d}(x) = E[|\mathbf{X} \cap B|] \tag{4.7}$$

In addition to the cardinality of the RFS, the PHD also contains information about the location of objects within $\mathcal{X}$ and the precision to which that location is known. Figure 4.1 illustrates this concept. It should be noted that while the PHD can be considered to be a density function, it does not fit the definition of a *probability density function*. Since its integral over $\mathcal{X}$ is the expected number of objects, and not necessarily equal to 1.

Since it is a function over a vector space, the PHD is much easier to manipulate than the full multi-object probability density. This means it is possible to obtain a Bayes' rule filter which propagates the PHD.

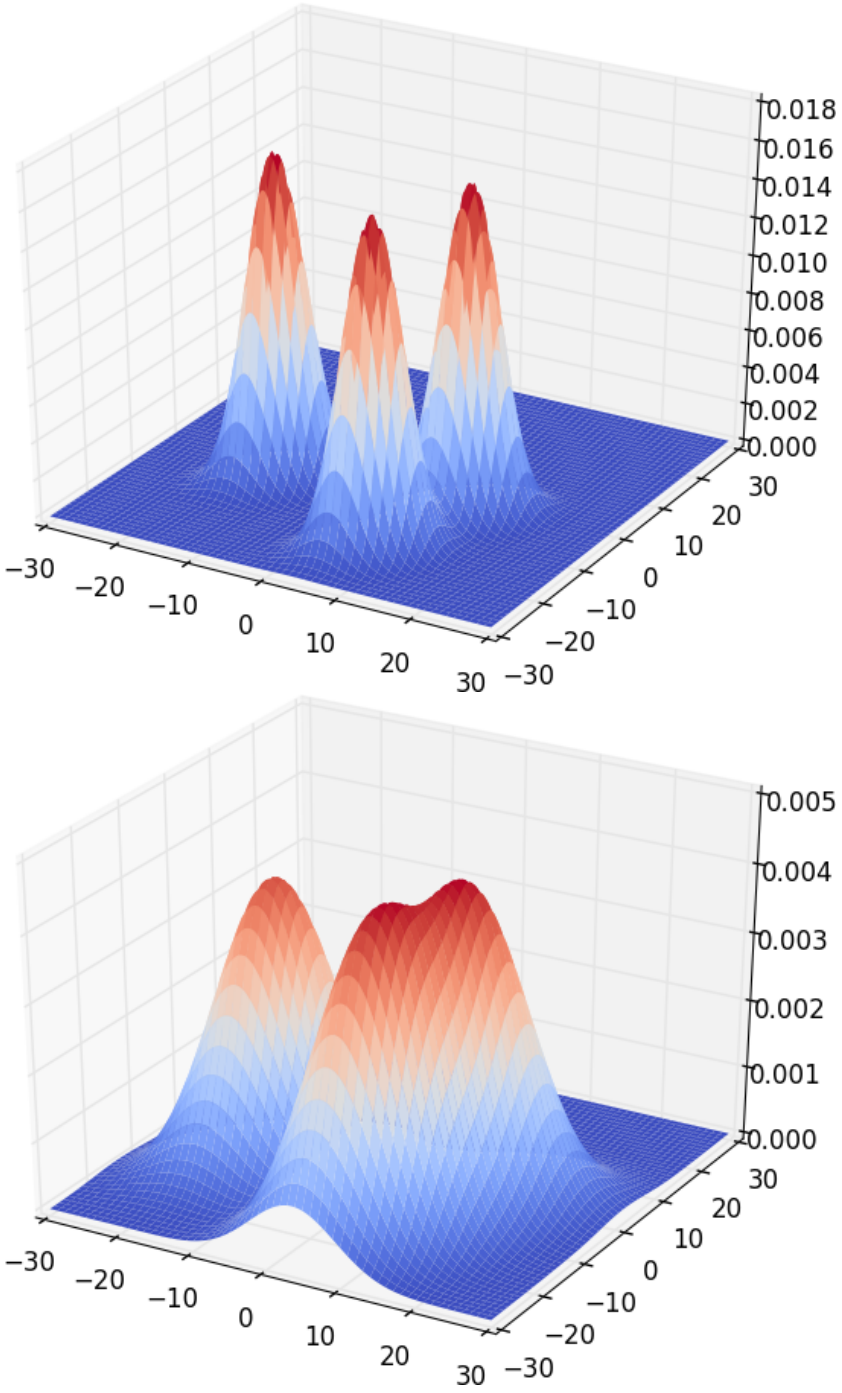


Figure 4.1: Example PHDs over a 2-dimensional state space. The integral of both these densities gives $n = 3$ as the expected number of objects. However, the one above left indicates a much higher certainty in the location estimate of the 3 objects.

While the definition of the PHD offered in (4.6) is perhaps the most accessible, for the purposes of deriving the filter, we will employ an alternative definition of the PHD:

$$D_{\mathbf{X}}(x) = \delta G(1; x) \tag{4.8}$$

That is, the PHD of the RFS $\mathbf{X}$ is equal to the functional derivative of its p.g.fl., evaluated at $h = 1$. A proof of this relationship can be found in [57, p.581-582].

4.2.1 PHD Filter Update

Let us begin with deriving the update equation for the PHD filter. The p.g.fl. for the Bayes updated multi-object probability density is [57, p. 757-758]:

$$G_{k|k}[h] = \frac{\delta F(0, h; Z_k)}{\delta F(0, 1; Z_k)} \tag{4.9}$$

with

$$F[g, h] = \int h^{\mathbf{X}} G_{k|k}[g|\mathbf{X}] p_{k|k-1}(\mathbf{X}) \, \mathrm{d}\mathbf{X} \tag{4.10}$$

$$G_k[g|\mathbf{X}] = \int g^{\mathbf{Z}} p_k(\mathbf{Z}|\mathbf{X}) \, \mathrm{d}\mathbf{Z} \tag{4.11}$$

We can observe that $G_k[g|\mathbf{X}]$ is the p.g.fl. for the measurement likelihood distribution $p_k(\mathbf{Z}|\mathbf{X})$. In order to obtain a more concrete expression, we make some modeling assumptions regarding the measurement process:

1. Each object $\mathbf{x} \in \mathbf{X}$ generates a measurement $\Upsilon(\mathbf{x})$ independently of the other objects. With a probability $p_D \leq 1$, $\Upsilon(\mathbf{x})$ will consist of a single measurement $\{\mathbf{z}\}$ or, with a probability of $1 - p_D$, a missed detection $\emptyset$. The singleton measurements are drawn from a single-object measurement likelihood distribution $p(\mathbf{z}|\mathbf{x})$.

$$\Upsilon(\mathbf{x}) = \begin{cases} \emptyset & \text{with probability } (1 - p_D) \\ \{\mathbf{z}\} \sim p(\mathbf{z}|\mathbf{x}) & \text{with probability } p_D \end{cases} \tag{4.12}$$

2. In addition to objects-generated measurements, $\mathbf{Z}$ includes a number of false alarm, or clutter measurements, generated independently of the objects. These clutter measurements are drawn from a Poisson process with rate parameter λ and spatial distribution $c(\mathbf{z})$. That is, the clutter measurements are an RFS $C = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ where each $\mathbf{z}$ is drawn from $c(\mathbf{z})$ and the cardinality m is drawn from a Poisson distribution with parameter λ.

Thus, the measurement set can be modeled as

$$\mathbf{Z} = \Upsilon(\mathbf{x_1}) \cup \cdots \cup \Upsilon(\mathbf{x_n}) \cup C \tag{4.13}$$

As we have specified that each component of this union is statistically independent of the others, the p.g.fl. of $\mathbf{Z}$ is simply the product of the p.g.fl.s for each individual term. Since the clutter process is Poisson, we have already shown in equation 3.24 that it's p.g.fl. is

$$G_c[g] = \exp(\lambda c[g] - \lambda) \tag{4.14}$$

We now turn our attention to the target-generated measurements. From the definition of the p.g.fl.:

$$G_z[g|\mathbf{x}] = \int g^{\mathbf{Z}} f(\mathbf{Z}|\mathbf{x}) \, \mathrm{d}\mathbf{Z} \tag{4.15}$$

$$= f(\emptyset|\mathbf{X}) + \int g(\mathbf{z}) f(\{\mathbf{z}\}|\mathbf{x}) \, \mathrm{d}\mathbf{z} \tag{4.16}$$

$$= (1 - p_D) + p_D \int g(\mathbf{z}) p(\mathbf{z}|\mathbf{x}) \, \mathrm{d}\mathbf{z} \tag{4.17}$$

Abbreviating $p_g = \int g(\mathbf{z}) p(\mathbf{z}|\mathbf{x}) \, \mathrm{d}\mathbf{z}$, we therefore obtain the p.g.fl. of the measurement set as:

$$G_k[g|\mathbf{X}] = G_z[g|\mathbf{x_1}] \times \cdots \times G_z[g|\mathbf{x_n}] \times G_c[g] \tag{4.18}$$

$$= (1 - p_D + p_D p_g)^{\mathbf{X}} \exp \lambda c[g] - \lambda \tag{4.19}$$

Substituting into $F[g,h]$:

$$F[g,h] = \int h^{\mathbf{X}}(1-p_D+p_Dp_g)^{\mathbf{X}} \exp(\lambda c[g]-\lambda) p_{k|k-1}(\mathbf{X})\,\mathrm{d}\mathbf{X} \tag{4.20}$$

$$= \exp(\lambda c[g]-\lambda) \int (h(1-p_D+p_Dp_g))^{\mathbf{X}} p_{k|k-1}(\mathbf{X})\,\mathrm{d}\mathbf{X} \tag{4.21}$$

$$= \exp(\lambda c[g]-\lambda) G_{k|k-1}[h(1-p_D+p_Dp_g)] \tag{4.22}$$

$G_{k|k-1}[h]$ is the p.g.fl. of the predicted RFS $\mathbf{X}$. The final modelling assumption necessary to obtain the PHD update equation is to specify that the predicted RFS behaves according to a Poisson process. This give us:

$$F[g,h] = \exp(\lambda c[g]-\lambda) \times \exp(\mu(s[h(1-p_D+p_Dp_g)]-1)) \tag{4.23}$$

$$= \exp(\lambda c[g]-\lambda+\mu(s[h(1-p_D+p_Dp_g)]-1)) \tag{4.24}$$

The foregoing equation can be expressed as a compounded functional:

$$F[g,h] = G[T_h[g]] \tag{4.25}$$

$$G[h] = \exp(\mu(s[h]-1)) \tag{4.26}$$

$$T_h[g] = h(1-p_D+p_Dp_g) + \mu^{-1}\lambda(c[g]-1) \tag{4.27}$$

Expressed in this way, the generalized chain rule for functional derivatives can be applied to obtain the numerator and denominator terms in Equation (4.9) to obtain the p.g.fl. of the Bayes updated multi-object distribution, and then further differentiated as in Equation (4.8) to obtain the updated PHD. A full derivation is outside the scope of this work, but can be found in [16]. The resulting expression for the updated PHD is as follows:

$$D_k(\mathbf{x}) = D_{k|k-1}(\mathbf{x})\left[(1-p_D) + \sum_{\mathbf{z}\in\mathbf{Z}_k} \frac{p(\mathbf{z}|\mathbf{x})p_D(\mathbf{x})}{\eta_z(\mathbf{x})}\right] \tag{4.28}$$

$$\eta_z(\mathbf{x}) = \kappa_k(\mathbf{z}) + \int D_{k|k-1}(\mathbf{x})p_D(\mathbf{x})p(\mathbf{z}|\mathbf{x})\;\mathrm{d}\mathbf{x} \tag{4.29}$$

with

$D_{k\|k-1}(\mathbf{x})$	predicted PHD
$p_D(\mathbf{x})$	detection probability of a feature state $\mathbf{x}$
$p(\mathbf{z}\|\mathbf{x})$	single-object observation likelihood for a measurement z, given an object state $\mathbf{x}$
$\kappa_k(z)$	PHD of the measurement clutter process evaluated at z

4.2.2 PHD Filter Prediction

The prediction step of the PHD filter propagates any possible changes in the multi-object state between measurements, typically the motion of individual objects. Derivation of the p.g.fl. for the multi-object Bayes prediction (4.1) is relatively simple compared to that of the update. It emerges from a straightforward application of the definition of the p.g.fl.

$$G_{k|k-1}[h] = \int h^{\mathbf{X}} p_{k|k-1}(\mathbf{X})\,\mathrm{d}\mathbf{X} \tag{4.30}$$

$$= \int h^{\mathbf{X}} \int f_k(\mathbf{X}|\mathbf{X}') p_{k-1}(\mathbf{X}')\,\mathrm{d}\mathbf{X}'\,\mathrm{d}\mathbf{X} \tag{4.31}$$

$$= \int p_{k-1}(\mathbf{X}') \int h^{\mathbf{X}} f_k(\mathbf{X}|\mathbf{X}')\,\mathrm{d}\mathbf{X}\,\mathrm{d}\mathbf{X}' \tag{4.32}$$

$$= \int p_{k-1}(\mathbf{X}') G_{k|k-1}[h|\mathbf{X}']\,\mathrm{d}\mathbf{X}' \tag{4.33}$$

Note that $G_{k|k-1}[h|\mathbf{X}']$ is the p.g.fl. of the multi-object transition density $f_k(\mathbf{X}|\mathbf{X}')$ To obtain a concrete expression for this p.g.fl., we begin by modeling the predicted RFS. Given a prior RFS $\mathbf{X}_{k-1} = \{\mathbf{x_1}', \ldots, \mathbf{x_n}'\}$, the predicted RFS will be union of the individual objects in $\mathbf{X}_{k-1}$ propagated forward in time, together with a birth RFS B, representing newly-appearing objects. Furthermore, the evolution of each individual object through time is statistically independent from all other objects, including ones newly appearing.

$$X_{k|k-1} = \Sigma(\mathbf{x_1}') \cup \cdots \cup \Sigma(\mathbf{x_n}') \cup B_k \tag{4.34}$$

Note that much of the literature involving the PHD filter also includes a *spawning process*, which describes the appearance of new objects dependent on existing ones. We will omit this piece for the sake of simplicity. We

will account for the possiblity that objects may cease to exist between time steps, characterized by a survival probability $p_S \leq 1$. For each object that survives, its predicted state will be sampled from a single-object transition density $f_k(\mathbf{x}|\mathbf{x}')$.

$$\Sigma(\mathbf{x}') = \begin{cases} \emptyset & \text{with probability } (1 - p_S) \\ \{\mathbf{x}\} \sim f_k(\mathbf{x}|\mathbf{x}']) & \text{with probability } p_S \end{cases} \tag{4.35}$$

This survival model is completely analogous to the detection model in Equation (4.12). Thus, we can use the same line of reasoning used to derive the p.g.fl. for object-generated measurements (4.17) to obtain the p.g.fl. for transitioned objects

$$G_{k|k-1}[h|\mathbf{x}'] = (1 - p_S + p_S p_h) \tag{4.36}$$

$$p_h = \int h(\mathbf{x}) f_k(\mathbf{x}|\mathbf{x}') \, \mathrm{d}\mathbf{x} \tag{4.37}$$

Let $G_b[h]$ be the p.g.fl. for the birth RFS B. The p.g.fl. for the multi-object transition density is therefore:

$$G_{k|k-1}[h|\mathbf{X}'] = G_b[h](1 - p_S + p_S p_h)^{\mathbf{X}'} \tag{4.38}$$

Substitute into Equation (4.33) to obtain the p.g.fl. for the predicted multi-object distribution.

$$G_{k|k-1}[h] = \int G_b[h](1 - p_S + p_S p_h)^{\mathbf{X}'} p_{k-1}(\mathbf{X}') \, \mathrm{d}\mathbf{X}' \tag{4.39}$$

$$= G_b[h] \int (1 - p_S + p_S p_h)^{\mathbf{X}'} p_{k-1}(\mathbf{X}') \, \mathrm{d}\mathbf{X}' \tag{4.40}$$

$$= G_b[h] G_{k-1}[(1 - p_S + p_S p_h)] \tag{4.41}$$

Taking the functional derivative, we obtain the PHD filter prediction equation.

$$D_{k|k-1}(\mathbf{x}) = \delta G_{k|k-1}(1; \mathbf{x}) \tag{4.42}$$

$$= \int p_S(\mathbf{x}) f(\mathbf{x}|\mathbf{x}') D_{k-1}(\mathbf{x}) \, \mathrm{d}\mathbf{x}' + \gamma(\mathbf{x}) \tag{4.43}$$

with the following definitions:

$D_{k-1}(\mathbf{x})$	prior PHD
$p_S(\mathbf{x})$	survival probability for a given object state $\mathbf{x}$
$f_k(\mathbf{x}\|\mathbf{x}')$	single-object observation transition density
$\gamma_k(x)$	PHD of the birth RFS

With the prediction and update equations defined, the PHD filter is complete. We will now move on to consider the SC-PHD filter.

4.3 SC-PHD Filter

The applications in the remainder of this work utilize a generalization of the PHD filter, called the SC-PHD filter. This filter considers the multi-object state to be generated by a cluster process, such as those described in Section 3.1.6, with the additional condition that the parent process consist of a single object. To derive the SC-PHD filter, we follow a procedure similar to the one outlined in the previous section. Where we encounter $G_{k-1}[h]$ and $G_{k|k-1}[h]$, the p.g.fl.s for the prior and predicted multi-object probability distributions, we substitute p.g.fl.s for cluster processes

$$G[h] = G_p[G_d[h]] \tag{4.44}$$

Given the requirement that the parent RFS be a singleton, the p.g.fl. of the parent process must be:

$$G_p[h] = \int h(\mathbf{x}) s(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{4.45}$$

Thus, the PHD of the parent process is identical to the spatial distribution $s(\mathbf{x})$, and the single cluster p.g.fl. is simply the inner product of the spatial distribution of the parent with the p.g.fl. of the daughter process.

$$G[h] = \int s(\mathbf{x}) G_d[h] \, \mathrm{d}\mathbf{x} \tag{4.46}$$

Here, we will present the SC-PHD filter equations, foregoing a full derivation. However, a complete discussion can be found in [89, Appendix B].

4.3.1 SC-PHD prediction

Under the assumption that the introduction of new daughter process objects is statistically independent of the dynamics of existing object, the predicted joint intensity for the single-cluster process is given by:

$$D_{k|k-1}(\mathbf{x},\mathbf{y}) = \int s_{k-1}(\mathbf{x}')\pi_{k|k-1}(\mathbf{x}|\mathbf{x}')\tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x}')\,\mathrm{d}\mathbf{x}' \tag{4.47}$$

where $\tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x})$ is the predicted PHD of the daughter process:

$$\tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x}) = \gamma_{k|k-1}(\mathbf{y}|\mathbf{x}) + \underbrace{\int \tilde{D}_{k-1}(\mathbf{y}'|\mathbf{x})p_S(\mathbf{y}'|\mathbf{x})\pi_{k|k-1}(\mathbf{y}|\mathbf{y}';\mathbf{x})\;\mathrm{d}\mathbf{y}'}_{\tilde{D}^p_{k|k-1}(\mathbf{y}|\mathbf{x})} \tag{4.48}$$

with the following definitions:

$s_{k-1}(\mathbf{x}')$ prior PHD of the parent state from time $k-1$

$\pi_{k|k-1}(\mathbf{x}|\mathbf{x}')$ Markov transition density of the parent

$\gamma_{k|k-1}(\cdot|\mathbf{x}')$ PHD for daughter birth process at time k, conditioned on parent state $\mathbf{x}'$

$\tilde{D}_{k-1}(\mathbf{y}'|\mathbf{x}')$ prior PHD of the daughter state from time $k-1$, conditioned on parent state $\mathbf{x}'$

$p_S(\cdot|\mathbf{x}')$ object survival probability, conditioned on parent state $\mathbf{x}'$

$\pi_{k|k-1}(\cdot|\cdot;\mathbf{x}')$ single-object Markov transition density for daughter process, conditioned on parent state $\mathbf{x}'$

$\tilde{D}^p_{k|k-1}(\mathbf{y}|\mathbf{x})$ portion of the predicted PHD corresponding to objects persisting from previous time step.

