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

RANDOM FINITE SETS IN VISUAL SLAM

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO ELÉCTRICO

ANGELO FALCHETTI PAREJA

PROFESOR GUÍA:
MARTIN ADAMS

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
MIGUEL TORRES TORRITI

ESTE TRABAJO HA SIDO AUSPICIADO POR CONICYT

SANTIAGO DE CHILE
ENERO 2017

RESUMEN DE LA TESIS
PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN ELÉCTRICA
Y AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: ANGELO FALCHETTI PAREJA
FECHA: ENERO 2017
PROF. GUÍA: MARTIN ADAMS

RANDOM FINITE SETS IN VISUAL SLAM

Este trabajo trata sobre el diseño e implementación de un sistema de Localización y Mapeo Simultáneo (SLAM) visual usando la teoría de Conjuntos Finitos Aleatorios (RFS), en el que un navegador (e.g. robot, auto, teléfono celular, etc.) utiliza una cámara de vídeo RGB-D para reconstruir la escena a su alrededor y al mismo tiempo descubrir su propia posición. Esta capacidad es relevante para las tecnologías del futuro, que deberán desplazarse sin ayuda externa.

Se considera la inclusión de modelos realistas de medición y movimiento, incluyendo la intermitencia de las detecciones de objetos, la presencia de falsos positivos en las mediciones y el ruido en la imagen. Para ello se analizan sistemas basados en la teoría RFS, que es capaz de incluir estos efectos de manera fundamentada, a diferencia de otras alternativas del estado del arte que se basan en heurísticas poco realistas como el conocimiento absoluto de las asociaciones de datos entre mediciones y puntos en el mapa.

Se incluye una amplia revisión de la literatura, desde Structure from Motion a Odometría Visual, a los distintos algoritmos para SLAM. Luego, se procede a explicar los detalles de implementación de un sistema flexible para el análisis de algoritmos de SLAM, así como la implementación particular del algoritmo Rao-Blackwellized (RB)-Probability Hypothesis Density (PHD)-SLAM. Se presentan análisis del desempeño de este algoritmo al cambiar las distintas estadísticas que pueden variar en su uso práctico. Se hace una comparación detallada con la alternativa Incremental Smoothing and Mapping (iSAM2), usualmente usada en otros sistemas del estado del arte. Luego, basado en la teoría de Modelos Gráficos Probabilísticos (PGM) que está detrás de iSAM2, se propone un nuevo algoritmo, Loopy PHD-SLAM, capaz de propagar información a lo largo del grafo inducido de manera eficiente, incluyendo las estadísticas de RFS. Con una implementación sencilla como prueba de concepto, se observa la capacidad de este nuevo método de cerrar ciclos y converger a soluciones correctas.

RESUMEN EN INGLÉS DE LA TESIS
PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN ELÉCTRICA
Y AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: ANGELO FALCHETTI PAREJA
FECHA: ENERO 2017
PROF. GUÍA: MARTIN ADAMS

RANDOM FINITE SETS IN VISUAL SLAM

This work presents the design and implementation of a visual Simultaneous Localization and Mapping (SLAM) system using the concepts of Random Finite Sets (RFS), in which a navigator (e.g. a robot, a car, a cellphone, etc.) uses a RGB-D video camera to reconstruct the scene around it and estimate its own pose at the same time. This capacity is relevant for the technologies of the future since they will need to move around without external assistance.

Realistic motion and measurement models are considered, including missed detections, the presence of clutter measurements and image noise. The RFS framework is well equipped to manage these effects in a mathematically sound manner, in contrast with alternative state-of-the-art solutions, which rely on unrealistic heuristics such as perfect knowledge of the data associations between measurements and map landmarks.

An extensive literature review is included, which goes from Structure from Motion to Visual Odometry to several alternative SLAM algorithms. Then, the implementation details for a flexible visual SLAM system are explained. This system has been designed to analyse the performance of SLAM algorithms. An implementation for the particular case of Rao-Blackwellized (RB)-Probability Hypothesis Density (PHD)-SLAM is detailed and its performance is evaluated for different model statistics configurations. A detailed comparison between PHD-SLAM and the popular Incremental Smoothing and Mapping (iSAM2) alternative is presented. Then, based on the theory of Probabilistic Graphical Models (PGM), which is behind the iSAM2 method, a novel algorithm is proposed, Loopy PHD-SLAM, which is capable of efficiently propagating information through the induced graph, including RFS statistics. A proof-of-concept implementation is used to analyse its performance in practice, observing its capacity to close loops and converge to correct solutions.

Dedicated to all those leaders that
are making their communities better

Acknowledgements

Special thanks to my advisor Martin Adams, who has helped me greatly to shape this thesis. Also to my fellow researchers, Felipe Inostroza, Keith Leung and Javier Correa, who have given me insightful feedback about my ideas.

Thanks to my family and friends, who have supported me throughout this journey and with whom I've shared so much along the way. Your company and opinion has been an evergreen source of joy and growth.

Also, thanks to CONICYT, who have believed in my potential and have been kind enough to give me funding for pursuing this research direction.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A key landmark of the modern human experience is the computer: a robust device capable of performing complex algorithms and tasks in an efficient manner. Nowadays everyone interacts with many computers several times a day, be it to read an email, pay with a credit card, talk on the phone or even change the radio station on the car. No one can deny that computers have improved the life of many people, making it more efficient, more expressive and even more connected. However, these machines are far from perfect, they improve every year, allowing their users to approach bigger, more abstract, productive and social challenges. Who would've thought a century ago that it would be possible to talk with a friend twenty thousands kilometres away instantaneously?

Today, this journey continues into uncharted territory: the autonomy of computer systems. Using research advancements, computers can be programmed to make decisions by themselves using appropriate goals, so they can interact with the world intelligently, in a manner similar to that of humans. Naturally, these decisions are much simpler than typical human interactions, but are good enough that the devices can react to new contexts without the need for a human to intervene and specifically program them for that particular context. This ability of generalization, abstract thinking and adaptation is key for collaborating with humans or for exploring new places where no human has gone before. It is the start of a world where machines are not just passive instruments, but become active participants on the solutions for the problems of the world.

Why active participation, one may wonder? In general, this trend responds to the intractability of predicting, defining and solving every situation a system may encounter; there are tasks that require so much detail that humans cannot do them by themselves. Very fine-grained input may be required, decisions may have to be made thousands of times faster than what a human can do and unforeseen dangerous obstacles may need attention; even the amount of work is sometimes just too much. Active participation alleviates these problems and opens a world of possibilities, where a lot of the tasks done today by people can be accomplished by autonomous computer systems.

A major field in the autonomization of systems is the planning and execution of movement. Movement is hard: it requires observing the environment, detecting obstacles and making quick decisions over the control of the navigator. Many things can change rapidly: obstacles may move, motors may not respond as specified, the expected position of the system may drift with time, noise can cloud the measurements; many times very fine precision is required to interact with objects or pass through narrow corridors; objects may enter and leave the sensing area; a previously designed

plan may be rendered invalid when new objects are observed, etc.. There are many details that have to be solved precisely for autonomous movement to work. Nevertheless, when it does work, astonishing results can be accomplished.

As the field moves forward, the market introduces new products that make use of the technology, such as Unmanned Aerial Vehicles (UAV, also known as drones) capable of autonomous hovering and flight [1]. Their use includes military applications, journalism, de-mining, surveying and recreational purposes. Even multi-vehicle swarms have been experimented with, collectively improving all of their estimates. Their ability to safely manoeuvre around other vehicles such as aeroplanes heavily relies on them understanding their environment.

These UAVs can be used to survey a terrain, i.e. generate a three-dimensional map of an area. This can be important for urban and rural planning, construction, geology and archaeology, and requires finding a precise map, obtained through the algorithms in this field*. The technique has been used for monitoring glacier dynamics [2], forest fires [3] and urban 3D reconstruction [4].

Rescue missions can be performed by autonomous navigators. They have better manoeuvrability than people and if they are small in size, they can move through debris in a collapsed building [5]; their expendability makes the rescue less critical and their inexpensive nature allows for big swarms to search big areas simultaneously. After a disaster, there are many sharp edges and exposed material that the rescuers must skilfully avoid using appropriate movement techniques.

Autonomy has also been tried as a manufacturing methodology known as "lights out", where factories run with no human presence on-site. FANUC, a japanese robotics company has factories using this methodology [6]. Sophisticated robots like Baxter (from Rethink Robotics) [7] can learn production line tasks and manage lightweight assemblies with a user-friendly interface for non-programmers, removing the need for humans in most of the manufacturing process.

Other robots have become very successful in the market, like Roomba, which can navigate around an entire floor in a house, cleaning everything on its way [8]; or the NAO, a small humanoid which can be taught to play football, around which an entire league has formed, namely RoboCup [9]. Of course the task is complex and requires good mapping of the field and localization of the player, appropriate tracking of the ball and tight coordination from the different players.

One of the most recent and promising applications of autonomous navigation is self driving cars, i.e. cars that do not require a driver, since they observe the road with many sensors and actuate all the controls usually performed by the drivers themselves, such as accelerating, braking, steering, gear shifting, path planning and avoiding obstacles [10]. A competition, the DARPA Challenge, has even developed to create a fully autonomous car. In this application there are additional challenges as the road has many rules and there are other drivers that the system must take into consideration. There's an elevated need for robustness, reliability and safety, since the vehicle transports people, whose physical integrity is trusted to the navigation algorithm, even in the presence of highly unexpected events such as bumps in the road, unresponsiveness of the wheels in the snow, inattentive animals that may cross the street or erratic behaviour of other cars in the road. A good implementation may save many lives that are lost every year to car accidents. The extra caution mandates a need for strict policies, which slowly but steadily are being enacted on different jurisdictions. Despite the many difficulties that surround them, self driving cars will probably reach the consumer market

*In this context, the stitching of different images to form a map is usually called aerial photogrammetry.

in the coming years, which puts them in a privileged position as the quintessential example of the autonomization field, capable of solving a real problem by applying state-of-the-art technology, all by themselves.

As shown, this field has a lot to give to the world, but of course, there are still challenges to overcome. This work addresses a specific problem that can still be improved upon: Visual SLAM. This is the problem of understanding *where* the navigator and the obstacles are using a visual system such as a video camera and a odometry source as data *input*. Planning and control strategies can be built on top of this solution.

A video camera shall be attached to the navigator so it can see the world as it moves around. From this stream, the navigator must process the images to estimate the three-dimensional position of *landmarks* in the world, which are easy-to-identify elements in the scene, such as the corners of a table or a printed logo in a book. Using these estimated landmarks, the navigator has to create a model of its surrounding area, known as the map. From this map, the system can try to reconcile different observations of the same landmarks, improving the quality of the map itself as new measurements come in, as well as improving the estimate of its own position in the map.

To reconcile the different measurements, this work uses the concept of *random finite sets* [11], which comes from the world of point processes, a branch of probability theory. The observed landmarks are a set of points in space and to understand the relationship between different observation frames, it would be nice to calculate the probability of a set of observations to come from a particular map model. Obtaining such probabilities is not trivial, though; elaborate mathematical arguments are required to derive useful formulas to calculate them. From this, known methodologies like Markov models can be applied to estimate the underlying map and navigator trajectory.

The world of SLAM is big and many different sensors have been used to deal with it, e.g. lasers, radars and sonar. Then, why use cameras? The answer is simple: cameras are very rich sensors, i.e. they contain a lot of useful information; they provide direct three-dimensional data, which can be used to construct more complete maps and better position estimates; and they are much cheaper devices compared with the alternatives. Also, there is massive amounts of video streams in the wild that can benefit from SLAM, retroactively. Note however that although rich, video streams are harder to process as the extra information requires higher bandwidth and processing power; this is becoming less of an issue each year, as processors capable of such work are cheaper and become more accessible.

This topic has received a lot of attention lately and many algorithms have been derived. Nevertheless, there are advantages to this approach around robustness to non-ideal environmental conditions.

1 Hypothesis

This work is guided by the following hypothesis:

> Current visual SLAM solutions can benefit from including Random Finite Set statistics and their integration may be done without compromising the sparsity benefits of graphical modelling.

2 Objectives

The general objective of this work is to implement and analyse the performance of visual SLAM with random finite sets. To do so, three specific objectives will be pursued. First, a flexible framework will be coded, which can take input from a RGB-D sensor and apply the Rao-Blackwellized PHD filter to estimate both the sensor pose and the map around it. This framework will be able to run simulations as a way to better understand the performance of the algorithm and will be appropriate to compare the results of this methodology against others.

Second, using this framework, a comparison will be performed between the PHD approach and the Incremental Smoothing and Mapping (iSAM2) methodology, which is one of the preferred state-of-the-art methodologies in the literature. This comparison will consider their performance in multiple simulated environments with controlled parameters to see how well they can face different kinds of challenges and with real data to validate the findings.

Third, using observations from the comparison, an extension to the random finite set approach will be proposed, which exhibits advantages from both approaches. This extension will be targeted towards improving the state-of-the-art performance on the visual SLAM problem, however, it will be an early step.

3 Thesis structure

The structure of this work is simple. In chapter 1, a short and smooth introduction has been presented to prepare the reader for the content of the work.

In chapter 2, a comprehensive review of the state of the art in the field is presented and the connections between different perspectives of the problem are exposed. Also, a theoretical framework and background is presented, which explains everything that may be required to understand the present work.

In chapter 3, a new visual SLAM system is proposed making use of recent advancements on the theory of random finite sets, in particular the Rao-Blackwellized PHD SLAM algorithm. Details on its design and implementation are discussed, as well its nuances and some extensions, along with the video stream processing and landmark extraction units.

In chapter 4, a comparison is performed between the random finite set approach and the alternative iSAM2 methodology. Big topics like data association and batch processing are addressed and compared between the algorithms.

In chapter 5, the method is extended using batch processing ideas inspired by alternative perspectives, using a Loopy Belief Propagation approach. Both theoretical and practical analyses of the extensions are presented, accompanied by a proof-of-concept implementation.

Finally in chapter 6, the work is concluded, giving an overarching summary of the work, along with the main insights and proposing new directions to continue this line of research.

Chapter 2

Review of the State of the Art

Navigating with cameras seems like quite a challenge to undertake. Like in any other endeavour, the first question shall be what options are there today to accomplish the task, i.e. what has already been done? It is important to stand on the shoulders of the community and use previous work as an advantage.

Visual SLAM, as its name clearly states, is at the intersection of two areas of research: on one side, there is the analysis of *visual* information to extract useful 3D representations, a technique known as *Structure from Motion*; on the other side, *SLAM*, or *Simultaneous Localization and Mapping*, comes from the notion of estimating the pose of a robot and the map surrounding it. Both ideas come from different places and objectives, but are very much related. The first one is very rooted on geometry, focusing on details including projections to the camera plane, camera transformations and image features. The second one is based on sensing theory and motion dynamics, focusing on information propagation, covariance estimation and system modelling. Both approaches try to reach an equivalent goal: understanding scenes and achieving robust navigation, albeit in different settings, as will be explained.

1 Structure from Motion (SfM)

Artists have long captured the essence of their times in paintings and sculptures. Using their eyes, they are able to replicate the world around them in stone. Structure from Motion seeks to transfer this ability to a machine, capable of generating a 3D representation from a set of images of a model. If enough images are available, and they have different perspectives of the scene, it is possible to triangulate points in the images and find their location in 3D space. Using this strategy with every point, a complete 3D model can be reconstructed. Its applications are many, from historical archiving [12] to making tiny personal statues [13].

The key notion to achieve this is keypoint extraction. This means finding easily-recognizable landmarks repeatedly in a set of images of the same scene, e.g. the corner of a table. If the projections of the same 3D point from several camera viewpoints are known, they give geometrical constraints on the viewpoints which can be solved numerically. After extracting a number of keypoints from an image, they have to be matched with the extracted points from other images. This

matching process is guided by a *descriptor* extracted around each keypoint that captures certain invariant properties of the landmark and will not vary much from image to image.

Many ways to find such keypoints and descriptors have been defined in the literature. The general idea is that relevant geometric points such as corners or complex texture details are constant regardless of viewpoint, so they can be used as reference. Harris [14] provided one of the first extractors which was based on finding locations where the irradiance gradient was strong in multiple directions, i.e. corners. Years later, Shi and Tomasi [15] developed better descriptors for corners that could be easily matched. One of the pinnacles of computer vision occured in 1999-2004 when Lowe described the SIFT (Scale-Invariant Feature Transform) keypoint routine [16], which is invariant to many transformations on the images such as scale, rotation in the image plane and mild rotations out of the image plane, and partially invariant to illumination changes. This means that keypoints can be well-matched even in distorted settings, unlike previous methods, which had difficulty if the viewpoints weren't parallel. The similar algorithm SURF (Speeded Up Robust Feature) was developed by Bay in 2008 [17], but the big drawback of both methods is they're slow and encumbered with software patents, making them unsuitable for open source, large or real-time settings. After these, a number of open technologies flourished, targeted at newly found constraints such as FAST (Features from Accelerated Segment Test) [18], FREAK (Fast Retina Keypoint) [19], ORB (Oriented FAST and Rotated BRIEF) [20] and KAZE [21]. These alternatives can be orders of magnitude faster than SIFT or SURF, although sometimes at a price in the quality of the matchings.

Once the keypoints have been found and matched between the images, all that's left is geometry. There are many mathematical details to solving this problem [22], but the main idea is simple. A camera projects the 3D world into the 2D space of the film or sensor. This transformation can be described by the pin-hole equation (2.1), which relates the underlying scene landmarks to the previously found keypoints,

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{2.1}$$

where (x, y) is the 2D projected point in the camera sensor plane, $(X, Y, Z)^T$ is the location in 3D space of a landmark and f is the focal distance to the sensor.

Note that the relationship can be expressed as a linear transformation if a dummy fourth dimension is added,

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix}_{local} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = C_t \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}. \tag{2.2}$$

This 4D extended space is known as homogeneous coordinates and is useful to express many statements succinctly. In here, the camera is defined by the 3x4 matrix C_t. To convert from homogeneous coordinates back to regular 3D coordinates, the components must be normalized: $(X', Y', Z')^T = (X/W, Y/W, Z/W)$. If W is zero, the vector represents a "point at infinity" or, equivalently, a direction. Analogous rules apply for 2D coordinates.

This equation is valid in the camera frame of reference but it is usually useful to use another

global origin. Then the camera matrix can be factored into two terms,

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix}_{global} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R_{3x3}} & \mathbf{t_{3x1}} \\ \mathbf{0_{1x3}} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = C_f C_p \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}, \tag{2.3}$$

where the first term C_f represent the internal calibration and usually doesn't change with time (unless in special circumstances, e.g. the focal length of the camera changes) and the second term C_p contains the position and orientation, which consists of a translation vector $\mathbf{t_{3x1}}$ and a rotation matrix $\mathbf{R_{3x3}}$ (see section 7.2).

From these projection equations, three constraints can be derived for every keypoint. Note however that every keypoint adds three new variables, namely its three coordinates in space. This means that adding keypoints alone will not help in determining the camera pose, which is an intuitive result. To solve this, some redundancy is required, which results from having the same keypoints in different cameras.

In theory, with perfect projection, there are ways to reconstruct the exact pose of the camera* from two images using eight keypoints [23]. Naturally, in real operations perfect reprojection is not possible. Instead, a typical minimization on the square error of the reprojection can be applied to find the best possible candidate,

$$\min_{p_i, c_k} \sum_{i=1}^{|p|} \sum_{k=1}^{|c|} v_{ik} |P(p_i, c_k) - x_{ik}|^2, \tag{2.4}$$

where p_i are the landmark locations, c_k the camera poses, $P(p_i, c_k)$ is the theoretical projection of the landmark p_i in camera c_k and x_{ik} the measured keypoint location. v_{ik} is a binary value that equals one if the landmark is visible from the camera and zero otherwise and is introduced to dismiss unknown data. The cardinalities $|p|$ and $|c|$ correspond to the number of landmarks and camera poses respectively.

This quadratic optimization is known as Bundle Adjustment (BA) [24] and serves as the fundamental framework for SfM. However, it has two major problems: it can be huge even with a moderate number of images and it is non-convex so a good initial approximation is required to converge to the real minimum. To find such an initialization a possible methodology is known as visual odometry [25]. This approach only works if the images are sequential and move smoothly, such as in a moving vehicle. Using this additional structure, it can be seen that consecutive frames will probably share many common keypoints, so that small optimizations can be performed throughout the sequence in pairs. Since the movement should be smooth, the viewpoint of the camera does not change too much, so exploring the camera pose space should be fast. This gives differential constraints between consecutive frames, which can then be followed globally to find a good initial approximation. The small optimizations are usually carried out through the Random Sample Consensus (RANSAC) voting scheme [26]. Variations that improve performance and accuracy exist [27, 28, 29, 30], but the general idea remains the same.

*Almost the exact pose: there will always be scale and Euclidean ambiguities, i.e. the absolute size of things can't be known just from images [22] and the origin is arbitrary.

2 Simultaneous Localization and Mapping (SLAM)

How is it possible to estimate the position of a robot as it walks around a city making errands? To inspire an answer, it can be noted that people do this all the time, and they use two big clues to solve the problem: first, they know what to expect from their muscular actuation, e.g. one footstep should move them about 20 cm in whatever direction they were pointing at; second, they can see objects, signs and buildings around the city that guide them.

These two clues are the key to SLAM. Walking expectation corresponds to grasping a little bit about how one's own motion works, this is usually called a *motion model* and its input is called *odometry*. So, taking one step is the odometry, and the rule "one step moves 20 cm" is the motion model.

Naturally, this model is not perfect and contains uncertainty. It is very important to consider this uncertainty when using the information. If a person closes their eyes and walks around the city, they may have some notion of where they are for a while but as time passes the uncertainty grows; after a couple of steps it is hard to know if the next step will still be on the sidewalk, the street or a wall.

The only thing required to fix the problem is opening the eyes. Immediately, people can locate themselves in the scene and stop before stepping into the dangerous road. This stage is called measurement or sensing, and consists of collecting data from the environment, *measurements*, and relating it to the pose, the *measurement model*. In this case, the observed curb of the road is the measurement, and "If an object is visually close to the feet, it is right next to the person in the real world" is the measurement model. Just like with motion models, measurement models are uncertain and require care and attention.

People make use of these two cues simultaneously in a tight feedback loop and this is the essence of SLAM. The moving robot needs to formalize this into concrete mathematical expressions.

2.1 Mathematical Formulation

To formulate the SLAM problem in a mathematical framework, it is modelled as in figure (2.1), where the vehicle follows a path through time and observes landmarks along the way. The map is represented by these point landmarks. A landmark can be an easily recognizable place in space like the corners of a table, just like in the SfM setting.

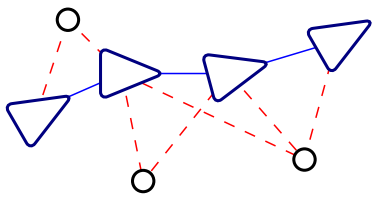


Figure 2.1: Elements and relationships in the SLAM problem. The vehicle trajectory is shown as blue triangles, the landmarks as black circles; the trajectory is the blue continuous line and the measurements are expressed as dashed red lines.

Unlike the optimization approach of Bundle Adjustment, SLAM is based on probabilities. This way, a suitable probability distribution must be proposed on the different robot trajectories and map landmarks, which then must be used to find the most likely trajectory and map. The solution is not straightforward and several alternatives exists, which will be listed later.

To discuss the different available algorithms, the use of a common terminology eases understanding. This work uses the following definitions: The robot pose at time k will be denoted as X_k, corresponding to an appropriate vector representation, e.g. in a 2D map this could be the vector $(x, y, \theta)^T$, of the vehicle's location and orientation. The complete trajectory between the starting time $t = 0$ and the final time $t = T$ will be $X_{0:T}$.

The map will be M, that may be represented as the concatenated vector of the landmark positions, e.g. $(x_0, y_0, x_1, y_1)^T$ could be a 2D map with two landmarks. The representation of the map is an important issue with subtle nuances that requires careful study and will be further explored through this work. When referring to a particular landmark inside the map lowercase m^i will be used to avoid confusion between time and positional indices.

In every time step k, the vehicle will sense some of the landmarks $m_k^1, m_k^2, \ldots, m_k^n$, generating measurements $z_k^1, z_k^2, \ldots, z_k^n$ from the measurement model (detailed later). Then, the vector Z_k will be the concatenation of all these z_k^i. Similarly to the case of the map, the representation for measurements will continue to be explored in this work. The measurements from time 1 to k will be $Z_{1:k}$.

The pose X_k and the map M will comprise the state of the model, $Y_k = (X_k^T, M^T)^T$.

In every time step, U_k is a vector representing the direct odometry information on the movement of the vehicle, e.g. (w_l, w_r) could be the measured rotation of the left and right rear wheels of a car from the last time step. The odometry readings from time 1 to time k will be $U_{1:k}$.

Note that both measurements and odometry readings start at time $t = 1$, unlike the trajectory which starts at $t = 0$. This is because the robot position needs to be fixed at the start, i.e. the global framework of reference is determined by X_0.

The relevant probability distribution is dependant on the observations performed by the robot and can be stated as [31, 32]

$$P(X_{0:T}, M|Z_{1:T}, U_{1:T}). \tag{2.5}$$

To use this approach in practice, estimates have to be calculated in *real-time* as the robot moves. In this setting, the solution at every time step has a lot in common so it would be wasteful to redo all the work at every step. Instead, previous results can be reused and only updated to account for the last odometry and measurement readings. This is known as filtering and has the following structure,

$$\cdots \to P_k(X_{0:k}, M|Z_{1:k}, U_{1:k}) \xrightarrow{Z_{k+1}, U_{k+1}} P_{k+1}(X_{0:k+1}, M|Z_{1:k+1}, U_{1:k+1}) \to \cdots \tag{2.6}$$

In some cases, real-time operation is not required, e.g. processing an old video. In this offline setting, the full problem can be solved at once, though it could be more computationally intensive. This is formulated as

$$\underset{X_{0:T}, M}{\arg\max}\, P(X_{0:T}, M|Z_{1:T}, U_{1:T}). \tag{2.7}$$

2.2 EKF SLAM

One of the seminal strategies to solve the filtering problem was proposed by Smith, Self and Cheeseman in 1987-1990 [33]. It follows the Bayesian inference approach, which is optimal when the model is known and exactly implemented.

The idea is simple: using the law of total probabilities and Bayes' theorem, the distribution $P_{k+1}(Y_{k+1}|Z_{k+1})$ can be optimally derived from the distribution $P_k(Y_k|Z_k)$ [34],

$$P_{k+1|k}(Y_{k+1}|Z_k) = \int P(Y_{k+1}|Y_k)P_k(Y_k|Z_k)\, dY_k, \tag{2.8}$$

$$P_{k+1}(Y_{k+1}|Z_{k+1}) = \frac{P_{k+1|k}(Y_{k+1}|Z_k)P(Z_{k+1}|Y_{k+1})}{P(Z_{k+1}|Z_k)}. \tag{2.9}$$

Equation (2.8) is known as the prediction step and moves the information from the previous pose to the new pose; equation (2.9) is known as the update step and incorporates new information from the measurements to the distribution.

These equations are optimal for any mathematical state process Y and measurement process Z of the form

$$Y_{k+1} = f(Y_k) \tag{2.10}$$

$$Z_k = h(Y_k), \tag{2.11}$$

and they underlie all the following filtering approaches. Note that both the motion model $f(\cdot)$ and the measurement model $h(\cdot)$ are arbitrary stochastic functions. This form is called a Hidden Markov Model (HMM). Y corresponds to the latent but unobservable state and Z is the observable measurements. Its graph representation can be seen in figure (2.2).

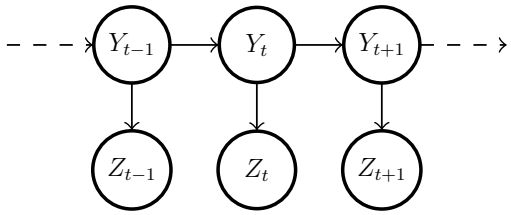


Figure 2.2: Hidden Markov Model.

Hidden Markov Models assume that the state Y_k only depends directly on its immediate predecessor Y_{k-1}, i.e. the state is history-blind given the previous state. Similarly, the measurements Z_k only depend directly on the current state Y_k. This is a big assumption, but it is reasonable in the context of SLAM.

Indeed, defining the state as the concatenation of the pose vector and the map landmarks, i.e. $Y_k = (X_k^T, M^T)^T$, the pose X_k only depends directly on the previous pose X_{k-1} and the odometry reading U_k in between, i.e. no matter what happened before, if the vehicle was at $(2, 3)\,[m]$ and moved by $(1, 2)\,[m]$, it is now at $(3, 5)\,[m]$ (the map components trivially only depend on themselves). The odometry reading imposes no problem because it is only a parameter of the motion model $f(X_k, U_k) = f_{U_k}(X_k)$ and the filtering equations for HMMs can deal with variant motion

models transparently. Similarly, the measurements only depend on the current pose and some landmark in the map, not in the historic trajectory of the pose.

For practical purposes, filtering equations (2.8) and (2.9) are too general and difficult to implement efficiently. The model can be simplified to the Kalman filter [35] by assuming Gaussian distributions for the state and the measurements and linear updates, i.e.

$$Y_{k+1} = \mathbf{F_k} Y_k + \mathbf{B_k} U_k + v_k \tag{2.12}$$
$$Z_k = \mathbf{H_k} Y_k + w_k, \tag{2.13}$$

where $\mathbf{F_k}$ is the linear motion matrix, $\mathbf{B_k}$ is the linear odometry matrix and $\mathbf{H_k}$ is the linear measurement matrix. The noise terms v_k and w_k are Gaussian distributed and follow $v_k \sim \mathcal{N}(0, \mathbf{Q_k})$ and $w_k \sim \mathcal{N}(0, \mathbf{R_k})$, where $\mathbf{Q_k}$ and $\mathbf{R_k}$ are the noise covariance matrices for the motion and measurement model respectively.

The Kalman filter only needs to propagate means $\bar{Y}_k$ and covariances $\mathbf{P_k}$ instead of relying in complex numerical approximations, giving a simplified prediction equation:

$$\bar{Y}_{k+1|k} = \mathbf{F_k} \bar{Y}_k + \mathbf{B_k} U_k \tag{2.14}$$
$$\mathbf{P_{k+1|k}} = \mathbf{F_k} \mathbf{P_k} \mathbf{F_k}^T + \mathbf{Q_k}, \tag{2.15}$$

and update equation:

$$D_k = Z_k - \mathbf{H_k} \bar{Y}_{k+1|k} \tag{2.16}$$
$$\mathbf{S_k} = \mathbf{H_k} \mathbf{P_{k+1|k}} \mathbf{H_k}^T + \mathbf{R_k} \tag{2.17}$$
$$\mathbf{K_k} = \mathbf{P_{k+1|k}} \mathbf{H_k}^T \mathbf{S_k}^{-1} \tag{2.18}$$
$$\bar{Y}_{k+1} = \bar{Y}_{k+1|k} + \mathbf{K_k} D_k \tag{2.19}$$
$$\mathbf{P_{k+1}} = (\mathbf{I} - \mathbf{K_k} \mathbf{H_k}) \mathbf{P_{k+1|k}}, \tag{2.20}$$

where D_k is the measurement innovation, $\mathbf{S_k}$ is the innovation covariance, $\mathbf{K_k}$ is the Kalman optimal gain, and $\mathbf{I}$ is the identity matrix. These equations are very efficient since matrix algebra is all that is required.

Despite its efficiency, the Kalman filters assume that the models are linear. This is a strong assumption and is not true in most SLAM applications, so an extension is required to work with non-linear dynamics. To solve this, Smith, Self and Chessman use the Extended Kalman Filter or EKF [34]. The models then become

$$Y_{k+1} = f(Y_k, U_k) + v_k \tag{2.21}$$
$$Z_k = h(Y_k) + w_k, \tag{2.22}$$

where $f(\cdot)$ and $h(\cdot)$ are arbitrary nonlinear model functions of motion and measurement respectively. The noise terms follow $v_k \sim \mathcal{N}(0, \mathbf{Q_k})$ and $w_k \sim \mathcal{N}(0, \mathbf{R_k})$ with $\mathbf{Q_k}$ and $\mathbf{R_k}$ being the covariances for the Gaussian motion and measurement noises respectively.

To manage the nonlinearities of the models, every time a linear matrix needs to be derived from it (analogous to $\mathbf{F_k}$ and $\mathbf{H_k}$ in the linear filter), the Jacobian of the non-linear models are used, $(Jf)_k$ and $(Jh)_k$.

The mean and covariance structure for the case of SLAM can be seen in figure (2.3). Both the estimate and the covariance are updated in time using the EKF equations. Note that to define an appropriate measurement model, a simplifying assumption that is made by this approach (and almost all others) is perfectly known *data association*. This means that when a measurement arrives, the system can identify to which map landmark it corresponds to, e.g. in the first time step in figure (2.1), the vehicle knows that the measurement was produced by the topmost landmark. The algorithm isn't responsible for figuring this out and takes it for granted. This is a critical assumption that can be dealt with in many ways and has its own section later in this chapter.