4.3.2 SC-PHD Update

The measurement update for the joint single-cluster PHD is:

$$D_{k|k}(\mathbf{x},\mathbf{y}) = \frac{s_{k|k-1}(\mathbf{x})L_{\mathbf{Z}_k}(\mathbf{x})}{\int s_{k|k-1}(\mathbf{x})L_{\mathbf{Z}_k}(\mathbf{x})\ \mathrm{d}\mathbf{x}}\tilde{D}_{k|k}(\mathbf{y}|\mathbf{x}) \tag{4.49}$$

where $\tilde{D}_{k|k}(\mathbf{y}|\mathbf{x})$ is the updated PHD of the daughter:

$$\tilde{D}_{k|k}(\mathbf{y}|\mathbf{x}) = \tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x})\left[(1-p_D(\mathbf{y}|\mathbf{x})) + \sum_{z\in\mathbf{Z}_k}\frac{g(z|\mathbf{y},\mathbf{x})p_D(\mathbf{y}|\mathbf{x})}{\eta_z(\mathbf{y}|\mathbf{x})}\right] \tag{4.50}$$

$$\eta_z(\mathbf{x}) = \kappa_k(z) + \int \tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x})p_D(\mathbf{y}|\mathbf{x})g(z|\mathbf{y},\mathbf{x})\ \mathrm{d}\mathbf{y} \tag{4.51}$$

with

$s_{k|k-1}(\mathbf{x})$ predicted PHD of the parent state

$\tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x})$ predicted PHD of the daughter state, conditioned on parent state $\mathbf{x}$

$p_D(\mathbf{y}|\mathbf{x})$ detection probability of a daughter state $\mathbf{y}$ given a parent state $\mathbf{x}$

$g(z|\mathbf{y},\mathbf{x})$ single-object observation likelihood for a measurement z given a daughter state $\mathbf{y}$ and parent state $\mathbf{x}$

$\kappa_k(z)$ PHD of the measurement clutter process evaluated at z

$L_{\mathbf{Z}_k}(\mathbf{x})$ the multi-object observation likelihood of the measurement set $\mathbf{Z}_k$ given a parent state $\mathbf{x}$

it is defined by:

$$L_{\mathbf{Z}_k}(\mathbf{x}) = \exp\left\{-\int p_D(\mathbf{y}|\mathbf{x})\tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x})\ \mathrm{d}\mathbf{y}\right\}\prod_{z\in\mathbf{Z}_k}\eta_z(\mathbf{x}) \tag{4.52}$$

We can see that the SC-PHD filter does not deviate very far from the PHD filter. The prediction and update equations for the daughter PHD are virtually identical to those of the PHD filter, the sole difference being the conditioning on the parent state. The parent prediction involves a straightforward inner product with a separate Markov transition density. The key result is the multi-object observation likelihood $L_{\mathbf{Z}_k}(\mathbf{x})$ for performing the parent update.

Chapter 5

SLAM with SC-PHD Filters

In this chapter, we describe the use of the SC-PHD filter derived in Chapter 4 for solving the SLAM problem. The filter is implemented with a hybrid particle/Gaussian mixture formulation, with modifications for measurement-driven birth, changing field of view, and features with heterogeneous dynamics. We present experimental results with synthetic datasets, and compare this algorithm with RB-PHD SLAM.

5.1 Problem Formulation

In this work, we consider a mobile sensor, or vehicle, whose state evolves according to a particular motion model such as constant velocity or coordinated turn. From these motion models, a probability density $\pi(\mathbf{X}_{k|k-1}|\mathbf{X}_{k-1}, \mathbf{u}_k)$, may be derived which describes the probability of transitioning from a previous state $\mathbf{X}_{k-1}$ to a current state $\mathbf{X}_{k|k-1}$, where $\mathbf{u}_k$ is the control input for the current timestep, which may or may not be present depending on the choice of motion model.

Furthermore, we assume that the environment is composed of a set of map features $\mathbf{M}$. The map is not necessarily completely static, so likewise its state evolves according to $\pi(\mathbf{M}_{k|k-1}|\mathbf{M}_{k-1})$. As the sensor explores the environment, it receives measurements $\mathbf{Z}_k$ relating the map features to the vehicle position through a possibly non-linear function $h(\mathbf{m}, \mathbf{X})$. The measurement

process is non-ideal, so that in addition to measurements being subject to additive noise, some map features may fail to be sensed (missed detections), and measurements may be received which do not correspond to any map feature (false alarms, or clutter).

We represent the vehicle state as a single random vector in the vehicle state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, and each map feature as a single random vector in the feature space $\mathcal{M} \subseteq \mathbb{R}^{n_m}$.

$$\mathbf{x}_k = [x_{k,1}, \ldots, x_{k,n_x}] \in \mathcal{X} \tag{5.1}$$

$$\mathbf{m}_{k,n} = [m_{k,n,1}, \ldots, m_{k,n,n_m}] \in \mathcal{M} \tag{5.2}$$

While it is certain that there is and always will be only one vehicle, no such certainty applies to the number of map features. Hence the map state is what we call an RFS on $\mathcal{M}$:

$$\mathbf{M}_k = \{\mathbf{m}_{k,1}, \ldots, \mathbf{m}_{k,n_k}\} \in \mathcal{F}(\mathcal{M}) \tag{5.3}$$

where $\mathcal{F}(\mathcal{M})$ denotes the set of all finite subsets of $\mathcal{M}$. The sensor measurements for each time step are modeled as a union of two independent RFS's on the measurement space $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$: one representing measurements generated by map features, and the other representing the clutter measurements:

$$\mathbf{Z}_k = Z(\mathbf{x}, \mathbf{M}) \cup C \in \mathcal{F}(\mathcal{Z}) \tag{5.4}$$

The goal in SLAM is to estimate the joint posterior distribution of the vehicle and map states given a history of control inputs and measurements: $p_k(\mathbb{X}_k) = p_k(\mathbf{X}_k, \mathbf{M}_k | \mathbf{u}_{1:k}, \mathbf{Z}_{1:k})$. This is accomplished through the prediction and update recursion that is ubiquitous throughout many Bayesian estimation applications, including SLAM. The predicted joint state is given by the Chapman-Kolmogorov equation:

$$p_{k|k-1}(\mathbb{X}_{k|k-1} | \mathbf{u}_{1:k}, \mathbf{Z}_{1:k-1}) = \int \pi(\mathbb{X}_{k|k-1} | \mathbb{X}_{k-1}, \mathbf{u}_k) \, \mathrm{d}\mathbb{X}_{k-1} \tag{5.5}$$

The updated joint state is the result of applying Bayes' rule:

$$p_k(\mathbb{X}_k|\mathbf{Z}_{1:k}) = \frac{g_k(\mathbf{Z}_k|\mathbb{X}_{k|k-1})p_{k|k-1}(\mathbb{X}_{k|k-1}|\mathbf{u}_{1:k}, \mathbf{Z}_{1:k-1})}{\int g_k(\mathbf{Z}_k|\mathbb{X}_{k|k-1})p_{k|k-1}(\mathbb{X}_{k|k-1}|\mathbf{u}_{1:k}, \mathbf{Z}_{1:k-1})\,\delta\mathbb{X}_{k|k-1}} \tag{5.6}$$

For all but the most trivial of cases, the above is computationally impractical. Because the joint state is defined over the space $\mathcal{X} \times \mathcal{F}(\mathcal{M})$, the integrals become set integrals [54], which are taken over all sets of all possible cardinalities. Even specifying and storing a probability distribution over such a space becomes a daunting task. These difficulties will be overcome in the following manner:

1. The SLAM scenario will be modeled as a cluster process, allowing us to separate the problem into estimation of a relatively simple parent process for the vehicle, and a conditional daughter process for the map.

2. Instead of propagating $p_k(\mathbb{X}_k)$, we propagate its first moment $D_k(\mathbf{x}, \mathbf{m})$. The first moment takes values over the single-object space $\mathcal{X} \times \mathcal{M}$ and is significantly easier to handle than a full multi-object probabilility distribution.

Cluster processes, illustrated in Figure 5.1, describe a hierarchical arrangement of point processes where the realization of a *daughter* process is conditioned on the realization of some *parent* process [88]. For modeling SLAM, we consider a special subset of cluster processes where the cardinality of the parent process is exactly equal to one. Such processes are termed *single cluster processes*. The map is represented by a daughter point process over the feature space whose realization is conditioned on the realization of the vehicle point process, which consists of a single individual in the vehicle state space. In this work, both the map and false positive measurements are considered to be realizations of Poisson point processes. Moreover, we assume that the introduction of newly-observed map features is statistically independent of the dynamics of existing map features. This allows us to apply the SC-PHD filter that was described in Chapter 4.

At first, cluster processes may appear to be incongruent to the SLAM problem. Indeed, the term "cluster process" evokes an image where daughter events are spontaneously generated by instances of the the parent process. This contrasts with the usual conceptualization of SLAM where environment features exist in perpetuity waiting to be "discovered" by the vehicle. To a vehicle with no *a priori* knowledge of its environment, this distinction is not important. At the risk of sounding somewhat existential, we may treat a feature as having come into existence the moment it is first sensed, as it is from that point onward that it becomes relevant to the operational decisions of the vehicle. Having overcome this conceptual hurdle, we find that the conditional relationship between observation of features and the trajectory of the vehicle satisfies the requirements of applying a cluster process model. Furthermore, by employing a model in which features may be born and die off, we create a SLAM framework that can cope with a dynamically changing environment.

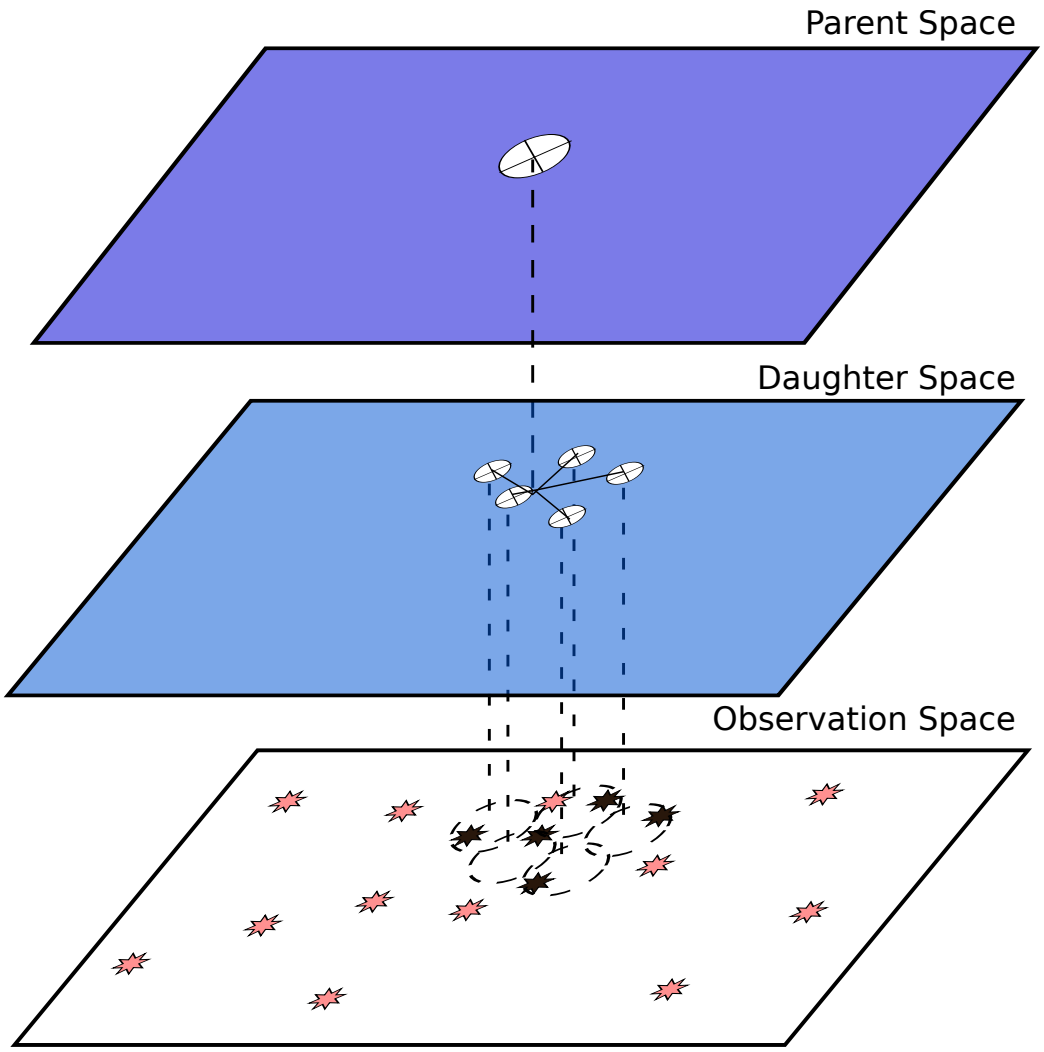


Figure 5.1: Visualization of a cluster process. The daughter process is conditioned on realizations of a separate parent process. Observations are collected only on the daughter process, and are mixed in with false alarms.

5.2 Implementing the SC-PHD Filter for SLAM

Thus far, our discussion of the PHD and SC-PHD filter has been couched in mostly symbolic terms. We will now turn our attention to how the SC-PHD filter is implemented numerically for application towards SLAM.

In implementing the SC-PHD filter, we couple a particle representation for the parent PHD with a Gaussian mixture representation for the PHD of the daughter state. To reflect the conditional relationship between the parent and daughter processes, each particle in the parent PHD is associated with its own Gaussian Mixture (GM) representing the daughter PHD conditioned on that particle's trajectory.

$$s_{k-1}(\mathbf{x}) = \sum_{i=1}^{N_{k-1}} \zeta_{k-1}^{(i)} \delta(\mathbf{x} - \mathbf{x}_{k-1}^{(i)}) \tag{5.7}$$

$$D_{k-1}^{(i)}(\mathbf{y}|\mathbf{x}_{k-1}) = \sum_{j=1}^{J_{k-1}^{(i)}} w^{(i,j)} \mathcal{N}(\mathbf{y}; \mu_{k-1}^{(i,j)}, \mathbf{P}_{k-1}^{(i,j)}) \tag{5.8}$$

Note that because the PHD is not a true probability distribution, the weights $w^{(i,1)}, \cdots, w^{(i,j)}$ do not necessarily sum to 1. Rather, the sum of these weights is the estimated number of map features. The particle representation was chosen for its ability to capture non-linearities in vehicle motion, while the Gaussian mixture representation was chosen for its computational economy and was deemed suitable under the assumption of relatively simple dynamics for objects in the daughter process. Each particle's conditional daughter PHD is propagated with a Gaussian Mixture PHD (GM-PHD) filter [96], amended with the measurement-driven birth strategy described in [43]. The GM-PHD filter relies on the further assumptions that the daughter process motion model and measurement models are linear Gaussian, or may be reasonably linearized as such, with matrices $\mathbf{F}$ and $\mathbf{H}$, and that the noise affecting these models is zero-mean Gaussian-distributed, with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$.

5.2.1 Measurement-driven Birth

In the general discussion of both the PHD and SC-PHD filters in Chapter 4, no constraints were placed on the RFS describing birth objects. Some works in the PHD filter literature, e.g. [97], sample the birth objects from a random point process. In reality, we have at our disposal a resource which allows us the construct the birth distribution in a more informed manner: the measurements from the current time step $\mathbf{Z}_k$. Even considering the presence of false alarms, some of the collected measurements can originate from previously unobserved map features and thus it is reasonable to derive the birth RFS from $\mathbf{Z}_k$. Let the PHD for the birth RFS at time k be defined by the following Gaussian mixture:

$$\gamma_k(\mathbf{m}|\mathbf{x}) = \sum_{\mathbf{z}\in\mathbf{Z}_k} w_b \mathcal{N}(\mathbf{y}; h^{-1}(\mathbf{z},\mathbf{x}), \mathbf{R}^*) \tag{5.9}$$

$$\mathbf{R}^* = \mathbf{H}^*\mathbf{R}\mathbf{H}^{*T} \tag{5.10}$$

Where $h^{-1}(\mathbf{z},\mathbf{x})$ is the inverse measurement function relating a measurement and vehicle state to a map feature location, and $\mathbf{H}^*$ is its linearization with respect to the measurement. The weight of the birth features w_b is typically tuned from the expected number of newly observed features each time step. In addition to these changes, the creation of the birth features to the daughter PHD is delayed until the update step. The reasons for this are twofold:

1. The measurements are already being used during the update step, and are otherwise not accessed during the prediction. Hence, it is more computationally expedient to create the birth features during the update;

2. As the birth features are created directly on top of the current measurements, they will be given disproportianate importance if they are passed through the PHD update normally, and so require special handling during the update. This will be elaborated upon in Section 5.2.5

It has been shown in that this strategy can provide superior performance when applied to the GM-PHD filter, as well as the SMC-PHD filter [43,

80]. However, it should be noted that this strategy can only be applied in scenarios where an inverse exists for the measurement model. For some sensor configurations, most notably bearing-only sensors, this is not true. In such cases the traditional measurement-agnostic birth strategy should be employed.

5.2.2 Prediction

Substitution of equations (5.7) and (5.8) into the SC-PHD prediction equation (4.47), and applying the sampling property of the Dirac delta function yields the following result:

$$D_{k|k-1}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_{k-1}} \zeta_{k-1}^{(i)} \pi_{k|k-1}(\mathbf{x}|\mathbf{x}_{k-1}^{(i)}) \tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x}_{k-1}^{(i)}) \tag{5.11}$$

Next, the Markov transition density of the parent is approximated by drawing M samples from it:

$$\{\mathbf{x}^{(i,1)}, \cdots, \mathbf{x}^{(i,M)}\} \sim \pi_{k|k-1}(\mathbf{x}|\mathbf{x}_{k-1}^{(i)}) \tag{5.12}$$

$$D_{k|k-1}(\mathbf{X},\mathbf{M}) \approx \sum_{i=1}^{N_{k-1}} \sum_{j=1}^{M} \frac{\zeta^{(i)}}{M} \delta(\mathbf{x}-\mathbf{x}^{(i,j)}) \tilde{D}_{k|k-1}(\mathbf{y}|\mathbf{x}_{k-1}^{(i)}) \tag{5.13}$$

For each parent particle from the prior, its associated conditional daughter PHD is replicated M times, and a prediction is generated for each replicate. With measurement-driven births, the incorporation of new targets into the daughter PHD is delayed until the update step. Therefore, the predicted daughter PHD describes only persisting daughter features propagated forward in time:

$$\tilde{D}_{S,k|k-1}(\mathbf{y}|\mathbf{x}_{k-1}^{(i)}) = p_S \sum_{j=1}^{J_{S,k|k-1}} w_{S,k|k-1}^{(j)} \mathcal{N}(\mathbf{y}; \mu_{S,k|k-1}^{(j)}, \mathbf{P}_{S,k|k-1}^{(j)}|\mathbf{x}) \tag{5.14}$$

$$w_{S,k|k-1}^{(j)} = w_{k-1}^{(j)} \qquad \mu_{S,k|k-1}^{(j)} = f(\mu_{k-1}^{(j)})$$

$$\mathbf{P}_{S,k|k-1}^{(j)} = \mathbf{F}_k \mathbf{P}_{k-1}^{(j)} \mathbf{F}_k^T + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^T$$

where $f(\mu)$ is the (possibly non-linear) motion model describing the motion of daughter features and $\mathbf{F}$ and $\mathbf{W}$ are the Jacobians of the motion model with respect to the daughter feature and process noise respectively. It should be noted that for static daughter features, the motion model is simply the identity, and so the prediction only needs to be performed for dynamic daughter features. The predicted joint PHD now consists of $N_{k|k-1} = M \times N_{k-1}$ particles corresponding to the parent state, and the same number of Gaussian mixtures representing the conditional daughter PHDs. Note that this means the number of particles grows geometrically with every prediction, but when the particles are resampled, we only draw a number of samples equal to the number of original particles, so that this growth does not continue unchecked.

5.2.3 Field of View

As the vehicle travels, the observable area of the feature space will change according to the position of the sensor. Let this area be $FOV(\mathbf{X}) \subseteq \mathcal{M}$. We consider map features within $FOV(\mathbf{X})$ to have a probability of detection equal to some nominal value p_D, and all other features to have a zero probability of detection.

$$p_D(\mathbf{y}|\mathbf{x}) = \begin{cases} p_D & \text{if } \mathbf{M} \in FOV(\mathbf{X}) \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

The PHD update in Eq. (4.28) may be thought of as the sum of a non-detection term and a detection term. When $p_D(\mathbf{y}|\mathbf{x}) = 0$ is substituted into (4.28), the equation reduces to only the non-detection term:

$$\tilde{D}_k(\mathbf{M}|\mathbf{X}) = \tilde{D}_{k|k-1}(\mathbf{M}|\mathbf{X})$$

This means that for the portion of the map that is outside of the vehicle's field of view, the updated PHD is the same as the predicted PHD. Therefore, the PHD update only needs to be performed on map features which are currently visible, which introduces an upper bound for the required computational

effort in this portion of the algorithm.
Clearly, this is a simplified model; for application in real-world systems, a feature's probability of detection could be affected by its range from the sensor, or by occlusion. The effects of mismatched detection probabilities should not be discounted. If a feature is assumed to be in the FOV when in reality it is not, its p_D will be overestimated. This means the non-detection term in Eq. (4.28) will be under-weighted, and because no measurement was received, the detection term will also have a very low weight. Depending on the state extraction scheme used, this may result in a map feature being incorrectly deleted. If the FOV proves too challenging to be parameterized, then it may be worthwhile to incorporate techniques for online estimation of the detection probability [56].

5.2.4 Dynamic Map Features

Traditionally, feature-based SLAM solutions operate under the assumption that the map consists only static features. There may be transient objects moving through the environment which are detected by the vehicle's sensors, but these measurements are excluded through some pre-processing step in order to fit the original assumption. We hypothesize that the SC-PHD SLAM framework is capable of discriminating between static and dynamic features, and of maintaining tracks for dynamic features, even when sensor measurements provide no information on the static or dynamic nature of the features. We can define a state space with both position and velocity components. For example in the Cartesian plane:

$$\mathbf{m} = [x, y, \dot{x}, \dot{y}] \subseteq \mathbb{R}^4$$

One might expect that given this state definition, we could model the feature evolution with some appropriate motion model, e.g. constant velocity or coordinated turn, and estimates for static features would coveniently converge towards points on the hyperplane $(\dot{x}, \dot{y}) = (0, 0)$. Unfortunately, this does not work as well as anticipated. The Markov transition densities formed from these motion models propagate the uncertainty in the velocity estimate as

uncertainty in the position estimate, so that even in the absence of process noise, the position uncertainty grows every time step during the prediction step unless the feature's velocity is known deterministically. This has serious implications for features outside of the field of view, whose position uncertainties will continue to grow until reobserved. Indeed, this approach is analogous to the tuning strategy for EKF-SLAM, where noise is injected into feature covariance matrices. A significant body of research has shown that this surely leads to inconsistent estimation [47, 95, 6, 44].

To address these issues, we instead represent the daughter process as the sum of two Poisson processes, one for static features, and one for dynamic features

$$\tilde{D}(\mathbf{M}|\mathbf{X}) = \tilde{D}_s(\mathbf{M}|\mathbf{X}) + \tilde{D}_d(\mathbf{M}|\mathbf{X}) \tag{5.16}$$

Because the sum of Poisson processes is still Poisson, the previously described equations are still applicable. We consider the positions for static features to reside in a different space than the positions of the dynamic features, so that $\mathbf{M}$ is now defined over the joint space $\mathcal{M} = \mathcal{M}_s \times \mathcal{M}_d$.

The prediction step will be performed separately for each process, according to (5.14), using their particular Markov transition density and survival probability.

5.2.5 Update

Each of the conditional daughter PHDs are updated individually when measurement inputs arrive. As mentioned previously, the incorporation of new targets into the PHD is postponed until the measurement update. Thus, the updated PHD can be considered as the sum of three terms: 1. Targets persisting from the previous time step which have failed to be detected (misdetection); 2. Targets persisting from the previous time step which have been detected (detection); and 3. Newly-appearing targets for the current timestep (birth). We assume that the static/dynamic nature of a daughter feature is not captured by the sensor, so for each measurement, we create two birth

terms: one in the static feature space and one in the dynamic feature space.

$$\tilde{D}_{k|k}(\mathbf{y}|\mathbf{x}) = (1-p_D)\sum_{j=1}^{J_{k|k-1}} w_{k|k-1}^{(j)}\mathcal{N}(\mathbf{y};\mu_{k|k-1}^{(j)},P_{k|k-1}^{(j)})$$
$$+\sum_{z\in\mathbf{Z}_k}\sum_{j=1}^{J_{k|k-1}} w_{k|k}^{(j)}\mathcal{N}(\mathbf{y};\mu_{k|k}^{(j)}(z),\mathbf{P}_{k|k}^{(j)}(z))$$
$$+\sum_{z\in\mathbf{Z}_k}\frac{w_0^{\gamma}}{\eta_z(\mathbf{y}|\mathbf{x})}(\mathcal{N}(\mathbf{y};z_s^*(\mathbf{x}),\mathbf{R}_s^*)+\mathcal{N}(\mathbf{y};z_d^*(\mathbf{x}),\mathbf{R}_d^*)) \tag{5.17}$$

$$w_{k|k}^{(j)} = w_{k|k-1}^{(j)}\frac{p_D g(z|\mathbf{y},\mathbf{x})w_{k|k-1}^{(j)}}{\eta_z(\mathbf{y}|\mathbf{x})} \tag{5.18}$$

$$\eta_z(\mathbf{y}|\mathbf{x}) = \kappa(z)+p_D\sum_{l=1}^{J_{k|k-1}} g(z|\mathbf{y},\mathbf{x})w_{k|k-1}^{(j)}+2w_0^{\gamma} \tag{5.19}$$

$$\mu_{k|k}^{(j)}(z) = \mu_{k|k-1}^{(j)}+\mathbf{K}(z-h(\mu_{k|k-1}^{(j)},\mathbf{x})) \tag{5.20}$$

$$\mathbf{P}_{k|k}^{(j)}(z) = (\mathbf{I}-\mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}^{(j)} \tag{5.21}$$

$$\mathbf{K}_k = \mathbf{P}\mathbf{H}_k^T\mathbf{S}_k^{-1} \qquad \mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T+\mathbf{R}_k \tag{5.22}$$

$z_s^*(\mathbf{x})$, $z_d^*(\mathbf{x})$, $\mathbf{R}_s^*$, $\mathbf{R}_d^*$ are the means and covariances for the measurement-driven birth terms, whose deriviation was discussed in Section 5.2.1. These feature states will have identical position components, and will differ only in the velocity components. The weights of the parent particles are updated using the multi-object measurement likelihood $L_{\mathbf{Z}_k}$.

$$\zeta_k^{(i)} = \frac{L_{\mathbf{Z}_k}(\mathbf{x}_{k|k-1}^{(i)})\zeta_{k|k-1}^{(i)}}{\sum_{l=0}^{N_{k|k-1}} L_{\mathbf{Z}_k}(\mathbf{x}_{k|k-1}^{(l)})\zeta_{k|k-1}^{(l)}} \tag{5.23}$$

$$L_{\mathbf{Z}_k}(\mathbf{x}_{k|k-1}^{(i)}) = \exp\{\sum_{j=0}^{J_{k|k-1}^{(i)}} w_{k|k-1}^{(j)}\}\prod_{z\in\mathbf{Z}_k}\eta_z(\mathbf{y}_{k|k-1}^{(i)}|\mathbf{x}_{k|k-1}^{(i)}) \tag{5.24}$$

Following each daughter update, the number of terms in the Gaussian mixture is equal to $J_{k|k} = (|\mathbf{Z}_k|+1)J_{k|k-1}+|\mathbf{Z}_k|$ To limit this exponential growth in mixture terms, a mixture reduction step is performed to eliminate terms with weights that are too low, and to merge similar terms. Specifically, we

employ the clustering algorithm described in [82]. The reduction procedure consists of pruning step, in which Gaussian terms are eliminated based on the criteria of minimum weight and maximum number of terms, followed by a mergingstep, in which Gaussians within a certain distance threshold are combined. Alternative reduction algorithms, such as Gaussian Mixture Reduction by Clustering [83] may be substituted here if higher fidelity is required.

5.2.6 Computational Complexity

The complexity of the SC-PHD SLAM algorithm is linear in terms of number of measurements m, number of map features n, and number of parent particles p. The prediction step is on the order $O(np)$, while the update is of the order $O(mnp)$. In fact, we have found that the largest performance bottleneck in our experiments comes from an externality: the Gaussian mixture reduction algorithm exhibits a complexity on the order of $O(n^3)$. We believe that continuing research in this area should prioritize alternative methods for this portion of the algorithm.

5.2.7 State Extraction

To obtain an estimate of the parent state, the weighted mean of the parent particles is used. The expected daughter PHD can be obtained by performing a weighted sum of all the Gaussian mixtures and then applying another Gaussian Mixture Reduction. However, because the reduction algorithm has a complexity of $O(n^3)$, this would scale poorly with the number of parent particles. For that reason, we simply select the PHD corresponding to the maximum-weighted particle. Empirical evidence has shown that the results do not differ significantly from the expected value map. Next, the Gaussian mixture weights are summed to give an estimate of the number of targets N_k. The N_k most highly-weighted terms are then taken to be the daughter estimate.

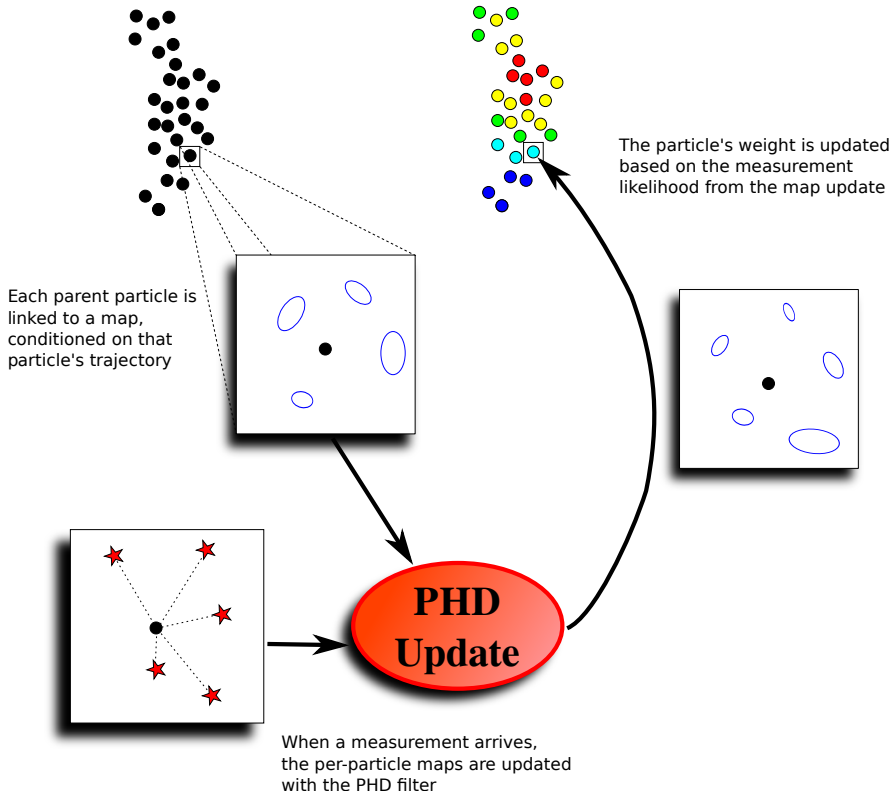


Figure 5.2: The single-cluster PHD SLAM process can be represented by the above hierarchy. Updates from landmark updates propagate upwards and reweight particles representing the vehicle position.

5.3 Simulation Results

The SC-PHD SLAM filter was verified via synthetic simulation in two scenarios: one to ascertain the functionality of mixed landmark estimation, and another to compare its performance against the previously proposed RB-PHD SLAM algorithm.

5.3.1 Dynamic landmarks

The first simulation was performed using the scenario depicted in Figure 5.3a. In this scenario, a vehicle explores a planar two-dimensional environment, and receives measurements generated by point features, several of which are themselves moving through the environment. The vehicle state is represented

by a three element vector consisting of absolute displacement in x and y, and orientation θ: $\mathbf{X}_k = [x_k, y_k, \theta_k]$. The dynamics of the vehicle adhere to the motion model described in [42, 69], where the next vehicle state depends on control inputs for velocity v_e taken from a wheel encoder, and steering angle α.

$$\begin{aligned}[x_k, y_k, \theta_k]^T = [x_{k-1}, y_{k-1}, \theta_{k-1}]^T \\ + \begin{bmatrix} \dot{x} + \dot{\theta}(a \sin\theta_{k-1} + b\cos\theta_{k-1}) \\ \dot{y} + \dot{\theta}(a\cos\theta_{k-1} - b\cos\theta_{k-1}) \\ \dot{\theta} \end{bmatrix} \Delta T \end{aligned} \tag{5.25}$$

$$\dot{x} = v_c \cos\theta_{k-1} \quad \dot{y} = v_c \sin\theta_{k-1} \quad \dot{\theta} = \frac{v_c}{L}\tan\alpha \tag{5.26}$$

where $a = 1.890$m, $b = 0.250$m, $H = 0.380$m, and $L = 1.415$m are parameters describing vehicle geometry and sensor placement, also

$$v_c = \frac{v_e}{1 - \tan\alpha\frac{H}{L}} \tag{5.27}$$

transforms wheel encoder velocity onto the vehicle centerline. The odometry inputs are reported with zero-mean additive Gaussian noise with standard deviations $\sigma_v = 1$ m/s and $\sigma_\alpha = 2^\circ$ for wheel velocity and steering angle respectively.

The measurements received by the vehicle from environment features consist of a range r and a relative bearing ϕ.

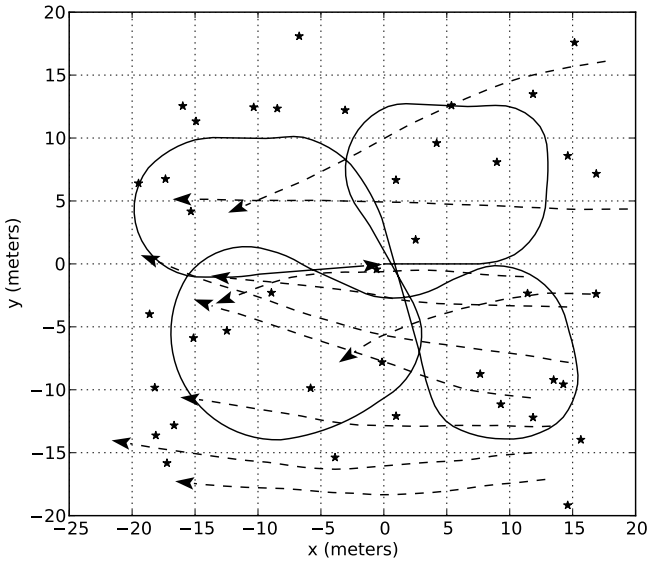
$$\begin{bmatrix} r_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} \sqrt{\Delta x^2 + \Delta y^2} \\ \arctan(\Delta y, \Delta x) \end{bmatrix} \tag{5.28}$$

The vehicle's sensor has a 15m range, and a 360° field of view. Measurements are subject to zero-mean additive Gaussian noise, with standard deviations $\sigma_r = 0.25$m and $\sigma_\phi = 0.5^\circ$ for range and bearing respectively. Moreover, false alarm measurements are received at a mean rate of $\lambda = 20$ per scan.

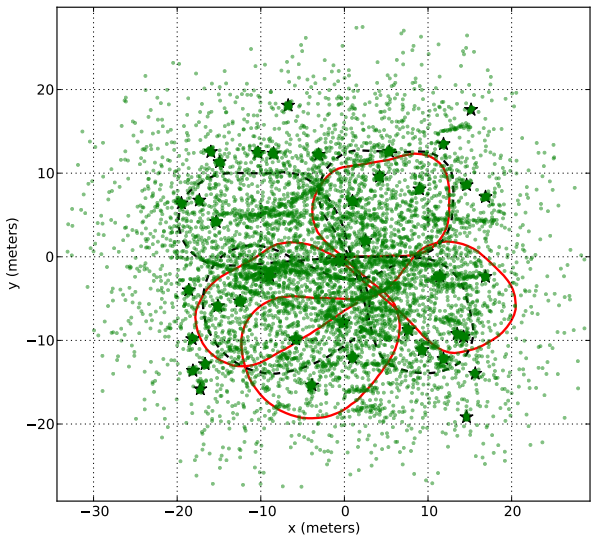
Odometry inputs arrive at a rate of 20 Hz, while measurements arrive at a rate of 10 Hz. The probability of detection was $p_D = 0.95$. Figure 5.3b gives an example of the dead-reckoning trajectory from the noisy odometry inputs, and cumulative sensor measurements which may be generated from these simulation parameters.