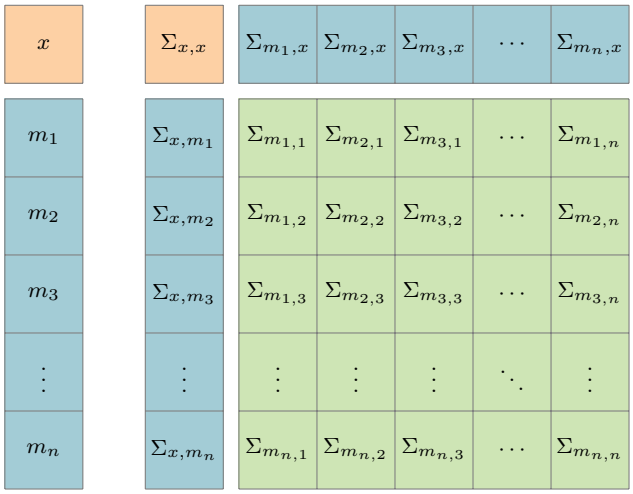


Figure 2.3: EKF vector and covariance representation.

As observed in figure (2.3), the space requirements are quadratic in the number of landmarks. This is a problem when dealing with large environments, since it becomes both too large to fit in memory and matrix operations become too slow as the map grows, so it's usually used on small maps of less than a couple hundred landmarks. Also, with time, numerical imprecision can affect the quality of the results and the linearization step may not be a good enough approximation if the models are highly non-linear.

2.3 FastSLAM

It is clear that the EKF solution can be improved. In 2003, Montemerlo and Thrun devised a major optimization on the problem called FastSLAM [36].

Their algorithm is based on the key insight that the problem can de decomposed in two sub-problems. Using the conditional independence properties that occur in the SLAM formulation, the desired probability distribution can be factorized as

$$P(X_{0:k}, M|Z_{1:k}, U_{1:k}, N_{1:k}) = P(X_{0:k}|Z_{1:k}, U_{1:k}, N_{1:k}) \prod_i P(m^i|X_{0:k}, Z_{1:k}, U_{1:k}, N_{1:k}), \quad (2.23)$$

where $N_{1:k}$ represents the known data association of the measurement at time k. For this decomposition to work, these associations are crucial, otherwise it becomes more cumbersome, as will be observed in later sections. Note that this approach assumes that one and only one landmark is

measured at each time step; this is only a mathematical covenience, multiple measurements can be processed sequentially.

From the factorization, it can be seen that every term is a distribution on only one of the unknowns. Solving separately for each one makes the process much more efficient. Each one of the landmarks will be filtered using Kalman filters, similar to the EKF model. Note however, that these factors are conditioned on the trajectory, which means that they require known localization. The trajectory term does not depend on the map so in theory would need to be marginalized over all possible maps.

To solve both complexities, a sequential Montecarlo method (or particle filter) is introduced [37]. The trajectory factor will be represented as a set of "particles" $\{s\}_i$, i.e. several hypothesized poses, which will follow the required pose distribution. Then, each particle has a known localization, so each particle can calculate all of the other terms efficiently.

For each localization, the map is represented as a vector of landmarks, expressed by their means μ_n^i and covariances Σ_n^i. Then, each landmark can be propagated independently as an extended Kalman filter.

There is still the question of how to make these particles follow the complex distribution of the pose. For this, at each time step, the system updates the particles using a stochastic motion model, i.e. it moves as expected from odometry and then adds Gaussian noise (these movements correspond to a *proposal distribution*) and then updates the landmarks for each particle. Finally, each particle is assigned an *importance weight*,

$$w_i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{P(X_{0:k}^i | Z_{1:k}, U_{1:k}, N_{1:k})}{P(X_{0:k}^i | X_{0:k-1}^i, Z_{1:k-1}, U_{1:k}, N_{1:k-1})}, \tag{2.24}$$

which through some mathematical work can be reduced to

$$w_i \propto \int \underbrace{P(Z_k | X_{0:k}^i, m^{N_k}, N_k)}_{\substack{\mathcal{N}(Z_k;\, h(X_k, U_k), \mathbf{R_k}) \\ \text{new measurement}}} \underbrace{P(m^{N_k} | X_{0:k-1}^i, Z_{1:k-1}, N_k)}_{\substack{\mathcal{N}(m^{N_k};\, \mu^i_{N_k,\,k-1}, \mathbf{\Sigma^i_{N_k}}) \\ \text{previous landmark estimate}}} \, dm^{N_k}. \tag{2.25}$$

Then, a new set of particles is sampled, picking sample s_i with probability proportional to w_i. From the definition of the weights, the new particles follow the desired distribution.

This way, the algorithm manages (in most cases) to greatly reduce the amount of work required. Where EKF SLAM memory requirements were quadratic in the number of landmarks in the map, $O(M^2)$, FastSLAM only requires $O(KM)$, where K is the number of particles. The processing is similarly reduced, since matrix algebra only deals with small dimensions. If trees are creatively used as data structures, the complexity can be even reduced to $O(K \log M)$. This allows the algorithm to deal with maps as large as 50,000 landmarks (at the time of publishing). Also, since every particle is more-or-less independent, different data associations could be introduced as in the case of MHT-FastSLAM [38].

The algorithm makes good use of the conditional independence of the map given the trajectory, but this separation can introduce numerical deviations. In EKF SLAM, the covariance between every landmark was clearly stored and updated; in FastSLAM, only the conditioned diagonal covariances are explicitly stored, the full covariance is implicit in the particle distribution. This means

that the particles have to be numerous enough to accurately track the distribution. Even further, as particles are stochastic in nature, they may slowly lose information, which could render the solution less accurate in long runs. Discussion on this issues will be expanded later as they are common to filtering approaches.

As a side note, this approach seems to decouple the stages of localization and mapping, which could appear to be inconsistent with the idea of *simultaneous* localization and mapping that was deemed so important at the beginning of the section (after all, this was the key insight that solved the problem in the first place). However, it is not so: both stages are very coupled through the importance resampling step, which takes information from the map estimation back to the localization step, which then gives information back to the map when updating its particles (and their poses).

In a subsequent work by the authors [39], they make a small but important change on the solution. In the resampling step, instead of using the motion model as proposal distribution, i.e. using the last odometry reading, they use both the motion and measurement models, i.e. using both the last odometry and measurement readings. Though theoretically the original algorithm is correct, it is wasteful since a bad odometry reading would give very poor particles, unnecessarily increasing the number of particles required to continue correctly. The new algorithm is much more efficient as the measurement can easily guide the proposal. The new weights are calculated as

$$w_i \propto \int\int \underbrace{P(Z_k|X^i_{0:k}, m^{N_k}, N_k)}_{\substack{\mathcal{N}(Z_k;\, h(X_k,U_k),\, \mathbf{R_k}) \\ \text{new measurement}}} \underbrace{P(m^{N_k}|X^i_{0:k-1}, Z_{1:k-1}, N_k)}_{\substack{\mathcal{N}(m^{N_k};\, \mu^i_{N_k},\, \mathbf{\Sigma^i_{N_k,\,k-1}}) \\ \text{previous landmark estimate}}} \underbrace{P(X_k|X^i_{0:k-1}, U_{1:k-1})}_{\substack{\mathcal{N}(X_k;\, \bar{X}^i_{k-1},\, \mathbf{P^i_{k-1}}) \\ \text{previous pose estimate}}} dm^{N_k} dX_k. \tag{2.26}$$

Using a linear approximation in the measurement model (not necessarily in the motion model),

$$h(X_k, m^{N_k}) \approx \bar{Z}^i_k + \mathbf{H_m}(m^{N_k} - \mu^i_{N_k,\,k-1}) + \mathbf{H_x}(X_k - \bar{X}^i_k), \tag{2.27}$$

the weight can be associated to a Gaussian with mean $\bar{Z}^i_k$ and covariance

$$\mathbf{H_x P_k H_x}^T + \mathbf{H_m \Sigma^i_{N_k,\,k-1} H_m}^T + \mathbf{R_k}. \tag{2.28}$$

Using this new approach the required number of particles is an order of magnitude less, requiring less memory and computation time. Note that this improvement has no practical disadvantage so should always be preferred over the original algorithm.

2.4 GraphSLAM

Although filtering approaches work well when real-time constraints are imposed, their theoretical optimality assertions may not hold because of numerical approximations needed to run them in reasonable times. If an operation will be performed offline, it seems useful to have slower but more accurate algorithms, which look at the entire data and optimize it.

There is a group of similar-spirited methods that use the key notion of conditional independence between different elements in SLAM to formulate the problem in the context of graphs, the

mathematical framework for networks and connectedness. These will be referred as GraphSLAM algorithms.

Using this viewpoint the sparsity of the problem is exposed and exploited. In fact, this exploitation can be efficient enough as to let some of these algorithms run in real-time. This may seem counter-intuitive, since they are solving a much bigger problem, but as it turns out, by having all the information *available* they may be able to make better decisions instead of having to generate a lot of redundancy to cope with the unknown future. The system may just try the best candidate in a difficult transition and just correct it later, when better information is available. Filtering approaches do not have this luxury so have to be cautious and prepared for all scenarios. Besides, although new information can change the past, it usually will only affect a very small part of it (the near past), so most of the time it is not really necessary to update most of the solution. GraphSLAM makes it possible to attain these benefits efficiently.

Thrun's GraphSLAM

In 2005, Thrun and Montemerlo used the natural sparsity of the SLAM problem by maintaining the system information in an adequate matrix-vector pair called the information matrix Ω and vector ξ. Unlike EKF-SLAM, which also uses a matrix-vector pair (joint covariance Σ and mean μ), GraphSLAM's pair directly corresponds to the gathered information; every entry can be associated to a particular constraint, corresponding to a particular odometry or measurement reading. As new information comes in, the structure is filled with new entries directly, no need for complex propagation, i.e. it is additive. In particular for each constraint of the form

$$(z - h(\mu, \theta) - H\mu)^T \mathbf{Q^{-1}} (z - h(\mu, \theta) - H\mu), \tag{2.29}$$

Ω and ξ corresponding entries are changed

$$\Omega_{x,y} = \Omega_{x,y} + H^T Q^{-1} H, \tag{2.30}$$

$$\xi_{x,y} = \xi_{x,y} + H^T Q^{-1} (z - h(\mu, \theta) - H\mu) \tag{2.31}$$

where the indices indicate submatrix. The system needs to be linearized for this constraints to be applicable.

This representation has a couple of advantages. On one hand, very few entries are added at each step (compared with the total number of entries in the matrix), i.e. the matrix is very sparse. This means lower memory requirement to run the system. Also, the update routine is simple and does not require complex matrix algebra that may take too much time (in EKF-SLAM, propagating the covariance becomes impractical for large systems). On the other hand, since every constraint is explicitly written in the information matrix, all the problem information is preserved through time, unlike filtering approaches which marginalize the past, making it inaccessible and which are prone to numerical drift since covariance propagation require many multiplications.

The fundamental question that remains to be solved is how to extract the usual covariance and mean from the information matrix. It turns out that both representations are related by the equations $\mu = \Omega^{-1}\xi$ and $\Sigma = \Omega^{-1}$. If the SLAM graph is a tree, it is easy to extract the mean for every variable, but under cyclic graphs (i.e. revisited landmarks), the general matrix inversion is too slow. To speed

things up, the authors propose to compact the information matrix by removing each landmark one by one from the problem. This means removing their corresponding column and row and adding new constraints between any nodes that were connected through the removed landmark (note the difference with the covariance, whose marginalization only requires removing the columns and rows). This procedure is nothing else than a variable elimination from linear algebra. Note that landmarks are eliminated, the matrix becomes more and more dense, since every time each variable has more neighbours. At the end, although the matrix may be quite dense, the number of variables is much smaller than at the start, since only the poses remain. Therefore, the matrix information becomes reasonable and a Maximum A Posteriori (MAP) estimate for the trajectory can be derived. A similar elimination is performed for each landmark with its neighbours, using the previous MAP estimate, deriving an estimate for each landmark.

Finally, the authors propose a novel data association scheme, where each measurement is associated to a new landmark and a statistical test is performed on close landmarks to merge them. To do this, the joint probability distribution between close-by landmarks is calculated and then used to find the distribution of the distance between the landmarks. From this distance distribution, the likelihood of zero distance $P(|m^1 - m^2| = 0)$ is computed and compared to a predetermined threshold ϵ to decide whether to merge or not, i.e. merge if $P(|m^1 - m^2| = 0) > \epsilon$. These decisions are not final, they may be changed if new information reveals they were suboptimal.

Square-Root Smoothing and Mapping

Dellaert and Kaess showed in 2006 [40] that SLAM can benefit greatly from efficient methods in the fields of sparse linear algebra and graph theory, from where they formulate their method, Square-Root Smoothing and Mapping ($\sqrt{\text{SAM}}$).

They factorize the full probability density as

$$P(X_{0:k}, M, Z_{1:k}) = P(x_0) \prod_{k=1}^{T} P(X_k|X_{k-1}, U_k) \prod_{k=1}^{M} P(Z_k|X_{i_k}, m^{j_k}), \tag{2.32}$$

where association is assumed known, so X_{i_k} is the pose associated to the measurement Z_k and L_{j_k} is the corresponding landmark. Note that due to Bayes' rule, maximizing the joint probability is equivalent to maximizing the original conditional probability,

$$\arg\max_{X_{0:k},M} P(X_{0:k}, M|Z_{1:k}) = \arg\max_{X_{0:k},M} P(X_{0:k}, M, Z_{1:k})/P(Z_{1:k}) = \arg\max_{X_{0:k},M} P(X_{0:k}, M, Z_{1:k}). \tag{2.33}$$

The factorization shown in equation (2.32) can be matched to a factor graph, where every term corresponds to an edge (as shown in figure (2.4)). There are three kinds of terms: one unary factor which encodes the prior information on the pose; binary factors in the first product, which relate consecutive poses (the motion model); and binary factors in the second product, which relate poses and landmarks (the measurement model).

Assuming Gaussian distributions for every term, the problem can be transformed into a least

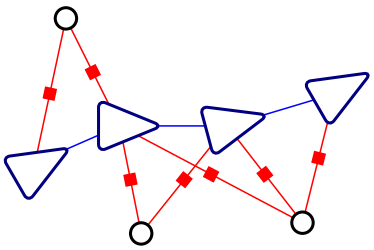


Figure 2.4: Factor graph representation of the SLAM problem. Every red square represents a factor.

squares optimization problem,

$$\arg\max_{X_{0:k},M} P = \arg\min_{X_{0:k},M} -\log P \tag{2.34}$$

$$= \arg\min_{X_{0:k},M} \Big\{ \sum_{k=1}^{T} ||f_k(X_{k-1}, U_k) - x_k||^2_{\mathbf{\Lambda_k}} + \sum_{k=1}^{K} ||h_k(X_{i_k}, m^{j_k}) - Z_k||^2_{\mathbf{\Sigma_k}} \Big\}, \tag{2.35}$$

where $f_k(\cdot)$ is the motion model, $h_k(\cdot)$ is the measurement model and $\mathbf{\Lambda_k}$ and $\mathbf{\Sigma_k}$ are their respective noise covariance matrices. Then it may be linearised,

$$\begin{aligned} \arg\max_{X_{0:k},M} P \approx \arg\min_{X_{0:k},M} \Big\{ & \sum_{k=1}^{T} ||\mathbf{F_k}^{k-1} \delta X_{k-1} + \mathbf{G_k}^{k} \delta X_k - a_k||^2_{\mathbf{\Lambda_k}} \\ & + \sum_{k=1}^{K} ||\mathbf{H_k}^{i_k} \delta X_{i_k} + \mathbf{J_k}^{j_k} \delta m^{j_k} - c_k||^2_{\mathbf{\Sigma_k}} \Big\} \end{aligned} \tag{2.36}$$

$$= \arg\min_{\delta} ||\mathbf{A}\delta - b||^2_2, \tag{2.37}$$

where $\mathbf{F_k}^{k-1}$ is the jacobian of the motion model, $\mathbf{G_k}^{k}$ is the negative of the identity matrix, $\mathbf{H_k}^{i_k}$ is the jacobian of the measurement model with respect to the pose, $\mathbf{J_k}^{j_k}$ is the jacobian of the measurement model with respect to the landmark and a_k and c_k are bias terms that come from the motion and measurement models respectively. Since the whole expression is a sum of squared norms of linear combinations, it can be succinctly written as the norm of one linear combination of all the poses and landmarks, as written in equation (2.37). Here δ corresponds to the concatenation of all the pose deltas and landmark deltas in the linearised equation and $\mathbf{A}$ and b have the form

$$\mathbf{A} = \begin{pmatrix} \mathbf{G_1^1} & & & & \\ \mathbf{F_2^1} & \mathbf{G_2^2} & & & \\ & \mathbf{F_3^2} & \mathbf{G_3^3} & & \\ \mathbf{H_1^1} & & & \mathbf{J_1^1} & \\ \mathbf{H_2^1} & & & & \mathbf{J_2^2} \\ & \mathbf{H_3^2} & & \mathbf{J_3^1} & \\ & & \mathbf{H_4^3} & & \mathbf{J_4^2} \end{pmatrix} \qquad b = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}, \tag{2.38}$$

where the Mahalanobis norms have been removed by using the identity $||e||^2_{\Sigma} = ||\Sigma^{-T/2} e||^2_2$.

This is a very large but sparse optimization problem. The solution can be found by solving a linear system,

$$\mathbf{A}^T \mathbf{A} \delta^* = \mathbf{A}^T b. \tag{2.39}$$

As in any least squares problem, it can be solved robustly using Cholesky or QR decomposition [41] on the information matrix $\mathcal{I} = \mathbf{A}^T\mathbf{A} = \mathbf{R}^T\mathbf{R}$, with straightforward back-substitution.

Note that Cholesky decomposition uses the information matrix $\mathcal{I} = \mathbf{R}^T\mathbf{R}$ while QR decomposition uses $\mathbf{A} = \mathbf{QR}$ itself. Both algorithms have an approximate time complexity $O(mn^2)$ as a function of the size of $\mathbf{A}$, $m \times n$, but QR decomposition is twice as slow [40].

To improve the performance of the algorithm, it is useful to show the implicit relationship between this algebraic treatment and the underlying graph structure of the problem. The Jacobian matrix $\mathbf{A}$ is the adjacency matrix for the factor graph associated with the model, and the QR decomposition is equivalent to removing edges and generating a new graph (the junction tree) along the way, corresponding to the $\mathbf{R}$ matrix. By removing edges, variables are eliminated; new links are introduced to "compensate", as shown in figure (2.5)[†].

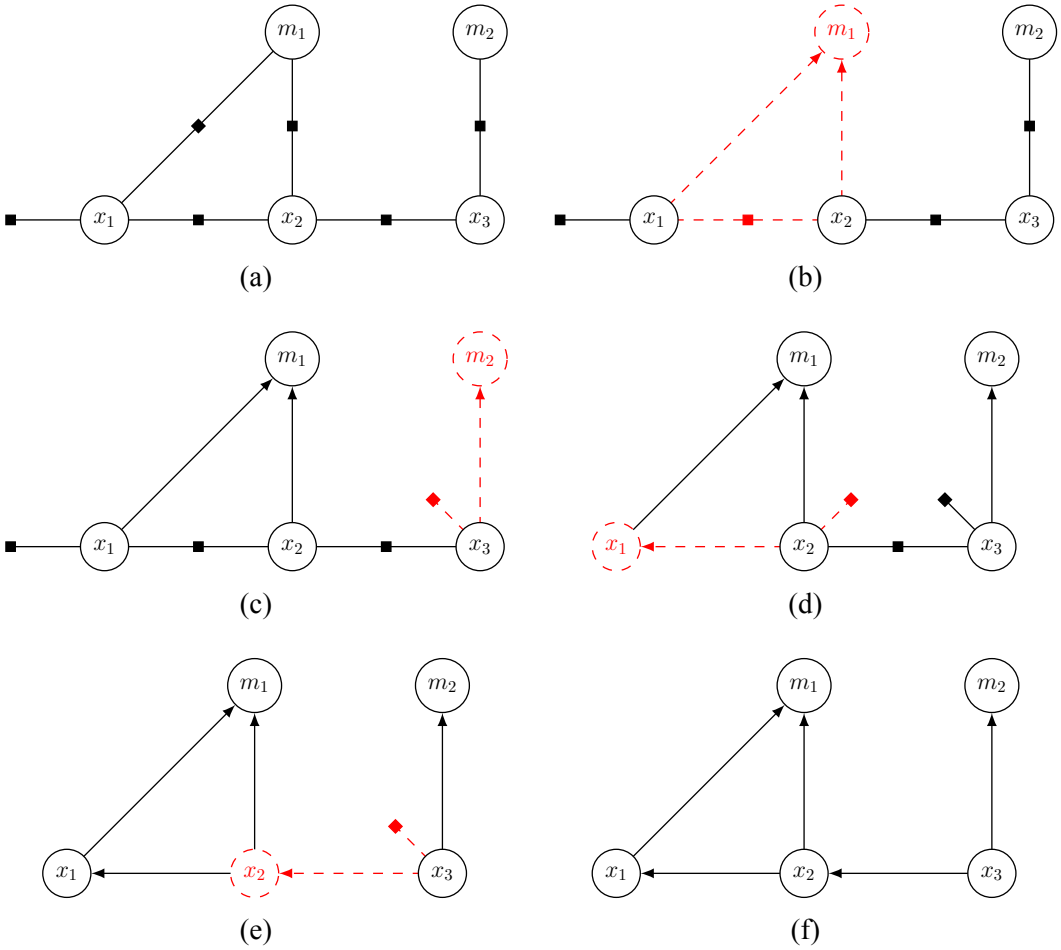


Figure 2.5: Variable elimination procedure. As the procedure removes nodes, a new directed graph associated to R is generated (as shown in [43]).

The order in which the variables are eliminated changes the sparseness of the final junction tree, and correspondingly, of the square root matrix R. So, if a good order can be selected, the result will be more amenable to the ensuing computation and the solution to the problem will be

[†]These operations are well known in the field of probabilistic graphical models [42].

efficient. Unfortunately, finding the optimal variable ordering is NP-complete; nevertheless, there are very good heuristic approaches which attain good results in practice (Square Root SAM uses the COLAMD algorithm).

The performance of the algorithm is much better than EKF, which requires quadratic memory growth in the number of stored features. If the ordering is good, $\sqrt{\text{SAM}}$ can grow linearly with the number of poses and landmarks. This by itself allows it to cope with much longer runs, but still grows without bound if the vehicle continues to move. Also, the optimization can be iterated multiple times using the previous solution as an initialization. This way the algorithm may relinearize variables and the linearization points become less constraining as they can be changed later.

Kaess, Dallaert et al. continued developing this technique in two subsequent algorithms, namely iSAM [44] and iSAM2 [43]. These algorithms focus the solution in the online real-time case. This is not to say they transformed it into filtering, but made further optimizations to allow the solutions to be incrementally improved frame by frame. After performing the full $\sqrt{\text{SAM}}$ algorithm, the information from the new frame can be incorporated much more efficiently than redoing all the optimization from scratch. In iSAM, a QR-updating method is proposed, where the new information is appended to the R factor (violating its triangular structure) and then correcting it by using Givens rotations, which solve QR entry-by-entry. Since the new data is small and sparse, this process is efficient.

This makes the algorithm run in $O(1)$ for most time steps, making it suitable for real-time operation. In loop closures, however, this can become much bigger as a lot of poses are affected (theoretically, the upper bound is $O(n^{1.5}) - O(n^3)$, but empirically the run-times are lower [43]). Note however that using this incremental approach the system slowly but steadily increases the amount of fill-in in the R matrix, its sparsity is lost; Givens rotations are blind to this effect since they are a general purpose tool. To correct this, the algorithm periodically reorders the variables to make the matrix sparse again; the more fill-in in R, the more this will take, so a trade-off on the spacing between such re-orderings must be balanced.

In iSAM2, the authors propose a new data structure called the Bayes tree, which is derived from the Cholesky and QR factorization algorithms and allows for better manipulation the graph as new information arrives. The main insight of this work is that all the operations that were usually implemented using matrices can be expressed as edits on the Bayes tree.

Storing the R matrix in this tree representation leads to more opportunities for efficiency. In fact, the new information can only affect the path of the tree that joins the nodes of the involved variables; Similarly, when adding new nodes only this critical path may require variable reordering, so it can be performed right after the inference step and only in the affected nodes.

The relinearization process no longer needs to be carried out on the whole structure. Instead every variable keeps track of its linearization quality status and if a relinearization is deemed necessary, the node is removed, relinearized from the original non-linear factor and reinstated in the tree along with all the factors that were changed in a non-negligible way, i.e. a continuous and fluid relinearization.

Note that the original factorization (from which the entire family of algorithms follows) can only be carried out with knowledge of the data associations between measurements and landmarks; otherwise the measurement factors would not be single terms, but sums over all possible associa-

tions, which would defeat the tractable nature of the subsequent optimization. Measurement factors would become $P(Z_k|X_{i_k}, M) = \sum_\theta P(Z_k|X_{i_k}, m^{j_\theta})P(\theta)$ (where θ is the data association), which then become log-sum-exp terms when taking the logarithm, $\log \sum_\theta \exp(||h_k(X_{i_k}, m^{j_\theta}) - Z_k||^2_{\Sigma_{k,\theta}})$. The whole expression is no longer a least squares optimization and the number of terms grows with the number of possible data associations.

2.5 RFS-SLAM

Every approach introduced so far represents the map as a vector of landmarks, which implies both a cardinality and an order for the map. If data association is known, the cardinality can be justifiable, but the order of the landmarks is always arbitrary. When data association is not known, the cardinality is uncertain and the algorithms require ad-hoc heuristics to force data associations on the measurements.

A more principled approach would be to treat the map and the measurements as what they really are: sets of points. The Random Finite Sets approach follows this concept [45], making few compromises in theory as the inherent set-like nature of the variables is maintained.

Using this idea, RFS-SLAM can propose a generalized [11, 46, 47] context in which to perform Bayesian updates or batch processing. The methodology is similar to other algorithms, but replaces vectors for sets where appropriate. The problem becomes how to define and implement the analogous operations on sets, e.g. extracting covariances or taking derivatives. This is by no means straightforward, but compact useful expressions can be derived.

Note that by using sets this approach solves data association through the SLAM solver itself and thus does not require external means. A similar benefit occurs for map management.

Now, solving the complete data association problem is combinatorial in nature, so no miracles can be expected in the worst case scenario: a more powerful method will require more resources on these extreme cases. In practice, though, data association is not completely uncertain; in fact uncertainty usually distributes at most over a few candidate landmarks, not the entire map, so the methodology is efficient. Moreover, the problem of representing association uncertainty becomes just a matter of numerical approximation and optimization; it can be dealt with using appropriate "resolution" thresholds on the implementation. Using its knowledge of the association situation, the concept has the potential for flexibility, using strict association policies when the data is clean, thus faster, and relaxing these policies when the data becomes corrupted and noisy, thus slower but more accurate.

As this methodology is the base for the present work, details and a more in-depth review will be studied later in this chapter.

3 SfM and SLAM

As will be shown here, both SfM and SLAM essentially solve the same problem with different methods. While the visual setting doesn't impose any defined order to the set of images, SLAM is

presented as a sequential problem. SfM's core step involves a huge dense quadratic optimization which becomes very hard to compute as the number of images increases. In SLAM there are many ways to simplify and guide the solution, making use of the structured nature of the problem, its sparsity and potentially marginalizing states out from calculations. Visual odometry is in the middle, using relationships between consecutive frames but disregarding global consistency requirements, which take too much time to fulfil.

This exposes the major difference between visual odometry and visual SLAM: the first one concentrates on the local scale, improving the coherence frame-by-frame; the second one includes the global scale, providing long-term effects such as loop closure, where the system can change the estimate for the pose trajectory using disparate landmark relationships. This also has implications on the processing time of each approach, which is important if an application needs to run in real-time.

Another big difference is that SLAM typically propagates the state covariance, which is usually absent in SfM methods. Knowing the uncertainty of an estimate is very relevant to evaluate the results: a decision may change depending on whether the navigator knows its position with high confidence or it is just pure guesswork.

Their differences do not mean both approaches can't complement each other. In fact, visual odometry can be used as a building block of a larger visual SLAM algorithm. In this respect, the advantages of full SLAM can be perceived, resulting in a better global estimate.

4 Visual SLAM

There have been some successful efforts to create visual SLAM solutions based on the previously exposed algorithms. Some of the most interesting are reviewed below.

In 2007, Davison et al. published MonoSLAM [48], where they used EKF-SLAM with monocular cameras (i.e. a typical single RGB camera). To avoid expensive tracking, they used active search, where landmarks are only matched near their expected pixel position in the image stream. To deal with the missing depth information, they represented a landmark as two objects: a point and a direction (this direction defines a line). At first the landmark may be anywhere along the line, but as different frames show the landmark from different viewpoints, they can be fused to bound the depth uncertainty; in particular, they use a particle filter over the line, i.e. put a number of hypothesis along it and remove them if they become inconsistent in consecutive frames. As this method uses EKF, it requires careful map management to maintain reasonable processing times and memory requirements.

In 2006, Eade and Drummond proposed Scalable Monocular SLAM [49], a work similar to Davison's but using FastSLAM instead. To diminish the effects of the missing depth information, they recover the depth before adding the landmark to the map, by using a separate Kalman filter over the landmark location in pixel coordinates (from the first frame it was seen) and an inverse depth distance (inverse because it follows a Gaussian more closely than the distance itself). After seeing the landmark enough times from different points of view, the depth uncertainty decreases and its 3D mean can be added as landmark. Through the time these landmarks are not-yet-added candidates,

they still give useful information about the camera pose, which the authors use to constrain it. Indeed, as a side effect of computing the inverse distance, the algorithm computes the epipolar reprojection error, which can then be incorporated as an artificial observation.

Alternatively, in their algorithm Parallel Tracking and Mapping (PTAM) [50], Georg Klein and David Murray approach the problem using bundle adjustment (BA, from SfM), exploiting the multi-threaded environment of modern processor architectures to separate mapping and camera tracking. By using separate threads, tracking does not need to wait for the lengthy mapping stage and therefore may use more time-consuming processing. Also, mapping is very redundant as consecutive frames show similar features from similar viewpoints and it can be performed sporadically, only in keyframes sufficiently distinct from one another, so it needs not be strictly real-time; it only has to be completed before the next keyframe. With this extra time, the incremental mapping strategies can be replaced for a heavier bundle adjustment. The tracking thread reprojects map points to the image to estimate the camera movement and uses active search to minimize calculations, but does nothing with unassociated measurements. The mapping thread associates keyframe measurements (using an epipolar search) and performs global BA, which takes cubic time with respect to features, $O(|M|^3)$. This becomes unacceptable in bigger maps, so the system falls back into a local BA, which only considers a subset of the keyframes (spatially close keyframes) if new features have been found (exploring a new area). This local version takes $O(T|M|)$ time, i.e. bilinear in pose number and map size. Note that if well-explored areas, the full version can run slowly, but it doesn't matter since tracking does not wait for it, and features are not expected to change much (static map). Also, should the mapping thread find itself idle, it may use the extra time to revisit old frames more carefully. To initialize the map, user cooperation is required by actively indicating good and easy pairs of images to perform a stereo reconstruction (five-point algorithm and BA).

PTAM works very well under constrained scenarios, but does not scale to larger environments. Mur-Artal, Montiel and Tardós extend some of its ideas and propose ORB-SLAM [51]. They augment the tracking/mapping solver with relocalization and loop closure stages. To perform such loop detection, they use place recognition through the Bag-of-Words concept [52]. Similarly to PTAM, tracking and mapping are in separate threads, and additionally a third thread performs loop closure; unlike PTAM, mapping is always solved with local BA. Global consistency is taken care by the loop closure thread: when it recognizes a familiar place, it aligns the endpoints and optimizes similarity constraints over the spanning tree of the strong covisibility graph (where every feature is connected if they are simultaneously visible in a keyframe) which they name *essential graph*, resulting in much faster times. They use ORB features instead of patch-correlation FAST from PTAM, which are extracted using the same procedure but filtered and matched differently; ORB features have the advantage of being useful in all the stages of the algorithm. Also, they are more liberal when adding features to the map, but they are filtered out if they do not pass well-behavior tests in the first few frames; similarly, redundant keyframes are also removed. This culling means the algorithm can be used in lifelong processing, since navigation in a familiar environment will not increase the map size nor the keyframe count. Unlike the PTAM approach, ORB-SLAM checks for degenerate cases when initializing the map and uses an alternative method in such case.

The success of works like PTAM prompted Strasdat to review the merits of filtering and batch approaches [53], concluding that batch methods make better use of the available time and memory resources.

4.1 Alternative features

Some authors have explored features different than points. Many [54] have used edges, others have used planes [55]. Both of these features can be more robust than keypoints since they represent higher-level structures, spanning over larger areas, but this also means they can be partially occluded, complicating detection. Salas et al. [56] goes a step further and uses complete objects from a previously prepared library as features, with complex bag-of-words matching to recognize partial objects. Note that using prepared models may not be plausible in unexplored environments. Also note that these features sometimes can be parametrized as keypoints in a higher dimensional space.

Following the recent trend in computer vision, some authors [57, 58, 59, 60, 61] have explored algorithms that work directly on the pixel representation, without any preprocessing to extract relevant features. These methods can reconstruct denser representations of the scene, since they are not constrained to work on pointwise information. Being able to run this kind of methods in real-time has been possible thanks to the major advances in the highly parallel computer architectures of GPUs.

In 2011, Newcombe et al. presented Dense Tracking and Mapping (DTAM), a direct dense reconstruction method. To do it, the scene is discretized into a 3D *cost* cubic voxel grid $\mathbf{C_r}$ as shown in figure (2.6) and as new keyframes are observed the cube values are updated to reflect the likelihood that an element in the cube is occupied by computing its consistency with the images, e.g. a voxel which projections into both images have different colors is probably not occupied. To find the values in the cube, the authors optimize the photometric errors of the volume projection into the images. Later in the tracking stage, the scene is aligned with the camera image at each frame to calculate its movement. As the authors explain, the system assumes constant lighting and its reconstruction space is fixed by the first keyframe, which limits its use to lab conditions.

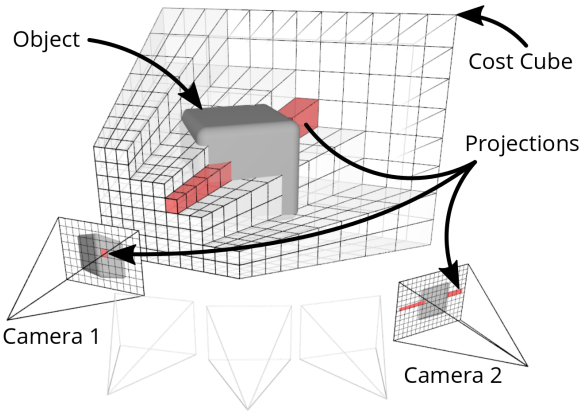


Figure 2.6: Cost cube discretization in the DTAM algorithm. Image taken from Newcombe's 2011 publication [58].

Three years later, Engel et al. continued the idea of direct reconstruction with their Large-Scale Direct Monocular SLAM (LSD-SLAM) [61]. In this work they propose a semi-dense scheme, which uses pixel information but only updates the most meaningful areas of the 3D model such as edges. The map reconstruction is similar otherwise to the Newcombe's idea (although they represent the inverse depth map as mean-covariance for each pixel instead of a discretized inverse

depth likelihood). To generalize the mapping to huge environments, the reference keyframe (which is fixed in Newcombe's method) can be updated by projecting the depth map into a new keyframe, so the system may still work if the original viewpoint is no longer visible. Special care is taken with scale drift, which is major source of error in monocular SLAM, by forcing the inverse depth map mean to be one and therefore moving any scale ambiguity into the camera tracking stage, which can use a regularized error to minimize scale-drift. Appearance-based matching is used with closest keyframes to perform loop closure. The system runs in real-time on a CPU.

In principle, these dense schemes should be able to always perform better since they are presented with more information, but removing all the redundancies is difficult and time-consuming and current state-of-the-art schemes are both slower and less accurate for camera tracking than keypoint-based alternatives [62]. Conversely, the reconstructed surfaces are denser, therefore easier to interact with since extrapolating the complete surface from points in space is not trivial.

It is important to note the general limitations of current methodologies for visual SLAM. Visual sensors are very sensitive devices and can present bad visibility in dark scenes, ISO noise, codec artifacts, defocusing, vibrations, tearing, sync inconsistencies, ... Moreover, all these problems can rapidly change as the environment changes, so any system needs to be able seamlessly adapt and reconfigurate to new conditions. For example, in slow motion and daylight, data association should only consider few matchings, while at night in a high speed vehicle it should consider more remote matches.

5 Estimation with Random Finite Sets

Given its qualities as an self-reliant concept, RFS-SLAM could be a good candidate to reduce the limitations that current visual systems have. This work will indeed present at length the implementation of such an extension, so a review of the theory behind the RFS concept is given here, which has been extensively developed by Mahler [63].

Within this method, map and measurements are sets of points. They will be noted in cursive to distinguish them to their vector counterparts, so $\mathcal{M}$ and $\mathcal{Z}_k$ are the map and measurement set respectively. This is illustrated in figure (2.7), where the map is represented as $\mathcal{M} = \{(0,1)^T, (1,1)^T, (1,-1)^T\}$ and the measurements as $\mathcal{Z} = \{(1.1, \pi/4)^T, (1.2, -\pi/4)^T\}$.

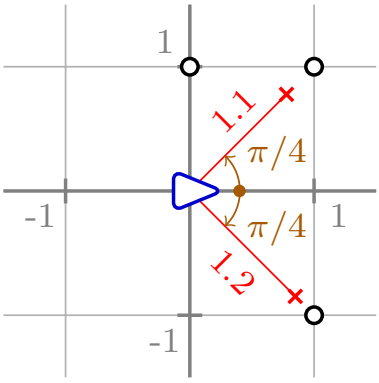


Figure 2.7: Set-based landmarks (white circles) and measurements (red crosses) in RFS.

These sets contain a finite number of points and their stochastic nature is imbued in a probability distribution over the set of finite sets $\mathcal{F}_{\mathcal{K}} = \{\mathcal{X} \in 2^{\mathcal{K}} : |\mathcal{X}| < \infty\}$, where $\mathcal{K}$ is the vector space in which the points exist and $|\cdot|$ is the counting metric. This space is huge.

Both filtering and smoothing can be re-expressed using these sets. For the filtering case [45], the governing equations become

$$\begin{aligned} P_{k|k-1}(X_{0:k}, \mathcal{M}_k) &= f_X(X_k|X_{k-1}, U_k) \\ &\quad \times \int f_{\mathcal{M}}(\mathcal{M}_k|X_k, \mathcal{M}_{k-1}) P_{k-1}(X_{k-1}, \mathcal{M}_{k-1}) \delta\mathcal{M}_{k-1} \end{aligned} \tag{2.40}$$

$$P_k(X_{0:k}, \mathcal{M}_k) = \frac{h_k(\mathcal{Z}_k|X_k, \mathcal{M}_k) P_{k|k-1}(X_{1:k}, \mathcal{M}_k)}{h_k(\mathcal{Z}_k|X_0, \mathcal{Z}_{1:k-1})}. \tag{2.41}$$

and for smoothing,

$$\underset{X_{0:T}, \mathcal{M}}{\arg\max}\, P(X_0, \mathcal{M}) \prod_{i=1}^{T} P(\mathcal{Z}_i|X_i, \mathcal{M}) P(X_i|X_{i-1}, U_i). \tag{2.42}$$

The main difference here is the presence of operators such as integration and optimization over the space of sets and the unavailability of the simplifying data association assumption.

Note that this approach is very related to the concept of point processes [64] in probability theory. Although the SLAM problem can be expressed in such a manner, the RFS conceptualization is more aligned to the rest of the SLAM algorithms (e.g. EKF-SLAM, FastSLAM, GraphSLAM) so it allows for a common language.

5.1 Filtering in the RFS framework

The first thing to do is define what will be the motion and measurement models using sets, i.e. how does the latent model evolve in time and how do measurements relate to it.

Note that there is a prime distinction between the map state (set-based) and the trajectory (vector-based); since the pose is always known to exist, no special treatment should be required to manage it. The more complex part is mapping since features may or may not exist. Therefore, the prediction of the trajectory follows a distribution similar to the non-set approaches,

$$f_X(X_k|X_{k-1}, U_k). \tag{2.43}$$

On the other hand, to model the map a set update equation is needed. In general, four things can occur in the map: a landmark can move, it can spawn additional landmarks, it can spontaneously appear in the map or it can die. This is captured by the RFS map update equation [65],

$$\mathcal{M}_k = \left[\bigcup_{\zeta \in \mathcal{M}_{k-1}} S_{k|k-1}(\zeta)\right] \cup \left[\bigcup_{\zeta \in \mathcal{M}_{k-1}} B_{k|k-1}(\zeta)\right] \cup \Gamma_k, \tag{2.44}$$

where $S_{k|k-1}(\zeta)$ corresponds to the movement of the landmark ζ in the map (and incorporates the death probability), $B_{k|k-1}(\zeta)$ to new landmarks that spawn from ζ and Γ_k to spontaneous births of

new landmarks. For the purposes of SLAM, though, only movement and births are relevant (no spawning). When the map is static (very usual), there's no movement either.

To model the measurements, there are two effects at play: the sensor will pick up real landmarks with some spatial and cardinality uncertainties (it may or may not see a landmark); and it may also record spurious readings which do not correspond to anything in the world, e.g. pixel noise. This is written as the measurement equation,

$$\mathcal{Z}_k = K_k \cup \left[\bigcup_{\zeta \in \mathcal{M}} \Theta_k(\zeta) \right], \tag{2.45}$$

where K_k is a spurious set and $\Theta_k(\zeta)$ the set of measurement from landmark ζ.

These models specify all the relevant information to compute the filtering equations in (2.40), but unfortunately are intractable to solve. Nevertheless, the equations can be simplified [63] given certain assumptions. First, for each RFS $\mathcal{X}$, an *intensity* v can be defined in the underlying $\mathcal{K}$ space such that

$$\int_S v(x)\, dx = E(|\mathcal{M} \cap S|) = \int |\mathcal{M} \cap S|\, P(\mathcal{M}) \delta\mathcal{M}. \tag{2.46}$$

It is a density which when integrated over a region, gives the expected number of landmarks contained in that region. It is also the first moment statistic of the set distribution, i.e. analogous to the mean in vector processes.

Second, an RFS is a Poisson RFS if it fulfils two properties: its cardinality distribution is Poisson with a given mean N and, conditioned on a given cardinality, the elements of the set are i.i.d. with the same distribution $v(\cdot)/N$, where $v(\cdot)$ is the intensity. Though a strong assumption, this model works well in practice. Less stringent alternatives have also been explored [66].

It turns out that for Poisson RFS, all statistics are captured by their intensity function (also referred as Probability Hypothesis Density), so the only thing that needs to be propagated in the filtering step is this PHD. This is the root of the PHD filter, which is able to track the complete set model efficiently.

The PHD equations are [63]

$$v_{k|k-1}(m|X_{0:k}) = v_{k-1|k-1}(m|X_{0:k-1}) + b(m|X_k) \tag{2.47}$$

$$\begin{aligned} v_k(m|X_{0:k}) = v_{k|k-1}(m|X_{0:k}) \Bigg[& 1 - P_D(m|X_k) \\ & + \sum_{z \in \mathcal{Z}_k} \frac{P_D(m|X_k) h_k(z|X_k, m)}{c(z|X_k) + \int P_D(\zeta|X_k) h_k(z|X_k, \zeta) v_{k|k-1}(\zeta|X_{0:k}) d\zeta} \Bigg], \end{aligned} \tag{2.48}$$

where $b(m|X_k)$ is the intensity of the spontaneous birth process, $c(z|X_k)$ is the intensity of the spurious readings and $P_D(m|X_{0:k})$ is the probability of detection in space. All these intensities are functions in the landmark field $\mathcal{K}$, contrary to the map set which exists in $\mathcal{F}_\mathcal{K}$.

Note that although the PHD filter assumes the different RFS to be Poisson, it can still be applied to any real model. Just as in the EKF, the solution continues to work, but the optimality condition is no longer valid.

One more step to complete an efficient implementation of this algorithm is required. So far, the PHD filter reduced the complexity from absolutely intractable to tractable. The next step is to make it fast. Similarly to other methods, Gaussians simplify the work. If the measurement readings h_k are Gaussian (as in the EKF) and the intensity functions are constrained to the family of Gaussian mixture models of the form $\sum_{i=1}^{N} w_i \mathcal{N}(\mu_i, \Sigma_i)$, this property will be invariant, thus allowing efficient explicit algebraic expressions [67].

Finally, there is one more loose end to solve. The vehicle trajectory needs to be incorporated. As proposed by Mullane in 2011 [45], a Rao-Blackwellization method can be employed, just like in FastSLAM. Several particles will be proposed from the vehicle transition model, each one carrying out its own mapping assuming known localization and then an importance resampling step will force them into the correct distribution. The localization then follows the equation

$$P(X_{1:k}|\mathcal{Z}_{1:k}, U_{1:k}, X_0) = h(\mathcal{Z}_k|X_{1:k}, \mathcal{Z}_{1:k-1}) \frac{f_X(X_k|X_{k-1}, U_{k-1}) P(X_{1:k-1}|\mathcal{Z}_{1:k-1}, U_{1:k-1}, X_0)}{h(\mathcal{Z}_k|\mathcal{Z}_{1:k-1})}, \tag{2.49}$$

which is approximated using particles with weights that update according to:

$$w_k^i \propto h(\mathcal{Z}_k|X_{1:k}^i, \mathcal{Z}_{1:k-1}) w_{k-1}^i. \tag{2.50}$$

The term $h(\mathcal{Z}_k|X_{1:k}, \mathcal{Z}_{1:k-1})$ is intractable to solve, but a trick can be employed to avoid having to calculate it directly. The term can be equated as

$$h(\mathcal{Z}_k|X_{1:k}, \mathcal{Z}_{1:k-1}) = \frac{h(\mathcal{Z}_k|X_k, \mathcal{M}_k) P(\mathcal{M}_k|X_{1:k}, \mathcal{Z}_{1:k-1})}{P(\mathcal{M}_k|X_{0:k}, \mathcal{Z}_{1:k})}. \tag{2.51}$$

As the reader may notice, the left hand side does not contain the map, while the right hand side contains it both in the numerator and the denominator. This implies that the equality must hold for all maps, so any of them can be used to compute the fraction. Using a zero-landmarks map it reduces to

$$h(\mathcal{Z}_k|X_{1:k}, \mathcal{Z}_{1:k-1}) \approx \kappa_k^{\mathcal{Z}_k} \times \exp\left(\hat{m}_k - \hat{m}_{k|k-1} - \int c_k(z|X_k) dz\right) \tag{2.52}$$

$$\kappa_k^{\mathcal{Z}_k} = \prod_{z \in \mathcal{Z}_k} c_k(z|X_k) \tag{2.53}$$

$\hat{m}_{k|k-1}$ and $\hat{m}_k$ are the expected cardinalities of the predicted and updated map respectively.

Alternatively using a one-landmark map $\mathcal{M} = \{\bar{m}\}$, it becomes

$$\begin{aligned} h(\mathcal{Z}_k|X_{1:k}, \mathcal{Z}_{1:k-1}) \approx \frac{1}{\Gamma}\Big[&(1 - P_D(\bar{m}|X_k))\kappa_k^{\mathcal{Z}_k} \\ &+ P_D(\bar{m}|X_k) \sum_{z \in \mathcal{Z}_k} \kappa_k^{\mathcal{Z}_k - \{z\}} h_k(z|X_k, \bar{m})\Big] v_{k|k-1}(\bar{m}|X_{0:k}) \end{aligned} \tag{2.54}$$

$$\Gamma = \exp\left(\hat{m}_k - \hat{m}_{k|k-1} + \int c_k(z|X_k) dz\right) v_k(\bar{m}|X_{0:k}), \tag{2.55}$$

Another approach is to fully compute it. To do so, the hard term is the first one, that may be

expressed [68] as

$$h(\mathcal{Z}_k|X_k,\mathcal{M}_k)=\sum_{\theta}\left(P(\mathcal{Z}_k^{\theta}|X_{0:k},\mathcal{M}_k^{\theta})\prod_{m\in\mathcal{M}_k^{\theta}}P_D(X_k,m)\prod_{m\in\overline{\mathcal{M}_k^{\theta}}}(1-P_D(X_k,m))P_{\kappa}(\overline{\mathcal{Z}_k^{\theta}})\right), \tag{2.56}$$

Most of these terms are negligible, since the data association uncertainty is usually pretty low compared to the number of landmarks. In [68], the authors provide a way to decompose it capitalizing on its sparse nature and approximate it using the largest n terms of the sum in $O(n(|\mathcal{M}|+|\mathcal{Z}|)^4)$ time.

It is clear that the zero-landmark alternative is the cheapest so why would anyone choose another? The reasons are numerical imprecisions and model mismatch. As usual, if the Poisson conditions are not met on every RFS, the assertions of the theory become less valid. In practice, the full approximation performs better than using one landmark, which performs better than using zero.

To improve the accuracy of the filter, other forms of random finite sets can be used. In [69], the Vo brothers and Cantoni propose an efficient implementation of the Cardinalized Probability Hypothesis Density (CPHD) filter, which generalizes the PHD filter by relaxing the requirement on the cardinality distribution, allowing for an arbitrary profile instead of Poisson. In this setting, both the PHD and the cardinality distribution are propagated in time. These equations are more convoluted than their PHD counterpart; to make the solution tractable, Gaussianity is assumed on the underlying processes. Nonlinear processes can be linearized similarly to the EKF setting. Note that when there are many landmarks, a Poisson distribution (PHD) variance grows with its mean while using other distributions (CPHD) can model both statistics independently.

Alternatively, the distribution can be assumed to be Multi-Bernoulli, as Mahler explains in [46], known as the MeMBer filter. Instead of propagating only the first moment, the full posterior is updated. This alternative approximates the density instead of the moments and is easier to interpret (statistics are easier to extract), but is confined to situation with good signal-to-ratio. As the authors noted in [70], this filter is biased towards large cardinalities, but can be counteracted to obtain an unbiased estimate known as the Cardinality Balanced Multi-Target Multi-Bernoulli filter or CBMeMBer.

These alternative formulations have been proposed in the tracking field, but have not been used in the SLAM literature. Their properties make them good candidates for future research.

So far, all the presented algorithms assume a static map. Dealing with dynamic landmarks is hard, because if a landmark is seen again in a different location, there is an ambiguity between the motion of the landmark and the vehicle. Defining an appropriate local relation to enforce is hard, the vehicle usually needs to be aware of the global situation (and sometimes for many time steps) to decide this ambiguity. A common solution is to separate dynamic and static landmarks and deal with them separately, with little communication [71, 72].

Nevertheless, the RFS framework is well equipped to deal with this Bayes-optimally, since in fact the PHD filter defines a transition distribution over landmarks, even if previous algorithms do not use it (and set it to be static). Lee, Clark and Salvi [73] use the single-cluster PHD filter to model both stationary and moving landmarks. Unlike other alternatives, the richness of the RFS

framework allows the different parts to communicate information more freely. Here single-cluster PHD is a hierarchical extension of the PHD filter, where the realization of the map and measurement RFS is conditioned on a parent RFS with cardinality one, in this case, the trajectory.

In [74], the authors use the RFS framework to estimate a vehicle's odometry from visual features. They do so by extracting SIFT features, matching them between consecutive frames and projecting them into the 2D space where the vehicle lies, i.e. the ground plane. To be able to project the features reliably, the height above the road and the tilt angle of the camera is assumed fixed and known. Then, the set of projected features (targets) are tracked jointly using a PHD filter. The motion of all targets is assumed to be the same and trivially related to the motion of the vehicle, so the latter one is calculated from the average motion of the targets, i.e. the vehicle motion is not modelled directly, but indirectly through the observed motion of the targets. Although it shares a similar goal as the methods introduced in this thesis, this method is limited in that it requires very strong constraints in the movement of the camera: it must move in a 2D plane, its height and angle must be constant and calibrated, so it is not suitable for freehand motion. In particular, any true 3D motion cannot be recovered completely, such as holding the camera with one's hands or the terrain of a mountain road. Also, as the features are projected to the ground plane, some 3D information of the landmarks is lost, which could have been used to improve the result. Further, only the vehicle's trajectory is estimated; the map is not part of the state, instead it is only used as intermediate information to help with the trajectory estimation. Additionally, as only consecutive frames are used to obtain an odometry value, no loop closure can be introduced, so the system would be expected to diverge with time.

5.2 Smoothing in the RFS framework

Filtering theory has been well developed under the RFS framework; smoothing, not so much, but recent efforts have focused on it. Note that no complete smoothing RFS SLAM solver has been proposed, but there is some work on the mapping and tracking areas, which are very related.

Before explaining the smoothing algorithms, a small remark about accidental smoothing while filtering. In theory, filtering doesn't provide information about the past, but since the state is not the pose of the vehicle but its trajectory, it can indeed change the distribution of past poses along the way. This means, as shown in many proofs of convergence, that with infinite particles the filtering approach can indeed use information to improve a previous estimate. Naturally in practice, resources are limited so this kind of smoothing happens but in a very minor degree than with algorithms designed to solve the full problem at once.

In 2011, Mahler and the Vo brothers proposed the Forward-Backward PHD smoother [75], which extends the Rauch-Tung-Striebel Bayesian smoother [76] to the RFS context. When filtering, the update equations only take into consideration information from the past, i.e $P(X_{0:k}, \mathcal{M}|\mathcal{Z}_{1:k})$ (the odometry parameters $U_{1:k}$ have been obviated since they can be considered parameters of the distributions and are never used in another way). Many times, ambiguities in the measurements cannot be resolved using past information alone, so the practical filter is forced to accept a bad proposal move with high probability. After filtering the whole trajectory, i.e. computing $P(X_{0:T}, \mathcal{M}|\mathcal{Z}_{1:T})$, a second pass can be done in reverse to condition each pose estimate on all the measurements, not only the past ones, i.e $P(X_{0:k}, \mathcal{M}|\mathcal{Z}_{1:T})$. This distribution can be backward-propagated from time

k to time $k' < k$ using the equation,

$$P_{k'|k}(\mathcal{M}_{k'}) = P_{k'}(\mathcal{M}_{k'}) \int f_{k'+1|k'}(\mathcal{M}_{k'+1}|\mathcal{M}_{k'}) \frac{P_{k'+1|k}(\mathcal{M}_{k'+1})}{P_{k'+1|k'}(\mathcal{M}_{k'+1})} \delta\mathcal{M}_{k'+1}, \tag{2.57}$$

where the update is from time k to time $k' < k$. This recursion, unlike the filtering one, does not require the posterior cardinality distribution to be Poisson.

As always, the expression is intractable so approximations are required. If the posterior cardinality distributions are Poisson, they can be reduced to

$$v_{k'|k}(x) = v_{k'|k'}(x) \left(1 - P_S(x) + P_S(x) \int f_{k'+1|k'}(y|x) \frac{v_{k'+1|k}(y)}{v_{k'+1|k'}(y)} \, dy\right). \tag{2.58}$$

where $P_S(x)$ is the probability of survival. The authors then proceed to implement this approach using particle filters.

In the backward smoothing update, no new particles are generated. Instead, the existing ones are reweighted, improving their represented distribution. This means that no new hypothesis would be introduced. This is bad if good hypotheses were removed for whatever reason, e.g. a bad odometry reading. The authors approach this by using an adaptive resampling scheme which reintroduces forgotten landmarks forcibly, so they have a chance to survive and become relevant if future measurements are consistent with them.

The authors note that the approach improves the state error, but not necessarily the cardinality error and point to possible extensions using the other techniques that have been discussed (CPHD and Bernoulli filters, etc.).

This could possibly be extended by using the smoothing stage to improve the filtering stage at each time step, i.e. using future information for proposing a better set of particles in the predict step of the filter. Naturally, the reweighting step would have to consider the fact that predicted particles no longer come from the odometry distribution (explored further in chapter 5).

Vu et al. [77] take a different approach, forgoing the entire filter development. The problem is proposed entirely as sampling strategy on the complete distribution, $P(\mathcal{M}|\mathcal{Z}_{1:T})$. To do so, a hybrid between Marginal Metropolis-Hastings and Sequential Montecarlo algorithms called Particle Marginal Metropolis-Hastings Montecarlo (PMMHMC) is used, in which one of the factors of the Metropolis-Hastings ratio is approximated using Sequential Montecarlo because it's intractable. This alternative has been found to work better in very high dimensional spaces. In this approach, there's no need for each sample to maintain more than one data association hypothesis. Each sample will represent a track hypothesis consisting of a set of tracks of the form $(r, t, x_0, \ldots, x_m)$, where r is a label (data association), t the time when a landmark was first observed and $x_0, \ldots, x_m$ the consecutive position of the landmark. To use the PMMHMC approach, the authors write down a proposal distribution of twelve possible moves, shown in figure (2.8).

The algorithm executes as follows: From a prior estimate for the map $\mathcal{M}_{k-1}$, e.g. extracted using the filtering approaches, perform a move from the proposal distribution to obtain $\mathcal{M}_k$. At the same time, from a prior measurement-to-landmark association θ_{k-1} propose a new association θ_k. Calculate the ratio

$$\eta = \frac{P_{\theta_k}(\mathcal{Z}) P(\theta_k) Q_{\theta_k}(\theta_{k-1}|\mathcal{Z})}{P_{\theta_{k-1}}(\mathcal{Z}) P(\theta_{k-1}) Q_{\theta_{k-1}}(\theta_k|\mathcal{Z})}, \tag{2.59}$$

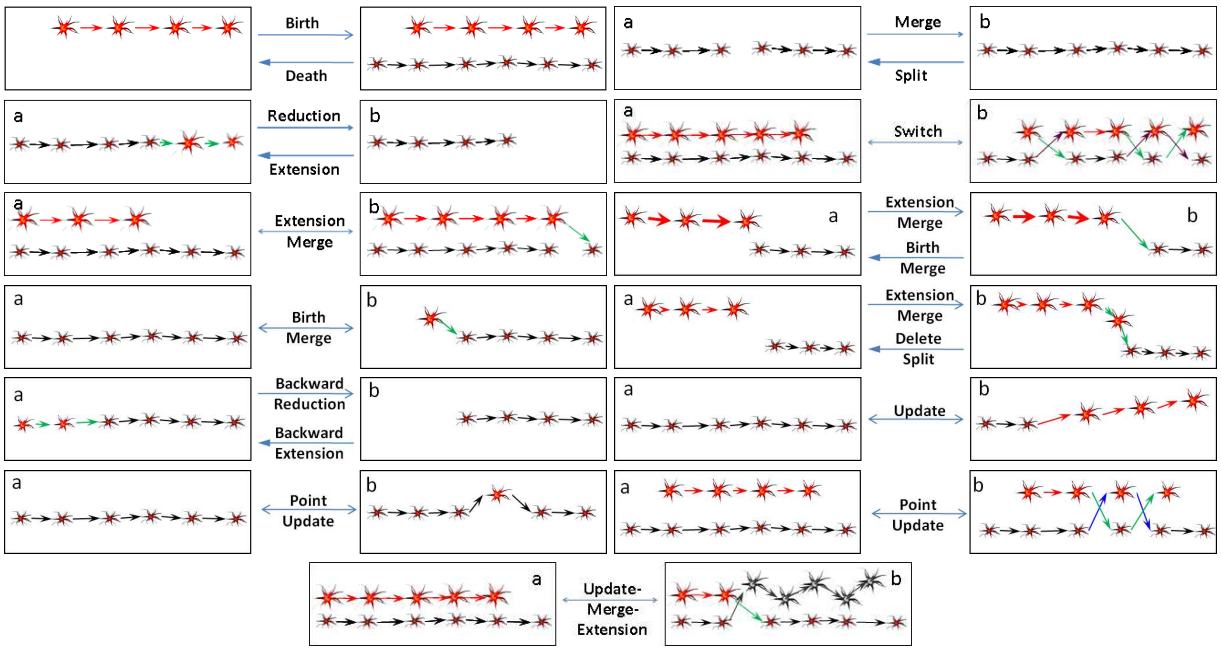


Figure 2.8: Vu's PM-MHMC proposal moves (extracted from their paper [77]).

where $P(\theta)$ is the prior probability of the measurement assignments (usually uniform), $Q_\theta(\theta|\mathcal{Z})$ is their proposal distribution and $P_\theta(\mathcal{Z}) = \int P_\theta(\mathcal{M}, Z)d\mathcal{M}$ is the marginal density of the proposal of landmarks that is too hard to compute, so is approximated with a particle filter. Accept this new sample with probability $\min\{1, \eta\}$ and iterate the procedure from this new map estimate $\mathcal{M}_k$ to obtain $\mathcal{M}_{k+1}$, etc.. The samples will theoretically converge to the desired distribution.

Using this methodology, the authors are able to significantly improve the results of the PHD filter, albeit at much higher time costs (e.g. from one minute to two hours and from 30 minutes to 5 hours).

The comparative table (2.1) is presented as a summary of the main solvers described in this chapter.

6 Data Association

A crucial step in most approaches is data association, i.e. matching different observations of the same landmark. To make any theoretical assertions over the performance of the algorithms, data association has been assumed correct; if it's not, the estimates will be suboptimal and may even diverge with time.

A first approximation to the problem is using a Nearest Neighbours (NN) approach [78], where each measurement is associated to the closest landmark, like in figure (2.9). This idea makes sense if landmarks are far away from each other, so measurements are unambiguous.

When a measurement does not correspond to any landmark, i.e. it is farther than ϵ to all of them (like the middle measurement in the figure), a new landmark must be generated. It is important to

Method	Feats	LC	Cov	Data Structures	Observations
EKF-SLAM	Keypoints		Yes	Array of Gaussians	Straightforward Bayesian filter.
Fast-SLAM	Keypoints		Yes	Particles	Factorizes by conditioning map on pose.
Graph-SLAM	Keypoints	Yes	Yes	Graph	Graph induces sparse linear algebra.
PHD-SLAM	Keypoints		Yes	RFS	Robust to spatial and cardinality noise.
Visual-specific					
BA	Keypoints	Yes		Poses + 3D points	Highly non-convex quadratic optimization.
MonoSLAM	Keypoints		Yes	Array of Gaussians	Uses EKF-SLAM.
Monocular SLAM	Keypoints		Yes	Particles	Uses FastSLAM.
PTAM	Keypoints			Poses + 3D points	BA considering only *key* frames.
ORB-SLAM	Keypoints	Yes		Poses + 3D points	Uses covisibility graph, ORB features and loop closure using Bag-of-Words.
DTAM	Dense			Voxel grid	Minimizes photogrammetric error of reprojection.
LSD-SLAM	Semi-dense			Semi-sparse voxel grid	Only updates interesting regions (edges).

Table 2.1: Comparative summary of the main methods presented in this review. It shows the nature of the features for the map (Feats), the ability to close loops directly (LC) , whether or not it estimates the covariance for the state components (Cov), the main data structures used in each method and additional observations.

avoid generating spurious landmarks (e.g. from sensor noise) as they could make future associations incorrect. To do so, extensive preprocessing should be performed on the measurements to remove any false positives. Alternatively, the landmarks should be periodically scanned and pruned.

Using the nearest neighbour method directly can lead to multiple measurements being paired with the same landmark. This can be suboptimal, for example in figure (2.10), where it can be expected that the two topmost measurements correspond to two different landmarks. To avoid this, a linear assignment [79] can be performed instead, using the Euclidean distance as the cost matrix. This way, even if there are many landmarks close together, they can be correctly matched if the measurements have the right geometric structure.

The NN approach may not work in real data, since many times there are a lot of uncertainty and landmarks can be near each other. As time passes by, the uncertainties may increase making the association less reliable, which in turn increases the uncertainties by making incorrect assumptions,

and so on in a vicious cycle.

Fortunately, there is usually additional information in the estimates that can be exploited: each landmark and measurement has an associated uncertainty covariance. Assuming Gaussianity on each component, these covariances can express the likelihood of the measurements given the landmarks. Then the maximum likelihood association [32] can be found by linear assignment using an appropriate Mahalanobis distance as a cost matrix. These distances are associated with the marginal pose-landmark covariances

$$d(z_i, m^j) = (z_i - m^j)^T \mathbf{\Xi_{ij}} (z_i - m^j), \quad \mathbf{\Xi_{ij}} = \mathbf{H}\mathbf{\Sigma}\mathbf{H}^T + \mathbf{R}, \quad \mathbf{\Sigma_{ij}} = \begin{pmatrix} \mathbf{\Sigma_{xx}} & \mathbf{\Sigma_{xm^j}} \\ \mathbf{\Sigma_{xm^j}} & \mathbf{\Sigma_{m^j m^j}} \end{pmatrix}, \quad (2.60)$$

where $\mathbf{R}$ is the measurement noise and $\mathbf{H}$ is the Jacobian of the measurement function $h(\cdot)$.

Calculating these may or may not be simple depending on the framework and the data representation (e.g. if covariances are implicit in a tree or a big dense information matrix, these marginals may be hard to extract).

If the complete marginal covariance is difficult to extract, sometimes a conservative approximation can be computed more easily by disregarding the pose-marginal joint component and estimating the landmark uncertainty from the noise level, i.e. assuming pose and map as independent. Note that this approach will always overestimate the covariance, as correlation implies a narrower cross-section in any Gaussian.

If uncertainty is too large or landmarks are too close, these methodologies, along with any preprocessing steps, are bound to face difficult choices and make mistakes. To face this issue alternative ideas have been proposed.

One idea is Joint Compatibility Branch-and-Bound (JCBB), proposed by Neira et al. in 2001 [80]. In this approach, the compatibility D_{ik} of a landmark hypothesis L_i and the measurements Z_k is defined as

$$D_{ik} = (Z_k - h(X_k, m^i))^T \mathbf{C_i}^{-1} (Z_k - h(X_k, m^i)) \quad (2.61)$$

$$\mathbf{C_i} = \mathbf{H_{ik}} \mathbf{P_i} \mathbf{H_{ik}}^T + \mathbf{R_k} \quad (2.62)$$

where $\mathbf{P_i}$ is the landmarks covariance, $\mathbf{H_{ik}}$ is the measurement Jacobian and $\mathbf{R_i}$ is the measurement noise. Note that these matrices are over the entire hypothesis and measurements, not on each

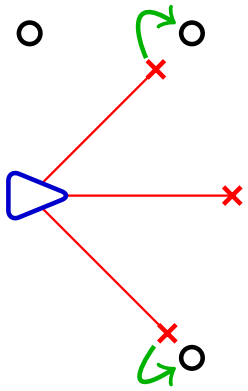


Figure 2.9: Nearest neighbours data association scheme.