Monte Carlo (MC) simulations were performed with 25 different sets of odometry and measurement inputs. To account for variability in sampling the Markov transition during the prediction, 4 runs were performed for each set of inputs, for a total of 100 MC runs. The filter was run with a nominal particle count of $N_0 = 32$, and $M = 2$ samples were drawn from the transition density every prediction step. The threshold for particle resampling was set to $N_{eff} = 0.05$. Figure 5.4a shows an example of the resulting map and trajectory estimates. Dynamic feature tracks are shown in Figure 5.4b. It can be seen that the majority of false tracks are located where there are closely-spaced static map features, and where successive measurements originating from these features could be construed as originating from a single dynamic feature.

Figures 5.5a and 5.5b show the average error in vehicle pose and map. Mapping performance was quantified using the OSPA distance [84], a metric which takes into account differences in both localization and cardinality. For computing the map error, the ground truth was constructed in the following manner: 1) Static features are included in the set of true features from the time step when they are first observed until the end of the simulation; and 2) Dynamic features are included in the set of true features only while they are within the field of view; this results in the segmented ground truth tracks in Figure 5.4b. The results indicate that the SCPHD SLAM algorithm produces a definitively superior trajectory estimate compared to odometry alone, and that map estimation error is non-increasing for the duration of the simulation. For the sake of expeditious computation in our MC runs, we used a reduced-size scenario of approximately 300 time steps in length. However, in order to ascertain the algorithm's viability in more extended operations, we also ran a single simulation on a scenario 3500 time steps in length, with identical parameters. The results are shown in Figures 5.6, and show that

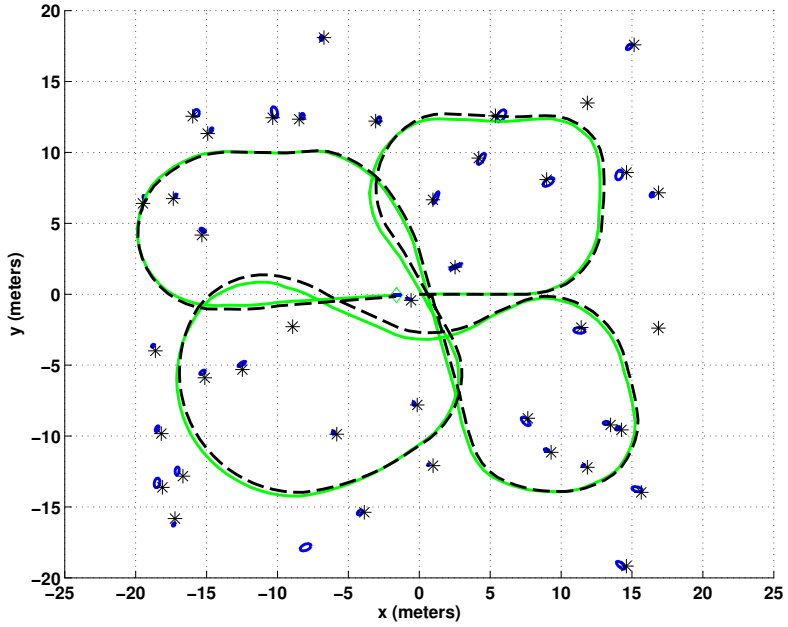


(a) Vehicle trajectory (solid line), static and dynamic map features (stars and dashed lines).

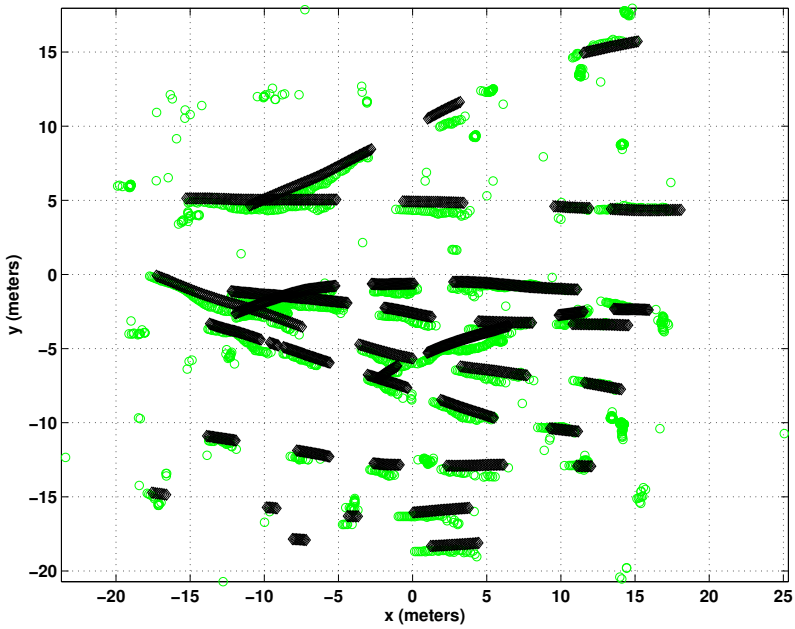


(b) Cumulative measurements and dead-reckoning trajectory (red, solid).

Figure 5.3: Simulation scenario

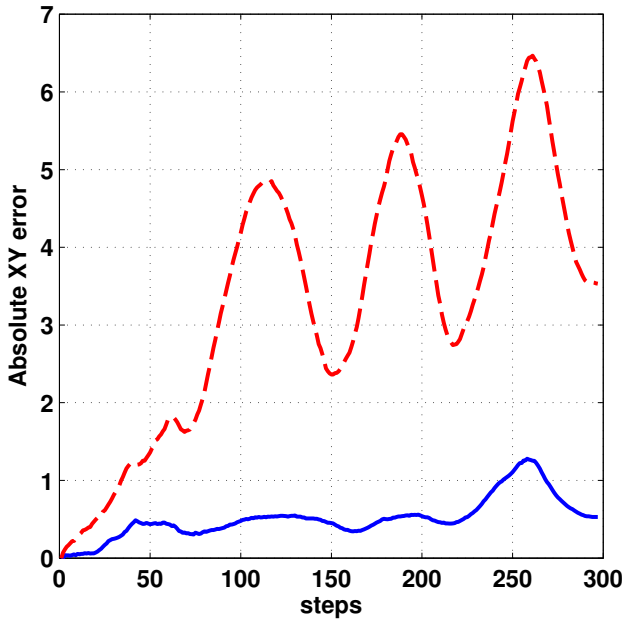


(a) Example map and trajectory estimate. Black stars and dashed line show true feature locations and vehicle trajectory.

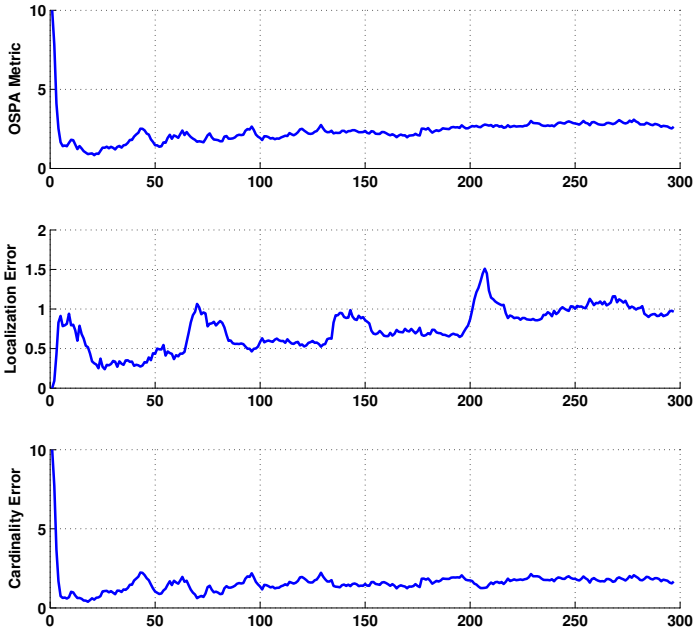


(b) Tracks of estimated dynamic targets (green), overlaid on true feature trajectories (black).

Figure 5.4: Simulation results: map and trajectory estimates



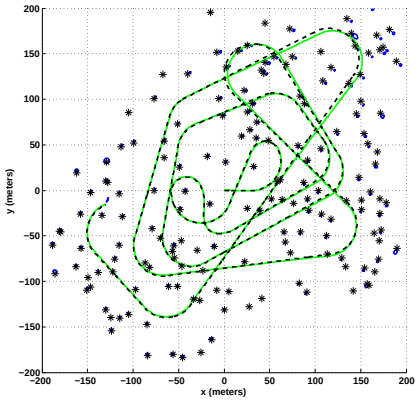
(a) Mean vehicle error over 100 MC runs (blue,solid) and dead-reckoning error (red, dashed).



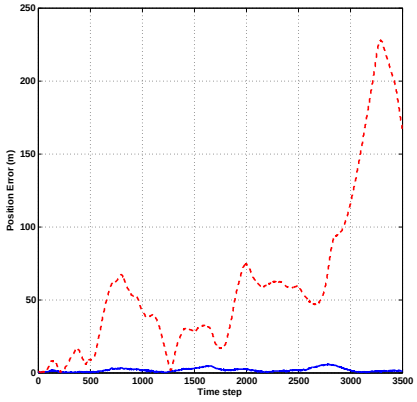
(b) Mean map error over 100 MC runs. A cutoff value of $c = 10$ was used in computing the OSPA metric.

Figure 5.5: Simulation results: error metrics

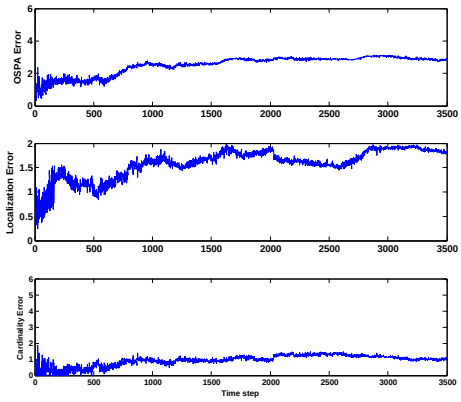
consistent estimation performance is sustained over longer running times. As each per-particle map is updated independently, the SC-PHD SLAM algorithm lends itself well to parallelization. Our implementation is based on the CUDA 4.0 parallel computing platform, and the simulations presented here were performed on an Nvidia Tesla C2070 GPU.



(a) Trajectory and map results. Black stars and dashed line show true feature locations and vehicle trajectory.



(b) Vehicle pose estimation error (blue,solid), compared to error from odometry alone (red,dashed).



(c) Map estimation OSPA error, with localization and cardinality components.

Figure 5.6: Results from 3500 timestep simulation of SC-PHD SLAM

5.3.2 Comparison with RB-PHD SLAM

In this set of experiments, we compare the performance our SC-PHD SLAM algorithm to that of a previous PHD filter SLAM algorithm: RB-PHD SLAM [64]. The motion and measurement models are identical to those used in the simulations in the previous section. The odometry noise is zero-mean Gaussian distributed, with standard deviations of $\sigma_v = 2$ m/s and $\sigma_\alpha = 5^\circ$ for the wheel encoder and steering angle respectively. Likewise, measurements are subject to additive zero-mean Gaussian distributed noise with standard deviations $\sigma_r = 1$m and $\sigma_b = 2^\circ$ in range and bearing respectively. Map features have a detection probability of $p_D = 0.95$ and false alarms arrive at a mean rate of $\lambda = 5$ per scan. Figure 5.7 illustrates the simulation scenario.

Using the same map feature layout and vehicle trajectory, 50 Monte Carlo runs were simulated, recreating odometry input and sensor measurements for each run. These inputs were then passed through both the SC-PHD and RB-PHD SLAM algorithms. The single-feature map approximation was used for RB-PHD SLAM. Figure 5.8 shows the pose and map errors averaged over the 50 MC runs.

These simulations show that the SC-PHD SLAM gives markedly better performance compared to RB-PHD SLAM. The most important difference between the two algorithms is the derivation of the multi-object likelihood $L_{\mathbf{Z}_k}(\mathbf{x})$, which is referred to as $g(\mathbf{Z}_k|\mathbf{X})$ in the RB-PHD SLAM literature. The RB-PHD SLAM likelihood is obtained by observing that it is equivalent to the normalization term in the daughter update. The Bayes' update for the conditional map is:

$$p_{k|k}(\mathbf{M}|\mathbf{X}) = \frac{g(\mathbf{Z}_k|\mathbf{M},\mathbf{X})p_{k|k-1}(\mathbf{M}|\mathbf{X})}{g(\mathbf{Z}_k|\mathbf{X})} \tag{5.29}$$

This can be re-arranged to become:

$$g(\mathbf{Z}_k|\mathbf{X}) = \frac{g(\mathbf{Z}_k|\mathbf{M},\mathbf{X})p_{k|k-1}(\mathbf{M}|\mathbf{X})}{p_{k|k}(\mathbf{M}|\mathbf{X})} \tag{5.30}$$

Because the $g(\mathbf{Z}_k|\mathbf{X})$ has no dependence on the map, any choice for the

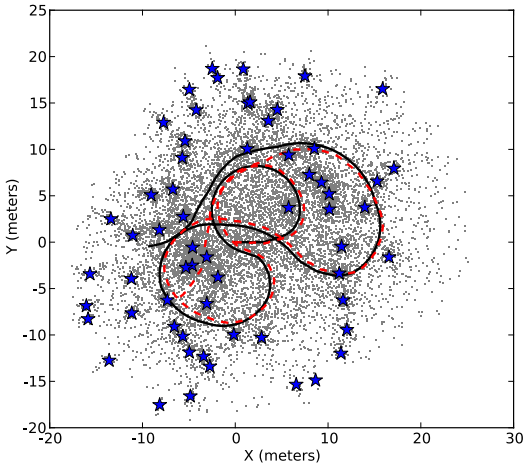
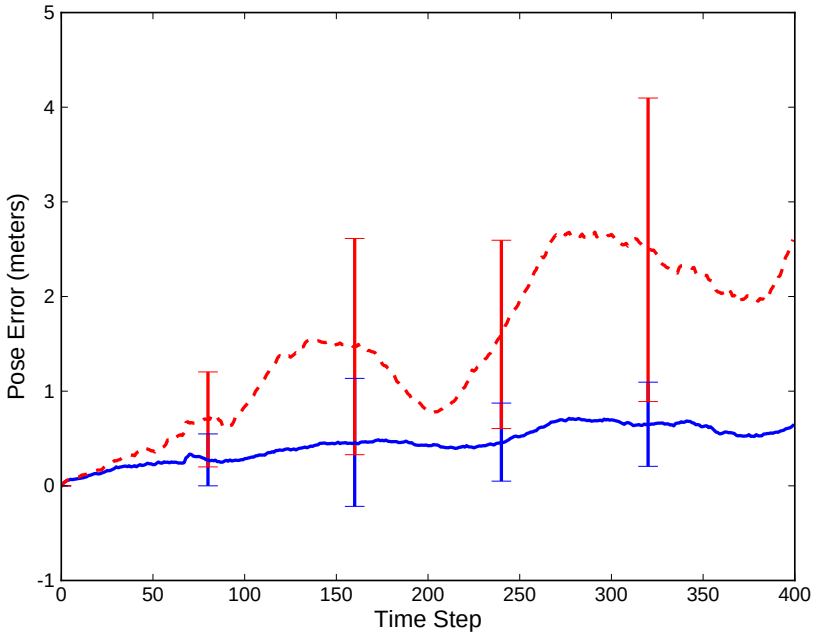


Figure 5.7: Illustration of the scenario used for RB-PHD comparison simulations, showing the true vehicle trajectory (black solid), dead-reckoning trajectory (red dashed), true map landmarks (blue stars), and cumulative sensor measurements (gray dots).

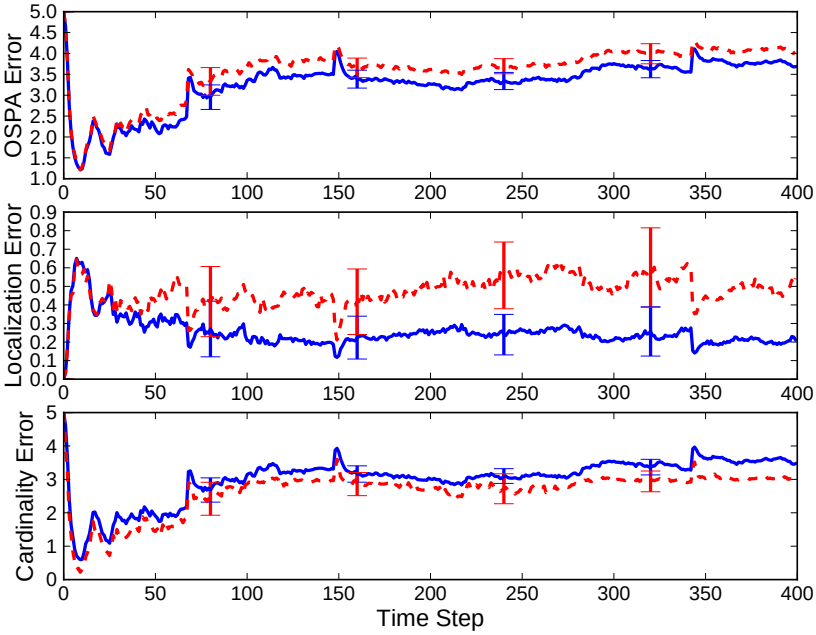
value of $\mathbf{M}$ will produce the same likelihood. Therefore, it is convenient to assume an empty map ($\tilde{\mathbf{M}} = \emptyset$) or a single-feature map ($\tilde{\mathbf{M}} = \{m\}$) for ease of computation. To obtain a closed-form expression for $g(\mathbf{Z}_k|\mathbf{X})$, the distributions $p_{k|k-1}$ and $p_{k|k}$ are assumed to be Poisson, and their first moments are evaluated at $\tilde{\mathbf{M}}$ rather than the true distribution.

The single-cluster PHD filter was derived with an approach more analogous to the classical PHD filter [57], where an assumption on the prior is used to derive a closed-form expression from the exact Bayes filter update. Specifically, we assume that the prior is the realization of a hierarchical Poisson process. The approach by Mullane, Vo and Adams required Poisson approximations on both the prior and posterior, and a further approximation on the number of features. Consequently, the single-cluster PHD filter more faithfully represents the true multi-object distribution since it requires fewer approximations in its derivation.

To further explore effect of this difference in practice, we carried out a small experiment. Using the same set of sensor inputs, the measurement updates for the two methods were executed on identical predicted states. Figure 5.9 shows the resulting parent distributions. It is apparent that the SC-PHD



(a) Vehicle Pose Error.

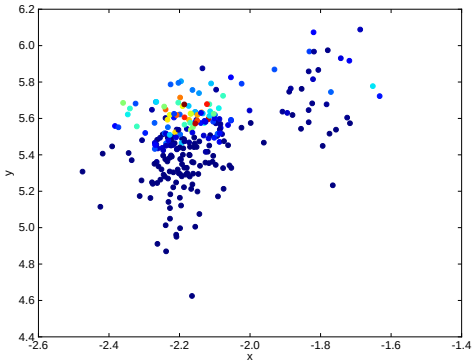


(b) Map OSPA Error, with localization and cardinality components.