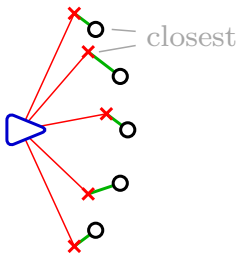


Figure 2.10: Linear assignment data association scheme. Even though the indicated pair is closer than the alternatives, choosing it would orphan other elements (expensive).

component. A hypothesis is said to be compatible if

$$D_{ik} < \chi^2_{|Z_k|,\alpha}. \tag{2.63}$$

where α is a suitable threshold. This is then used to find the largest compatible hypothesis by using an efficient recursive method which avoids calculating the compatibility for most configurations (hence the branch-and-bound). Note that unlike the naive NN case, this expression contains the full covariance between all the relevant variables, i.e. instead of assuming independence, the joint problem is optimized. By maintaining the extra joint data, the association is better, but more expensive to compute.

Another idea is Joint Probabilistic Data Association Filter (JPDAF), proposed by Fortmann and Bar-Shalom in 1980 [81, 78, 82]. Here, the notion is to take the whole distribution over associations into consideration, from which a maximum likelihood estimator can then be extracted (or alternatively, any other aggregate estimator). Unlike in JCBB, this distribution may be modelled beyond simple covariances with richer methodologies including non-linearities.

A third idea is Multi-Hypothesis Tracking (MHT), proposed by Blackman in 2004 [83]. In this approach, different tracks are spawned as measurements come in, each one with a possible data association. At every step, each one of these tracks is updated using some of the SLAM methodologies, associating a probability to each hypothesis using Bayes inference. As time goes by, some of these hypotheses are rendered unlikely given new information, so they may be discarded.

Selecting appropriate associations to spawn may be difficult, since the number of possible associations is combinatorial in the number of landmarks and measurements. To follow the best candidate paths, MHT uses Murty's algorithm [84] to select the top n associations, which runs in $O(n(|M| + |Z|)^4)$. To speed it up even more, landmarks and measurement can be clustered into smaller regions far away from each other, each one being processed independently.

7 Modelling tools

7.1 Sensors

To implement any of the SLAM methodologies a sensor model is required. Many different models exist, which are appropriate for different kinds of sensor. Usual sensors are lasers, radars, sonars and cameras.

Cameras are intrinsically 3D sensors, but in their raw form they only produce 2D projections directly, so some processing is required to fuse the information into 3D measurements. Here it is important to distinguish between RGB, Stereographic and RGB-D cameras: the first one only contains color information, while the second one consist of an arrangement of two cameras in static relative configuration between each other, and the latter one includes an additional depth map. Both stereo and RGB-D configurations allow for a complete determination of a landmark in 3D space by triangulating them from the difference of perspective in the stereo cameras or the depth stream in the RGB-D cameras. In the RGB case, measurements do not give enough information to pinpoint landmarks in space, but rather give an infinite ray of possible locations.

Not knowing the depth in RGB cameras means that in this configuration there will always be a scale degree of freedom that cannot be fixed only with the sensor. Besides that, using several video frames the configuration can be adapted to work similarly to the other approaches by locally triangulating keypoints, i.e. using multiple video frames as stereo pairs.

Stereo images require careful calibration [22] to produce accurate models; triangulation heavily depends on the geometric position of both cameras.

Auto-calibration methods exist for both stereo and RGB cameras using multiple frames, which can simplify the workflow and approach dynamic conditions such as camera movement due to vibrations.

RGB-D are the simplest to work with from the mathematical point of view, but they are more limited, since besides calibration, they require particular conditions to work properly, such as small environments and no sunlit scenes. Note that the results in the RGB-D setting usually can be extended to the other settings by using dense tracking algorithms in the literature like [58].

7.2 Rotations

Vehicle poses consist of a translational and a rotational part. The translation aspect is simple, as it is a vector space where addition and scaling are easily defined, so additive white Gaussian noise can be implemented with a simple euclidean sum, $X' = X + w_x$. The rotational aspect is not so trivial, though, since rotations do not form a vector space, but a non-linear space known as special orthogonal group $SO(3)$, which corresponds to the unit hypersphere of dimension 4 [85]. A parametrization must be selected, of which there are many, each one with different benefits and drawbacks.

In two dimensions the rotational space has one degree of freedom ($SO(2)$ corresponds to the unit circle), which makes it simple enough to be captured as an angle $\theta \in [0, 2\pi]$. In three dimensions, the parametrization is more nuanced. The space $SO(3)$ has three degrees of freedom, so a minimum of three parameters are required. A typical parametrization uses three Euler angles, which act upon the rotated object consecutively over different axes.

Rotations are linear operators so they can be written as matrices, $\mathbf{R}$, and act on vectors by matrix multiplication, $Rot_{\mathbf{R}}(X) = \mathbf{R}X$. In the case of 3D, these matrices are in $\mathbb{R}^{3x3}$. However, not every matrix is a rotation; they have to be unitary, orthogonal and its determinant must be +1 to correspond to a rotation. They are another parametrization for rotations, with nine parameters (one

per entry).

In the matrix context, Euler angles can be viewed as a factorization into three different rotation matrices (one per axis), as in

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.64}$$

There exists 24 different possible factorizations, depending on the chosen axes.

Although more compact, this simpler Euler parametrization has a big drawback: there are configurations which lose one degree of freedom, a problem known as gimbal lock. For example, in the previous factorization, if $\beta = \pi/2$, the multiplication can be reduced to

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.65}$$

$$= \begin{pmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{pmatrix}. \tag{2.66}$$

In this configuration, both α and γ do the same thing, i.e. the system can't directly rotate in one of the three axes (it can do it by changing β at the same time than the other angles but this is unstable). In practice this means that numerical calculations may diverge or perform arbitrarily poorly around this point.

It turns out that any parametrization with three arguments presents this problem, so to avoid it alternative representation must be embedded in higher dimensional spaces. Two of such parametrizations are common: matrices (already explained) and quaternions.

Quaternions come from algebra, where they generalize the notion of complex numbers. They can be viewed as a scalar-vector pair $(w, \mathbf{x}, \mathbf{y}, \mathbf{z})$ with a special multiplication rule

$$\begin{aligned} (a_w, a_x, a_y, a_z) \times (b_w, b_x, b_y, b_z) = (&a_w b_w - a_x b_x - a_y b_y - a_z b_z, \\ &a_w b_x + a_x b_w + a_y b_z - a_z b_y, \\ &a_w b_y - a_x b_z + a_y b_w + a_z b_x, \\ &a_w b_z + a_x b_y - a_y b_x + a_z b_w). \end{aligned} \tag{2.67}$$

If the quaternion is unitary, then multiplying it to a pure vector corresponds to a rotation and can be decomposed as $(\cos(\theta/2), \sin(\theta/2)\,\hat{\mathbf{u}})$, where θ is the rotation angle and $\hat{\mathbf{u}}$ is the rotation axis.

Quaternions are a more compact representation than matrices, but matrices directly expose the linear nature of rotations (so they are helpful in derivatives). The number of scalar multiplications/additions to operate on a vector is higher for quaternions, but concatenation of several rotations requires more operations in matrices.

Since these parametrizations exist in higher dimensional spaces, not all elements from the space are valid rotations. As operations accumulate, slight truncation errors can deviate the result from the valid subset, e.g. matrix multiplications could slowly make the result non-unitary or

non-orthogonal. To correct these deviations, quaternions should be renormalized by dividing by the magnitude. Matrices should first be decomposed into its SVD (Singular Value Deocmposition) form, which corresponds to the product UDV^T, where U and V are orthogonal matrices and D is diagonal (it can be computed with a variant of the QR method), where each element in the diagonal D is called a singular value of the original matrix. A rotation matrix should have all singular values equal to one (otherwise it performs an additional scaling operation in some axis), so to correct any deviations, the matrix is corrected to UIV^T, where I is an appropriately sized identity matrix.

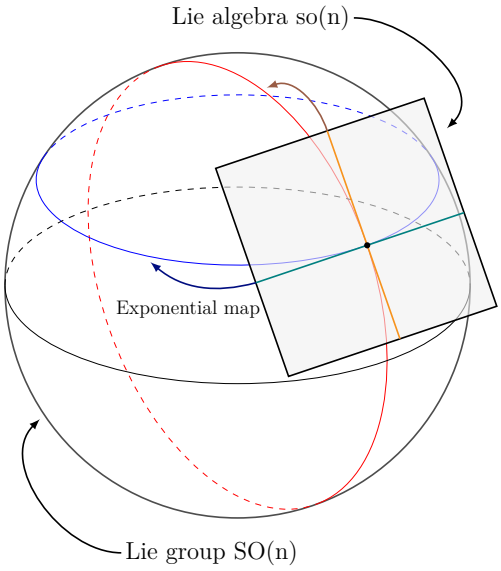


Figure 2.11: Relationship between Lie groups their associated Lie algebras. For SO(3), the hypersphere lies in $\mathbb{R}^4$ and the algebra vector space is a 3D hyperplane.

Adding noise to these representations is not straightforward. The noisy result should distibute over the valid subspace of rotations. One way to accomplish this is by adding noise in the 4D embedding space and then renormalizing, i.e. reprojecting into the valid subspace.

Another parametrization alternative uses the notion of a Lie algebra [86, 87, 85]. To construct it, a point is chosen in the unit hypersphere $SO(3)$ (usually the identity rotation, which does nothing). The lie algebra is then defined as the vector space of all the vectors ω that are tangent to the hypersphere at the chosen point, as in figure (2.11). Then, there exists a parametrization of the entire hypersphere from this tangential space, which is known as the exponential map, e^{ω}. In this case, this map follows the rule

$$e^{\omega} = \mathbf{I_3} + \frac{\sin\theta}{\theta}[\omega]_\times + \frac{1-\cos\theta}{\theta^2}[\omega]_\times^2, \tag{2.68}$$

known as Rodriguez' formula. Here the angle $\theta = |\omega|$ and the skew symmetric matrix

$$[\omega]_\times = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}.$$

Also, its inverse mapping, the logarithm, can be defined:

$$\log \mathbf{C} = \omega \iff e^{\omega} = \mathbf{C}.$$

Both the exponential and the logarithm can also be used with quaternion representations,

$$e^{\omega} = (\cos|\omega|, \sin|\omega|\hat{\omega}) \tag{2.69}$$

$$\log \mathbf{q} = \omega \iff e^{\omega} = \mathbf{q}. \tag{2.70}$$

Using these defintions, a natural interpolation function can be defined, known as the spherical interpolation function,

$$\text{Slerp}(\mathbf{p}, \mathbf{q}, \eta) = \mathbf{p}(\mathbf{p}^{-1}\mathbf{q})^{\eta} \tag{2.71}$$

$$= \mathbf{p}e^{\eta \log(\mathbf{p}^{-1}\mathbf{q})}, \tag{2.72}$$

which corresponds to an interpolation along the unitary hypersphere of the quaternions.

This Lie algebra can be useful to define noises and deltas in the "correct" number of dimensions, allowing to update the rotation part of the pose elegantly. However, since it uses three parameters, it can be affected by gimbal lock. A more robust approach is to use quaternions or matrices to store the rotations, while performing the updates (which are usually small around the identity, so no gimbal lock) using the Lie space.

7.3 Map distances

To evaluate the performance of the SLAM algorithms, some metrics should be defined to compare the trajectories and maps. The trajectories are simpler: since they always have the same structure, they can be compared frame-by-frame using an Euclidean distance. The map however is more complex. Since the map is a set of points, there are primarily two kinds of errors that can occur: cardinality errors and localization errors, i.e. missing, extra and misplaced landmarks.

Maps may be evaluated using the Hausdorff metric, which is defined as

$$d_H(X, Y) = \max\Big(\max_{x\in X}\min_{y\in Y} d(x, y), \max_{y\in Y}\min_{x\in X} d(x, y)\Big), \tag{2.73}$$

and intuitively finds the *required* distance to travel from one set to the other in the worst case. Although generally useful, it doesn't deal so well with cardinality errors and outliers. The outlier problem has been addressed in [88], where the algorithm is extended by replacing the internal max-min operations for a L^p average of the form $(\sum(\cdot)^p)^{1/p}$.

To try and solve the drawbacks of the Hausdorff metric, Hoffman and Mahler proposed the Optimal Mass Transfer (OMAT) metric [89], which is defined as

$$\begin{aligned} d_p(X, Y) &= \min_{C}\Big(\sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}\, d(x_i, y_j)^p\Big)^{1/p} \\ d_\infty(X, Y) &= \min_{C}\max_{\substack{1\le i\le m\\ 1\le j\le n}} \delta_{\text{Dirac}}(i-j)\, d(x_i, y_j), \end{aligned} \tag{2.74}$$

where the minimization is over all transportation matrices C. Each transportation matrix corresponds to a data association between maps X and Y. This minimization domain helps the algorithm

to better represent the cardinality error, which will be observed as the sum of the bad matchings in the given association, i.e. after all other points could be matched together, the remaining "unassociable" ones will have to be matched together, heavily increasing the distance, instead of just finding an extreme case as the Hausdorff does.

This definition also introduces the L^p norm $(\sum(\cdot)^p)^{(1/p)}$ which alleviates the outlier problem by capping their ability to artificially deviate the result. Indeed, the outliers will affect the the output, but their effect will be only a small number of components in the L^p average. In the case of Hausdorff, one outlier could deviate the min-max value and no matter how many other associations were or how good they were, the extreme nature of the operator means the value would be effectively locked into the wrong place.

Schuhmacher and the Vo brothers [90] observed a number of limitations of the OMAT approach, such as inconsistency between similar scenarios, difficult interpretation, geometry dependence and undefined corner cases. Then they provided a new metric that addresses these issues known as the Optimal Subpattern Assignment (OSPA) metric. To define it, they first introduce a truncated distance between points,

$$d^{(c)}(x, y) = \min(c, d(x, y)) \tag{2.75}$$

and then define the metric as

$$d_p^{(c)}(X, Y) = \Big(\frac{1}{n}\Big(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)})^p + c^p(n - m)\Big)\Big)^{1/p} \tag{2.76}$$

$$d_\infty^{(c)}(XY) = \begin{cases} \min\limits_{\pi \in \Pi_n} \max\limits_{1 \le i \le m} d^{(c)}(x_i, y_{\pi(i)}) & \text{if } m = n \\ c & \text{if } m \neq n, \end{cases} \tag{2.77}$$

which they study closely. The metric has an intuitive interpretation and its effects can be clearly separated between cardinality and spatial errors. Its computation can be performed by linear assignment, as detailed in the original paper.

Chapter 3

Visual SLAM

To start this work, a complete visual SLAM system has been implemented. It is capable of extracting a video stream from either a Kinect sensor, a pre-recorded session or a simulation, using one of several SLAM algorithms and present the user with the solution along with some performance metrics.

1 System Design

The system is separated into two main components: a visual perception module and a SLAM module. As their names indicate, the first takes the sensor input and converts it into useful measurements and the second uses these measurements to solve for the vehicle pose and generates the map. An overview of the system structure is shown in figure (3.1). Details on the design of each subsystem are presented in the rest of this section.

The system has been primarily programmed in C# for its clean syntax, expressiveness, relative high speed and cross-platform portability, which allows a robust development. Several useful libraries exist for the platform: AForge.Net, Accord.Net and accord-net-extensions have been used for linear algebra and image processing, MonoGame for real-time simulation and visualization, and NDesk for command line parsing. Tests were implemented using the NUnit framework.

Different laguages were employed were they made sense. Python was used for its plotting and numerical capabilities in the post-processing stage and C++ was used for bridging libraries that were already written in this language (e.g. to make comparison against other SLAM implementations) and to use OpenCV when computation intensive processing was required (e.g. video pre-processing).

1.1 Visual Perception

The visual perception module's first responsibility is the communication with the RGB-D driver to obtain usable data. Then, it must transform this data into a set of landmark measurement to feed

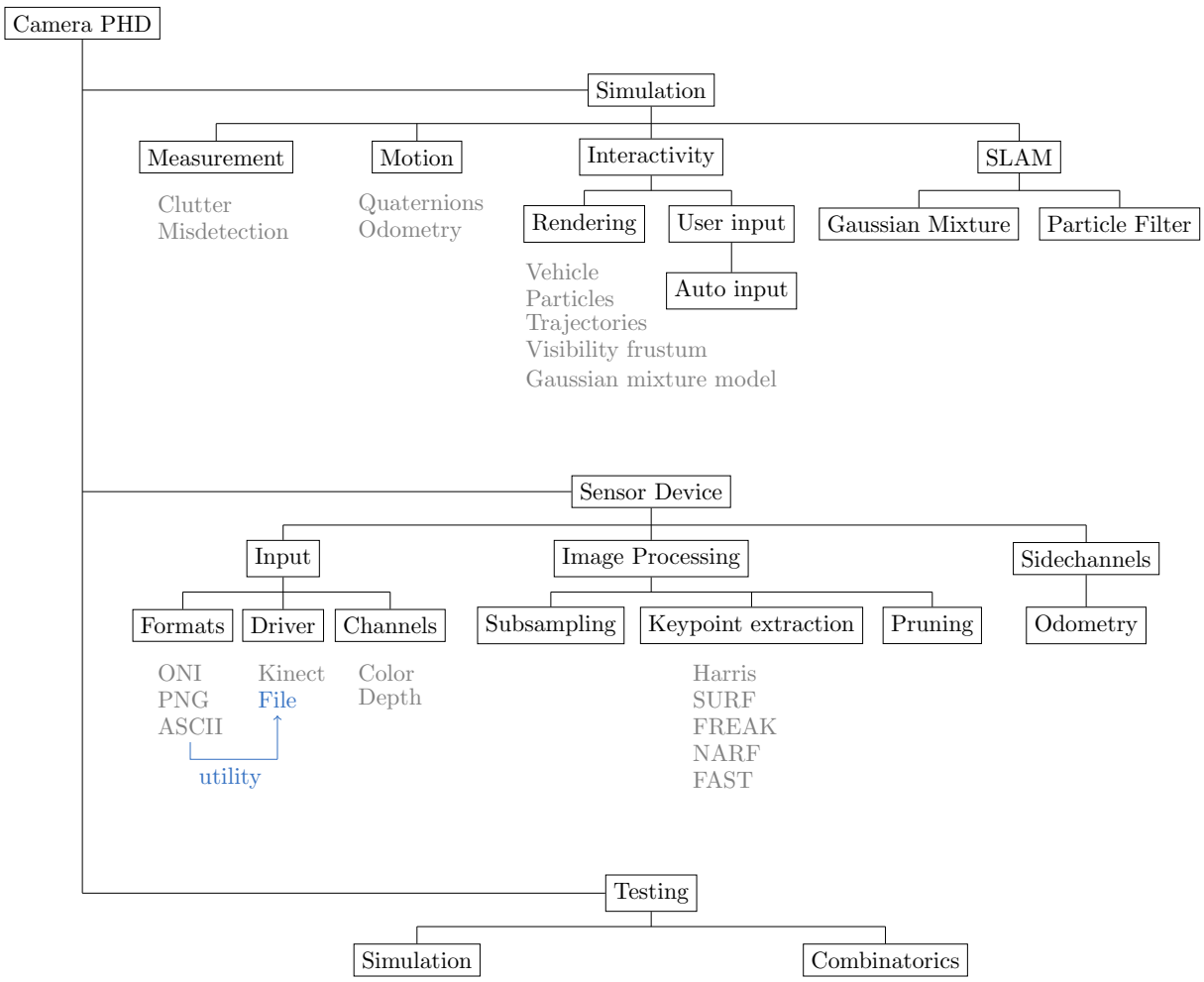


Figure 3.1: Visual SLAM system overview.

into the SLAM solver. For that, it performs image processing to the color image and depth map from the sensor.

Sensor Data Acquisition

The input sensor is a Microsoft Kinect, which uses two cameras: a color one, which gives a typical image, and an active infrared one, which illuminates the scene with patterns to calculate a depth map. Its nominal depth resolution is 1 [mm] and has a usable range from 0.4 to 3.4 [m].

To acquire data from this device, the OpenNI v2.2 driver has been selected as it is the most compatible and has a sensible interface. It can deal with real sensors and recorded files transparently, so most of the work can be done once. The 'oni' recording format is defined in the same library. It manages synchronized color and depth streams and can perform depth-color registration.

Unfortunately, not all datasets are in this format so a conversion utility is implemented in C++ (to take advantage of its raw speed since transcoding is a expensive task) to convert from a set of input options to ONI streams. Although the architecture for this utility is universal, for this

work, PNG sequences are enough, since the Technische Universität München (TUM) dataset is in this format. Note that reading directly from PNG images bloats the code by introducing multiple parsers and performing synchronization, while at the same time being slower because of repeated access to the hard drive. Separating this aspect into a utility converter is more efficient.

Transcoding attention has been paid to both the timestamps for each frame and the scale factors for the depth maps, which are subtle but important details. If they are not carefully maintained across formats, the system may silently fail by associating depth maps to the incorrect RGB images, or by giving depths inconsistent with the odometry readings. To ensure correctness, a 3D depth projection can be observed later in the system GUI to check that the depth map is consistent through time from different viewpoints.

The depth map contains a dense number of measurements throughout the visible area. Much of this information is redundant and not point-like, so they are not easily matchable between frames, so a keypoint extraction routine must be implemented. To do so, both the depth and the color maps can be used. For this work, the color channel gives better results thanks to the scene textures. However, in other situations the depth map could help find untextured landmarks.

To decouple the acquisition stage from the keypoint extraction, the system implements a cache, which can extract several frames before starting extracting features and performing SLAM. This can give a better sense of the timing requirements for each stage.

Image Processing

Feature extraction is a well-studied problem and there are plenty of algorithms to find good representative points. For this work FAST (Features from Accelerated Segment Test) [18] and FREAK (Fast Retina Keypoint) [19] features have been found to be fast enough for real-time operation and to provide points of reasonable quality to satisfactorily track the images. In the final implementation, ORB (Oriented FAST and Rotated BRIEF) was selected, since it's based in FAST (i.e. relatively fast) but can filter out edges since it uses Harris corner detection as the score and is more robust to size changes given its pyramidal scheme [20].

To improve the selection of keypoints, frame-to-frame descriptor matching is performed between subsequent images and outliers are removed. For this matching, a descriptor is extracted for each point, using the respective algorithm, and lowest descriptor distances to define the matches. RANSAC is used to detect the outliers.

Once the keypoints have been extracted and filtered from the color image, the corresponding range measurement is taken from the depth map and all three coordinates are fed into the SLAM solver as a point in the measurement set.

The sensor gives a relatively high resolution image (640x480 [px^2]). This is slow to process and gives too much detailed points so a subsample operation is executed on the image previous to capturing the keypoints.

The extracted candidates are pruned to eliminate invalid pixels, such as those that have zero depth (a depth shadow) or that are too close to the borders of an image (be it the boundaries of the image or a shadow). Zero depth pixels are points where the distance was out of the limits of the

sensor or where the sensor didn't get any information at all such as in low reflection surfaces. After this, only the best measurements are kept.

The Kinect depth sensor is by no means perfect. Its range is relatively small so parts of the scene are obscure to it and some surfaces, like monitors or windows, can cause it to misbehave, generating noise or blank spaces. These problems can be explained by the active light configuration it uses, which relies on infrared light reflecting back to the device. In certain materials this reflection does not occur and in sunlit areas the infrared is drowned by the sun emissions. The second version of the Kinect (not used for this project) can sidestep these problems since it uses a different configuration based on time-of-flight.

Odometry

Along with the visual input, the system requires a 3D six-dimensional odometry reading. This usually will come from an Inertial Measurement Unit (IMU), but can also be estimated directly from the video stream by matching consecutive frames. This idea has been presented earlier as visual odometry and there are several options to implement it. However, it is not an essential aspect of the system, so for this work only an odometry file is used (it may be a device file coming from a sensor). Note also that the system may work even without any odometry (using a random walk motion model), but that would take much more computation time and memory so it may be infeasible in real systems.

The testing dataset contain two sources of odometry information: an accelerometer reading and a groundtruth trajectory. Unfortunately, the accelerometer reading does not have any angular information so it is incomplete. So, the odometry is randomly derived from the groundtruth by adding Gaussian noise to the readings.

1.2 3D RFS SLAM

Simulation

As part of the random finite set SLAM solver, it is required to model the motion and measurement processes and in Rao-Blackwellized RFS SLAM (RB-RFS SLAM), to spawn vehicle particles from the motion process, so a vehicle emulator has been implemented. Note that this component can also simulate a real vehicle for testing under controlled settings. The two models are detailed below.

Motion model

Since the vehicle can move in 3D space, its state is represented by a three-component translation vector and a quaternion for the rotation, totalling seven coordinates with six degrees of freedom. Quaternions are used because they're less redundant, more memory efficient than matrices and more stable than angles, as this representation does not suffer from gimbal lock.

The vehicle motion model assumes an observable noisy velocity odometry reading, corrupted

with Gaussian noise. The odometry format is $\Delta x, \Delta y, \Delta z, \Delta\psi, \Delta\phi, \Delta\theta$, where the first three coordinates reflect local translations, i.e. from the point of view of the camera, and the last three reflect local rotations in the three possible rotation axes.

Note that if no odometry is present, all of these parameters can be set to zero so the Gaussian noise takes over and the system model follows a random walk. However, as previously stated, the solver would take too much time to be practical. If necessary, it is better to extract this odometry from the images.

An important aspect to consider is that local odometry has six dimensions, while the internal state for the pose has seven. This mismatch can be tricky for two reasons: first, operations on different sized vectors is not straightforward, appropriate transformations must be done where necessary; and second, the pose intrinsic dimension is lower than its representation, which means the representation is rank-incomplete. This entails that covariances can have null singular values, which is logical, since there will always be one direction where the quaternion can't move (the direction normal to the four-dimensional unit hypersphere).

In this work two paths have been tried. In one hand, noise has been introduced in the seven dimensions, with an small artificial boost on the quaternion components, so as to generate small but not zero singular values. This means the quaternion can become non-unitary, but the covariances work fine. To resolve the unitary requirements, the quaternion is renormalized after the operations. This approximation works well in practice.

On the other hand, a more subtle approach is to work with two different kinds of representation using Lie algebra. For general representations quaternions are employed, but for differentials Lie elements are used (equivalent to the odometry readings). Then appropriate sums, subtractions and other transformations are defined on these elements. This way whenever a covariance is required, it is extracted on the Lie space, i.e. with six dimensions. Since the dimension corresponds with the state rank, there's no need for complicated noise heuristics; at the same time, using appropriate nearby linearisation points, gimbal lock cannot occur, since all operands are far from the locking points. This approach works equally well for modelling, but is easier to reason about when doing inference with covariances (for the later parts of the project).

Both approaches use the same deterministic prediction equations. Given the vehicle pose $(X_k, Q_k)^T = (x_x, x_y, x_z, q_w, q_x, q_y, q_z)^T$, it is updated as

$$\delta q = \text{YawPitchRoll} \rightarrow \text{Q}(d\psi, d\phi, d\theta) \tag{3.1}$$

$$Q' = Q_k \delta q \tag{3.2}$$

$$Q_{\frac{1}{2}} = \text{Slerp}(Q_k, Q', 0.5) \tag{3.3}$$

$$\begin{pmatrix} X' \\ 0 \end{pmatrix} = \begin{pmatrix} X_k \\ 0 \end{pmatrix} + Q_{\frac{1}{2}} \begin{pmatrix} dX \\ 0 \end{pmatrix} Q_{\frac{1}{2}}^{-1}, \tag{3.4}$$

where $d\psi, d\phi, d\theta$ are, respectively, the differential yaw, pitch and roll odometry observed in one frame and δq is the corresponding differential odometry quaternion. Q' is the predicted orientation of the vehicle and $Q_{\frac{1}{2}}$ is the halfway spherical interpolation between the predicted and previous orientations. dX is the translation odometry reading. Products must be done in the correct space (quaternions should use quaternion multiplication). For the noise, the 7D quaternion-space repre-

sentation uses a simple addition in the final space,

$$\begin{pmatrix}\mathbf{X}\\ \mathbf{Q}\end{pmatrix}'' = \begin{pmatrix}\mathbf{X}\\ \mathbf{Q}\end{pmatrix}' + v_k \tag{3.5}$$

$$\mathbf{X}_{k+1|k} = \mathbf{X}'' \tag{3.6}$$

$$\mathbf{Q}_{k+1|k} = \text{Normalize}(\mathbf{Q}''), \tag{3.7}$$

where $v_k \sim \mathcal{N}(0, \mathbf{R})$ is motion noise in seven dimensions and the $\mathbf{R}$ motion noise covariance is rank-deficient due to the over-representation of the quaternion space. Alternatively, the Lie representation introduces noise in the Lie space, which is equivalent to running the deterministic equation again with a zero-mean noisy odometry.

Measurement model

The measurement model uses the pinhole camera equation to define the relationship between a landmark position and the pixel position and range that it maintains with respect to a camera,

$$\begin{pmatrix}x\\ y\\ w\end{pmatrix} = \begin{pmatrix}f & 0 & 0 & 0\\ 0 & f & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix} \begin{pmatrix}\mathbf{R_{3x3}} & t_{3x1}\\ \mathbf{0_{1x3}} & 1\end{pmatrix} \begin{pmatrix}X\\ Y\\ Z\\ W\end{pmatrix}, \tag{3.8}$$

where $(x, y, w)^T$ are the feature coordinates in homogeneous 2D coordinates, $(X, Y, Z, W)^T$ are the landmark coordinates in homogeneous 3D coordinates, f is the focal length of the camera, $\mathbf{R_{3x3}}$ is the camera rotation and t_{3x1} is the camera position in space.

Noise can come from two sources: the depth map uncertainty and the keypoint extraction jitter. This last one occurs because keypoints are not always found at the exact same location, due to viewpoint or illumination changes. Keypoints may also disappear (misdetection) or not correspond to any real landmark (clutter). The measurement noise covariance is set at 2 [px^2] for each "film coordinate" and 1 [mm^2] for the range coordinate.

A detail in the measurement model has to do with the edge of the visible space. While in theory the system is assumed to have a hard edge, in practice the keypoint detection does not work so reliably at the border. So, a visibility ramp has been introduced near the edge, where the probability of detection falls from P_D (e.g. 0.9) to zero in the last tens of pixels. Also, it is important to note that occluded objects are not visible, so if a landmark is behind the depth map, its probability of detection is set to zero.

SLAM Solver

Using these models, the PHD solver is based on the previous works of Vo [67], Mahler [63] and Leung [91]. The practical system implements the equations (2.47), (2.48) and (2.50) using particles; a mapping routine solves the problem with known localization and the reweighting scheme explained in section (5.1) of the second chapter is used to update the localization particles.

When implemented with Gaussian mixtures, the correction step requires the Jacobian of the measurement model to calculate uncertainty covariances, which follows the rule in equation (3.9),

$$J = J_{\text{projection}} J_{\text{rotation}} \tag{3.9}$$

$$J_{\text{projection}} = \begin{pmatrix} f/Z & 0 & -fx/z^2 \\ 0 & f/Z & -fy/z^2 \\ x/|\mathbf{X}| & y/|\mathbf{X}| & z/|\mathbf{X}| \end{pmatrix} \tag{3.10}$$

$$J_{\text{rotation}} = \text{Quaternion} \rightarrow \text{Matrix}(\mathbf{Q}). \tag{3.11}$$

An implementation consideration must be made for the prediction step, as theoretically the "birth volume" should be the whole visible space, but in practice this is too wasteful as it requires too many Gaussians to fill the space and most of these components are removed immediately in the next step. To counteract it, only the "unexplored new measurements areas" are used as birth volume. This can't be done trivially: just using the measurement points would lead to comparing them with themselves in the correction step, which produces an artificial overconfidence in the observations. A better approach is to use the unexplored measurements from the previous update cycle, as it doesn't contaminate as much the correction step.

1.3 Pipeline

Using these definitions, the program follows the pipeline described in figure (3.2). The main components are the Vehicle, the Navigator and the Manipulator. The first one is in charge of defining the motion and measurement models; the second one must define the SLAM solver algorithm and apply it to the vehicle; and the third one defines the global flow of the program.

Each component has been designed to be easily replaced for another implementation making use of object oriented techniques. This feature has been extensively used in three areas, which correspond to the three main components: in the input area the system can be a simulation or a device acquisition module; in the navigator class, the algorithm can be PHD, iSAM2 or other alternatives; and in the visualization tool, the program can either solve the system or show a pre-recorded run for further analysis. The same flexibility allows the system to use different kinds of motion models, measurement models and internal vehicle state representations. For this thesis, two such models have been implemented: the aforementioned 3D visual model and a simpler 2D linear model that will be used in chapter 5. The user may decide between these different implementations using command line arguments to the program.

Input

To directly interact with the program, the simulation reacts to user input through the keyboard and the mouse. Using the keyboard, they may move the simulated vehicle in its z-axis (front/back) and may rotate it around its three main axis: yaw, pitch and roll. All these motion are performed at constant rate for simplicity. Only four of the six degrees of freedom are manipulable; otherwise the control would become too cumbersome.

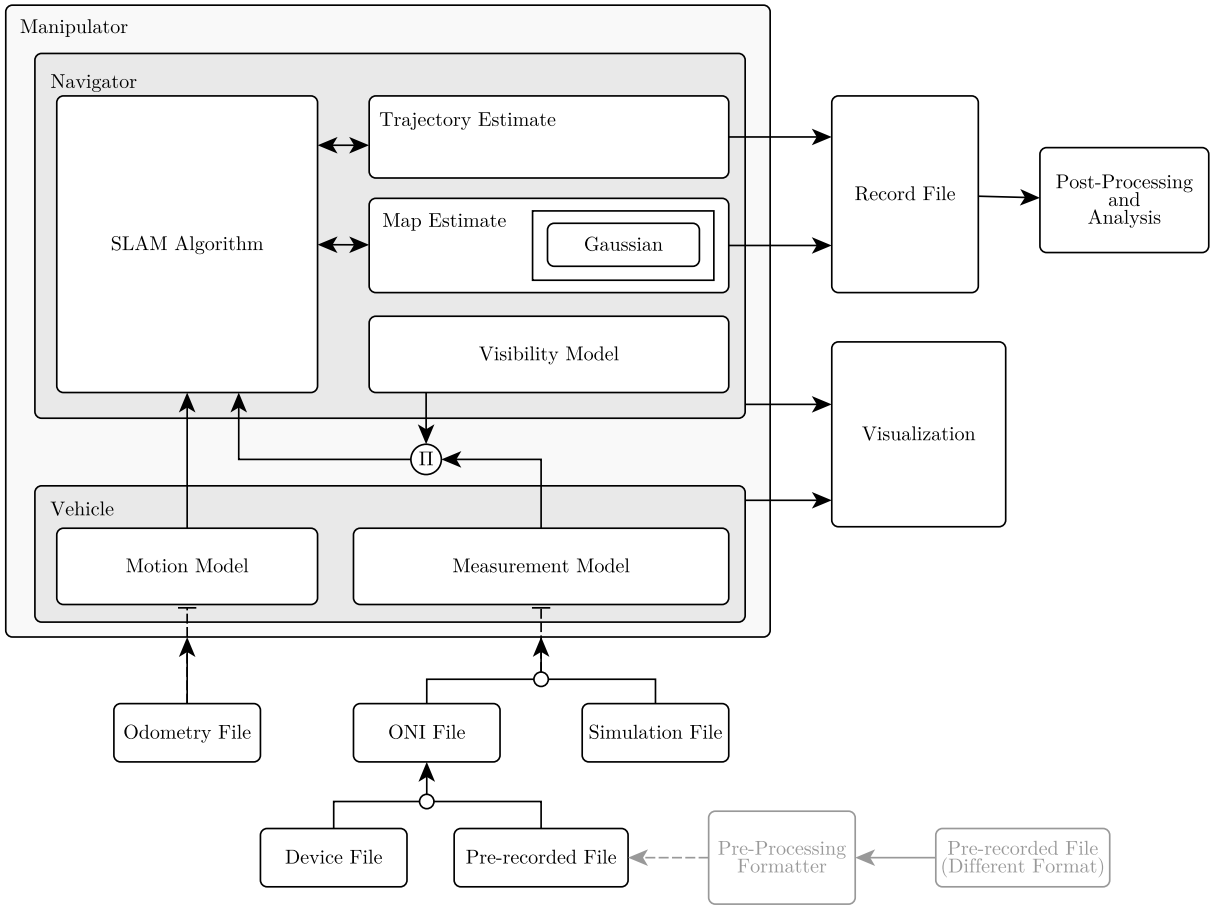


Figure 3.2: System pipeline.

Since manual input is not useful when running many tests, the system can use an automatic input script, which reads commands from a file and execute them. In this mode, the script has more freedom to move the vehicle, since all six degrees of freedom can be adjusted and at any dynamic rates.

Additionally, while simulating, the user can move the camera to observe different perspectives in the 3D space. The camera can be zoomed, rotated and centred around key objects in the scene for fine analysis. The mouse is used to centre the camera onto a target object.

Other toggle keys configure the environment and will be explained later in this section.

Output

The user receives feedback about the SLAM solution in two ways: first, an immediate visualization shows the results in an interactive manner and second, the final solution is compared against the groundtruth data (in the case of the simulation) and plotted.

The rendering contains all the information needed to understand the solver state. It shows the real pose of the vehicle in time, its trajectory so far, the SLAM estimate for this pose, the real

map, the estimate for the map and the measurements locations along the way. A sample from this visualization can be seen in figure (3.3), along with an explanation of the geometrical representation. Note there is also a frustum indicating an approximate field of view for the vehicle.

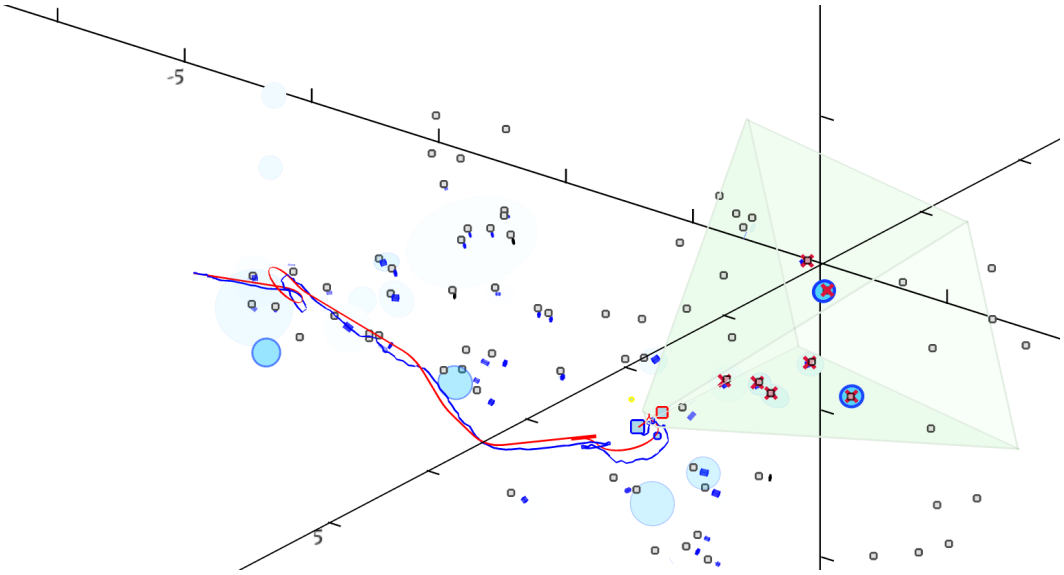


Figure 3.3: System real-time rendering. The vehicle is represented as a four squares (the big blue one is the center, the smaller red one is the front and the small yellow and blue ones indicate the rotation); the groundtruth trajectory is in red; the estimated trajectory is in blue; the landmarks are light-gray squares; the current measurements are red crosses; the best map estimates are blue/black ellipsoids; and the visible area is in a translucent frustum (shown in green).

In some cases, some of this information is not available, so it is not shown. In particular, when using real sensor data, there's usually no map groundtruth to compare. Conversely, when using the RGB-D sensor, the full depth map can be optionally projected onto the visualization area. This does not make sense on the point-map simulations, although the framework allows them to be easily implemented. This visualization can help the user to visually detect errors or drift.

The chosen implementation of PHD-SLAM uses particles, so several pose/map estimates are present at every time. For simplicity, there are two modes of presentation: either the Maximum A Posteriori (MAP) estimate trajectory or all of them can be shown. For the map, the one associated with the MAP trajectory particle is shown.

This map estimate is formed by a mixture of Gaussians, so sigma-ellipsoids are used to represent each component. The ellipsoids are projected onto the camera plane and their associated weights influence the way they are drawn. As the weight becomes close to zero, they become transparent, and their border becomes thin. When they pass one, their border color changes from blue to black and when they pass two, it becomes red. This gives the user a visual feedback around the confidence on the map.

The post-processing plots are generated after the main system runs. Python and matplotlib are used since they have much better plotting capabilities. With them, two plots are generated: one giving the pose estimation error and one giving the map estimation error.

For the pose error, the Euclidean distance between the real location and the estimated location is calculated in time. There are three ways to show this distance: in filtering mode, at every time step, the SLAM algorithm produces a best estimate, which is compared against the groundtruth; in smoothing mode the whole trajectory is processed with the PHD algorithm and the final estimate for every time step is compared against the groundtruth; and in average mode at every timestep all the errors in the partial estimation are averaged. These three modes expose different properties in the system. The filtering approach shows the quality of the information that the system has when working in real-time; the smoothing approach shows all the information present in the final solution, which can help identify difficult places which couldn't be well observed; the averaged approach gives more information about how the global information changed over time, indicating for example precisely when the system was able to close a loop or when it slowly drifted apart.

For the map error, the OSPA metric is used as explained in the previous chapter. Note that to generate a point-map from the RFS distribution, the means of all the components with weight higher than 0.8 were used.

The user can introduce time bookmarks to pinpoint specific events, which show as vertical bars in both plots. Also, since the system outputs all its information as raw files, additional processing can be performed as well.

Optimizations

The primary optimization in any project is always choosing the right algorithms and data structures, so care has been put into really thinking and refactoring all the structures on the code to fulfil their role as efficiently as possible, but there are a few of notable cases worth mentioning.

The one thing to mention about "general optimizations" is that they were performed with the help of the SlimTune profiler, which allows the programmer to check and pinpoint the code bottlenecks. This way there's no need for micro-optimizations that only complicate the code without real benefits. An example of its use was the observation that many Gaussian evaluations were duplicated or could be swapped with faster Mahalanobis distances comparisons. This optimization was relatively simple and small but was so profusely repeated through the code that the slight change made a real difference.

The most important optimization was to use parallelization to distribute the bulk of the work into multiple CPU cores. To do so, a high-level parallel-for construction was used in the SLAM recursion call. This works very well, since these calls are relatively long, so the overhead is negligible and the work performed on different particles is trivially independent, which means that the speed-up is basically eight-fold on an eight core machine.

The code also uses kd-trees instead of lists to keep the map models. Trees can query for close neighbours much faster than lists: instead of searching the whole $O(n^2)$ space, trees can do the same task closer to $O(n \log n)$.

Another simple optimization has been to tabulate the exponential function inside the Gaussian evaluation routines. The approximation doesn't reduce map quality and gives some improvement in efficiency.

One important thing to consider is the language in which the system is programmed. In this case, C# gives several options to represent matrices. The most common are multidimensional arrays and jagged arrays (array[a,b] and array[a][b] respectively). Due to the JIT compiler, these two alternatives differ vastly in terms of performance, since the jagged version allows smart check bounding elision that gives the system speed without compromising safety. This option is used throughout the project.

Two interesting optimizations in the code are the elimination of *far away* comparison in the corrections step, where the relationship between measurement and landmarks too far away from each other (Mahalanobis distance) is obviated; and sparse matrix implementation for the combinatoric routines of the weight update method. This allows, for example, to drastically reduce the time of the linear assignment problem, since the full problem must deal with $O(n^2)$ elements, while the sparse version only deals with a typically linear subset of the whole matrix $O(n)$, an improvement both in memory and in execution time.

For computational reasons, the birth density has been approximated around the actual measurements, since all the rest would just disappear anyway (note there's a one-frame delay to avoid overconfidence).

Re-runs and Configurability

To compare different algorithms, the system takes advantage of the full raw data that is output after every run to allow for re-runs of the exact same problem multiple times. These re-runs are the same, from having the same groundtruth movement to reading the same noisy odometry and measurements. This makes the comparison fair, since different alternatives face exactly the same problem.

Since there are many parameters in the algorithms and the simulation settings, it can be tedious to specify them every time the program is run. This is why, all non-essential shared configuration details are described in a file, which can be reused in different runs. This keeps the comparisons fair, since it's less prone to typing errors than specifying the files each time.

A special class called Config is used to hold all configurable data. Using reflection, the program understands the structure of this class and can automatically generate adequate file parsers, which are transparent to the coder. This way, adding new configurable constants is as easy as adding a new field to the class, without breaking any backwards compatibility.

The more obvious configurations are all the algorithm parameters, like the birth rate, birth covariance, probability of detection and clutter rate. There are others that are more nuanced and have to deal with efficiency, like merge thresholds and video undersampling. Some of them control parts of the process like filtering the image keypoints with RANSAC or not, disregarding odometry input and assuming incorrect noise estimation values. Finally, there are visualization choices, like showing all particles or only the best one, and coloring the map by visibility or confidence.

The system can also choose between rendering all pose particles or only the MAP one; the map always corresponds to the MAP particle.

Apart from these upfront configurations, the user can change the behaviour of the program as

it progresses. When these events occur, appropriate bookmarks are added to the timeline, which show in the plots. This way, the user can know exactly how the system reacted when they pressed a key.

In particular the user may reset the estimates and the whole simulation at any point, to easily correct mistakes or try from a different initial state. Additionally, they can switch between solving the entire SLAM problem, or only its mapping component, i.e. assuming localization is perfectly known. This helps understand the effect of each stage separately.

The user may also pause the simulation, explore the world by focusing the camera on any particular important element in the scene.

Fair start

Since in the first few frames there is very little information as the map is just forming, the vehicle can drift without notice, making the ensuing trajectory and map estimation highly volatile in a uninteresting way, since it corresponds to the Euclidean free parameter. To avoid any random advantages in the comparisons, the first few frames are assumed perfectly localized so a minimal and coherent initial map can form that may guide further estimations.

Testing and Reliability

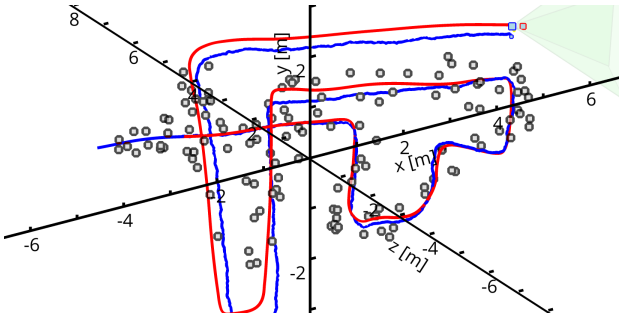
Correctness of code is essential for research. To be able to guarantee it, it is important to take adequate measures such as testing and careful structural design.

In this case, extensive testing has been performed on the output of every component in the modules. Some of it is in the form of automated tests, which consider unit tests, which ensure the correct functioning of complex routines like Murty's algorithm, and mock simulations, which have easy properties which can be asserted.

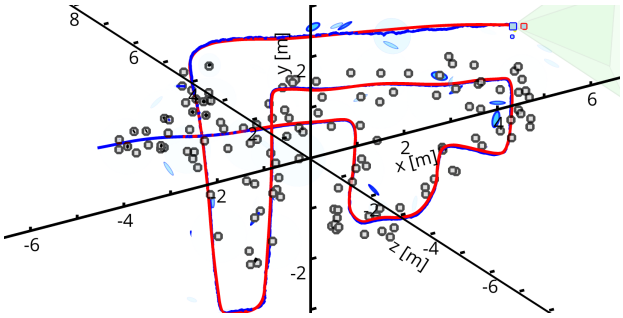
Besides these automated tests, some manual tests have been performed on many trial runs, including edge cases, all of which give the expected results. During the development of the inner code of the PHD filter, which is conformed of several stages, all intermediate output could be bypassed, so as to analyse the behaviour of every stage separately. Also, since the system has both a visual output and a text output, any behaviour can be inspected both visually and numerically. Further, the system can record previous runs for later deeper inspection and comparison against other algorithm sources. Consistency checks between different runs of the same pre-recorded session are correct, giving the exact same output for the same input and algorithm (for deterministic algorithms).

Along with testing, the program has been designed to be scientifically robust against code bugs. Every component has been well isolated so errors can't easily propagate and information flow is limited quickly after the contact point between objects. For example, when creating a Navigator that tracks a certain Vehicle the developer could inadvertently make illegal use of the real vehicle trajectory on some of the SLAM solver routines, making it seem as the solver works better than it really does; this is avoided by creating an unlinked copy of the Vehicle in the Navigator constructor. This way, any error involving the wrong Vehicle can't use privileged information, since there's no

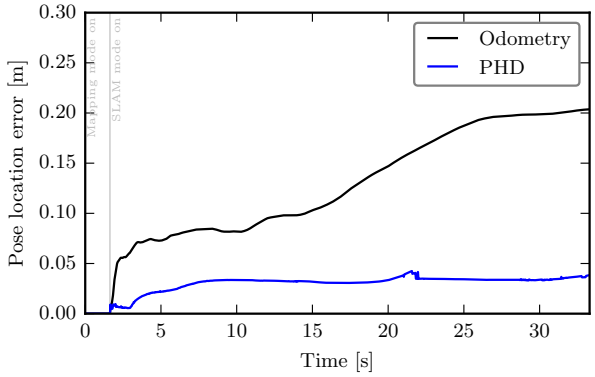
simple access to it past the constructor. This property is easy to test and therefore any incorrect improvement of performance must be due to accidental improvement of the method instead of inappropriate information access. Similarly, base structures such as Gaussians, which may be subtly shared between different components, have been defined as immutable so they can be used without worrying about improper information flow.



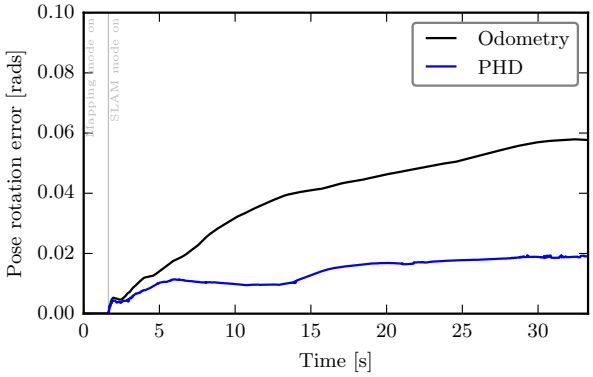
(a) Odometry dead reckoning estimate. The red trajectory is the groundtruth and the blue one is the estimate.



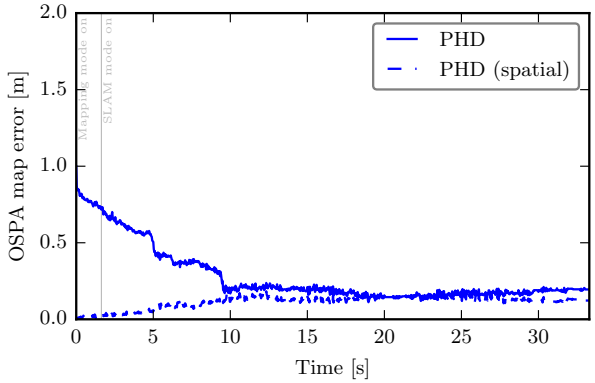
(b) PHD estimate. White rectangles are landmarks and blue/black ellipses are the map estimate.



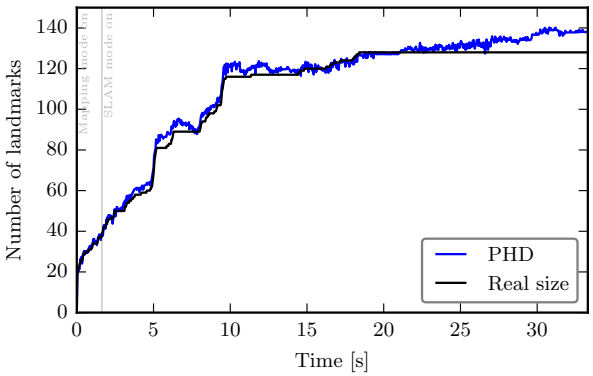
(c) Location error (average Euclidean distance between groundtruth and estimate).



(d) Rotation error (average absolute angle between groundtruth and estimate).



(e) Mapping error (OSPA, $c = 1\,[m]$ and $p = 1$). OSPA and disregards the difference in cardinality.



(f) Map size estimate (integral of the PHD over the entire 3D space).

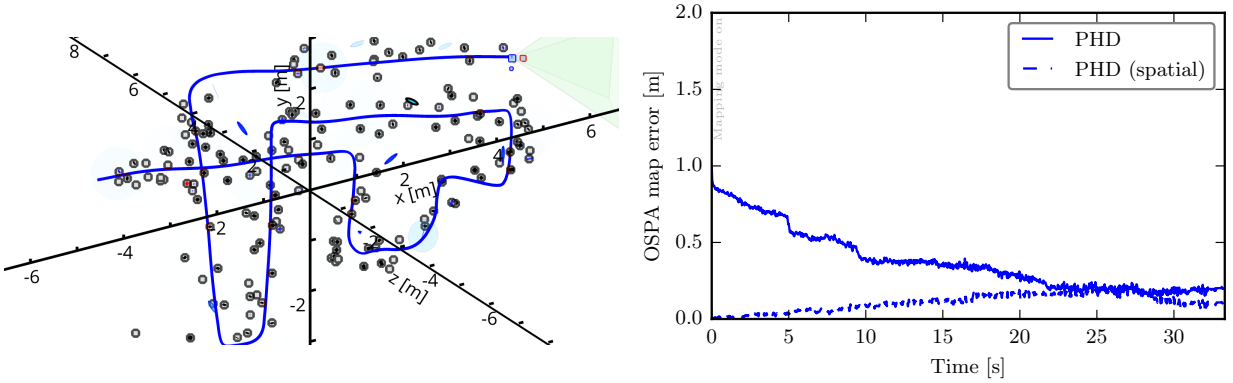
Figure 3.4: PHD filter performance on a simulated environment. 'Spatial' corresponds to the spatial component of the PHD over the

2 Results

The PHD filter is tested using the implementation detailed above. To do so, its output is compared with the raw odometry reconstruction in a simulated baseline case. To evaluate its performance, the Euclidean distance between the groundtruth location and the estimated location is calculated. Similarly, the mean angle between rotations is also plotted. Additionally, the OSPA distance between the groundtruth map and the estimate is obtained. To extract a point map from the RFS estimate, the expected map size $N_m = \int v(x)\, dx$ is calculated and the $[N_m]$ most likely landmark locations are used. In the following experiments, unless noted, 800 particles are used, with a measurement covariance per unit of time of $2\,[px^2]$ in the pixel components, $0.001\,[m^2]$ in the range component, a motion covariance per unit of time of $0.05\,[m^2]$ in the location components and $0.002\,[rad^2]$ in the rotation components. The sensor statistics contemplate a probability of detection of 0.9 and a clutter rate of $3 \cdot 10^{-7}$, which integrates to an average of 0.912 clutter measurements per frame.

The first experiment (shown in figure (3.4)) consists of an easy scenario, where the model statistics are near ideal. The filter performs much better than the odometry; in many segments, the pose error does not even go up, which means the system can remain correct despite odometry noise. The mapping error goes down as landmarks are observed, as it is expected.

A second experiment assumes known localization, so as to understand how well the SLAM mapping stage performs alone (shown in figure (3.5)). Although the map error steadily goes down in time, it doesn't reach zero. One might assume that the error should converge to zero, but this is not the case. For example, if a landmark was only seen once, the corresponding map density component will be centred around its unique measurement. Since the measurement has noise, the landmark estimate will be invariably incorrect and there can be no estimator which gives a better estimate, since there is no other information than the noisy measurement. However, measurements are unbiased so in average they give enough information to improve on the odometry, especially when the landmark is seen multiple times. This is why the map error can go very low, but it cannot be expected to reach zero (also, there are numerical optimization induced errors). The results are good nevertheless and it can be seen that this map error is similar to the one on the previous experiment,



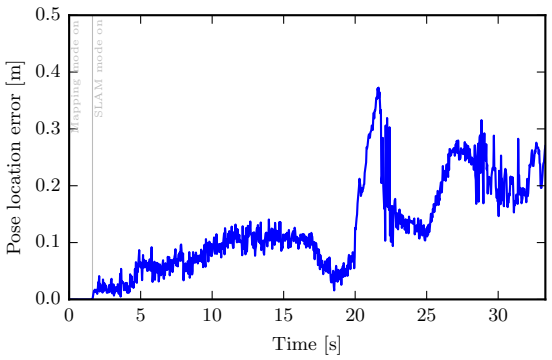
(a) Map estimate.

(b) Mapping error (OSPA, $c = 1\,[m]$ and $p = 1$). The error decreases as new landmarks become visible and are correctly estimated.

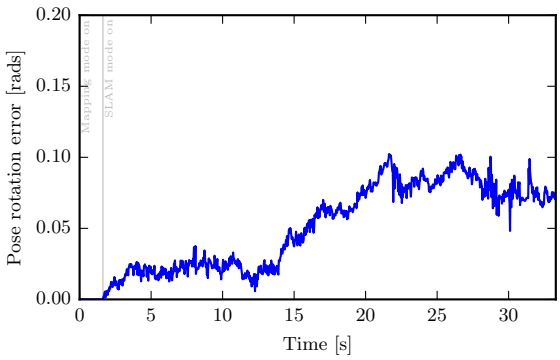
Figure 3.5: PHD filter performance with known localization

which considered an unknown localization. This demonstrates the capacity of the PHD algorithm to give a good map estimate despite of localization uncertainty.

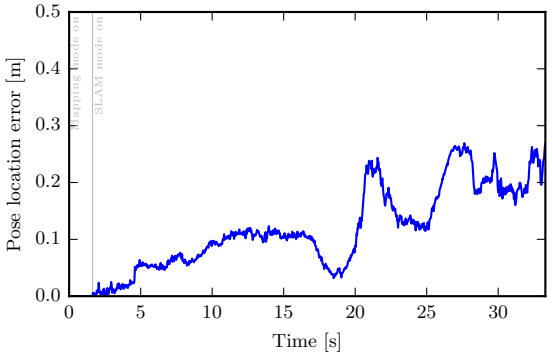
As designed, the system can output different pose plots ('modes'), each one of which remarks a different aspect of the solution. In figure (3.6) the three modes are compared. Observe that the filter mode is noisy as the estimate bounces between particles in time. Both the filter and the smooth modes vary heavily in time, while the time-average one is steady. This may be attributed to the time-local aspect of these plots; if the rotation of the estimate is incorrect but the location odometry is correct, the estimated trajectory can move closer and farther from the groundtruth. Averaging over time can marginalize these effects since it averages over parts of the trajectory that are close together



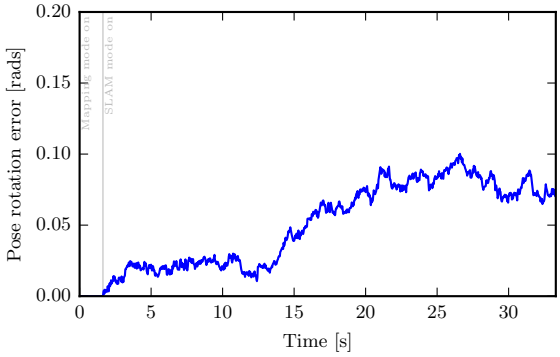
(a) Filtering mode location error.



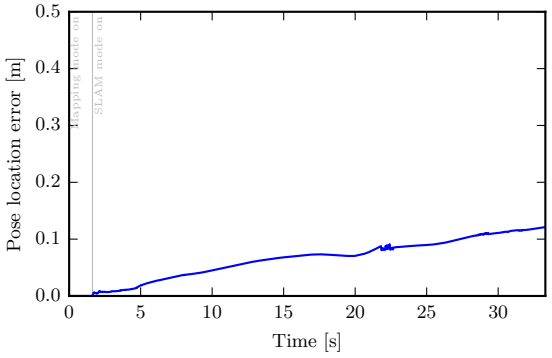
(b) Filtering mode rotation error.



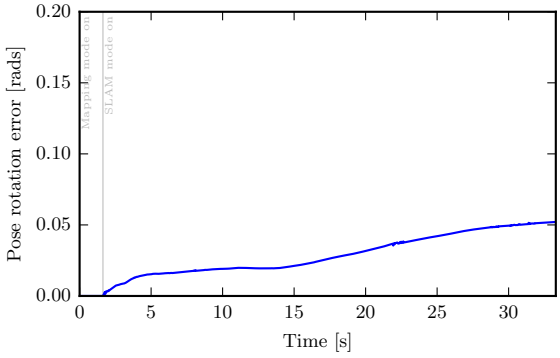
(c) Smoothing mode location error.



(d) Smoothing mode rotation error.

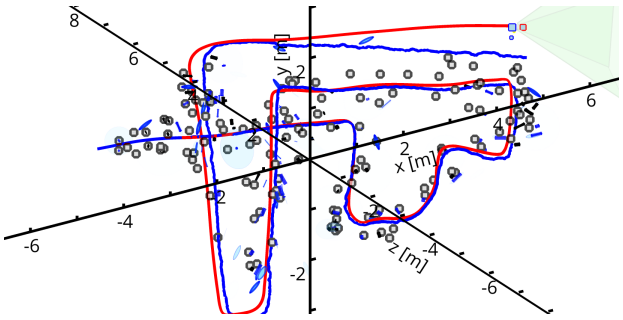


(e) Averaged mode location error.

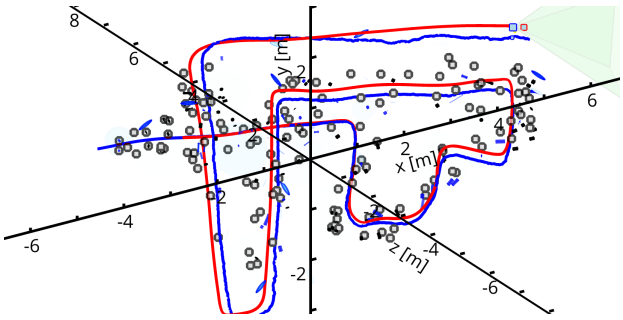


(f) Averaged mode rotation error.

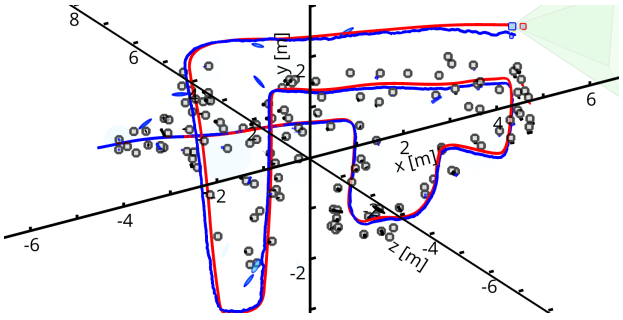
Figure 3.6: Localization plotting modes.



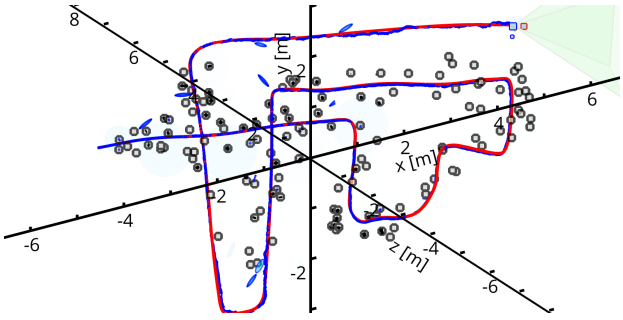
(a) PHD estimate with 20 particles.



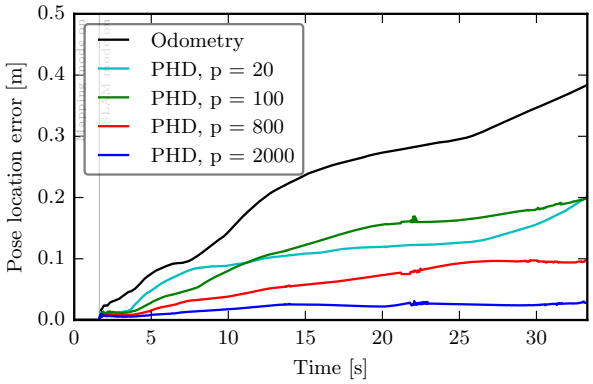
(b) PHD estimate with 100 particles.



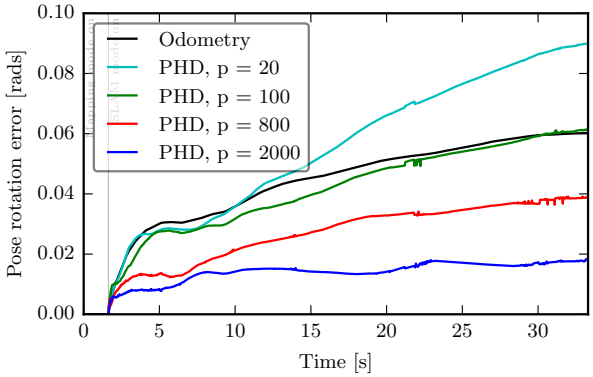
(c) PHD estimate with 800 particles.



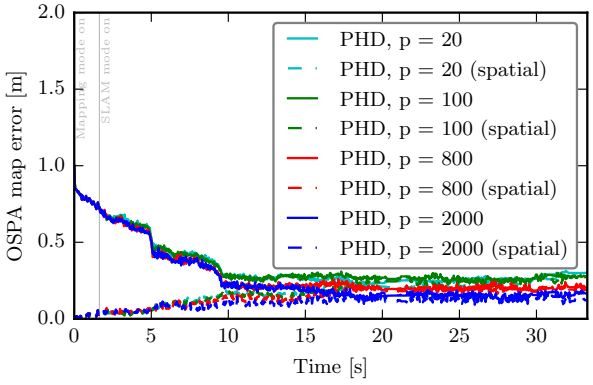
(d) PHD estimate with 2000 particles.