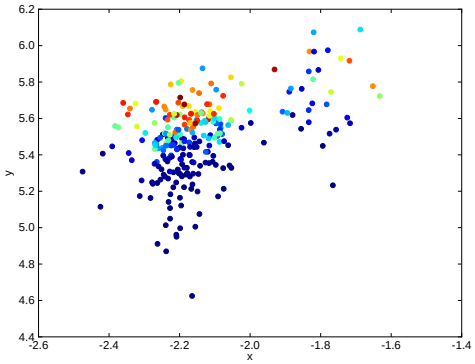
Figure 5.8: Monte Carlo simulation results, with $1-\sigma$ bounds indicated. SC-PHD SLAM: blue, solid. RB-PHD SLAM: red, dashed.

SLAM update generates a significantly more focused parent distribution than the RB-PHD SLAM update. This suggests that the multi-object likelihood used in the single-cluster derivation is more discriminating, and is better able to concentrate the parent particles about the true vehicle position.

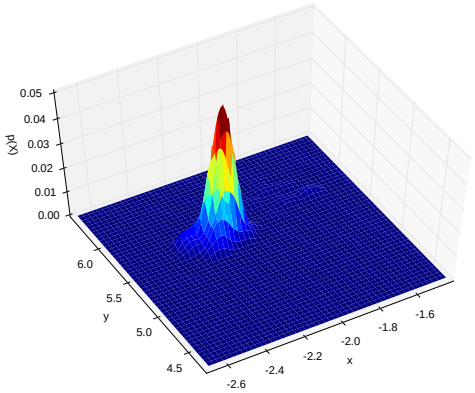
Lastly, we can draw a comparison based on computational complexity between the SC-PHD and RB-PHD SLAM filters. Most of the adaptations we have included in our algorithm, such as measurement-driven birth and prediction with multiple parent samples, are equally valid for the RB-PHD framework. Therefore, it will suffice to compare the relative complexities for the multi-object likelihood computation. Using the same notation as Subsection 5.2.6, we find that the likelihood computation in (5.24) has a complexity of $O(m+n)$. For RB-PHD SLAM, the complexity of the likelihood computation using both the empty map and single feature map approximations is $O(m)$. However, without applying approximations, the complexity returns to $O(m+n)$.



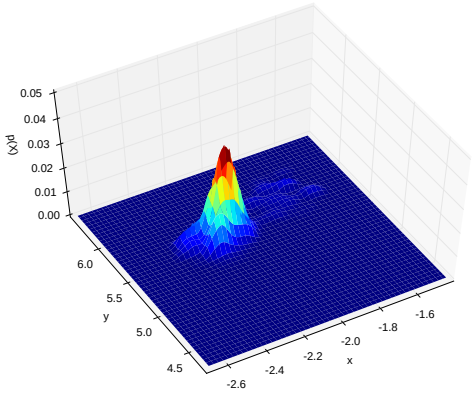
(a) SC-PHD SLAM particles .



(b) RB-PHD SLAM particles .



(c) SC-PHD SLAM smoothed distribution .



(d) RB-PHD SLAM smoothed distribution .

Figure 5.9: Comparison of updated vehicle distributions. Smoothing was performed using a Gaussian kernel with a bandwidth $h = 0.05$.

5.4 Discussion

The results presented in this chapter demonstrate that the SC-PHD filter is suitable to Simultaneous Localization and Mapping applications. We believe that its ability to cope with measurement clutter without the need for data association is an attractive property of the SC-PHD filter, and with further development, it could be successfully applied to real-world scenarioss. In addition, this framework can correctly discriminate and track dynamic map features mixed in with static features, even when the sensor measurements provide no information about the movement of the feature. Whereas in the past moving objects in the environment have been regarded as a nuisance for SLAM algorithms, we have shown that they can in fact be exploited for localization purposes.

Nevertheless, there is room for improvement. We anticipate that the ambiguity between dynamic map features and closely-spaced static features can be resolved by introducing more complex birth models for the dynamic features. Like other particle filter SLAM algorithms such as FastSLAM and RB-PHD SLAM, the SC-PHD SLAM filter has a tendency towards degeneracy in the vehicle state estimate, but this could be ameliorated through the use of techniques such as particle filter regularization. Furthermore, if the non-linearity of the vehicle motion is not significant, then Gaussian implementations for the parent filter are also possible. It may be worthwhile to explore alternative modelings of the SLAM process. The Poisson assumption on the map feature process imposes a high variance on its cardinality estimate, also the single-cluster process model only encodes dependencies between the map features and the vehicle, but not between map features. Recent contributions in Finite Set Statistics [20] will allow us model SLAM in the context of general hierarchical interacting population processes, where such inter-object relationships are taken into consideration.

Chapter 6

GPU Implementation of the SC-PHD Filter

Due to the combinatorial nature of the multi-object likelihood, single-threaded implementations of the SC-PHD filter can have very long running times: a MATLAB implementation can run for several hours to complete relatively simple simulations. For this reason, a multi-threaded implementation is critical for obtaining results in a timely fashion. Fortunately, the SC-PHD algorithm is quite amenable to parallel implementation. The maps for each parent particle are conditionally independent of each other, and so may be propagated entirely in parallel. We created a GPU implementation of the algorithm using the CUDA programming platform. The purpose of this chapter is to explain this implementation, and the design considerations that went into creating it.

6.1 The CUDA Architecture

CUDA (formerly Compute Unified Device Architecture) is a programming platform developed by NVIDIA for performing general purpose computing on the company's' graphics processing units (GPUs). The CUDA platform consists primarily of software libraries and tools such as compilers, linkers, and debuggers, as well as a software development kit and documentation li-

brary. In general, CUDA programs will contain a combination of instructions which are executed on the GPU, or *device code*, and instructions which are executed on the CPU, or *host code*. Host code can be written in full C/C++, while device code only supports C and a limited number of C++ features.

The heart of a CUDA application is a special C function called a *kernel*. This is the code that will be executed in parallel by all the GPU threads. As a consequence, every single thread receives the same set of instructions, and therefore, we require some means to differentiate the behavior of each thread whether by sending threads down different code paths, or by causing them to operate on different parts of the input and output data. Millions of threads performing the same operations on the same data could hardly be thought of as parallel computation! In order to see how threads may be coerced into behaving differently while following the same set of instructions, we must first consider how the threads are organized (Figure 6.1). Threads are first grouped into *blocks*, which are then arranged as a *grid*. Each thread has access to its position within its block through a special variable called `blockIdx`, its block's position within the grid (`gridIdx`), and the dimensions of the blocks (`blockDim`). From this information, each thread can determine its unique index:

$$\texttt{threadIdx} = \texttt{gridIdx} \times \texttt{blockDim} + \texttt{blockIdx}$$

This index can then be used to access different parts of the input and output data, or used in conditional statements for branching code paths. The block-grid hierarchy serves two purposes. First, it facilitates scalability to GPUs with different numbers of multiprocessor cores. The CUDA runtime automatically and transparently distributes the workload across the number of available cores at a block-level granularity. The second purpose is that threads are only able to share information with other threads in the same block, through a shared memory mechanism. This is a particularly important consideration for designing the SC-PHD update.

Setup and execution of a kernel function is performed by host-side operations. A barebones kernel launch consists of the following steps:

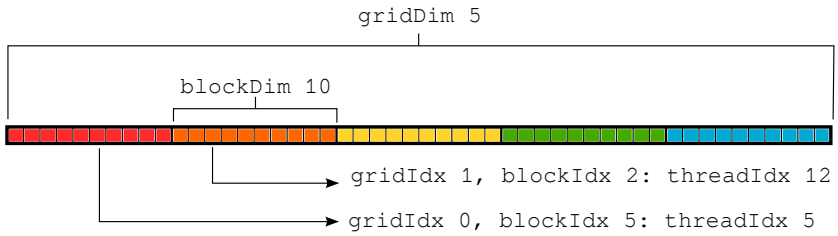


Figure 6.1: Example configuration of a grid of GPU threads. Here, the grid is a one-dimensional array, but two and three dimensional layouts may also be created.

1. Allocate GPU memory for storing inputs and outputs with `cudaMalloc`
2. Copy input data to global GPU memory with `cudaMemcpy`
3. Launch the kernel
4. Copy output data from GPU to host with `cudaMemcpy`
5. Free allocated GPU memory with `cudaFree`

The kernel launch is a modfied function call with syntax extensions for specifying the number of blocks, and the number of threads within each block. For example, to launch a kernel called `myKernel` with 100 blocks and 256 threads per block:

```
myKernel<<<100,256>>>(args...)
```

Scalar arguments to the kernel are passed by value, vector arguments must be passed pointers to *device* memory. Synchronization between the host and the device is handled transparently. The host program will continue to execute after launching the kernel until it encounters an instruction that depends on the output of the GPU operation. It will then block until the kernel as finished execution.

6.1.1 Performance Considerations

Specification sheets for contemporary GPUs quote computational capabilities in the neighborhood of 1 teraflop (trillion floating point operations per

Table 6.1: Compute Capability 2.0 specifications

Max grid dimensionality	3
Max x,y, or z dimension of grid	65535
Max block dimensionality	3
Max x or y dimension of block	1024
Max z dimension of block	64
Max number of threads per block	1024
Warp size	32
Max number of resident blocks per multiprocessor	8
Max number of resident warps per multiprocessor	48
Max number of resident threads per multiprocessor	1536
Number of 32-bit registers per multiprocessor	32K
Max amount of shared memory per multiprocessor	48 KB
Number of shared memory banks	32
Local memory per thread	512 KB
Constant memory size	64 KB

second). These figures represent an ideal case, where the GPU's entire complement of threads is fully engaged in computation. In practice, launching and executing a GPU kernel involves several forms of overhead which must be managed in order to approach specified performance. This discussion aims to cover only a few major design considerations. For a more complete treatement, we refer the reader to the CUDA C Best Practices Guide [72]. The various generations of GPU hardware are subject to different limitations, or *compute capability* in CUDA parlance. Our Tesla C2050 GPU is a compute capability version 2.0 device, the specifics of which are detailed in the Table 6.1

```
if (tIdx is odd) then
  | Do x
end
else
  | Do y
end
```

```
if (⌊tIdx/WARPSIZE⌋ is odd) then
  | Do x
end
else
  | Do y
end
```

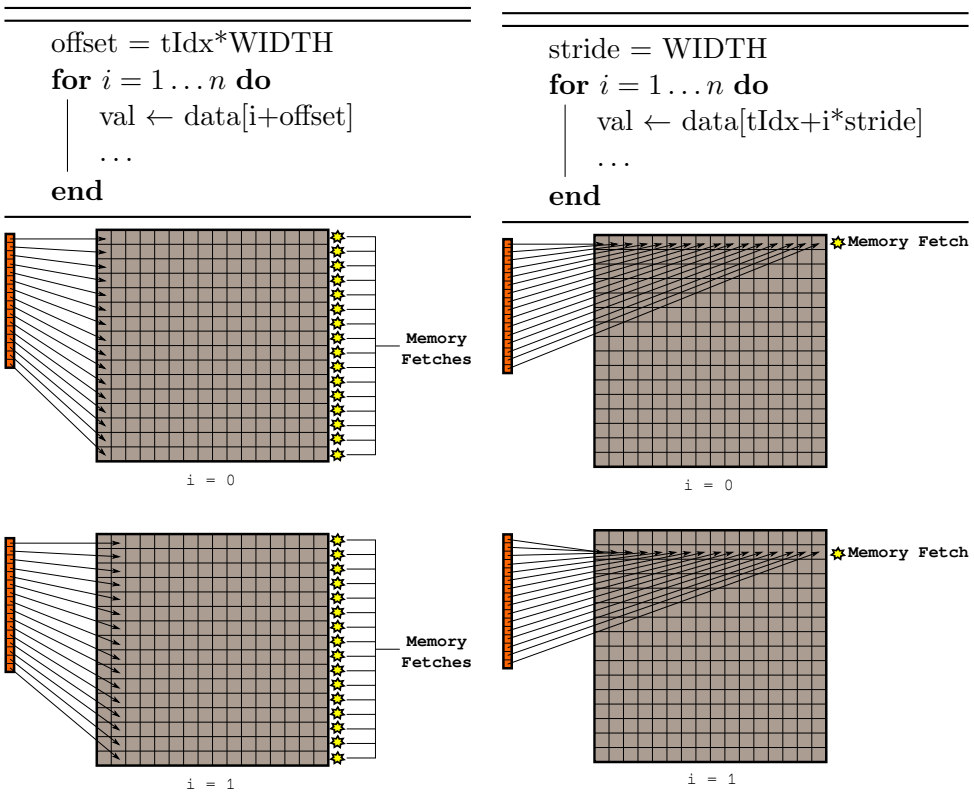
Figure 6.2: Both of these code listings cause half of the threads to perform an action x and the other half to perform y. The code on the left will cause divergence, as all the odd-indexed threads within a warp will perform x and all the even-indexed threads will perform y. In contrast, the code on the right does not produce divergence. The threads in each warp will all be performing either x or y.

Control Flow

Within each block resident on a GPU multiprocessor, the threads are scheduled and executed in groups of 32 consecutive threads called *warps*. The threads within the warp execute one instruction at a time. It is most efficient when every thread within the warp is performing the same instruction. If some threads are at a different location within the program, then the warp has diverged, and the different instructions must be executed sequentially. Divergence usually occurs as a result of conditional branching within the kernel code. Because the assignment of of threads to warps is completely deterministic, it is possible to re-implement these branches so that no warp divergence occurs, as shown in Figure 6.2.

Coalesced Global Memory Access

All but the most trivial GPU kernels operate on some input and output data. Accessing these data invokes some overhead costs that need to be minimized in order to optimize the performance of the kernel. The GPU global memory is divided into aligned 64-byte segments. When a thread accesses a particulary memory address, all the other memory addresses in the same segment are fetched at the same time. To maximize effective memory bandwidth we should ensure that when threads within a warp perform memory transac-



(a) Non-coalesced access pattern (b) Coalesced access pattern

Figure 6.3: Having each thread loop through its own contiguous section of memory (left) results in wasted memory bandwidth, as each thread requires its own fetch operation. Modifying the loop to the example on the right keeps memory accesses within the same segment each loop iteration, reducing the required number of fetches.

tions, they take place at addresses within the same segment. In situations where each thread must loop over several global memory locations, a strided access pattern is therefore recommended (Figure 6.3). It should be noted that newer generation GPUs are equipped with a modest amount on-chip L1 and L2 cache, which can help reduce the number of extra fetches. Nevertheless, designing the kernel with coalesced memory access in mind will ensure good performance regardless of the target hardware.

Shared Memory

Each block of threads has at its disposal an amount of on-chip memory called *shared memory*. This memory resides on the processor die, and therefore can be accessed much more quickly than global GPU memory, which is located off-chip on the GPU board. Moreover, shared memory accesses do not need to be coalesced. Because of this, the input data relevant to the thread block should first be copied to shared memory before further manipulation in order to reduce memory overhead.
Shared memory is divided into a number of banks (16 or 32 depending on the generation of GPU). Accesses to seperate banks are serviced in parallel, but if multiple threads in the same warp attempt to access different addresses within the same bank, a *bank conflict* will occur and the conflicting transactions will have to be serialized.

Block Sizes

To fully utilize GPU resources, all of the multiprocessors should be occupied. The computation tasks are allocated to multiprocessors on the basis of thread blocks, so to ensure full occupancy, the kernel should launch with a number of blocks greater than or equal to the number of multiprocessors on the device. Furthermore, each multiprocessor can actively run multiple blocks at once. While it waits for a warp in one block to finish executing, it will switch over to another warp in another block. Maximizing the number of these *resident* blocks has an important effect on computational performance. In our Tesla C2050,

6.2 Prediction

The SC-PHD prediction involves two kernel functions, which are launched sequentially. The first performs the prediction of the parent particles. Each thread is tasked with computing a single predicted parent particle state, which essentially consists of propagating the prior state through the vehicle motion model with added noise. Implementing a per-thread random number

generator would be prohibitively complicated, so the random noise samples are generated by the host, and passed to the kernel as input arguments.

Algorithm 1: Kernel function for parent particle prediction. Because the number of predicted particles can be a multiple of the number of prior particles, the indices for the prior and predicted arrays may be different.

Function `parentPredictKernel`(*priorParticlesArray,noiseArray, predictedParticlesArray*)

```
    idxPrior = ⌊tid/numParticlesPerPredict⌋
    oldState = priorParticlesArray[idxPrior]
    noise = noiseArray[tid]
    newState = computeMotion (oldState,noise)
    predictedParticlesArray[tid] = newState
end
```

The second kernel computes the predicted map features. Stationary features need not be predicted, so there is some host-side preprocessing done to first separate the dynamic features. Because the map features are Gaussians rather than particles, no random samples need to be generated for the prediction noise. In practice, the number of features may be larger than the total number GPU threads, so the kernel loops over the feature array with a stride size equal to the thread count. Because we have adopted the adaptive birth scheme, introduction of newborn targets is postponed until the update step.

Algorithm 2: Kernel function for map feature prediction.

Function `mapPredictKernel`(*priorFeaturesArray,predictedFeaturesArray*)

```
    for j = 0,nThreads,2nThreads,...,nFeatures do
        idx = tid + j
        predictedFeaturesArray[idx] = computeMotion
        (priorFeaturesArray[idx])
    end
end
```

6.3 Update

The SC-PHD update consists of updating the parent particle weights and the conditional daughter Gaussian mixtures. Because the number of parent particles is only on the order of several hundred, the reweighting and resampling is best implemented on the host side. On the other hand, updating the conditional daughters typically involves the creation of hundreds of thousands of Gaussian components, each consisting of a weight, mean, and covariance matrix. Thus, daughter PHD update comprises the bulk of the computational expense in the SC-PHD filter recursion. We first determine the amount of memory that will need to be allocated to perform the update. A predicted SC-PHD consists of K parent particles and their associated Gaussian mixture daughters. Each Gaussian mixture contains $J^{(k)}$ components. Performing an update with M measurements will result in an updated SC-PHD containing the following number of Gaussian components:

$$N_{predict} = \sum_{k=1}^{K} J^{(k)} \tag{6.1}$$

$$N_{update} = N_{predict} + M \times N_{predict} + M; \tag{6.2}$$

Where the three terms within the summation represent non-detection, detection, and measurement-driven birth respectively. Therefore, the update kernel must be allocated `Nupdate` $\times$ `sizeOf(Gaussian)` bytes for the output. In addition, `Npredict` $\times$ `sizeOf(Gaussian)` bytes are required for the predicted PHD and `M` $\times$ `sizeOfMeasurement` bytes are required for the measurements. We will store the measurements in constant memory because they will be accessed by all blocks of the kernel.

To determine the thread geometry for the update kernel, we must consider what operations we wish to perform at the thread level, and at what level the threads will need to communicate with each other. Each individual thread shall be tasked with the creation of a single Gaussian term in the updated PHD. However, as shown in Equation (5.19), computing the normalization term for each daughter PHD requires that the weights of all Gaussian terms

be summed. Thus, we will partition the kernel such that each thread block corresponds to a single parent particle and it's associated Gaussian mixture, and so that shared memory within each block may be used for computing the weight normalizer term. Because the predicted PHD is input as a single large array of all daughter Gaussian mixtures concatenated together, the kernel will require as an additional input an array of indexing offsets so that each thread block can find its corresponding Gaussian mixture. These offsets are computed on the host side prior to kernel launch.