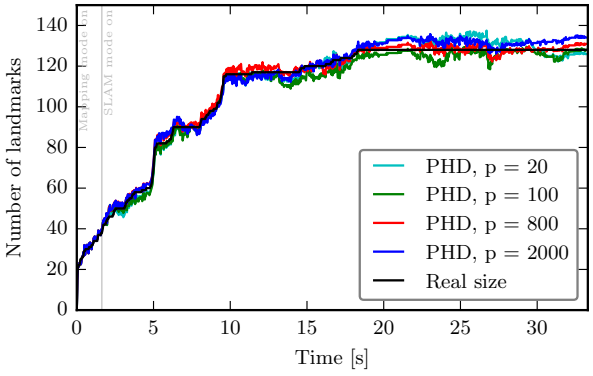
(e) Location error for different N_p.



(f) Rotation error for different N_p.



(g) Mapping error (OSPA, $c = 1\,[m]$ and $p = 1$) for different N_p.



(h) Map size estimate for different N_p.

Figure 3.7: PHD filter performance with different number of particles (localization step).

and those which are far away. The end result is that average plot is smoother and gives a general view of the error behaviour.

Also, it can be observed that the smoothing mode curve is quite similar to the filtering one, since they are reflecting the same places in the spatial trajectory. However, the smoothing mode is smoother due to using information of the future to estimate the trajectory (the system can "backtrack" and correct previous estimate). The level of smoothness is related to the number of particles used for the filter, since more particles means more memory to backtrack. So, the comparison between these two plots is a useful metric of particle depletion. It shows the typical time frame where smoothing occurs; if there are too few particles, they become rapidly depleted and both plots would be pretty much the same; if there are many particles, the estimate can change freely and the smoothing width would be large.

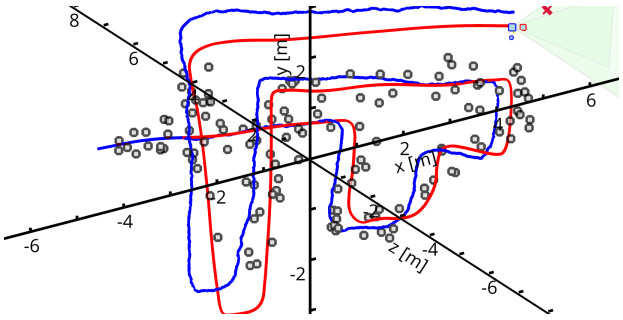
The plot in figure (3.7) shows the performance of the filter with different number of particles. As particles are added, the system localization performs better but the gain in the mapping stage is modest. This is consistent with the previous results of known localization mapping, which gave similar results to having an uncertain localization. Using many particles would be ideal but increasing this value requires more computational resources, both time and memory, which limits the use of the system in real-time or embedded conditions. Note that for low number of particles (e.g. 20), the solution can get randomly lucky sometimes and unlucky at other times due to the underrepresentation of the pose distribution. They can also present "indecision" effects (not shown on this experiment), where two good solutions are present and they fight for the top spot, which shows as wild oscillation between two error levels in the plots.

All of these runs have assumed a perfect knowledge about the system statistics. This is unrealistic in most real situations, which would have to estimate them from data, so a couple of imprecise runs have been plotted in (3.8). In the experiments two alternate configurations are used, where the measurement and motion covariances are over and under estimated by 20%. Incorrect statistics result in somewhat worse performance, but the resulting estimates are still good and well below the odometry error in the case of location. The rotation angle error seems to be a less reliable signal in all experiments and is the only one of five in this one to show deterioration (but looking at the trajectories it is clear that all configuration reached a satisfactory result). This shows that the algorithm is robust against uncertainty in the model statistics, which is very relevant in real applications.

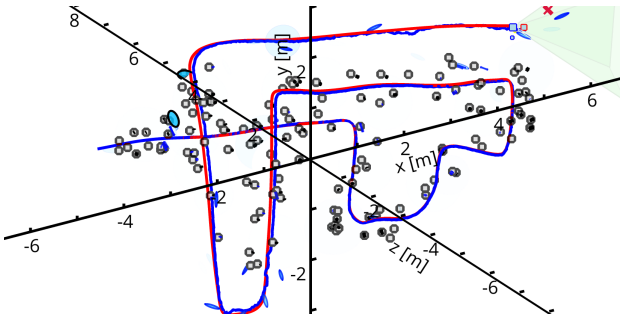
Similar experiments on over and under estimation of the probability of detection and the clutter rate show that underestimating P_D makes the system overconfident about the landmark weights, which can reach values higher than two (incorrectly) if P_D is low enough. In the inverse case, an overestimated P_D can make the system remove landmarks too heavily when they are not seen, usually removing them completely. The clutter rate has a similar but inverse effect.

In any case, the complete estimate suffers but, similarly to the covariance variations, they are still much better than the odometry.

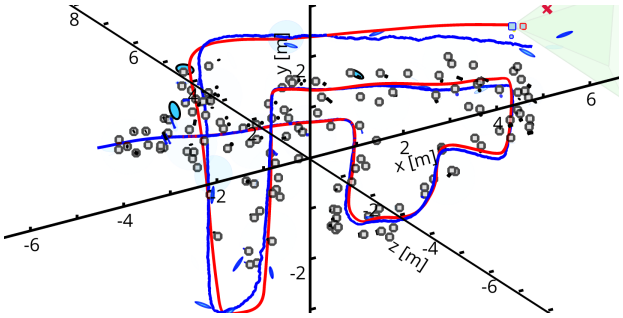
Finally, the output of the system working on a real sensor is shown in figure (3.9). In this case, apart from the estimated trajectory, an image processing sample output is shown, with the extracted keypoints. Since there's no map groundtruth, the map error can't be calculated and its quality has to be visually estimated from the screen and the scene. By inspection, the map looks reasonable as the computed landmarks resemble the spatial configuration of the room in the video.



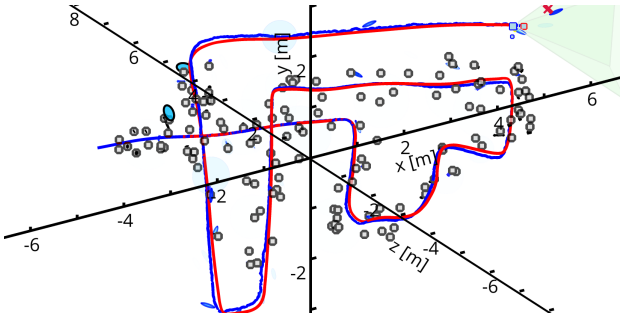
(a) Odometry estimate.



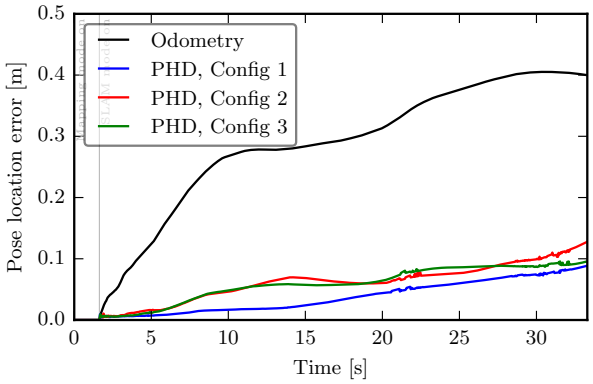
(b) PHD estimate with correct statistics.



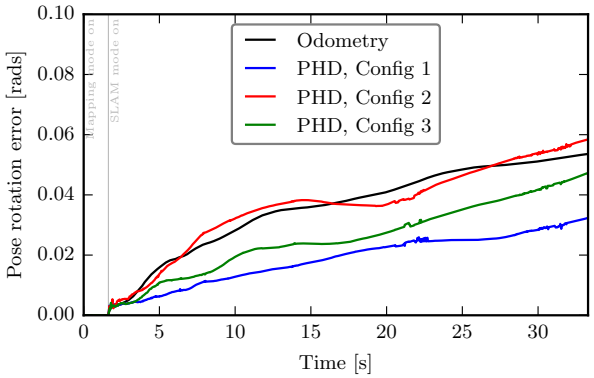
(c) PHD estimate with overestimated noises.



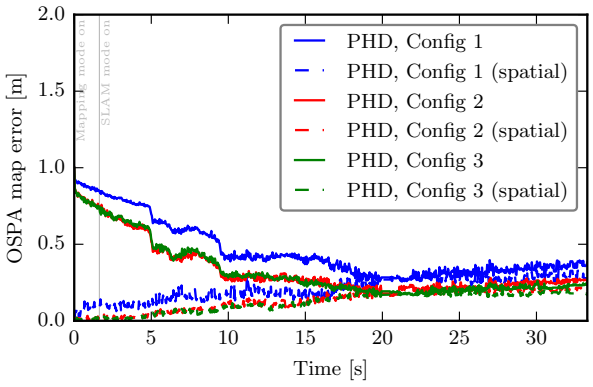
(d) PHD estimate with underestimated noises.



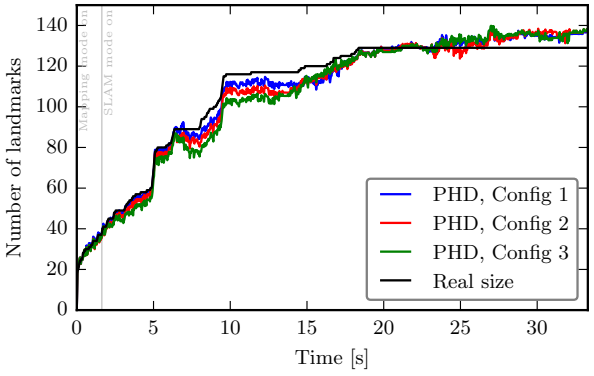
(e) Location error with incorrect statistics.



(f) Rotation error with incorrect statistics.



(g) Mapping error (OSPA, $c = 1$ $[m]$ and $p = 1$) with incorrect statistics.



(h) Map size estimate with incorrect statistics.

Figure 3.8: PHD filter performance with incorrect statistics. Config 1) Correct statistics, Config 2) 20% overestimation of noise covariances, Config 3) 20% underestimation of noise covariances.

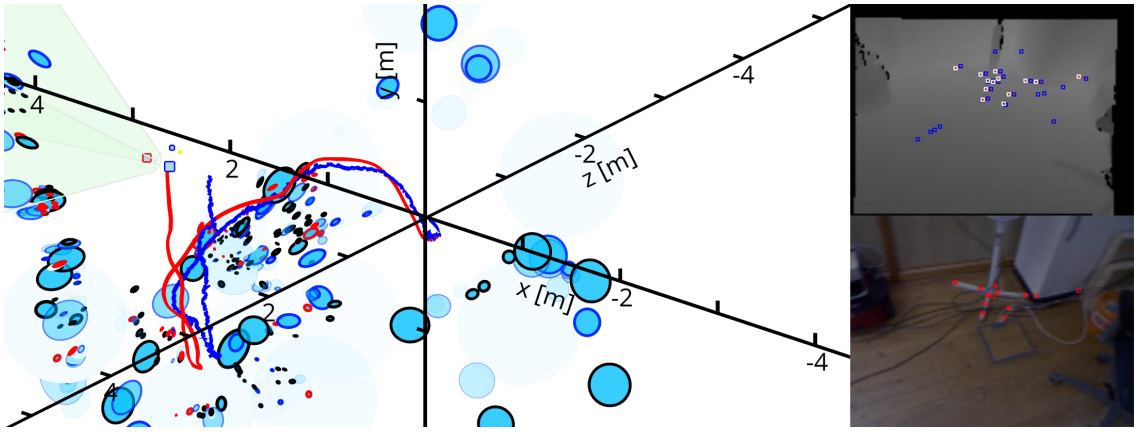
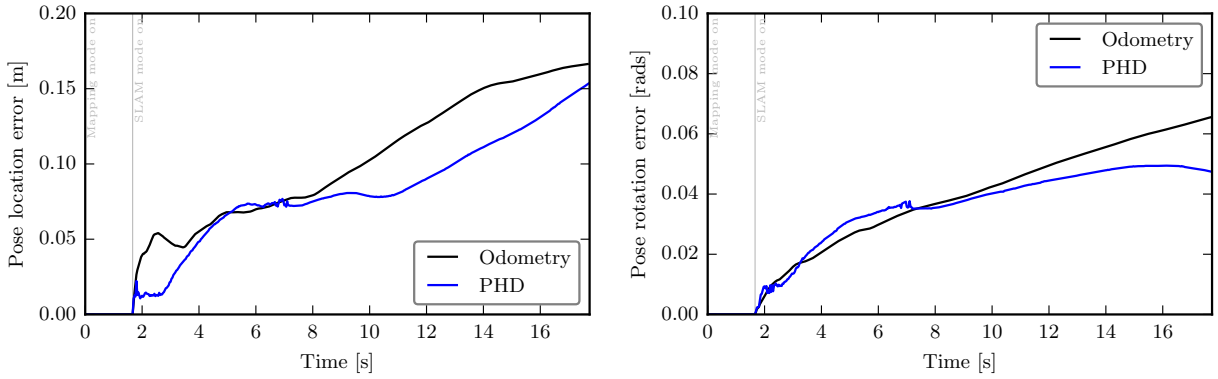


Figure 3.9: PHD estimate on a real video. On the right, the extracted keypoints can be seen on both the color and depth images, as well as a matching between frames (for filtering).

The SLAM solution is accurate at first in this scenario (figure (3.10)), but four issues have been detected which deteriorate its performance in time. First, the number of particles seems to be too low (as observed by the smoothing behaviour). Second, there is a short time period around 4 [s] containing too many measurements due to a highly textured scene. Third, as the experiment progresses, the number of components in the map PHD mixtures grows to accommodate for the newly observed landmarks. By the end of the experiment, these Gaussian mixtures have reached the maximum number of 600 components (since every one of the 2000 particles has a map estimate, adding more components becomes too computationally expensive), so the algorithm has problems including new landmarks to its estimate. Fourth, in cameras, landmarks are not so well defined because looking at the same object from different perspective shows a different projection of it which may have inconsistent landmarks, e.g. a extreme case is a sheet of paper that has different printed logo on each side (and occupy approximately the same place in space).



(a) Location error in a real video.

(b) Rotation error in a real video.

Figure 3.10: Pose error in a real video. Note there's no map groundtruth available, so OSPA can't be plotted.

3 Discussion

As seen from the results, the system works as intended, managing to correctly track the vehicle's location and the map despite the noise, clutter and missed detections.

However, loop closure was not perfect since there's only a limited number of particles; during time gaps where no landmarks are visible, the representation power of the particles decays rapidly since the probability density becomes flatter and requires more particles. As observed in the first experiment, this effect can occur even if new *different* landmarks are observed (see figure (3.11)); if the observed sets are disjoint, the new observations will only improve the new area estimate and the vehicle pose in regard to this new area, so the particle selection will poorly consider the different hypothesis for the transition period, i.e. the no-landmark period will get stuck with a random motion*. Later on, if one of the first landmarks is revisited, i.e. loop closure, the system will not be able to fix the estimates perfectly since the no-landmark transition segment will contain too few, if any, options.

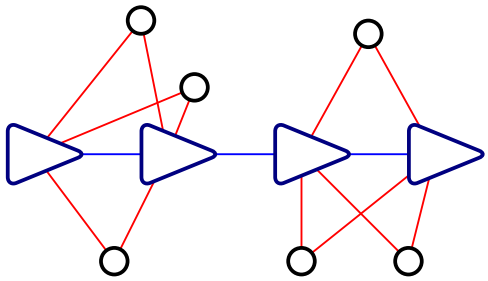


Figure 3.11: Disjoint measured-landmarks sets between time steps decouples the estimates for both sections. If this property holds for too long, the system cannot recover a good estimate for the transition relationship and the estimate will diverge. This is one reason why RFS cannot solve loop closure satisfactorily under current techniques.

Since only one particle's map is rendered (MAP trajectory), its density is conditioned on the pose, i.e. it does not correspond to the global probability for each landmark. Alternatively, the Expected A Posteriori (EAP) estimator could have also been used, averaging every particle and merging their maps proportionally to their weights, but this alternative is more expensive and MAP gives the "best" hypothesis, even though it may not be the most representative of the distribution.

When the vehicle is lost (due to lack of measurements), the MAP pose estimation quickly jumps as the best particle changes radically due to new measurements (loop closure) or due to competing hypothesis. The user can then visually see the precise re-localization moment with ease.

The MAP estimation, though, lacks information; it does not contain any sense of uncertainty. The alternative ensemble render mode, which shows every particle pose, gives the user some feeling for the uncertainty on the pose as the particles spread out as they track the probability density. This way it is easy to see when the vehicle is becoming increasingly lost and when the particles deplete. However, in this visualization mode there is a causality issue. As all the particles are drawn and

*As seen in the graph, the only way to move information between the old and new areas is through the poses, i.e. the particles. If these particles become flat, they give no useful information, hence the pose estimate becomes reliant only on the new area.

bad ones die off along the way, the final estimated trajectory correspond to that of the MAP particle *at the end* of the process, i.e. the poses are better than what the vehicle thought they were at the time, there are no jumps like in the point estimator. This mode does not represent an online process so caution must be taken with the results: the point estimator evaluates an online system and the ensemble estimator evaluates a slightly smoothing (offline) system.

Note that this work does not concern itself with efficiently extracting the odometry. Either an IMU reading can be integrated to obtain velocities or one of the many visual odometry routines can be used from the camera. This work is orthogonal to such subsystem and assumes a given odometry (e.g. from the dataset or by adding artificial noise to the groundtruth); an extension would be straightforward. Even further, since the system is based on Bayes filtering, both inputs may be used together.

Also, the keypoint extraction is relatively simple and could be upgraded later on. The extraction works fine, but when the camera motion is high, motion blur makes keypoints disappear since textures are lost. This is a difficult unmodelled phenomenon that occurs in cameras due to the shutter time, which does not occur in other sensors. Since the filter does not expect all landmarks to vanish for several frames, estimates can lower their quality: map components can be removed and weights flattened; the bad case is that good particles get dropped because of deceitfully low weights and the system would not be able to recover. To solve this problem, a motion blur detector could be used to disregard the map update in a difficult frame ($P_D = 0$).

Note that unlike the vector alternatives, the RFS framework uses both positive and negative information from the scene, i.e. both the presence of a landmark and its absence influence the map estimate. Since the vector system requires data association, a missing landmark is not associated, does not enter the process and therefore does not affect the estimates. In the PHD filter, such a missing landmark will reduce the density around it, eventually flattening it to zero. This implies a strict requirement for a good visibility model; in particular, if an object can't be observed but the system believes it should be observable (e.g. an occluded object in the scene), the density will incorrectly decrease. Such a visibility modelling has been introduced in the system based on the RGB-D sensor.

In RGB-D sensors this visibility model is given by the depth map: if a landmark is behind the depth map it is unobservable. A caveat is that in PHD the map is represented not by points but by a continuous density (landmarks are extracted after the fact), even though Gaussian components are used to represent such density (one should not confuse mixture components for landmarks). The visible space intersection with the Gaussian components is arbitrary, hence half of a component could be visible and the other half occluded. Practically, this is complicated since the Gaussian Mixture PHD update equations do not hold exactly around the visible edge; the density no longer can be approximated by a Gaussian mixture, since one half of the Gaussian is affected by the update and the other not.

To solve this problem there are a couple of options: first, a simple approximation could just associate an unobservable status to a whole mixture component based on its mean, which could work if most component are well within the visible range; second, associate each close-to-the-edge component with a probability of detection P_D proportional to its intersection area with the visible space, which should be statistically correct; third associate each "unobservable" component with a P_D proportional to its distance to the edge, i.e. making the edges fuzzy, softly decreasing P_D along

them.

The third option is similar to the second one in that in both cases P_D decreases as the landmarks moves into the unobservable region, but in the third option, the process is common to all landmarks regardless of their covariance and easier to compute. This system uses this third option.

A fourth method is to break the mixture components into smaller elements (maintaining the density integral, i.e. the map estimate size) and apply one of the three other options to each element. This way, the update can indeed affect only half of the original Gaussian. However, in this way, the number of components can become much bigger with time. Importantly, note that in the visual SLAM setting, the observable landmarks are always on the surface of objects, i.e. on the edge of the visible space, so this situation occurs all the time.

Additionally, the camera sensor border also introduces a secondary issue: the image keypoint extraction routine may become unreliable because the textures are not completely visible. Even though the center of a keypoint patch is within the image, the patch may be undetectable because half of the patch is outside the boundaries of the sensor. This can be dealt similarly, by using an appropriate decreasing P_D around the edges, or alternatively by artificially decreasing the sensor size while modelling, i.e. assume anything closer than ϵ to the border (or to a depth shadow) is unobservable. The alternative will never incorrectly decrease the density but it may lose the sporadic positive information. The previously mentioned third option is a reasonable trade-off.

A useful addition for future work could be to augment the measurement space to also consider the normal direction from the landmark, as a way to differentiate between close landmarks, but on different surfaces, e.g. two landmarks on opposite faces of the same piece of paper. In this case, the visibility model will have a hard time with no normals context.

This work has chosen RGB-D sensors because they are simpler; the system can be extended to stereo and mono vision later, as well as more exotic sensors like 3D lasers. This way, the work focuses on the most interesting aspect: SLAM. Note however that stereo and mono vision will have a harder time defining a visibility model since they do not have direct access to a depth map. An artificial depth map would be required, using depth estimation, optical flow, deep learning or recent dense and semi-dense techniques [92, 93, 94, 95, 58, 96] to gather it.

Note that compared to laser ranging, cameras are more difficult because of motion blur, small field-of-view and high computational requirements. Nevertheless, they can be solved and can provide richer information.

Finally, in the real case scenario the algorithm shows potential, but there are still some improvements required to use it in production. The use of more particles would be useful but expensive; in some cases such as big cars, it may be reasonable to use high-end computers, but for embedded devices other solutions are required.

The highly textured observations mean that the pose triangulation is very "good" from the measurements, i.e. the posterior probability is narrowly peaked. Although this should be good news, it means that almost all particles (which were sampled only from the motion model) lie on a low-probability space, which produces the particle depletion observed in the estimate. It may be approached by a method similar to FastSLAM 2, using a better predict sampling which considers the measurement, so as to avoid particle depletion (this would also help diminish the required num-

ber of particles for good convergence). Alternatively some heuristics can be applied on top of the methodology: measurement covariances can be artificially inflated or some of the measurements could be omitted randomly (changing P_D appropriately) so as to make the peaks more reasonable. This solutions are simpler but throw away information.

Also, camera inconsistent landmarks (e.g. the two-faced sheet of paper problem indicated at the end of the results section) can be improved by adding normal information to the measurement space, so the two sides of a paper are effectively separable. However, it should be noted that even with these inconsistencies, the algorithm can still work locally in time.

Improving any of these issues is useful, but in the remainder of this manuscript a more fundamental development will be pursued: the use of graphical methodologies along with RFS for estimating the state globally. As has been discussed in previous chapters, this approach has the benefit of observing the full picture at the same time and making global decisions, being able to solve loop closure more efficiently than filtering methods.

Chapter 4

PHD and GraphSLAM

To put the proposed visual SLAM system results into perspective, it is necessary to compare them against other well-established methodologies. In the recent literature, the GraphSLAM family of SLAM solver has become prevalent due to its global consistency properties and its new found efficiency. For this work, iSAM2 [43] has been chosen as the reference for comparison, since it is a fairly recent development with one of the best performances of the GraphSLAM family and it manages to be highly efficient.

The algorithms will be compared both in various simulated environments, where many parameters can be adjusted to understand their performance, and a real visual dataset provided by the Computer Vision Group at the Technical University of Münich (TUM) [97], which can help understand their applicability to real problems.

1 Setup

The setting for the simulation has 140 landmarks dispersed in a $10 \times 6 \times 3$ [m^3] volume. The vehicle receives projections into a centered camera sensor with a size of 640×480 [px^2], a focal distance of 575.8 [px] and a visible range between 0.1 [m] and 10 [m].

As the baseline configuration, the motion noise covariance was set to $0.05\,[m^2]$ for each translation component and $0.002\,[rad^2]$ for each angle axis. The measurement spatial covariance was set to $3\,[px^2]$ for each pixel component and $0.0002\,[m^2]$ for the range component. The probability of detection P_D was 0.9 and the clutter density κ was $3 \cdot 10^{-7}$ which integrates to $N_c = 0.912$ spurious measurements over the entire visible frustum every frame, which is fairly low, so both algorithms should have no trouble solving SLAM.

Eight sets of experiments are conducted on this setting, which includes varying the clutter rate, the probability of detection, the spatial motion and measurement noise and comparing with alternative versions of iSAM2 (using no preprocessing and with known data association).

For PHD filtering, 800 particles have been used, unless specifically noted. For iSAM2, the maximum likelihood estimator (using Mahalanobis distance) has been employed for data associ-

ation, with some preprocessing explained later. Both models are given the same model statistics concerning noises.

Data association

The iSAM2 algorithm expects correct data association. To associate known landmarks, a maximum likelihood estimator works well but cannot resolve features that are seen for the first time, since they may be real or just noise. Adding noise to the map can generate some inconsistencies later on when solving the estimates, so it must be avoided. Also, the inner structure of iSAM2 requires a very graph-specific Bayes tree, which would need to be rebuilt if a landmark were to be removed, wasting precious time.

Instead, a preprocessing step is developed, in which any new unassociated measurement generates a candidate landmark which does not yet participate in the SLAM optimization, but which can still be associated to new measurements and updated (using a simple average, since every one of these new measurements is assumed to have the same initial uncertainty). Once a predetermined number of consecutive sightings (3 in these experiments) of the landmark has been collected, it is deemed probable enough to be incorporated into the iSAM2 graph.

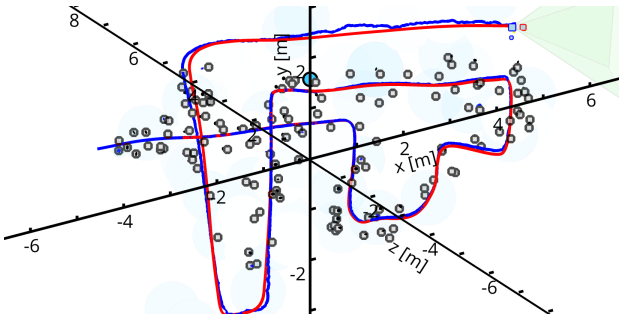
Note that to perform the maximum likelihood Mahalanobis association, it is necessary to obtain appropriate marginal covariance approximations on the map landmarks, which take more time than the updates themselves.

Fair start

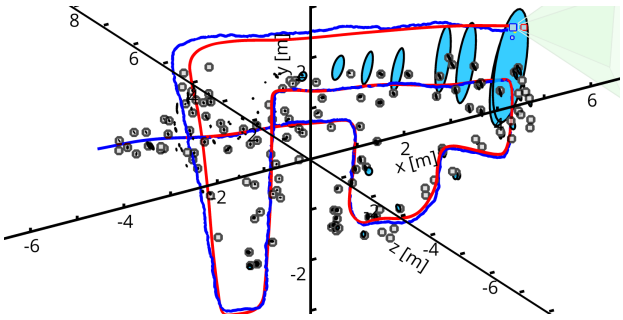
Since in the first few frames there is very little information, as the map is just forming, the vehicle can drift without notice, making the ensuing estimations highly volatile in an uninteresting way, since it is only an Euclidean free parameter. To avoid any random advantages in the comparisons, the first few frames are assumed perfectly localized, so a minimal and coherent initial map can form and guide further estimates.

2 Results

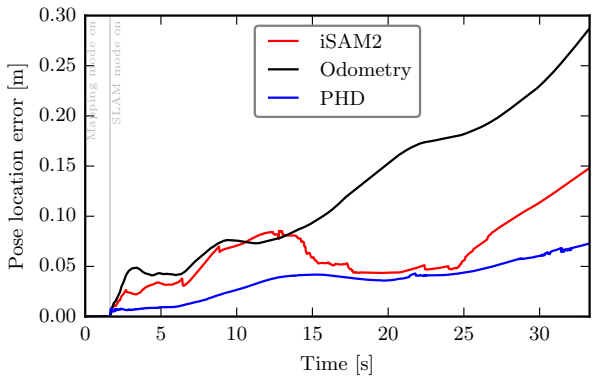
In the first (easy) simulation, the algorithms have no trouble finding an appropriate solution to the problem, as can be seen in figure (4.1). However the PHD solver provides a better estimate. It starts much better than iSAM2, until 15 [s] (which corresponds to the first loop closure) where iSAM2 is able to rapidly correct its pose and gives a similar error, although at the cost of an increasing map error. Later at 25 [s] (which corresponds to the second loop closure) its accumulated error is too large (some sections of the trajectory in this second loop have few measurements so the odometry error accumulates) and cannot close the loop correctly. The map error continues to increase, but the spatial component drops for the first time below that of the PHD filter, which can be explained by the fact that some of the additional landmarks that make up the cardinality error are in fact previously correct estimates that have drifted due to the incorrect loop closure.



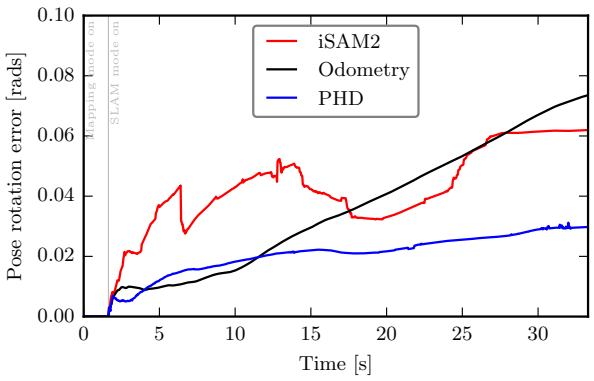
(a) PHD estimate. The red trajectory is the groundtruth and the blue one is the estimate.



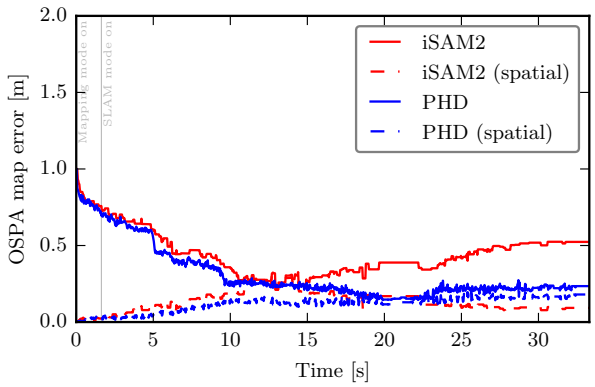
(b) iSAM2 estimate. White rectangles are landmarks and blue/black ellipses are the map estimate.



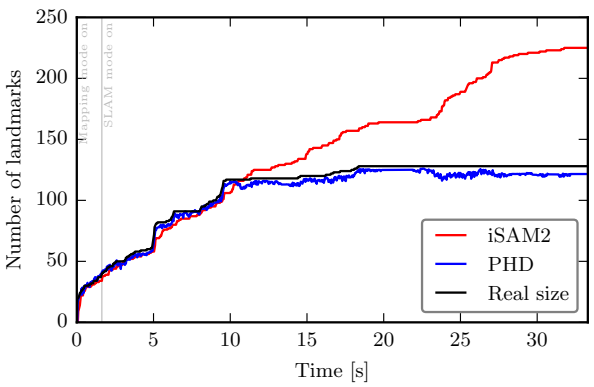
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$). 'Spatial' corresponds to the spatial component of OSPA and disregards the difference in cardinality.

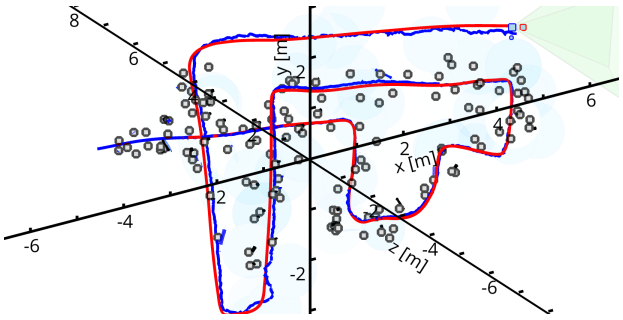


(f) Map size estimate.

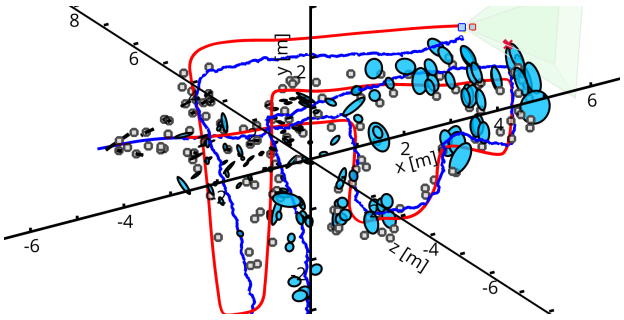
Figure 4.1: Comparison between the PHD filter and the iSAM2 solver in a case with easy statistics, ($\sigma_x^2 = 0.05\,[m^2], \sigma_\theta^2 = 0.002\,[rad^2], \sigma_{lx}^2 = 3\,[px^2], \sigma_{lz}^2 = 0.0002\,[m^2], N_c = 0.912, P_D = 0.9$).

As the spatial noise of the simulation is increased, the task becomes harder and data association becomes more fragile. Therefore, iSAM2 becomes unreliable. As shown on figure (4.2), it becomes worse than odometry and the map error is very high. The PHD filter degrades less and continues to be reasonable, as seen in the same figure.

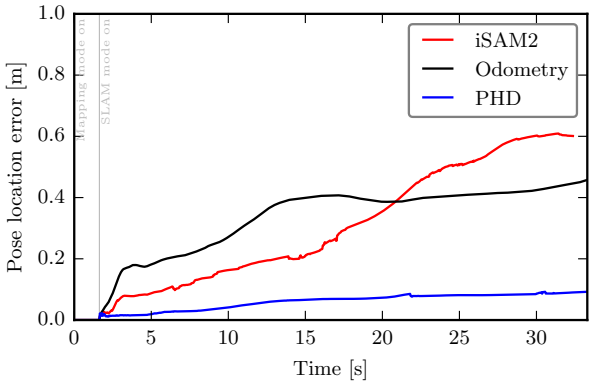
Alternatively, if the clutter rate goes up, as shown in figure (4.3), the clutter measurement can easily distort the distribution of the landmarks, since they can be influenced by false information.



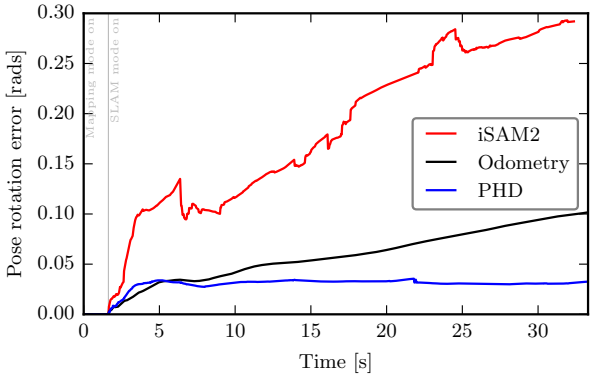
(a) PHD estimate.



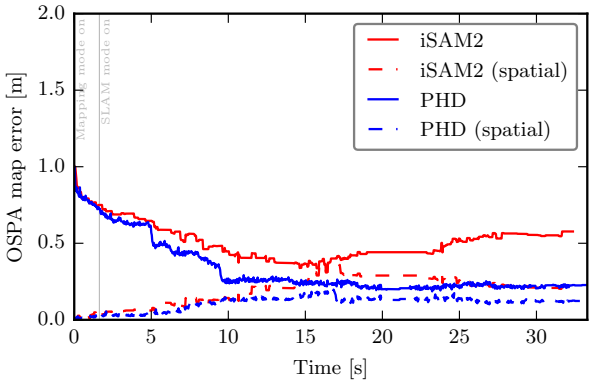
(b) iSAM2 estimate.



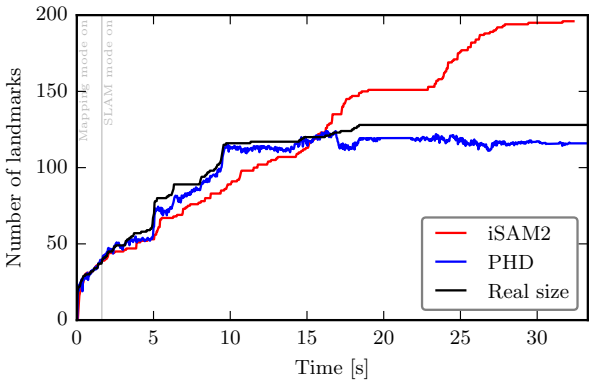
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).

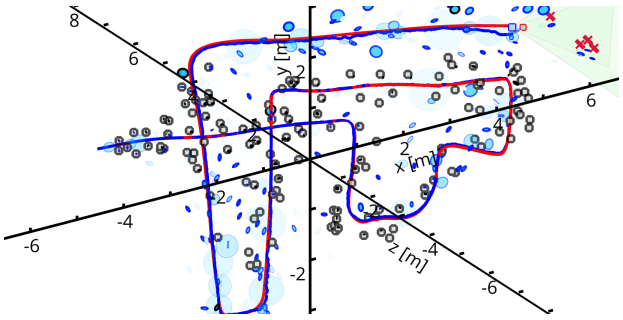


(f) Map size estimate.

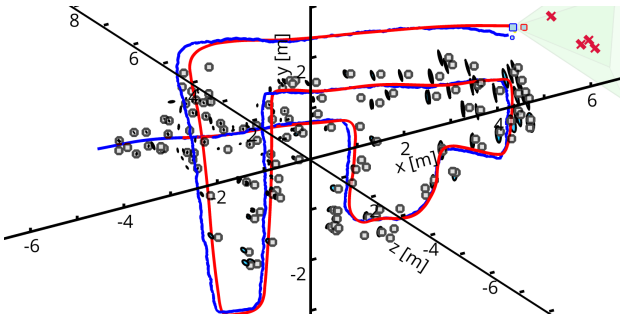
Figure 4.2: Comparison between the PHD filter and the iSAM2 solver with higher measurement and motion model noise, ($\sigma_x^2 = 0.2\,[m^2], \sigma_\theta^2 = 0.008\,[rad^2], \sigma_{lx}^2 = 8\,[px^2], \sigma_{lz}^2 = 0.003\,[m^2]$).

Both map errors are larger, but the PHD one remains to be better. The preprocessing step for iSAM2 is good at ignoring most of these clutter measurements, if they are sparse enough. Again, the PHD filter is less affected, since it doesn't need to make a decision on the association right away, and therefore can wait until more information is present to solve the problem robustly.

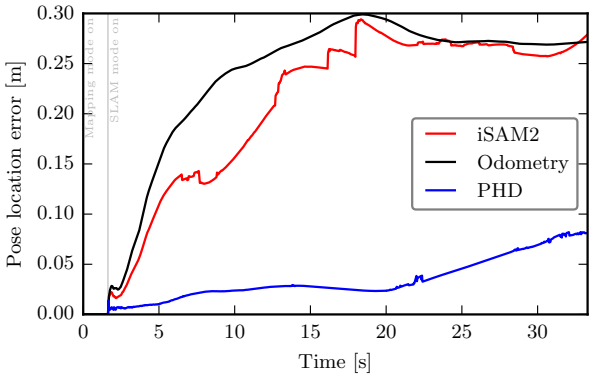
Another parameter that may vary is the probability of detection. This should be less of an issue for both algorithms, since it removes useful data instead of adding false information. This means the algorithm can't be as confident about their measurement corrections, but it does not generate a conflict to be resolved. In figure (4.4) the behaviour of the PHD continues to be reasonable, while iSAM2 seems to be incapable of using the sparse information provided by the landmarks to improve



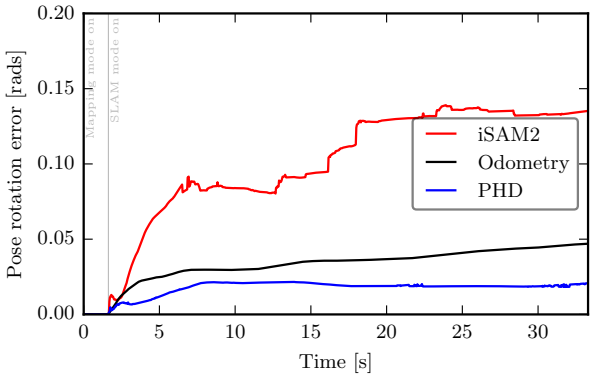
(a) PHD estimate.



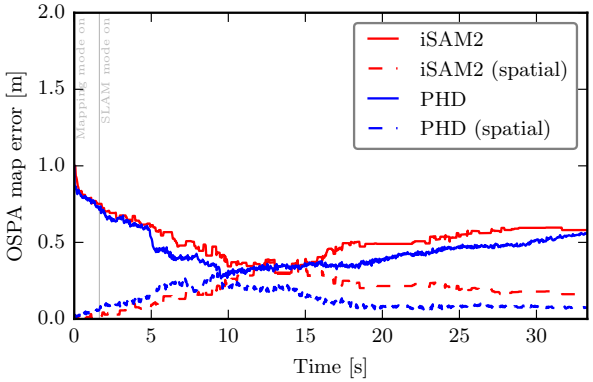
(b) iSAM2 estimate.



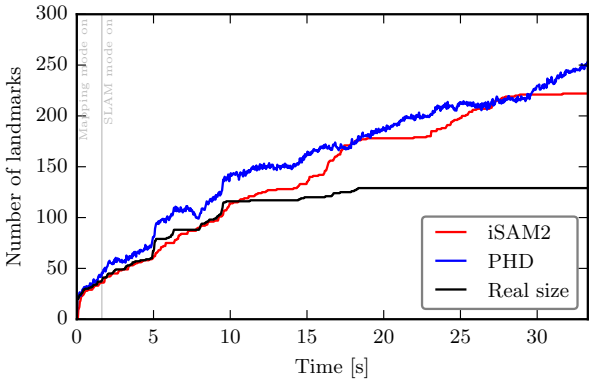
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).

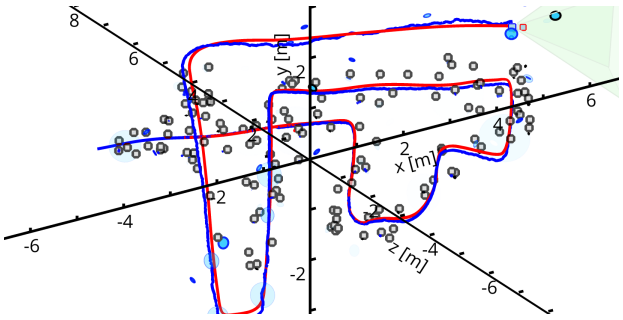


(f) Map size estimate.

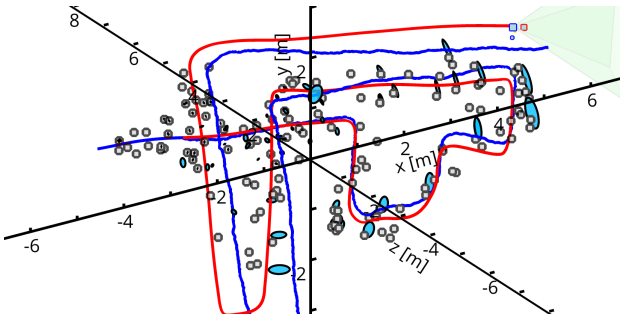
Figure 4.3: Comparison between the PHD filter and the iSAM2 solver in a cluttered environment, ($N_c = 9.12$).

the pose estimate. In both cases, the estimated map sizes are smaller than the real one since missing a landmark induces death in the PHD filter and removes it in the iSAM2 preprocessing step.

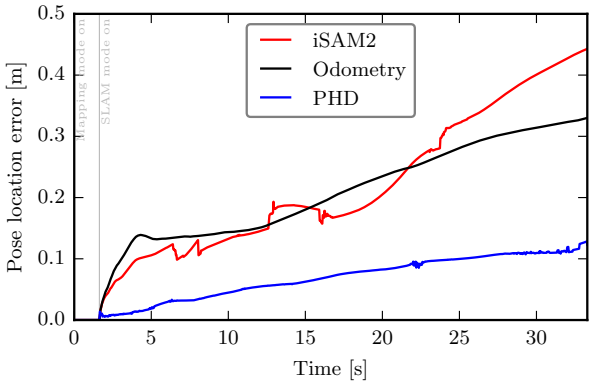
The preprocessing step in iSAM2 does not help in this scenario, because it can be too stringent and prematurely removes good candidates. So, it must be calibrated before each run. This is not satisfactorily in practice because clutter and low detection can occur at the same time, which require opposite calibration values, so this preprocessing stage can't accommodate both. A more complex one is required. Of course, both systems would require additional machinery to change the calibration if the parameters change in time.



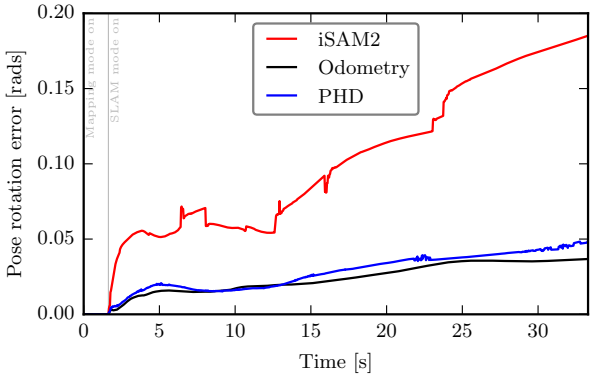
(a) PHD estimate.



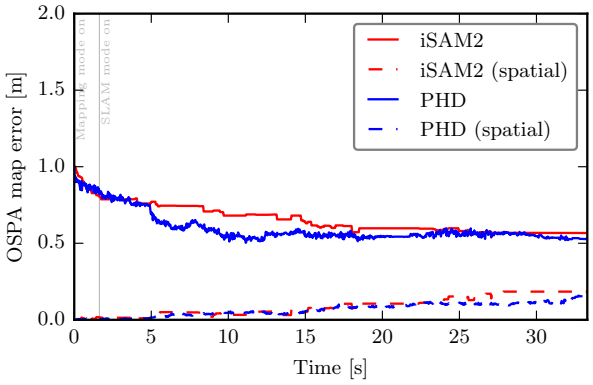
(b) iSAM2 estimate.



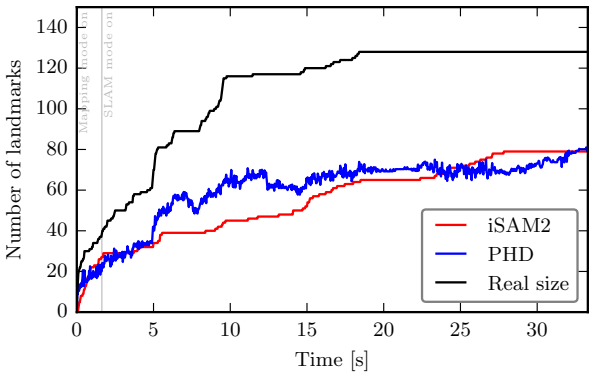
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).

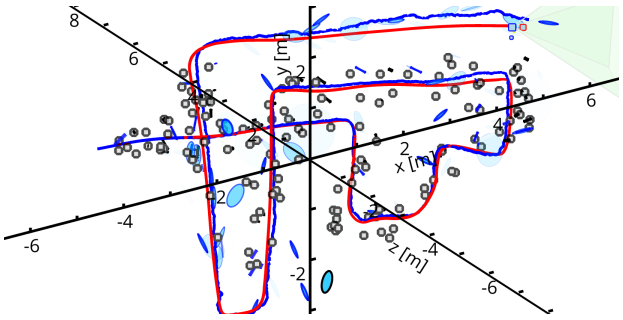


(f) Map size estimate.

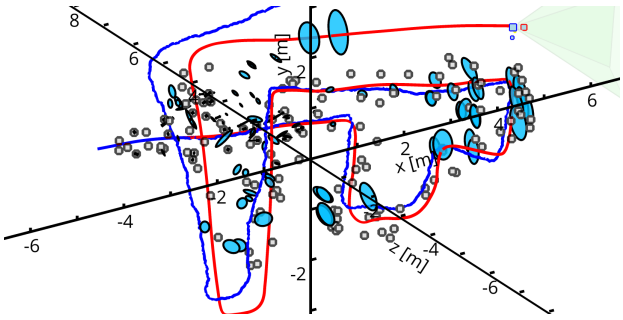
Figure 4.4: Comparison between the PHD filter and the iSAM2 solver when the probability of detection is low, ($P_D = 0.5$).

The different effects of the parameters can be compounded if more than one are relevant, such as in figure (4.5). Here both the clutter rate and the spatial noise are large making the iSAM2 algorithm diverge at the time of the second loop closure, while PHD SLAM remains reasonable.

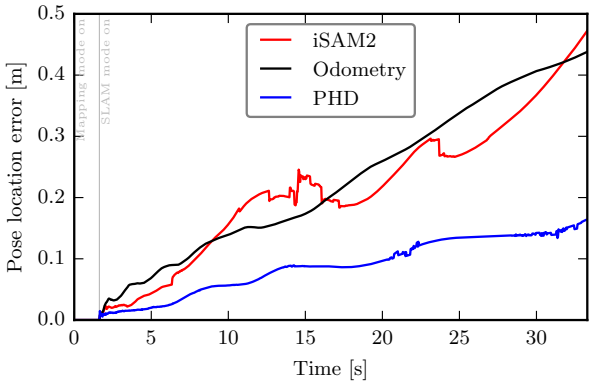
The preprocessing step is very important for iSAM2. The algorithm can deal with unprocessed inputs for a short time, but the map becomes increasingly cluttered, which finally makes the system diverge (when it tries to close a loop with spurious measurements from the past). Since the map size increases very fast, even during the time where the vehicles' pose has converged to its correct value, the processing time becomes too long even for controlled offline environments, which is demonstrated in figure (4.6), the experiment for which had to be prematurely terminated because



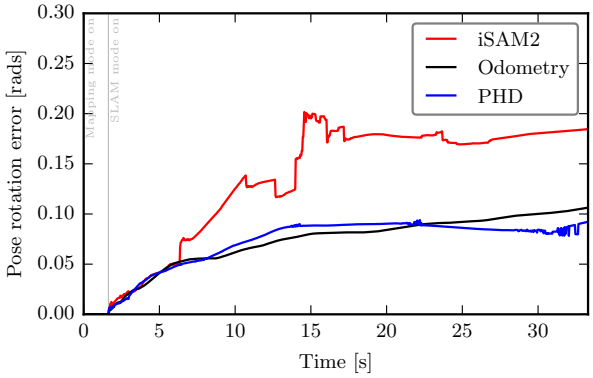
(a) PHD estimate.



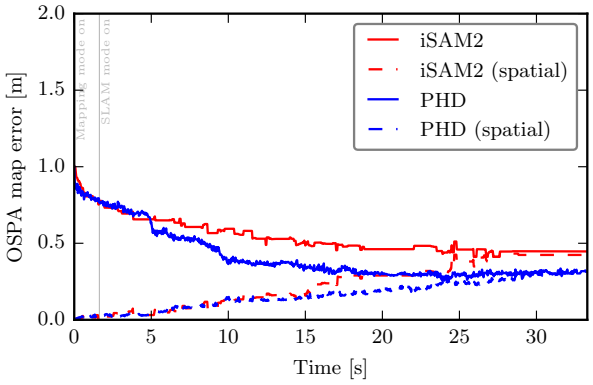
(b) iSAM2 estimate.



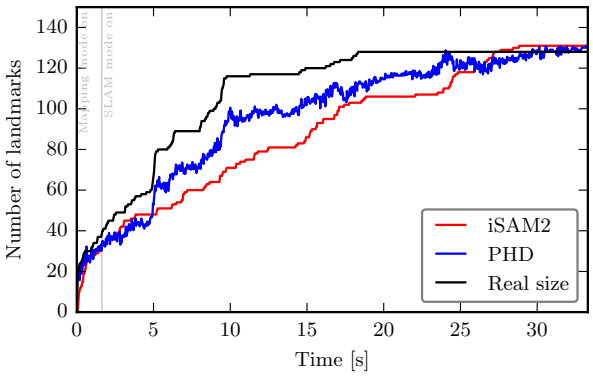
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).

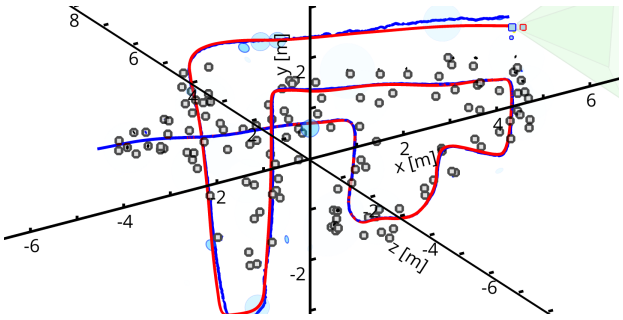


(f) Map size estimate.

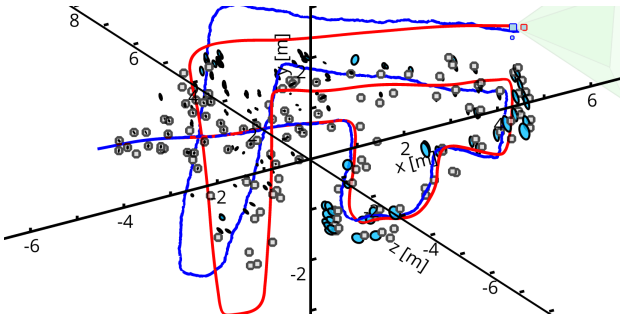
Figure 4.5: Comparison between the PHD filter and the iSAM2 solver when several kinds of errors are compounded, ($\sigma_x^2 = 0.1\,[m^2], \sigma_\theta^2 = 0.009\,[rad^2], \sigma_{lx}^2 = 5\,[px^2], \sigma_{lz}^2 = 0.02\,[m^2], N_c = 2.13, P_D = 0.7$).

every frame took more than thirty minutes to process. Divergence has been observed in other runs with smaller maps (which would happen in the first loop closure at 15 [s] in this scenario).

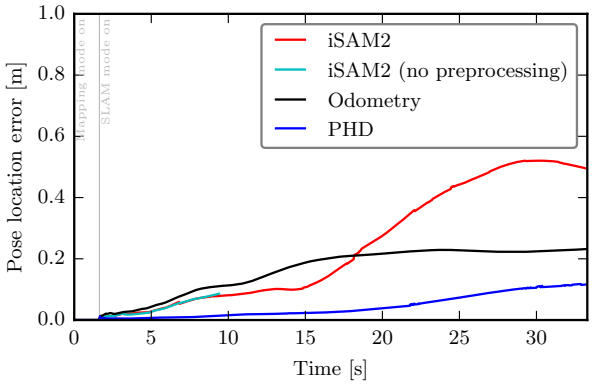
As was stated, the most important requirement for iSAM2 is good data association. If it were known, the algorithm should perform very well, since it is optimal in the Gaussian approximation. In figure (4.7), a comparison of the PHD filter against both a known association iSAM2 (KA-iSAM2) and a unknown data association iSAM2 is presented. Here it is confirmed that knowing the correct association gives the alternative algorithm enough information to show the best results. PHD SLAM manages to be very close to it and in the map error they are so similar that the green



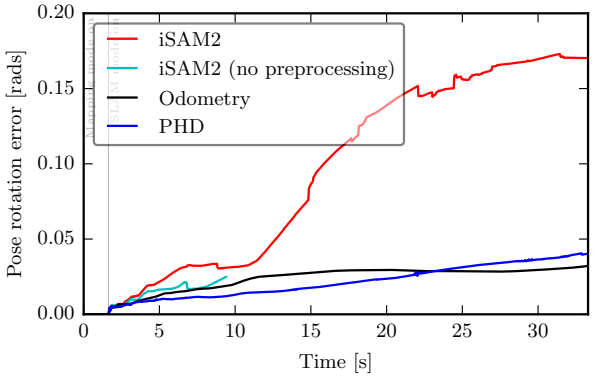
(a) PHD estimate.



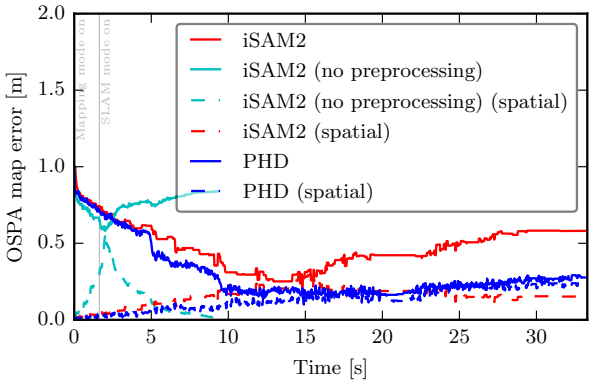
(b) iSAM2 estimate.



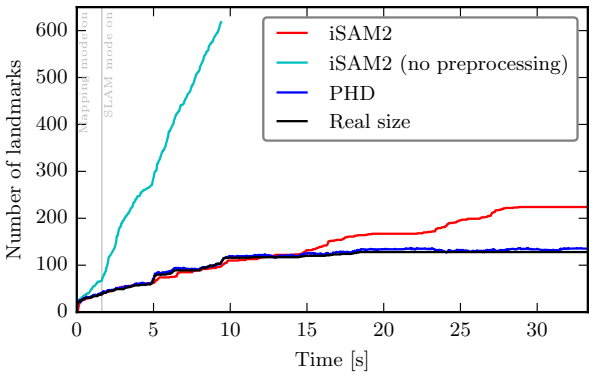
(c) Location error.



(d) Rotation error.



(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).



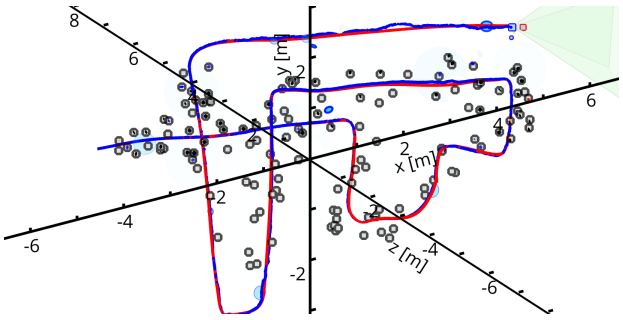
(f) Map size estimate.

Figure 4.6: Effect of preprocessing in the iSAM2 algorithm, (baseline configuration).

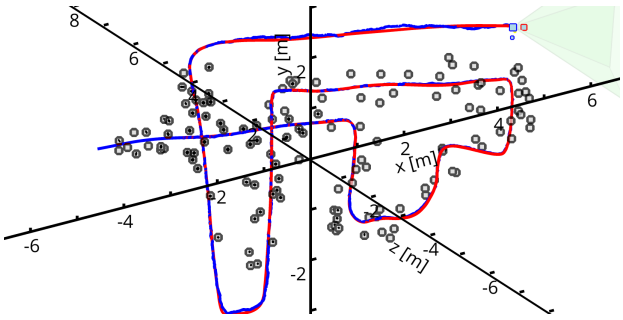
line (known iSAM2) is almost entirely occluded by the blue one (PHD SLAM).

Finally, both algorithms have been run in the context of real sensor data, as shown in figure (4.8). Here it can be seen that they struggle to converge, for the same reasons that were explained in the previous chapter. Nevertheless, both manage to generate visually reasonable maps. However, iSAM2 cannot use it to help the pose estimation because there's too much clutter. Note there's no map groundtruth available so the OSPA error cannot be plotted.

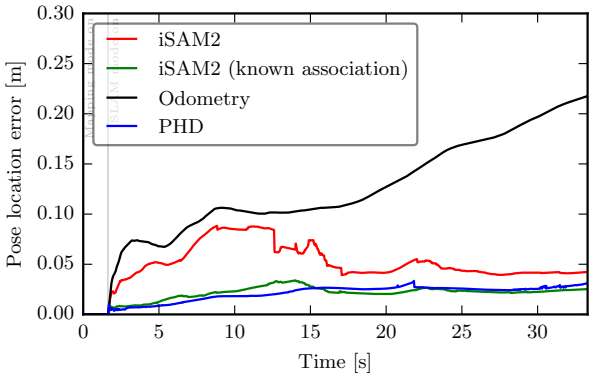
At 4 [s] the PHD filter has difficulty in finding the true SLAM solution because of the high number of measurements as explained in the previous chapter. To visualize how well the algorithm performs within short time frames, figure (4.9) shows "differential" locations errors, which have



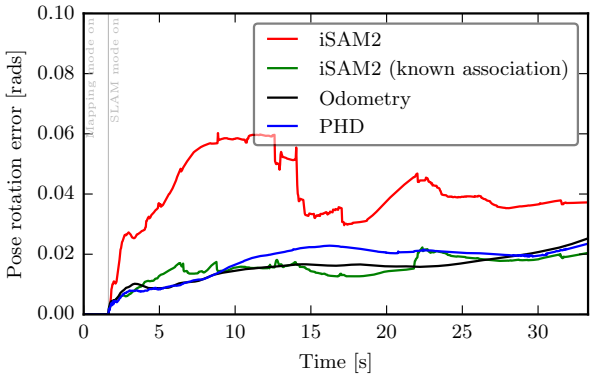
(a) PHD estimate.



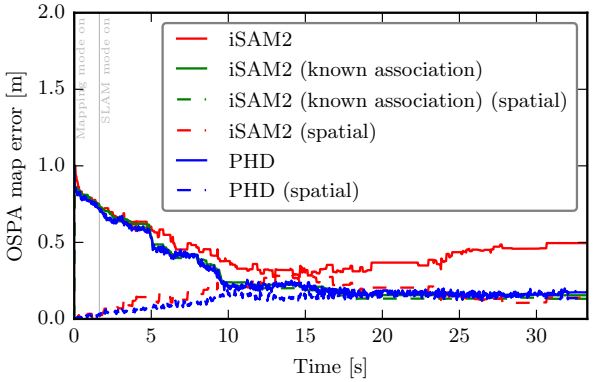
(b) iSAM2 estimate with known association.



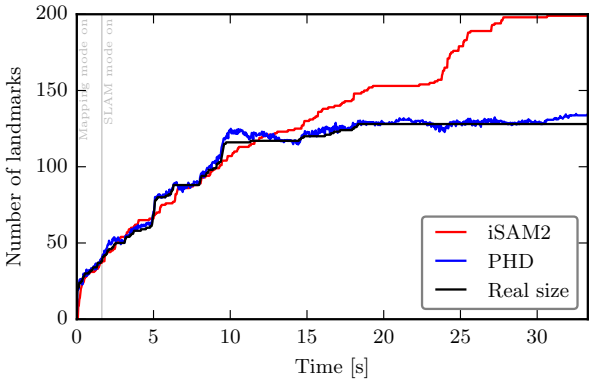
(c) Location error.



(d) Rotation error.



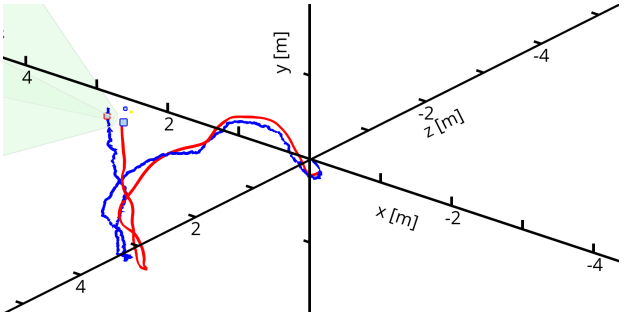
(e) Map error (OSPA, $c = 1\,[m]$ and $p = 1$).



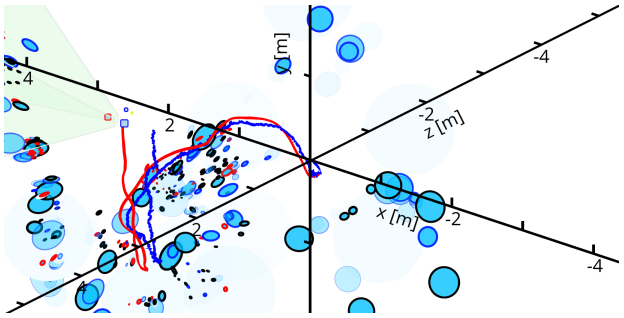
(f) Map size estimate.

Figure 4.7: Effect of perfectly known association in the GraphSLAM alternative, (baseline configuration).

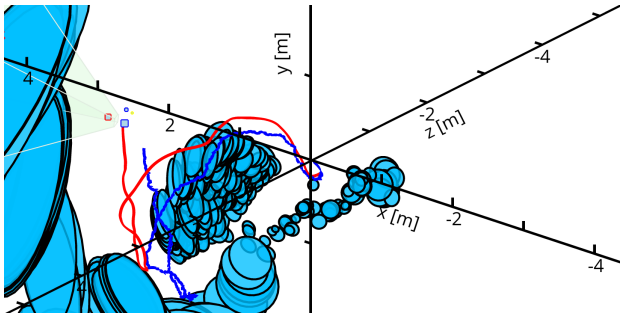
been computed by sliding the reference and comparison times as explained in the figure, i.e. these errors $e_{\Delta}(t)$ correspond to the error introduced in the short period of time between $t - 0.33[s]$ and t. Here it can be seen that for some time PHD SLAM can make improvements over odometry while iSAM2 cannot.



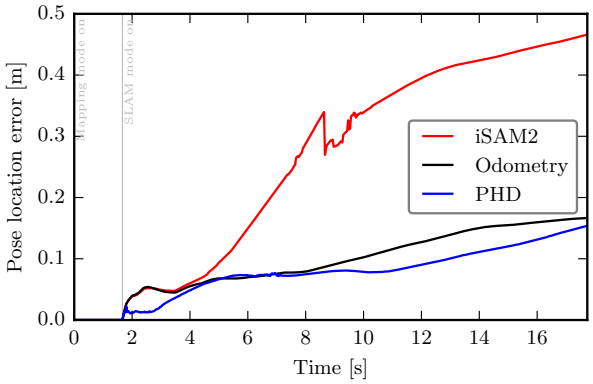
(a) Odometry dead reckon estimate.