The following pseudocode listing illustrates the general structure of the SC-PHD update kernel:

Algorithm 3: Abbreviated pseudo-code for the PHD update GPU kernel

```
Function PHDUpdateKernel(predictedGaussians,offsets,measurements)
    [idxIn,idxOut,nPredict,blockOffset] =
    computeIndices(offsets,threadIdx,blockIdx)
    predicted = predictedGaussians[idxIn]
    i = idxOut - blockOffset
    if i >nPredict then non-detection
        weight = predicted.weight
        updatedGaussians[idxOut] = predicted
    else if nPredict <= i <(nPredict*(1 + nMeasurements)) then
    detection
        m = floor((i - nPredict)/nPredict)
        updated = preUpdate(predicted,measurements[m]) weight =
        updated.weight
        updatedGaussians[idxOut] = updated
    else if i >= (nPredict*(1 + nMeasurements)) then birth
        m = i - (nPredict*(1 + nMeasurements))
        birth = computeBirth(measurements[m])
    end
    sharedmem[threadIdx] = weight
    normalizer = sum(sharedmem)
    updatedGaussians[idxOut] /=normalizer
end
```

6.4 Source Code

The source code for this GPU implementation of the SC-PHD filter is freely available under the Apache 2.0 license at the following address:

`https://github.com/cheesinglee/cuda-PHDSLAM`

Chapter 7

Underwater Vehicle Application of SC-PHD SLAM

This chapter discusses an application of the SC-PHD SLAM algorithm with a real-world dataset that was collected during an experiment at the Underwater Robotics Research Center in Girona, Spain. The purpose of this work is to demonstrate that the SC-PHD filter can be successfully applied toward solving real world problems in underwater robotics. The work presented in this article is to our knowledge, the first application of RFS techniques in underwater robotics.

7.1 SLAM for Underwater Vehicles

Underwater SLAM is an active and mature area of research, with nearly two decades' worth of contributions. Because satellite navigation like GPS or Gallileo is unavailable , and acoustic transponder systems like] (usbl) further complicate the already challenging logistics of deploying an AUV, autonomous navigation is more important to underwater vehicles than to terrestrial or aerial vehicles. A number of EKF-based SLAM algorithms have been demonstrated using both sonar [78, 71, 3] and camera sensors [99, 31]. Sonar-based navigation using least squares methods have also been proposed [35]. Underwater SLAM has advanced to the point that meaningful missions can

be executed by an AUV. A particle filter localization algorithm was employed in the autonomous survey of flooded sinkholes [34]. The ESDF was applied to autonomously survey the wreck of RMS Titanic [32]. Although PHD filter SLAM methods have been demonstrated for surface water platforms [64], underwater applications remain unexplored.

7.2 Implementing the Single Cluster PHD SLAM

To assess the suitability of the finite set techniques for underwater navigational applications, we developed an implementation of an SC-PHD SLAM algorithm for use on an underwater robotic vehicle, the Girona 500. In this section we will discuss a number of adaptations made to the algorithm, as well as the experiment performed to evaluate its performance.

7.2.1 The Girona 500 Vehicle

The Girona 500 (Figure 7.1) is an underwater robotic vehicle developed by Underwater Robotics Research Center (CIRS) at the University of Girona. It is a triple hull design, with each torpedo-shaped hull measuring 0.3 m in diameter, and 1.5 m in length. Its overall dimensions are 1.5m×1.0m×1.0m (L×W×H), and its weight is less than 200 kg. Its maximum operating depth is rated at 500 m. The triple hull design provides a high separation between the vehicle's centre of buoyancy and centre of gravity, which in turn affords stability in pitch and roll. The Girona 500's design is highly modular. It can be equipped with between 3 to 8 thrusters in various configurations to allow for motion in 3 to 6 redundant degrees of freedom. In addition to the typical navigational and survey sensors, it has space dedicated to mission-specific payloads such as robotic arms for intervention tasks.

7.2.2 Hybrid Particle PHD SLAM

As the single-cluster PHD can be separated into a parent and conditional daughter term, we adopt a hybrid particle and Gaussian mixture approach

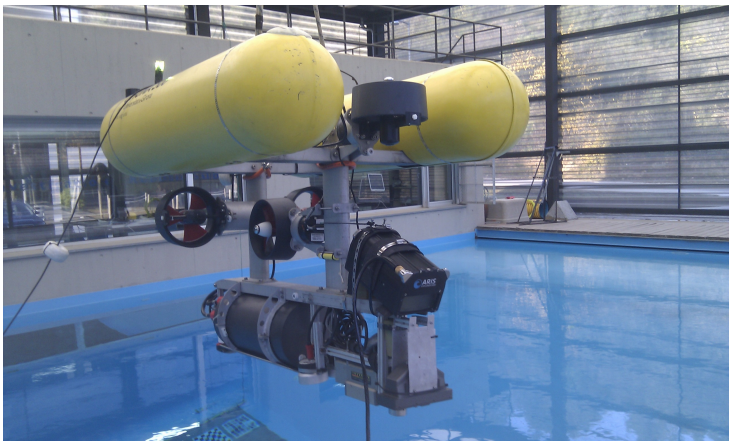


Figure 7.1: The Girona 500 Vehicle at CIRS.

for implementing the filter on our underwater vehicle. A particle vehicle state allows us to cope with a non-linear observation model, while a Gaussian mixture map state keeps the computational expenses in the map update manageable. The filter is updated using velocity measurements from the Doppler velocity log (DVL) and relative positional measurements from landmarks. The particles representing the vehicle state are propagated forward in time using a 3D constant velocity motion model.
The vehicle state is defined by a 12-dimensional vector consisting of cartesian and angular positions, and their respective velocities.

$$\mathbf{X} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}]^T$$

The cartesian displacements x, y, z are relative to the global reference frame, while the respective velocities are given in the vehicle reference frame.
Typically, this would require a particle state space consisting of positions and velocities in both linear coordinates and roll/pitch/yaw angles, for a total of 12-dimensions which would require an enormous number of particles. To mitigate the curse of dimensionality, we first treat the roll and pitch angles and velocities from the inertial measurement unit (IMU) as a trusted source, thus eliminating a third of the dimensions in the particle state space. However, the compass observations show significant inaccuracies and therefore yaw values must remain in the filter state. We continue even further by

implementing a pipelined approach for the linear velocities. We maintain a separate EKF to filter the vehicle-frame velocities based on a random-walk prediction, and linear measurements from the DVL. These filtered velocities are then transformed to the global reference frame and used to predict the vehicle particles with the constant velocity model. In this manner, we need only to maintain a particle state space of five dimensions: $(x, y, z, \theta, \dot{\theta})$. Using this pipelined approach leads to estimation in a lower dimension state-space with the particles, as well as lower variance on the estimated state [81]. Other options to improve the filter are the use of optimal sampling strategies [94] and multiple model filters [79] to better characterize different behaviours of the vehicle.

The reduced-dimensionality state is propagated forward through time using a constant velocity motion model:

$$\mathbf{X}_{k|k-1} = \begin{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{pmatrix} + \mathbf{R} \begin{pmatrix} \dot{x}_{k-1} \\ \dot{y}_{k-1} \\ \dot{z}_{k-1} \end{pmatrix} \Delta t \\ \theta_{k-1} + \dot{\theta}_{k-1}\Delta t + n_\theta \frac{\Delta t^2}{2} \\ \dot{\theta}_{k-1} + n_\theta \Delta t \end{pmatrix} \tag{7.1}$$

where $\mathbf{R}$ is the 3D rotation matrix computed from the three orientation angles, n_θ is a process noise parameter on the yaw, and Δt is the length of the time interval.

The vehicle particles represent the parent state in the context of the SC-PHD filter, and to each particle, we associate a Gaussian mixture which represents the map, or daughter PHD conditioned on that particle's trajectory. Updating the filter with landmark observations thus consists of performing the PHD filter update for each map [96], which is then propagated upwards to the parent process. This is reflected by the weight-update of the parent particles in the single cluster process. Figure 5.2 depicts this hierarchical update.

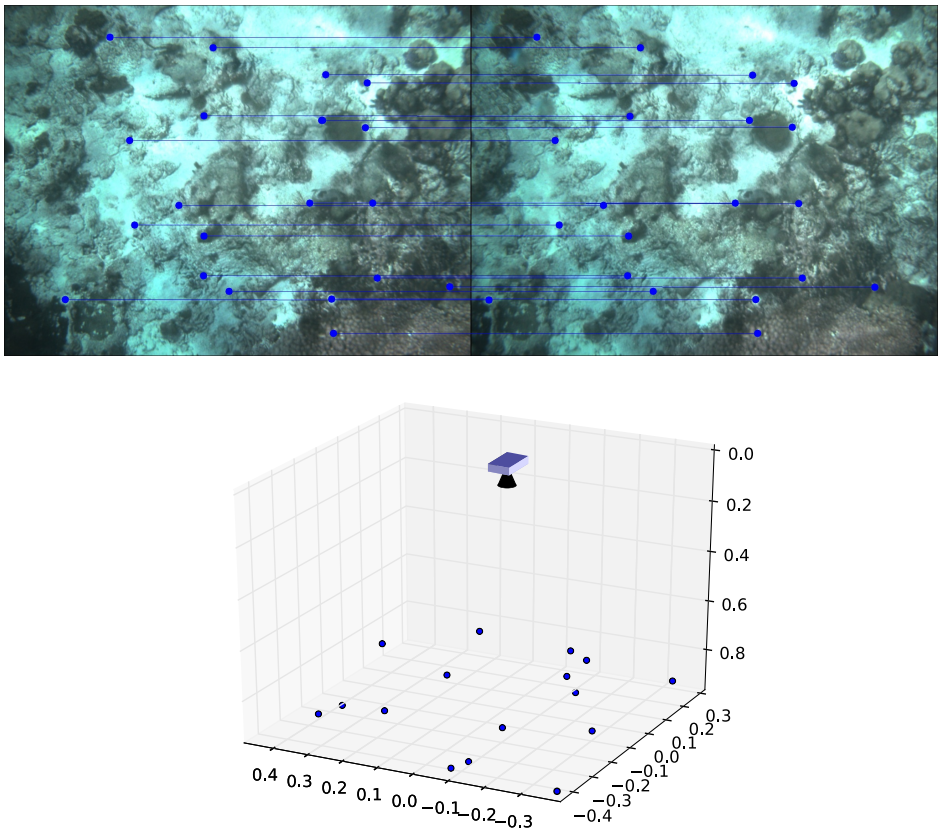


Figure 7.2: Keypoints are matched between the left and right images (top). Triangulation gives the 3D coordinates of the points with respect to the camera (bottom).

7.2.3 Detecting Features of Interest

In the absence of position measurements, the increase in uncertainty on vehicle position is unbounded. This can be reduced by incorporating measurements with respect to static targets under water. Typically, these targets will correspond to features in man-made structures such as risers, pipes, anchor chains, or naturally occuring rock formations and/or vegetation. In our experiments, we utilise a stereo camera to detect and triangulate such points of interest with respect to the camera. While our experiments here rely solely on vision-based features, sonar is also an important sensing regime for underwater robotics, although feature detection can be more problematic with sonar imagery compared to vision [45].

Points of interest in the sensor field of view (FoV) can correspond to physically meaningful structures with semantic meaning, or merely an abstract set of points. For example, an underwater riser in the camera FoV may represent a point of interest. In this case, we can attribute a semantic tag to the structure and possibly other identifying information. In contrast to this, a lower level analysis of the structure may decompose the riser into a set of corners, lines and/or circles, with each of these representing a point of interest. The use of semantic tags is only possible in the presence of prior information. The vehicle is unable to assign tags to unrecognizable landmarks, and in such instances, points of interest such as points and lines are more appropriate.

Our approach parameterizes features as points in $\mathbb{R}^3$, and involves the use of standard image feature detectors such as speeded up robust features (SURF) [8] to detect a small number of "strong" features or keypoints in the image. By detecting and matching keypoints in a pair of stereo images, we can approximately triangulate the position of keypoints in the camera coordinate system (Figure 7.2). Our measurements consist of pixel coordinates $[u_l, u_r, v_l, v_r]$ from pairs of corresponding SURF keypoints in the left and right camera images. Given a map feature $\mathbf{m} = [xyz]$ in 3D, its predicted measurement can be computed from the following model:

$$[u_l, v_l, 1]^T = g(\mathbf{m}, \mathbf{P}_l) \tag{7.2}$$

$$[u_r, v_r, 1]^T = g(\mathbf{m}, \mathbf{P}_r) \tag{7.3}$$

$$g(\mathbf{m}, \mathbf{P}) = \mathbf{P} \begin{bmatrix} \mathbf{R}(-\phi_k, -\psi_k, -\theta_k) & -\mathbf{T}_k \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ 1 \end{bmatrix} \tag{7.4}$$

Here, $\mathbf{P}_l$ and P_r are the 3×4 projection matrices for the left and right cameras, $\mathbf{R}(\phi, \psi, \theta)$ and $\mathbf{T}$ are the extrinsic rotation matrix and translation vectors respectively. To obtain a 3D feature location from a pair of corre-

spondences, we invert the measurement model:

$$\mathbf{m} = \begin{bmatrix} \mathbf{R}(\phi_k, \psi_k, \theta_k) & \mathbf{T}_k \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} g^{-1}([u_l, u_r, v_l, v_r], P_l, P_r) \tag{7.5}$$

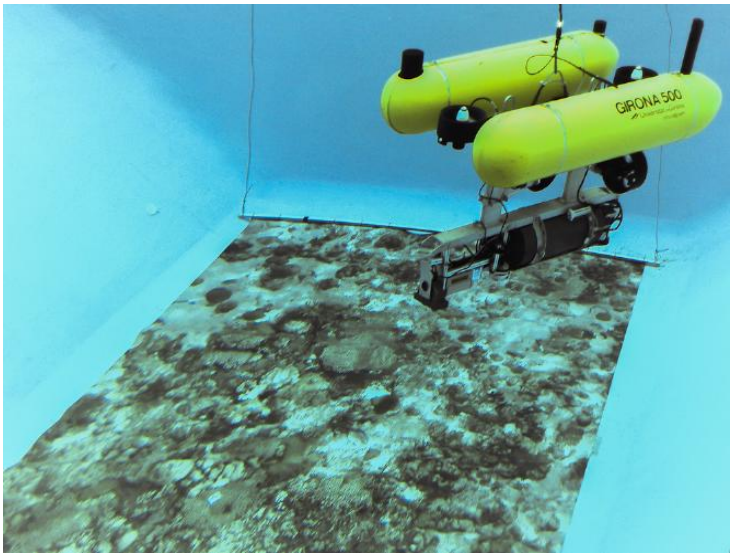
In practice, we use the OpenCV library function `triangulatePoints` to accomplish this.

Due to changes in illumination and contrast, the set of detected keypoints can change significantly between images. This gives rise to a situation with numerous unstable keypoints with poor probability of detection. The more stable keypoints will be tracked by the filter, while keypoints that are visible only briefly will be modelled by the clutter RFS in the PHD filter. In our model, we assume a uniform density of clutter measurements over the entire map, and a constant probability of detection $p_D < 1$ for features within the camera's view frustrum and $p_D = 0$ for features outside.

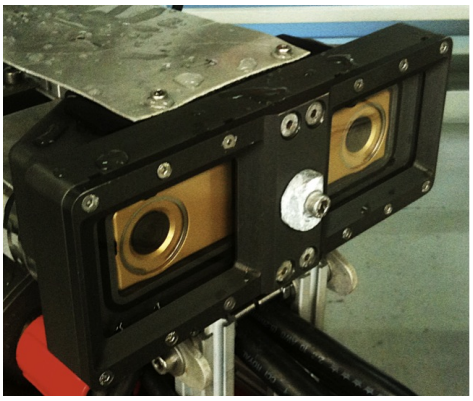
7.2.4 Underwater SC-PHD SLAM

To test the SC-PHD SLAM algorithm, we conducted an experiment in the test tank at CIRS, which measures 8m×8m×5m. The bottom of the test tank is overlaid with a known pattern simulating the sea-floor (see Figure 7.1), and the vehicle was teleoperated in a lawnmower survey trajectory over this pattern.

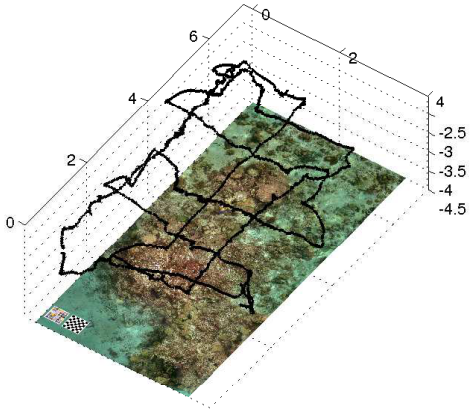
The vehicle is equipped with a downward looking stereo camera recording at approximately 10 frames per second. SURF features are extracted from the camera images to provide measurements for the SC-PHD SLAM algorithm. For the purpose of the filter update, the map features are projected onto the camera image plane, and the likelihoods are evaluated in the image domain. This allows for better characterisation of the measurement noise. Due to the fact that the appearance of the surface over which the vehicle is travelling is known *a priori*, it is also possible to obtain a ground-truth trajectory for this scenario by reprojecting the collected camera images onto the pattern. This is shown in Figure 7.3c.



(a) The Girona 500 hovering above the test tank floor. The known pattern of the floor is used to estimate the ground-truth trajectory of the vehicle.



(b) The housing for the Girona 500's stereo camera



(c) Camera-centered ground truth trajectory constructed by matching collected image with *a priori* seafloor pattern

Figure 7.3: Elements of the underwater SC-PHD SLAM experiment

Determining a ground truth for the map is far less straightforward. As we have the image file on hand for the map mosaic, it is possible to run a feature extractor on the image and predict where the most salient points of the environment would be. However, these features correspond to a top-down, whole-scene view under ideal lighting conditions. It is not certain that the features extracted with this approach would be consistently observed from the various viewing angles, depths, and illumination conditions over the course of the vehicle's trajectory. Therefore, we instead opt with a vehicle-oriented approach for locating the map features. Using the ground-truth trajectory described previously, we project the SURF features from each camera frame onto the mosaic and match the projected features against features detected in the mosaic. We divide the mosaic into grid and sum the successful matches within each grid cell to construct a 2D histogram showing the locations of consistently observed features. These histograms allow a qualitative comparison of the maps constructed by the SLAM algorithm (Figure 7.6).

The raw images from the camera exhibit a high degree of distortion (see Figure 7.4). Although corrected for, this distortion can manifest as errors in triangulation if camera calibration is inexact. We account for this in the experiment by assigning a relatively high level of additive noise to the feature measurements.

We used the sensor data from this experiment as inputs the SC-PHD SLAM filter. The RB-PHD SLAM filter was also run for comparison. The parameters used for both algorithms are listed in Table 7.1. Due to the limited computing capacity onboard the vehicle, the SLAM algorithms are run offline. Clearly, autonomy is of paramount importance for robotics applications. We anticipate that through a combination of algoirthm optimisation, improvements in available computing resources, and energy storage efficiency, real-time and online execution of SC-PHD SLAM is not too far in the future.

Figure 7.5 depicts the trajectory outputs of the SLAM algorithms compared to dead reckoning and the ground truth. Without SLAM, localization is performed using an EKF filter using velocity measurements from the DVL. It is clear that there is significant drift in the estimated vehicle position when

	Parameter	Value
Noise Standard Dev.	Yaw	0.1 rad
	IMU	0.2 rad/sec
	DVL	0.1 m/s
	Pixel	3.0 px
Particle Filter	Number of Particles	400
	N_{eff} resampling threshold	0.5
PHD Filter	Birth intensity	0.05
	Clutter intensity	20
	Probability of survival	1.0
	Probability of detection	0.85
GM reduction	Minimum pruning weight	0.0005
	Merging distance threshold	0.5
	Maximum number of Gaussian terms	12000

Table 7.1: SC-PHD SLAM parameters for underwater vehicle experiment.

only the DVL is used to perform dead reckoning. Aside from accumulated error in the velocity readings, the dead reckoning estimate also suffers from inaccurate heading measurements. As mentioned previously, the IMU values are trusted completely, but in reality the sensor is affected by soft-metal effects due to proximity to the test tank's concrete walls. This is evidenced by the significant deviation following the initial bend in the vehicle trajectory. In some portions of the trajectory, the IMU heading was off by up to 15 degrees. For operations in open sea settings, this effect should be far less pronounced, as the sensor will be well-separated from sources of interference. Introducing the seafloor pattern as a navigational reference improves matters. Compared to dead reckoning, both SLAM algorithms result in a better estimates of the trajectory. The multiple loop closures in the trajectory serve to keep the localization error bounded and eliminates the cumulative drift which otherwise occurs. A qualitative assessment suggests that the SC-PHD trajectory matches the ground truth better. This is confirmed through the examination of squared trajectory error shown in Figure 7.7.

The map state is taken from the map PHD belonging to the most highly-weighted particle. As we have opted for a Gaussian mixture implementation for the map PHD, we can obtain the expected number of features by summing weights of all the mixture components; this is equivalent to taking the integral of the PHD over the entire map. We then select the most highly-weighted mixture components as the locations of the map features. This is shown in Figure 7.6. Because our map "ground truth" is given as a histogram, a quantitative comparison is not given, since the optimal sub-pattern assignment (OSPA) metric is used for comparing point sets.

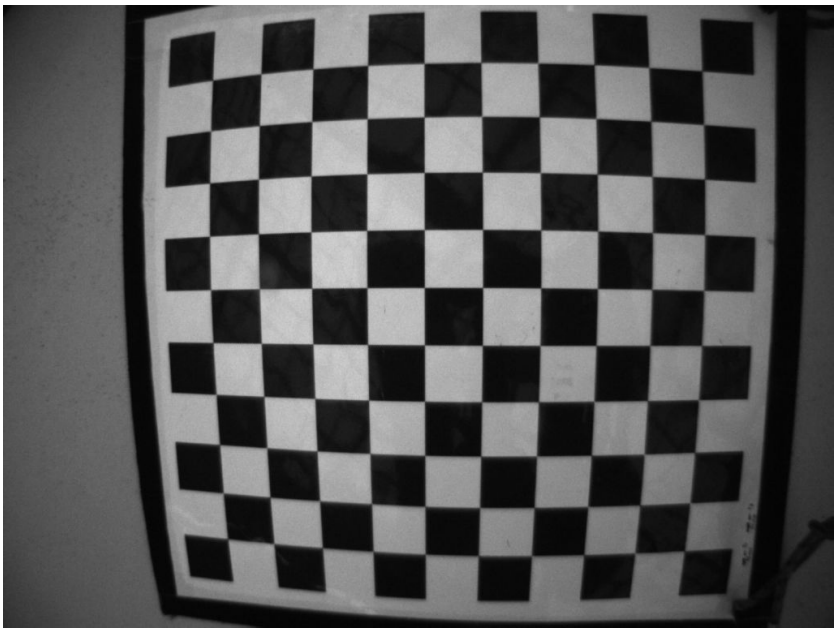
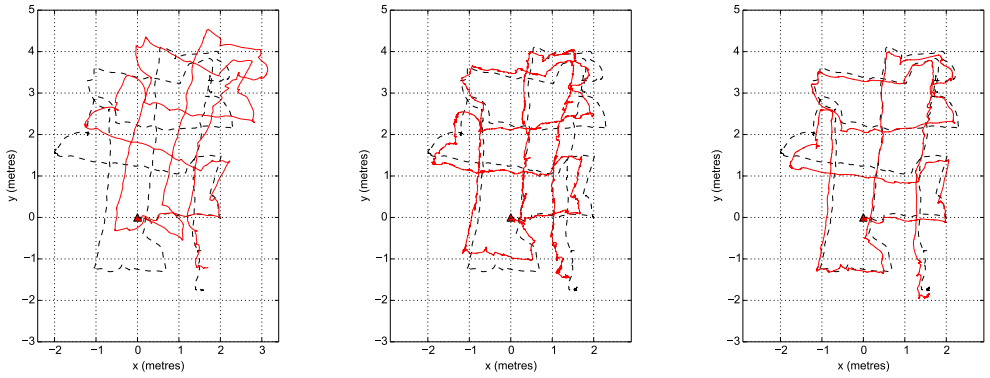


Figure 7.4: Distorted images from the downward looking stereo camera.

7.3 Outlook

We have seen here that PHD filter methods are suitable for application to the SLAM problem. There are of course, some refinements that can be made to our implementation. For example, the suboptimal performance of the IMU in the constricted test tank environment warrant a more conservative handling of its readings. Our research in this avenue will continue as we integrate the SC-PHD SLAM algorithm into a larger autonomous control framework, using its navigational estimates to inform higher-level tasks and missions.
The scenario presented here was in a high light environment where cameras may be used and a myriad of computer vision techniques can be brought to bear. Underwater vehicles operating at greater depths will be without this luxury and will need to rely on sonar, and feature detection will be even less straightforward. We expect that given the PHD filter's capability to excel in difficult sensing regimes, it can rise to meet these challenges, and we are looking forward to seeing many interesting applications in the underwater arena.

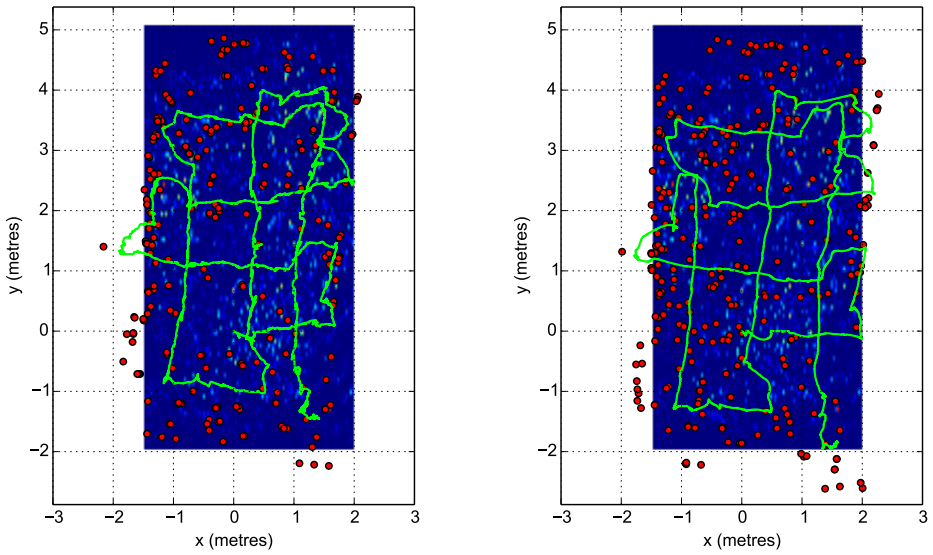


(a) Dead reckoning trajectory.

(b) RB-PHD SLAM trajectory.

(c) SCPHD SLAM trajectory.

Figure 7.5: Resulting trajectories from SLAM. Start of trajectory indicated by triangle. SC-PHD SLAM exhibits lower trajectory error compared to RB-PHD SLAM, while both SLAM methods are superior to odometry alone.



(a) RB-PHD SLAM landmarks.

(b) SCPHD SLAM landmarks.

Figure 7.6: Estimated landmarks for RB-PHD SLAM and SCPHD SLAM are overlaid on ground truth histogram described in Section 7.2.4. Landmarks outside the histogram area originate from features detected in the test tank environment, outside of the mosaic.

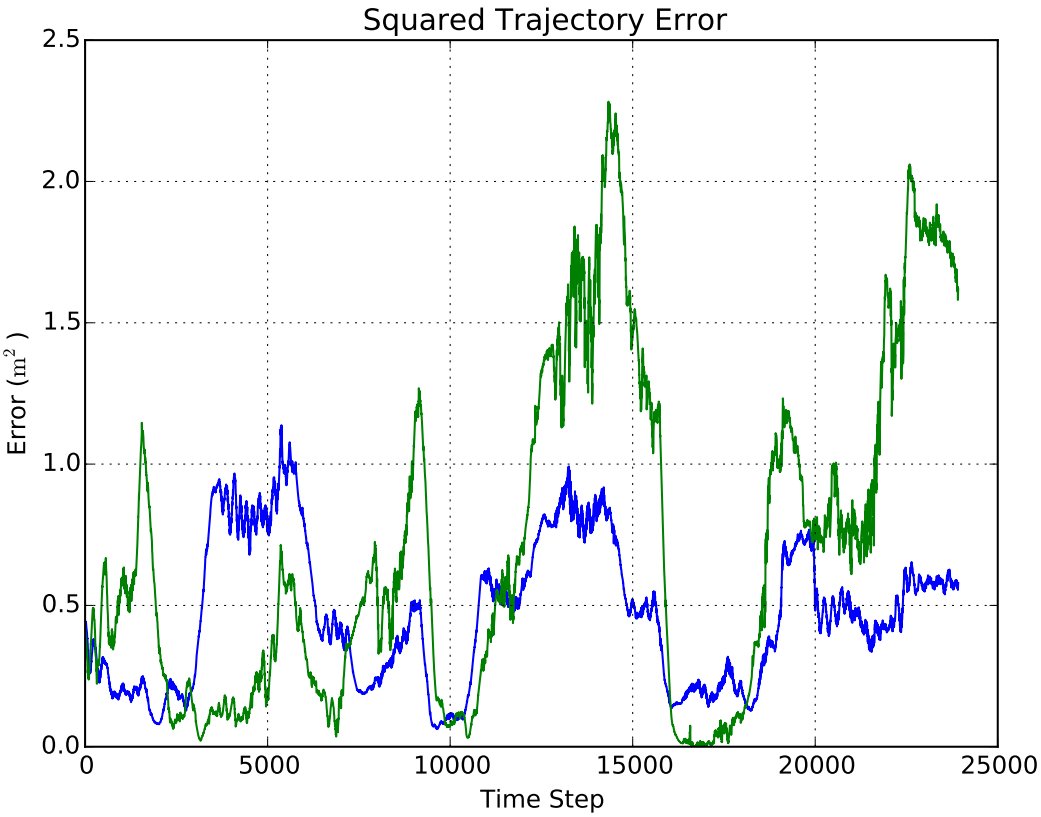


Figure 7.7: Squared trajectory error for SC-PHD SLAM (blue), and RB-PHD SLAM (green).

Chapter 8

Concluding Remarks

8.1 Summary

Multi-object moment filters have a number of strengths which can be applied to the challenging domain of mobile autonomous vehicles. Namely:

- Coping with less than ideal measurement conditions (false alarms and missed detections).
- Maintaining and updating estimates of multiple objects from multiple measurements without the need for data association heuristics.
- Tracking objects with heterogeneous dynamic characteristics.
- A well-established modeling and algorithm design methodology which can be traced back to first principles.

This work has presented an application of multi-object estimation methods to the area of feature-based simultaneous localization and mapping.
In Chapter 2, we reviewed the current state of the art in feature-based SLAM and RFS methods. The body of research in feature-based SLAM classified into several categories: information filter based methods, particle based methods, and submapping methods; all of these methods shared the common weaknesses of attempting to bridge a multi-object scenario with single-object estimation techniques.

In Chapter 3, we covered the mathematical background for finite set statistics and multiobject calculus. Topics covered were set integrals, functional derivatives, probability generating functionals, point and cluster processes, and the OSPA metric.
In Chapter 4, we built upon the material covered in Chapter 3 to illustrate the derivation of first-moment multi-object filters, namely the PHD filter and the SC-PHD filter. Modelling the object dynamics and measurements as RFSs allows us to derive concrete expressions for the p.g.fl. prediction and update, which we can then differentiate to obtain the PHD filter equations.
In Chapter 5 developed an algorithm based on the SC-PHD filter for SLAM with dynamic landmarks, and validated it in simulation. The algorithm is centered on a hybrid particle/Gaussian mixture implementation of the SC-PHD filter, with adaptations for measurement-driven birth and varying FoV. The simulation results showed that our algorithm compares favorably with the RB-PHD filter, a prior PHD filter based SLAM algorithm.
In Chapter 7, we demonstrated that the SC-PHD SLAM algorithm is viable for real-world applications by running it with the Girona 500 underwater vehicle. The experiment described a seafloor survey, with map features extracted via SURF from stereo imagery. The SC-PHD SLAM algorithm was again shown to perform better than RB-PHD SLAM and EKF SLAM.

8.2 Contributions

This thesis makes the following contributions:

- A novel SLAM algorithm derived from the SC-PHD filter which is capable of operating under difficult sensing conditions characterized by the presence of false alarms rates and missed detections. The algorithm is also capable of navigating in a map composed of a mixture of stationary and moving landmarks.

- A successful implementation of this algorithm in an underwater robotic platform with stereo vision.

- An implementation of the PHD and SC-PHD filters on a GPU, which achieves a computational rate that is orders of magnitude faster than MATLAB implementations.

8.3 Outlook

We believe that this field of research will become particularly fruitful as multi-object estimation methods move beyond their roots in target tracking and statistical signal processing. Areas such as astronomy, forestry, and statistical geography, where point process theory has already been a well-established tool, contain numerous opportunities to apply methods such as the SC-PHD filter.

In the domain of mobile autonomous vehicles and feature-based SLAM, there remain many challenges to which this work could be applied. The map features we worked with described only position and velocity. Augmenting these with richer descriptions (shape,extent,semantic tags), would facilitate a greater level of autonomy and presents and interesting challenge for modelling and filter development. In addition, autonomous vehicles measure their environment with a wide range of sensors modalities including radar,laser scanners, and sonar. As seen in our work with the underwater vehicle dataset and camera calibration, each type of sensor involves a unique set of challenges to which novel solutions can be developed.

Appendix A

SC-PHD Filter Pseudocode

This section contains pseudocode for our implementation of the single cluster PHD (SC-PHD) filter. In this implementation of the filter, the joint PHD is defined by the parent particle states and their accompanying Gaussian mixture probability hypothesis density (PHD)s. The pseudocode listings will make use of the following abbreviations:

$$D_k(\mathbf{x}, \mathbf{y}) \doteq \{\mathbf{x}^{(i)}, \omega^{(i)}, \tilde{D}_k^{(i)}(\mathbf{y}|\mathbf{x})\}_{i=1}^{N_k} \tag{A.1}$$

$$\tilde{D}_k^{(i)}(\mathbf{y}|\mathbf{x}) \doteq \{\mu^{(j)}, \mathbf{P}^{(j)}, w^{(j}\}_{j=1}^{J^{(i)}} \tag{A.2}$$

Each of the subroutines listed here accepts as input and returns as output a single parent particle and its corresponding map. For numerical stability, it is recommended that weights and likelihoods be replaced with their log-equivalents, and that the corresponding computations be modified accordingly.

Function SC-PHD($D_{k-1}, \mathbf{Z}_k$)

Input: Prior joint state estimate, current measurements

Output: Posterior joint state estimate, measurements

for $i = 1 \dots N_{k-1}$ **do**

$D^{(i)}_{k|k-1}$ =`Predict(`$D^{(i)}_{k-1}(\mathbf{x}, \mathbf{y})$`)`

$\hat{D}^{(i)}_{k|k-1}$ =`PreUpdate(`$D^{(i)}_{k|k-1}(\mathbf{x}, \mathbf{y}), \mathbf{Z}_k$`)`

$[\tilde{D}^{(i)}_{k|k}, L_{\mathbf{Z}_k}(\mathbf{x})]$ = `Update(`$\tilde{D}^{(i)}_{k|k-1}, \hat{D}^{(i)}_{k|k-1}, \mathbf{Z}_k$`)`

$\omega^{(i)} = \omega^{(i)} \times L_{\mathbf{Z}_k}(\mathbf{x})$

$\tilde{D}^{(i)}_{k|k}$ = `PruneAndMerge(`$\tilde{D}^{(i)}_{k|k}$`)` `// [96, Table II]`

end

`// Particle resampling according to [1, Algorithm 2]`

$\mathbf{x}^{(1\dots N_k)}$ = `Resample(`$\mathbf{x}^{(1\dots N_k)}, \omega_k^{(1\dots N_k)}$`)`

$\omega^{(1\dots N_k)} = 1/K$

return $D_k = \{\mathbf{x}^{(i)}, \omega^{(i)}, \tilde{D}^{(i)}_k(\mathbf{y}|\mathbf{x})\}_{i=1}^{N_k}$

The top-level filter iteration should look familiar to those experienced with other Bayesian filter methods such as the Extended Kalman Filter (EKF). The update step is split into two subroutines: `PreUpdate` computes auxiliary terms which are needed for the computation of the SC-PHD update and `Update` implements the actual update equations. In this work the `PruneAndMerge` subroutine was implemented as described in [96], but an alternative Gaussian mixture reduction algorithm can be substituted at the user's discretion. The same may be said about the `Resample` subroutine for resampling the parent particles.

Function Predict(D_{k-1})

Input: Prior parent state particle and conditional daughter PHD

Output: Predicted sensor state and object PHD

`// Sample sensor transition density`

for $n = 1 \dots M$ **do**

$\mathbf{x}_{k|k-1}^{(n)} \sim \pi(\tilde{\mathbf{x}}|\mathbf{x})$

$\omega_{k|k-1}^{(n)} \leftarrow \omega$

`// Predict daughter PHD`

forall the $\{\mu_{k-1}^{(j)}, \mathbf{P}_{k-1}^{(j)}, w_{k-1}^{(j)}\}$ *in* $\tilde{D}_{k-1}$ **do**

$w_{k|k-1}^{(j)} = w_{k-1}^{(j)}$

$\mu_{k|k-1}^{(j)} = f(\mu_{k-1}^{(j)})$

$\mathbf{P}_{k|k-1}^{(j)} = \mathbf{F}^{(j)}\mathbf{P}_{k-1}^{(j)}\mathbf{F}^{(j),T} + \mathbf{W}^{(j)}\mathbf{Q}^{(j)}\mathbf{W}^{(j),T}$

end

$\tilde{D}_{k|k-1}^{(n)} = \{\mu_{k|k-1}^{(j)}, \mathbf{P}_{k|k-1}^{(j)}\}_{j=1}^{J}$

end

return $D_{k|k-1} = \{\mathbf{x}_{k|k-1}^{(n)}, \omega_{k|k-1}^{(n)}, \tilde{D}_{k|k-1}^{(n)}\}_{n=1}^{M}$

The `Predict` subroutine generates a prediction for a parent particle and its conditional daughter Gaussian mixture PHD. The outer `For` loop implements sampling of the parent state transition density and replication of the daughter PHD. However, depending on the nature of the parent transition model, good performance can be achieved with $M = 1$, simplifying the implementation of the filter.

Function PreUpdate($D_{k|k-1}, \mathbf{Z}_k$)

Input: Predicted sensor particle and multi-object PHD, measurements

Output: Pre-update terms for SC-PHD daughter update

forall the $\{\mu^{(j)}, \mathbf{P}^{(j)}, w^{(j)}\}$ *in* $D_{k|k-1}$ **do**

- $\hat{z} = h^{-1}(\mathbf{X}, \mu^{(j)})$ `// Predicted measurement`
- $\mathbf{S} = \mathbf{H}\mathbf{P}^{(j)}\mathbf{H}^T + \mathbf{R}$ `// Innovation covariance`
- $\mathbf{K} = \mathbf{P}^{(j)}\mathbf{H}^T\mathbf{S}^{-1}$ `// Kalman Gain`
- $\mathbf{P}^{(j|\cdot)} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}$ `// Updated covariance`
- **for** $i = 1 \dots |Z|$ **do**
  - `// single-object likelihood`
  - $p(z^{(i)}|\mu^{(j)}, y) = \mathcal{N}(z^{(i)}; \hat{z}, \mathbf{S})$
  - $\mu^{(j|i)} = \mu + \mathbf{K}(z - \hat{z})$ `// Updated mean`
- **end**

end

$\hat{D}_{o|s} = \{\mu^{(j|i)}, \mathbf{P}^{(j|i)}\}_{\substack{j=1\dots J \\ i=1\dots|Z|}}$

return $p(\cdot|\cdot, \mathbf{x}), \hat{D}_{k|k-1}$

In the `PreUpdate` subroutine, single-object measurement likelihoods for each measurement and each term within the daughter Gaussian mixture are computed. The Gaussian mixture terms are also updated with the measurements via a Kalman update, but not yet reweighted. Depending on the numerical stability properties of the implementation, the updated covariance may not be symmetric. Using the alternative, Joseph formulation of the update can help address this issue, as well as further subroutines to enforce symmetry.

The remainder of the SC-PHD update is performed by the `Update` subroutine. Here, the predicted and preupdated daugther PHD terms, along with measurement-driven birth terms are concatenated and reweighted to form the final updated PHD.

Function Update($D_{k|k-1}, \hat{D}_{k|k-1}, p_{\mathbf{z}|\mu,\mathbf{x}}$)

Input: Predicted multi-object PHD and pre-update terms

Output: Updated multi-object PHD

forall the $\mu^{(j)}, \mathbf{P}^{(j)}, w^{(j)}$ *in* $D_{k|k-1}$ **do**

`// Non-detection terms`

$\mu_{nd}^{(j)} = \mu^{(j)}; \quad \mathbf{P}_{nd}^{(j)} = \mathbf{P}^{(j)}; \quad w_{nd} = w^{(j)}$

`// Detection terms`

for $i = 1 \dots |Z|$ **do**

$\mu_d^{(j|i)} = \mu^{(j|i)}; \quad \mathbf{P}_d^{(j|i)} = \mathbf{P}^{(j|i)};$

`// from` $\hat{D}_{k|k-1}$

$w_d^{(j|i)} = \tilde{w}^{(j)} p_D p_{z|s,o}(z_i|\tilde{\mu}^{(j)}, y)$

end

`// Measurement-derived birth terms`

for $i = 1 \dots |\mathbf{Z}_k|$ **do**

$\mu_0^{(i)} = h^{-1}(y, z_i); \quad \mathbf{P}_0^{(i)} = \mathbf{R}^*; \quad w_0^{(i)} = w_0$

end

`// Normalize weights`

`// compute multi-object likelihood`

$\tilde{N} = \sum_{j=1}^{J} \tilde{w}^{(j)}$

$L_{\mathbf{Z}_k}(\mathbf{x}) = \exp(\tilde{N})$

for $i = 1 \dots |Z|$ **do**

$\eta_{z_i} = \kappa(z_i) + \sum_{j=1}^{J} w_d^{(j|i)} + 2w_0$

for $j = 1 \dots J$ **do**

$w_d^{(j|i)} /= \eta_{z_i}$

end

$w_0^{(i)} /= \eta_{z_i}$

$L_{\mathbf{Z}_k}(\mathbf{x}) = L_{\mathbf{Z}_k}(\mathbf{x}) \times \eta_{z_i}$

end

end

`// Concatenate terms`

$\mu_{k|k} = [\mu_{nd}^{(1\dots J)}, \mu_d^{(1\dots J_{k|k-1}|1\dots|\mathbf{Z}_k|)}, \mu_0^{(1\dots|\mathbf{Z}_k|)}]$

$\mathbf{P}_{k|k} = [\mathbf{P}_{nd}^{(1\dots J)}, \mathbf{P}_d^{(1\dots J_{k|k-1}|1\dots|\mathbf{Z}_k|)}, \mathbf{P}_0^{(1\dots|\mathbf{Z}_k|)}]$

$w_{k|k} = [w_{nd}^{(1\dots J)}, w_d^{(1\dots J_{k|k-1}+1\dots|\mathbf{Z}_k|)}, w_0^{(1\dots|\mathbf{Z}_k|)}]$

return $\{\mu_{k|k}, \mathbf{P}_{k|k}, w_{k|k}, L_{\mathbf{Z}_k}(\mathbf{x})\}$

Bibliography

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174 –188, Feb. 2002.

[2] J. Aulinas, C. S. Lee, J. Salvi, and Y. R. Petillot. Submapping SLAM based on acoustic data from a 6-DOF AUV. In *AUV", 8th IFAC Conference on Control Applications in Marine Systems (IFAC/CAMS), RostockWarnemnde*, pages 15–17, 2010.

[3] J. Aulinas, X. Llado, J. Salvi, and Y. Petillot. Selective submap joining for underwater large scale 6-DOF SLAM. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2552–2557, Oct 2010.

[4] J. Aulinas, X. Lladó, J. Salvi, and Y. Petillot. SLAM base Selective Submap Joining for the Victoria Park Dataset. In *7th IFAC Symposium on Intelligent Autonomous Vehicles (IFAC/IAV), Lecce (Italy) September*, pages 6–8, 2010.

[5] B. Bacca, J. Salvi, and X. Cufí. Long-term mapping and localization using feature stability histograms. *Robotics and Autonomous Systems*, 61(12):1539–1558, 2013.

[6] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE, 2006.

[7] S. Barkby, S. B. Williams, O. Pizarro, and M. Jakuba. Incorporating prior maps with bathymetric distributed particle SLAM for improved AUV navigation and mapping. In *IEEE OCEANS*, pages 6–12, October 2009.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[9] P. Bernhard. Chain differentials with an application to the mathematical fear operator. *Nonlinear Analysis: Theory, Methods & Applications*, 62(7):1225–1233, 2005.

[10] M. Bosse, P. Newman, J. Leonard, and S. Teller. SLAM in large scale cyclic environments using the atlas framework. *International Journal Robotics Research*, 23(12):1113–1139, 2004.

[11] F. Bourgeois and J.-C. Lassalle. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM*, 14:802–804, December 1971.

[12] M. Cannaud, L. Mihaylova, N.-E. E. Faouzi, R. Billot, and J. Sau. A probabilistic hypothesis density filter for traffic flow estimation in the presence of clutter. In *Proc. from the IEEE Sensor Data Fusion Workshop: Trends, Solutions, Applications*, 2012.

[13] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for extended kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.*, 27:609–631, September 2010.

[14] D. Clark. Faa di Bruno's formula for Gateaux differentials and interacting stochastic population processes. *Arxiv preprint arXiv:1202.0264*, 2012.

[15] D. Clark, C. Lee, and S. Nagappa. Single-cluster PHD filtering and smoothing for SLAM applications. In *ICRA*, 2012.

[16] D. Clark and R. Mahler. Generalized PHD filters via a general chain rule. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 157–164. IEEE, 2012.

[17] D. Clark, I. Ruiz, Y. Petillot, and J. Bell. Particle PHD filter multiple target tracking in sonar image. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(1):409–416, 2007.

[18] D. Clark, I. T. Ruiz, Y. Petillot, and J. Bell. Particle PHD filter multiple target tracking in sonar image. *IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS*, 43(1):409–416, JAN 2007.

[19] D. Clark, B. Vo, and J. Bell. GM-PHD filter multi-target tracking in sonar images. *Proc. SPIE Defense and Security Symposium. Orlando, Florida [6235-29]*, 2006.

[20] D. E. Clark and J. Houssineau. Faa di Bruno's formula for Gateaux differentials and interacting stochastic population processes. *ArXiv e-prints*, Feb. 2012.

[21] D. E. Clark and J. Houssineau. Faa di Bruno's formula for chain differentials. *arXiv preprint arXiv:1310.2833*, 2013.

[22] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes. Vol. I.* Probability and its Applications (New York). Springer-Verlag, New York, second edition, 2003. Elementary theory and methods.

[23] S. J. Davey. Simultaneous localization and map building using the probabilistic multi-hypothesis tracker. *Robotics, IEEE Transactions on*, 23(2):271–280, 2007.

[24] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[25] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[26] G. Dissanayake, P. Newman, H. Durrant Whyte, S. Clark, and M. Csorba. A solution to the simultaneous location and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(2):229–241, May 2001.

[27] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *in Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03*, pages 1135–1142. Morgan Kaufmann, 2003.

[28] A. I. Eliazar and R. Parr. Hierarchical linear/constant time slam using particle filters for dense maps. In *Advances in Neural Information Processing Systems 18*, pages 339–346, 2006.

[29] O. Erdinc, P. Willett, and Y. Bar-Shalom. The bin-occupancy filter and its connection to the PHD filters. *Signal Processing, IEEE Transactions on*, 57(11):4232–4246, 2009.

[30] C. Estrada, J. Neira, and J. Tardós. Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, August 2005.

[31] R. Eustice, O. Pizarro, and H. Singh. Visually augmented navigation in an unstructured environment using a delayed state history. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 25–32. IEEE, 2004.

[32] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[33] R. M. Eustice, H. Singh, and J. J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006.

[34] N. Fairfield, G. Kantor, D. Jonak, and D. Wettergreen. Autonomous exploration and mapping of flooded sinkholes. *The International Journal of Robotics Research*, 29:748–774, 2010.

[35] M. F. Fallon, M. Kaess, H. Johannsson, and J. J. Leonard. Efficient AUV navigation fusing acoustic ranging and side-scan sonar. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2398–2405. IEEE, 2011.

[36] J. Folkesson, P. Jensfelt, and H. Christensen. Graphical SLAM using vision and the measurement subspace. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, Aug. 2005.

[37] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research*, 22:2003, 2003.

[38] U. Frese. Treemap: An o(log n) algorithm for simultaneous localization and mapping. In *IN SPATIAL COGNITION IV, C. FREKSA, ED*, pages 455–476. Springer Verlag, 2005.

[39] K. Granstrom, C. Lundquist, and O. Orguner. Extended target tracking using a gaussian-mixture PHD filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(4):3268–3286, October 2012.

[40] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[41] J. Guivant and E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *In IEEE International Conference on Robotics and Automation*, pages 2731–2736, 2002.

[42] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics*, 17(3):242–257, 2001.

[43] J. Houssineau and D. Laneuville. PHD filter with diffuse spatial prior on the birth process with applications to GM-PHD filter. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1 –8, july 2010.

[44] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended Kalman filter based SLAM. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 473–479. IEEE, 2008.

[45] N. Hurtos, X. Cufí, Y. Petillot, and J. Salvi. Fourier-based registrations for two-dimensional forward-looking sonar image mosaicing. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5298–5305, Oct. 2012.

[46] S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar 2004.

[47] S. J. Julier and J. K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243. IEEE, 2001.

[48] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics, TRO*, 24(6):1365–1378, Dec 2008.

[49] J. Knight, A. Davison, and I. Reid. Towards constant time SLAM using postponement. In *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems, Maui, HI*, volume 1, pages 406–412. IEEE Computer Society Press, Oct. 2001.

[50] C. Lee, D. Clark, and J. Salvi. SLAM with single cluster PHD filters. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2096–2101. IEEE, 2012.

[51] C. S. Lee, D. Clark, and J. Salvi. Slam with dynamic targets via single-cluster PHD filtering. *Selected Topics in Signal Processing, IEEE Journal of*, 7(3):543–552, June 2013.

[52] J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 1442 –1447 vol.3, nov 1991.

[53] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *AUTONOMOUS ROBOTS*, 4:333–349, 1997.

[54] R. Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Trans. Aerospace and Electronic Systems*, 39(4):1152–1178, October 2003.

[55] R. Mahler. PHD filters of higher order in target number. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(4):1523 –1543, 2007.

[56] R. Mahler, B.-T. Vo, and B.-N. Vo. Cphd filtering with unknown clutter rate and detection profile. *Signal Processing, IEEE Transactions on*, 59(8):3497 –3513, aug. 2011.

[57] R. P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA, 2007.

[58] R. Martinez-Cantin and J. Castellanos. Unscented SLAM for large-scale outdoor environments. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3427 – 3432, aug. 2005.

[59] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598, Edmonton, Canada, 2002. AAAI.

[60] 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.

[61] D. Moratuwage, B.-N. Vo, S. Wijesoma, and D. Wang. Extending the bayesian RFS SLAM framework to multi-vehicle SLAM. In *ICRA*, 2012.

[62] J. Moyal. The general theory of stochastic population processes. *Acta Mathematica*, 108:1–31, 1962. 10.1007/BF02545761.

[63] J. Mullane, S. Keller, and M. Adams. Random set versus vector based SLAM in the presence of high clutter. In *IEEE ICRA 2012: Workshop on Stochastic Geometry in SLAM*, May 2012.

[64] J. Mullane, B.-N. Vo, M. Adams, and B.-T. Vo. *Random Finite Sets for Robot Mapping and SLAM - New Concepts in Autonomous Robotic Map Representations*, volume 72 of *Springer Tracts in Advanced Robotics*. Springer, 2011.

[65] J. Mullane, B.-N. Vo, M. D. Adams, and W. S. Wijesoma. A random set formulation for bayesian slam. In *IROS*, pages 1043–1049, 2008.

[66] K. Murphy. Bayesian map learning in dynamic environments. In *Advances In Neural Info. Proc. Systems (NIPS)*, pages 1015–1021. MIT Press, 2000.

[67] C. Musso, N. Oudjane, and F. Le Gland. Improving regularised particle filters. In *Sequential Monte Carlo methods in practice*, pages 247–271. Springer, 2001.

[68] S. Nagappa and D. Clark. On the ordering of the sensors in the iterated-corrector probability hypothesis density (phd) filter. *Proceedings of SPIE*, 8050:80500M, 2011.

[69] E. M. Nebot. Ute data parameters. `http://www-personal.acfr.usyd.edu.au/nebot/experimental_data/modeling_info/Ute_modeling_info.htm`, February 2003.

[70] J. Neira and J. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890 – 897, December 2001.

[71] P. M. Newman, J. J. Leonard, and R. J. Rikoski. Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar. In *Robotics Research. The Eleventh International Symposium*, pages 409–420. Springer, 2005.

[72] Nvidia Corporation. CUDA C Best Practices Guide. Technical report, NVIDIA Corporation, 2011.

[73] N. Oudjane and C. Musso. Progressive correction for regularized particle filters. In *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, volume 2, pages THB2–10. IEEE, 2000.

[74] M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1157–1164, San Francisco, CA, 2003. Morgan Kaufmann Publishers.

[75] L. M. Paz, J. Tardós, and J. Neira. Divide and conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107–1120, October 2008.

[76] P. Piniés, L. M. Paz, and J. D. Tardós. Ci-graph: An efficient approach for large scale slam, May 12-17 2009.

[77] P. Piniés and J. D. Tardós. Large scale SLAM building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics*, 24(no. 5):1094–1106, October 2008.

[78] D. Ribas. *Underwater SLAMfor Structured Environment Using an Imaging Sonar*. PhD thesis, University of Girona, 2008.

[79] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter*. Artech House, 2004.

[80] B. Ristic, D. Clark, B.-N. Vo, and B.-T. Vo. Adaptive target birth intensity for PHD and cphd filters. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(2):1656 –1668, april 2012.

[81] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer, 2nd edition, 2004.

[82] D. Salmond. Mixture reduction algorithms for point and extended object tracking in clutter. *Aerospace and Electronic Systems, IEEE Transactions on*, 45(2):667 –686, april 2009.

[83] D. Schieferdecker and M. Huber. Gaussian mixture reduction via clustering. In *FUSION '09. 12th International Conference on*, 2009.

[84] D. Schuhmacher, B.-T. Vo, and B.-N. Vo. A consistent metric for performance evaluation of multi-object filters. *IEEE Transactions on Signal Processing*, 56(8-1):3447–3457, 2008.

[85] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *UAI*, pages 435–461, 1986.

[86] N. Sünderhauf and P. Protzel. Towards a robust back-end for pose graph slam. In *ICRA*, pages 1254–1261, 2012.

[87] A. Swain and D. Clark. First-moment filters for spatial independent cluster processes. *Proceedings of SPIE*, 7697:76970I, 2010.

[88] A. Swain and D. Clark. The single-group PHD filter: an analytic solution. In *International Conference on Data Fusion*, 2011.

[89] A. Swain and D. Clark. The PHD filter for extended target tracking with estimable extent shape parameters of varying size. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1111–1118. IEEE, 2012.

[90] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, 21(4):311–330, 2002.

[91] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2001.

[92] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7/8):693–716, 2004.

[93] M. Uney, D. Clark, and S. Julier. Information measures in distributed multitarget tracking. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8. IEEE, 2011.

[94] R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems 13*, Nov 2001.

[95] T. Vidal-Calleja, J. Andrade-Cetto, and A. Sanfeliu. Conditions for suboptimal filter stability in SLAM. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 27–32. IEEE, 2004.

[96] B.-N. Vo and W.-K. Ma. The gaussian mixture probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 54(11):4091–4104, 2006.

[97] B.-N. Vo, S. Singh, and A. Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):1224 – 1245, oct. 2005.

[98] B.-T. Vo, B.-N. Vo, and A. Cantoni. Analytic implementations of the cardinalized probability hypothesis density filter. *IEEE Transactions on Signal Processing*, pages 3553–3567, 2007.

[99] S. Williams, G. Dissanayake, and H. Durrant-Whyte. Efficient simultaneous localisation and mapping using local submaps. In *Australian Conference on Robotics and Automation (ACRA)*, pages 128–134, 2001.

[100] S. B. Williams, G. Dissanayake, and H. F. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *ICRA*, pages 406–411, 2002.

[101] T. M. Wood, C. A. Yates, D. A. Wilkinson, and G. Rosser. Simplified multitarget tracking using the PHD filter for microscopic video data. *IEEE Trans. Circuits Syst. Video Techn.*, 22(5):702–713, 2012.