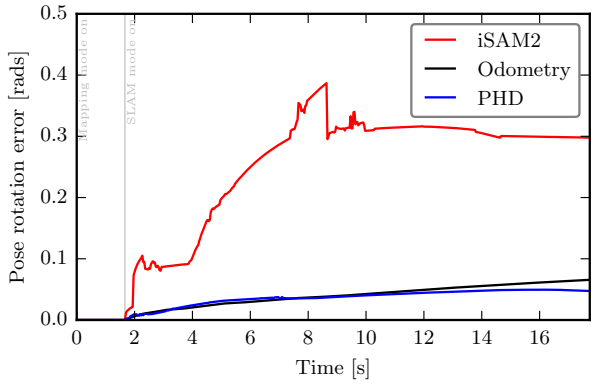
(b) PHD estimate.



(c) iSAM2 estimate.



(d) Location error.



(e) Rotation error.

Figure 4.8: Comparison between the PHD filter and iSAM2 for real data, ($\sigma_x^2 = 0.05\,[m^2], \sigma_\theta^2 = 0.002\,[rad^2], \sigma_{lx}^2 = 2\,[px^2], \sigma_{lz}^2 = 0.001\,[m^2], N_c = 4.864, P_D = 0.7$).

3 Discussion

These results show the potential that the random finite set approach can make to the SLAM problem, since it can obtain better results than GraphSLAM under realistic circumstances. It also shows some of its limitations. One big difference comes from the very nature of both algorithms: the PHD approach is a filtering method and iSAM2 is a batch method. They are solving different variations of the problem and therefore present different properties. The PHD filter processing is very localized and has trouble closing very large loops, since it does not have the required information at the required time to do so. iSAM2, being a batch processing technique can optimize for this case more easily. It can also resolve model misfits better since it can relinearise as needed, whilst the filter approach can't and is stuck with the approximation it did the one time it gets to fit the data. This

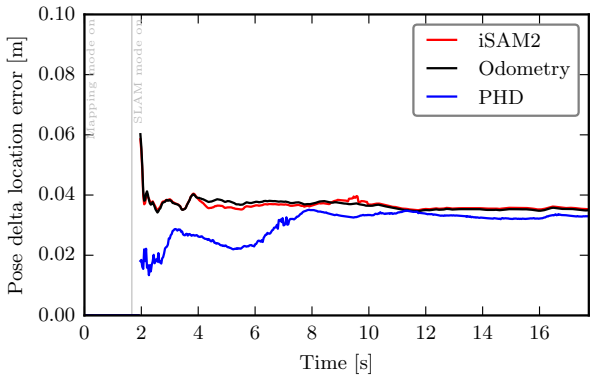


Figure 4.9: Local location error for PHD and iSAM2, where Δx is compared between the estimate and groundtruth, i.e. fixing the poses at t, how far have the poses at $t + \Delta t$ diverged ($\Delta t = 10$ frames).

means information is lost as time goes by and can't be recovered. A similar effect happens due to the discrete nature of particles. On the other hand, it requires data association, which is not needed in the PHD case and it is very sensitive to it. Additionally, PHD filtering finds the entire distribution for the problem, while iSAM2 only finds the mode and the marginal covariances.

This data association can be very hard to get right in difficult settings such as high clutter, high noise or low probability of detection. In contrast to the PHD filter, iSAM2 can easily diverge in these situations. This is an important advantage as this setting is realistic. Further, these settings may occur dynamically during a vehicle ride (e.g. when going through a tunnel) and the PHD filter may be easily adapted to such dynamic changes [73]. This makes the approach more robust in real conditions.

Due to the optimization nature of iSAM2, when false data association occurs, the algorithm may diverge to infinity, since the underlying problem can become ill-posed. This means that failure in this framework may halt the processor, which is unable to continue operating as there's no straightforward way to recover from a numerically invalid state*. This is unacceptable in any real scenario outside of a lab. Additional structure would be required to let the system continue to work under these circumstances. The PHD approach does not suffer this problem, since it can't be fooled by erroneous associations and would remain locally consistent even if the past becomes inconsistent with new measurements. This means that although global estimates may drift due to a failure, the system can continue to operate as well as possible afterwards. As shown in the perfect association comparisons, GraphSLAM could theoretically work better if it knew the exact associations but that is often not realistic.

In fact, iSAM2 exposes a fundamental property of the problem: if data association is perfectly known, there is a efficient and optimal solution, which relies on the inner sparse structure of the map. From this sparsity, linear algebra methods become faster, which allows them to become suitable for real-time operation. This inner sparsity would be interesting to exploit in other frameworks too. The data association route has a big problem though, which can't easily be addressed by vector SLAM methodologies: separating it from the main SLAM routine loses opportunities for clever

*As a analogy, a system that sums numbers can't continue operating after receiving an undefined value, i.e. given $F(X) = \sum_i x_i$, what should $F((1, 2, 5, 4, \text{Undefined}, 5, 6))$ be?

optimizations and throws away information that may be useful to guide the SLAM solver itself. This is an inefficient use of the resources which the PHD filter avoids by combining both stages into one global proposition. Nevertheless, note that although fast in principle, iSAM2 slows down considerably when extracting covariances, which are necessary to perform data association. As shown in the original iSAM2 article [43], conservative estimates for the covariance can be calculated more quickly, but this become less suitable when difficult conditions are present. The PHD filter, on the other hand, is relatively fast, but does not scale well on the number of particles necessary for localization. Indeed, as more particles are added, they contribute much less to the quality of the estimate since they do more and more redundant work as they become more densely distributed. More particles are required to search a finer grid for the optimal solution.

Note that the PHD filter can make good use of the system visibility model statistics, but that also means it needs to know them. As was pointed out before, there are methods to deal with unknown statistics, but they are still an additional burden to manage. In particular, the probability of detection implies a visibility model for the vehicle which can be difficult to use if landmarks are close to the visibility edges. This is an example of the difference in used information of the methods: while iSAM2 uses positive information about the detection of landmarks, the PHD filter uses both the same positive information and negative information about where the landmarks are not, i.e. a missed detection can change the estimate, unlike in the iSAM2 case. This can be both good and bad, since it can eliminate clutter but at the same time can't easily distinguish between empty space measurements and uninformative measurements, e.g. around the edges of the screen.

From these results and discussion it has been shown that PHD shows better overall results on the studied experiments, although both algorithms have their advantages. Nevertheless, even though PHD can be an improvement over the GraphSLAM alternative, it still has many rough edges that could benefit from iSAM2 notions. Therefore, extending it to use some of iSAM2 ideas is an enticing proposition.

In particular, the batch setting can be much more efficient given the sparse inner structure of the problem. Naturally, in the case of random finite sets, this sparsity is not as strong, since data associations haven't been resolved and therefore measurements can affect several landmark estimates. Also, being able to use information from the future can also help close long loops more efficiently, without requiring as many particles.

Chapter 5

Going Beyond Filtering

Filtering approaches have an intrinsic disadvantage over batch methods, since they can't make use of all available information as efficiently, since they only have the opportunity of one pass over the data. This difference has been well established by the results of batch methodologies like iSAM2.

Inspired by the results of both methodologies, RFS-SLAM and iSAM2, it is worthwile to explore the idea of extending the RFS framework to a batch method. To do so, the first step is to analyse previous efforts.

1 Extending Previous Work

As expressed in the literature review, in the RFS framework there are two alternatives that have developed around the idea of using all the information to make better SLAM estimates.

The first alternative, the forward-backward PHD smoother, proposed by Mahler et al. [75] and used by Clarke [98] in the SLAM context, is able to back-propagate future information to change the estimate of previous poses. As already indicated, practical implementations only change the weights and can't propose new hypothesis for the trajectory, which means they cannot "fix a closure problem" if the solution wasn't already in the pool.

A way to obtain better hypotheses could be to run the forward-backward propagation algorithm multiple times as shown on figure (5.1). For each iteration, the weights for the state distribution are corrected to include both past information (already included by filtering) and future information (from the forward-backward smoother, as shown at the top of the figure). Considering a vehicle trajectory of T steps, each round k would require $O(2 \times (T-k))$ calculations, for a total of $O(2 \times (T-k) \times T) = O(T^2)$ time complexity. The Forward-Backward filter would change weights from $w_k^i \tilde{P}(X_k|\mathcal{M}, \mathcal{Z}_{1:k})$ to $w_k^i \tilde{P}(X_k|\mathcal{M}, \mathcal{Z}_{1:T})$.

Note that the algorithm would linearise each update based on whatever state the particles happen to be at. Hence, the pose distribution estimates would be correct only around these linearisation points. So, if the backward pass changes the state estimates too much (e.g. by closing a loop), the final estimates may not be valid as they are too far away from the linearisation point and

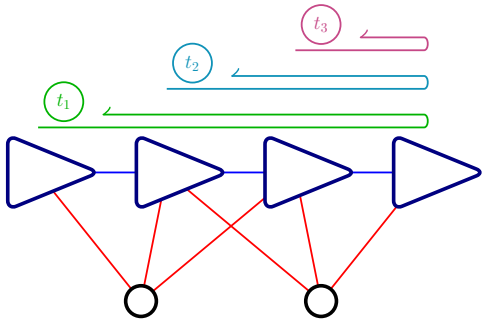


Figure 5.1: The forward-backward propagator could propose pose movements based on the future.

$P(X_k|\mathcal{M}, \mathcal{Z}_{1:T}) \not\approx P_{\text{linearised}}(\delta X_k|\mathcal{M}, \mathcal{Z}_{1:T})$. Also, it is observed that as time progresses, particles forget their history due to pruning. As particles are discarded, the representation of old pose distribution becomes worse. Since the earlier pose states have been around the longest, they have experienced more pruning than newer nodes, eventually having only one effective particle and therefore being unable to fix any loop closure due to particle depletion. Despite these details, it's an interesting path to pursue.

The second alternative comes from the mapping community, where Vu [77] uses Metropolis-Hastings Montecarlo (MHMC) [99] to sample from the complex probability distribution induced by the measurements. Since the estimated state distribution is complex, Vu uses an internal particle filter to evaluate the Metropolis-Hastings ratio, a technique usually called Particle Marginal Metropolis-Hastings Montecarlo (PMMHMC) [100].

Note however that the structure of this algorithm is framed to deal with highly dynamic independent targets, which differs from the typical SLAM structure. Indeed, in SLAM, landmarks are generally static in a global reference coordinate system. Then, if such a reference coordinate system (e.g. ground) is known, there is no real justification for the complex proposal moves in Vu's work. Nevertheless, it could prove useful when extending the SLAM problem to dynamic maps.

Also, in SLAM there's an additional constraint which does not occur in mapping: loop closure. Landmarks may be out of the field of view of the robot's sensors for quite a while and then be recaptured. In mapping, such a situation is easier to deal with. The landmark can be associated to different indices the first and second time it is seen, but in SLAM, recognizing they're the same feature is crucial to solving the localization part of the problem.

The Metropolis-Hastings algorithm can still be used to solve SLAM, factoring the probability distribution similarly to Vu's idea, but separating pose and map instead of the map and feature-to-measurement associations. Then, a proposal distribution needs to be defined over the trajectories, and the map can be solved using any of the mapping methods in the literature, e.g. the PHD filter or Vu's PMMHMC tracker. Using the trajectory proposer, the MHMC generic solver would create a new trajectory and accept it or reject it by evaluating its likelihood using the PHD filter. The structure of the MHMC sampler guarantees (under typical MHMC assumptions) that the collection of samples eventually converges to the real state distribution. Note that obtaining the likelihood in this model takes a lot of time, requiring the computation of the map RFS updates for the entire path, so a good trajectory proposal distribution is required.

A useful metric to measure the quality of a pose-to-pose link is the entropy of the particles

at that timestep $H = \sum w_i \log w_i$, where w_i are the particle weights, since when the data is not helpful, e.g. due to lack of measurements for a while, this entropy will tend to increase and when the system can localize itself, the entropy quickly drops. This metric can't really capture long-term discrepancies, since an incorrect localization will decrease the entropy too. Nevertheless, entropy should have increased before any erroneous localization takes place.

Errors can be local to a particular pose (and corrected afterwards) or be a drift that affects all future poses. So, a plausible proposal distribution could be selecting a random pose node (local error) or pose edge (drift) proportional to its entropy and change it following a Gaussian distribution.

This may work but, similarly to Vu's work, is computationally expensive so it is probably not suitable for real-time operation. There ought to be a more efficient alternative, as demonstrated by GraphSLAM approaches for the vector case.

2 Extending the RFS framework

There's nothing preventing the random finite set approach from being extended in the same manner than vector alternatives, although such extensions may be more complex than their vector counterparts. In particular, it is harder to draw a parallel between set-based states with linear algebra, since entries in a matrix correspond to pairwise relationships between elements, but in set-based SLAM, many elements may relate at a time (e.g. one pose and three close landmarks through two measurements). Working with the graph itself, like in iSAM2, can be extended more easily.

2.1 Probabilistic Graphical Models

Before delving into details of the RFS extension, it is necessary to go over probabilistic graphical models (PGM) more in depth [101]. In particular, the different types of networks, their nuances and the relevant algorithms for inference will be explained.

Structures

In a PGM problem, every variable to be studied will be represented by a node in a graph (e.g. 'Difficulty' and 'Ranking' in figure (5.2)). Among these variables, there are relationships of influence or cause, i.e. a change in one variable will directly induce a change in another (in this model, it is expected that 'Intelligence' directly influences 'Grade'). This information may be represented as directed edges between nodes (e.g. 'Intelligence' and 'Difficulty' are parents of 'Grade' in figure (5.2)). Note that in general most nodes can influence each other, albeit indirectly, i.e. only through an edge path. For this application, it does not make sense for a variable to influence itself (these models are static systems, not dynamical ones), so cycles are disallowed; otherwise, following the edge cycle would make a variable depend on itself (this would be like saying "the student approved the course" because "the student approved the course").

This kind of PGM is called Bayesian network and can be seen as a representation of conditional

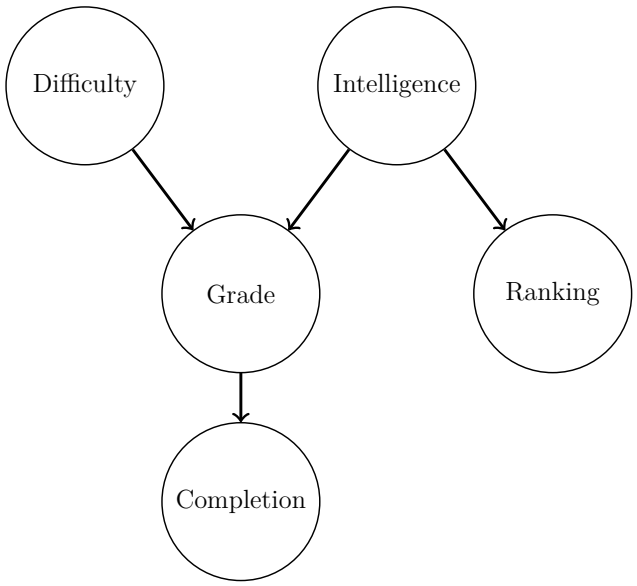


Figure 5.2: Example of a probabilistic graphical model representing the relationships between variables in the performance of an academic student in a given course.

independences in the problem. If certain node values were known, other ones become independent, e.g. in figure (5.2) if the grade is known, the difficulty of the course can't give any more information on the completion status of the student; all information flows through the grade. The rules of independence can be tricky, since there are several ways nodes can be connected. The elemental configurations are shown in figure (5.3); in the first three cases, careful consideration shows that *x* and *z* are dependent (through *y*), but they are independent in the last case. Conditioning on *y*, however, make the other two nodes become independent in the first three cases, but dependent in the last case [101]. In the first case, x affects z, which then affects y, so by fixing z, no matter what x is, y's distribution will not change. The second case is analogous. In the third case, y only depends directly on z so again, if z is fixed, y's distribution is also fixed. The fourth case is reversed because y does not depend on z, but the other way around. Marginalizing over z makes x and y independent because the do not depend on anything. But if z is fixed, both x and y's distributions are constrained by their inverse models and may become correlated, e.g. if $z = x + y$, given $z = 2$, then $x + y = 2 \Rightarrow y = 2 - x$, x and y are completely determined by one another.

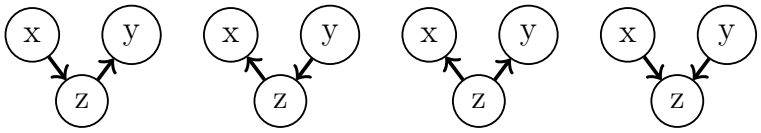


Figure 5.3: Elemental relationships between triplets found on any graphical model.

In probability theory, independence can be associated with a factorization of the joint distribution of variables. If two sets of variables are independent, their distribution will factor into separate terms for each set. So, the Bayesian network induces a joint probability distribution over the variables which can be factored out appropiately. In particular, using the chain rule for probabilities, a

factorization can be found with the formula

$$P(X) = \prod_{x_i \in X} P(x_i|\text{parents}(x_i)). \tag{5.1}$$

For nontrivial problems, this factorization and associated graph is not unique. For example, the chain rule could be applied in a different order to get a different result. Each time, some of the terms may be simplified using inner knowledge of the problem and therefore simplify the final factorization. This means that maybe a particular factorization doesn't expose *all* the independences in the problem. It is tempting to find the best factorization, which captures all the independences of the problem. As it turns out, in general this is not possible: there are some graphs which can't be reduced to a perfect factorization, i.e. there will be variables or sets of variables which are independent, but its not implied by the graph. On the other hand, if the graph does say they are independent, that is definitely so. Nevertheless, for many problems (among them, SLAM) there is a perfect graph and can be found efficiently.

Another kind of PGM is called a factor graph, which is directly derived from a distribution factorization. In this model there are two kinds of nodes: variables and factors (see figure (5.4)). Each factor node corresponds to a factor in the factorization, and each variable node is connected to every factor in which it appears. Converting Bayesian networks into factor graphs is straightforward by looking at the induced factorization. The SAM family works with this representation.

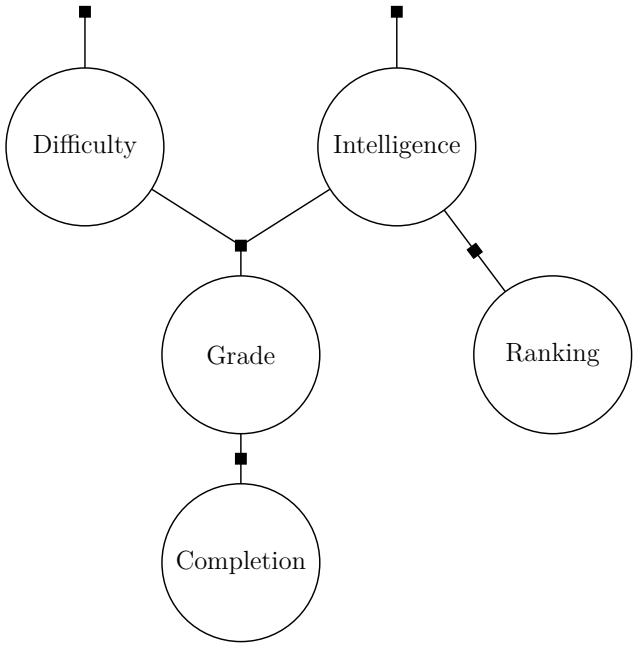


Figure 5.4: Example of a factor graph. Each circle corresponds to a variable node and each filled square is a factor node.

A third kind of model is called Markov Random Field (MRF), which extends Markov models. The Markov model assumes that a chain of events holds the property that "the future is independent of the past given the present". Similarly, MRF nodes are independent of each other given the nodes in between them, e.g. in figure (5.5), the a_i sub-graph is independent of the c_i sub-graph given b_0.

This is very similar to Bayesian networks, but in this case no direction is given for the influence. For some MRF, there is a compact factorization over cliques (fully connected subsets):

$$P(X) = \prod_{c \in \text{cliques}(X)} \psi(c). \tag{5.2}$$

A special case are Pairwise Markov Random Fields, whose factorization can be decomposed into unary and binary potentials,

$$P(X) = \prod_{x_i} \phi(x_i) \prod_{x_i, x_k} \psi(x_i, x_k) \tag{5.3}$$

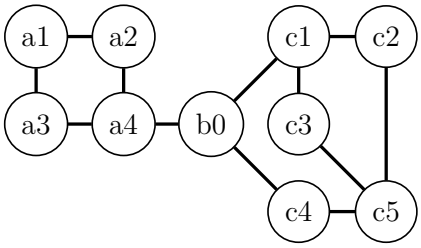


Figure 5.5: Example of a pairwise Markov random field. The edges are not directed so the influence between variables is symmetrical. Every node corresponds to a unary potential and every edge corresponds to a binary one.

The network for SLAM has been presented before, but is repeated for clarity in figure (5.6). The variables correspond to pose states, landmarks and measurements. The typical factorization is

$$P(X_{0:k}, M, Z_{1:k}) = P(x_0) \prod_{k=1}^{T} P(X_k | X_{k-1}, U_k) \prod_{k=1}^{M} P(Z_k | X_{i_k}, L_{j_k}). \tag{5.4}$$

Measurements may be considered parameters in the pose-landmark relationship instead of variables. Since they are not really interesting (they are never changed), their role as variable is not very relevant.

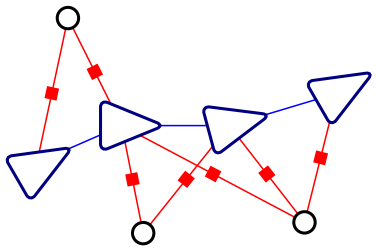


Figure 5.6: Factor graph representation of the SLAM problem. Every red square represents a factor.

Inference

The two most interesting questions to ask about these models are those of marginalization and maximization, i.e. what is the distribution of a single variable if all others are marginalized, and

what is the single most likely value configuration for all variables. These problems are referred to as inference. Both their solutions are similar (except the swap of integrals by maximizations), so only the marginalization will be detailed.

In principle, marginalization is just a very large integral, but it is usually intractable in practice. So to numerically solve it, it is required to exploit the structure of the particular problem at hand. A direct approach to accomplish this is to marginalize each node one-by-one until only the desired node is left. This procedure is called variable elimination. At each step, one node is removed from the graph, but the distribution of each of its neighbours needs adjusting to account for the marginal integral. Additional edges must be added between the neighbours. Although correct, this approach is not really suitable for SLAM, since the resulting integrals are still intractable.

Belief Propagation

There are many alternatives to variable elimination for performing inference. For this work, a suitable one is Belief Propagation (BP) [102, 103].

In this scheme, neighbour nodes will send messages to each other. The messages will consist of what each node thinks the distribution of the other node should be, given its own information. Then, each node can combine the messages it received using Bayes rule to estimate its own marginal distribution. Using this new improved distribution, the nodes can send messages again, which will be better informed, leading to better distributions and so forth. This way, information can flow throughout the graph, moving from node to node through messages. Eventually, each node's distribution should converge to the real marginal distribution.

Mathematically, this is expressed as follows. Let $m_{i\to k}$ be the messages from node x_i to node x_k. Then the marginal distribution for each node will be

$$P_i(x_i) \propto \phi(x_i) \prod_{x_k \in N(x_i)} m_{i\to k}, \tag{5.5}$$

where $\phi(x_i)$ is a prior on the variable (usually just a constant) and $N(x_i)$ is the set of neighbouring nodes.

Each message will be of the form

$$m_{i\to k}(x_k) = \int \psi(x_i, x_k)\phi(x_i) \prod_{x_h \in N(x_i)\setminus\{x_k\}} m_{h\to i}\, dx_i \tag{5.6}$$

$$m_{i\to k}(x_k) = \int \psi(x_i, x_k) P_{i|\overline{k}}(x_i)\, dx_i. \tag{5.7}$$

where $\psi(x_i, x_k)$ is the binary factor that relates both nodes, and an auxiliary distribution has been defined as

$$P_{i|\overline{k}}(x_i) = \phi(x_i) \prod_{x_h \in N(x_i)\setminus\{x_k\}} m_{h\to i}. \tag{5.8}$$

The message resembles a local marginalization over the sender node. $P_{i|\overline{k}}$ is almost the (current estimate for the) sender distribution, except one term is missing: the message from the receiver x_k.

This is important since otherwise the information from node x_k would flow to x_i and then back to x_k, artificially inflating its relevance because the node would think everyone is telling it to use it, as if it was external new information.

Finally, there are two more things to define: initial distributions and messages for each node and edge (usually just 1) and a message schedule protocol, i.e. in what order should the messages be sent. A simple protocol could be to send all messages at the same time and update the distributions accordingly.

Note that these equations correspond a MRF model, which is the simplest and therefore easiest to understand. Bayesian networks and factor graphs's BP equations are analogous.

As previously stated, these messages are expected to converge for the procedure to work. However the general conditions under which they converge are not fully understood. General convergence has only been proven for restricted cases, e.g. Gaussian distributions.

On one hand, belief propagation is exact for the simple case of a tree (no cycles in the graph). In such cases there exists a simple message schedule which converges in linear time: send the message from the leaves of the tree inwards and after reaching the root send them in reverse order, from the root to the leaves. There is no need to do any more propagation, the result will be exact after the inward-outward pass. This case corresponds to a vehicle that never sees a landmark twice, which is not realistic. On the other hand, if the graph contains loops, the algorithm is referred to as Loopy Belief Propagation (LBP) and its result is an approximation of the real inference. It may not converge and many factors influence its quality, e.g. schedule protocol, initialization, complexity of distribution and number of cycles. In practice, it works well and has been applied in many different areas (even SLAM [104]).

For the case of maximization, the messages become

$$m_{i\to k}(x_k) = \max_{x_i} \psi(x_i, x_k) P_{i|\overline{k}}(x_i), \tag{5.9}$$

and the final estimate

$$x_i^* = \arg\max_{x_i} \phi(x_i) \prod_{x_k \in N(x_i)} m_{i\to k}. \tag{5.10}$$

2.2 RFS Graph SLAM

The main idea proposed in this chapter follows from the flow of information through the graph. As shown in figure (5.8a), filtering information mainly flows through the trajectory in time and then from the trajectory to the map.

Let's say there's a time t^m such that the vehicle does not see any of the landmarks from times $t < t^m$ for a while, until time $t^c > t^m$, i.e. the observed landmark subsets $M_1 = M_{t=0:t^m-1}$ and $M_2 = M_{t=t^m:t^c-1}$ are disjoint as shown in figure (5.7). Then, the estimated trajectory $X_{0:T}$ can be split in two: one which is intertwined with measurements of the first set of landmarks, X_1 and another which is intertwined with measurements of the second set of landmarks, X_2.

The second trajectory segment is only related to the first one and the first landmarks by the odometry link $f(X_{t^m-1}, X_{t^m})$. This link has an associated uncertainty (odometry noise) and nothing

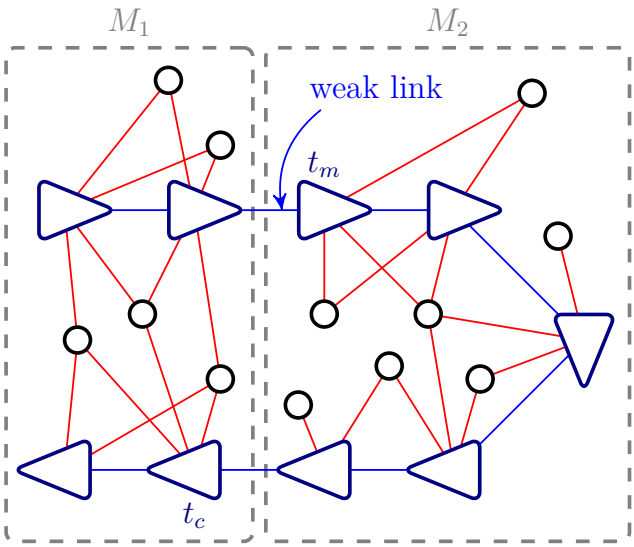


Figure 5.7: Effect of disjoint observable sub-graphs in inference quality.

that occurs later on the second segment can affect this uncertainty so no matter how accurate the estimation continues to be, this link will be uncertain. This is bad for any system that relies on particles, because the system will need to allocate some of its resources to represent the weak link uncertainty. As each particle represents an entire trajectory, i.e. the estimate is not local on each pose, but global on the entire trajectory, if there are N such weak links, the number of required particles to maintain a reasonable estimate is $O(k^N \cdot P)$ where P is the number of particles required to make the particle filter work under normal conditions and k is the number of particles required to model the link distribution alone.

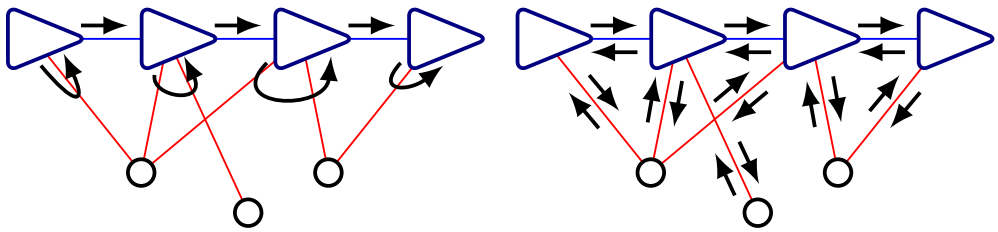
In practice what will happen is that many samples will be obtained from the weak link odometry model and they will be propagated through to the second segment. Some of them will correspond well with the new measurements, but some will not. As the effective number of particles decays due to the measurement model weight updates, the worst particles will be pruned, the best ones will be duplicated and so, the history of the particles will become more uniform. Since the weight update of the particles is not affected by odometry, the historic estimate of the weak link will be just one of the original proposals chosen at random, i.e. the system will incorrectly converge to a random value for every weak link (random from the odometry distribution). Later, when the vehicle sees the old landmarks again, the previous weak link will have only one particle so the system can't repair it.

Even if there is only one weak link in the entire trajectory, given a long enough second segment, the history will be lost from pruning. Memory has always been limited in filtering methods, but it's usually fine because the estimate is usually good for old poses. However, if there's a weak link as explained above, its distribution estimate does not get better for a long time and eventually it is forgotten, which is critically damaging. The number of particles to solve this problem becomes of the form $O(k^N \cdot r^{t^c - t^m})$, where r is the number of particles required to represent one time step of the trajectory.

In more extreme conditions, such as if the second segment has very few measurements, the trajectory estimate can also suffer from other kinds of losses due to approximations, linearisation errors, particle, odometry drift, etc., which compound the problem. Therefore, trying to fix these

kinds of difficult cases with particles is not scalable because the inference path through the trajectory is too long (exponential requirements). Instead, it is interesting to notice that information could flow through the landmarks and reach the future pose much faster (e.g. information could flow from the first pose in figure (5.8b) to the first landmark and then to the third pose directly, without going through the second pose), therefore being less prone to losses. Such pose-landmark-pose paths can be extrapolated to very long intermediate trajectories (replacing the second pose), which corresponds to the typical loop closure setting, and the path length difference becomes arbitrarily large.

Trying to take advantage of inference through these multiple paths is not straightforward, since every path will provide different beliefs about the nodes. For example, the path going through second pose in the previous example will consider the odometry reading between the second and third poses, while the pose-landmark-pose path will not. Naturally, the pose-landmark-pose path can be extended to include the second pose and then, in theory, every path should say the same thing, since they consider the same information; but in practice, due to numerical inaccuracies, model misfits, linearisations and other approximations, they will not. Each path will consider different approximations and thus will capture different relationships more strongly in their beliefs. The traditional filtering path will be more appropriate for correctly weighting odometry and measurements locally whereas the pose-landmark-pose paths described above will be able to model long loop closures. A principled way to mix the beliefs of both update strategies is necessary. This is where graphical models come into play, since there's already a well established algorithm to propagate information in such a way in a graph known as loopy belief propagation. As already noted, its convergence is not guaranteed, but in practice it works well.



(a) Filtering. (b) Loopy Belief Propagation.

Figure 5.8: Information flow in the variations of the SLAM problem.

Since there is no data association, the graph can't be created over individual landmark nodes. A simple way to solve this is to use the entire map as one node, as is shown figure (5.9). As per the definition of loopy belief propagation, nodes will message each other trying to influence their neighbours' beliefs.

To make the computation tractable, each pose node will be approximated by a normal distribution and the map will be assumed a Poisson RFS*.

*Alternatively, poses could be approximated using particle filters and the map could be modelled using a CPHD or Bernoulli RFS.

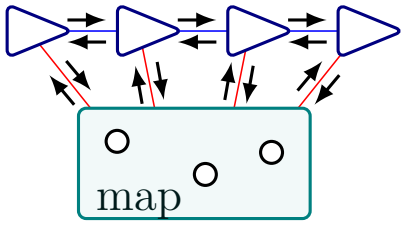


Figure 5.9: RFS Loopy Belief Propagation formulation.

On one hand, beliefs over poses become

$$P(X_k) \propto P(X_k, Z) \propto \phi_k(X_k) m_{k-1\to k}(X_k) m_{k+1\to k}(X_k) m_{\text{map}\to k}(X_k) \tag{5.11}$$

$$\sim \mathcal{N}(\mu_k, \Sigma_k) = \mathcal{N}^{-1}(\eta_k, \Lambda_k), \tag{5.12}$$

where μ_k corresponds to the mean of the pose distribution, Σ_k to its covariance, η_k to its information vector and Λ_k to its information matrix[†]. This corresponds to the prior and three messages: one from the previous pose, one from the future pose and one from the map, all of which are assumed Gaussian.

There are two special cases: the first and last poses, which do not have a previous or future pose respectively. For the last pose, the future message will be omitted (completely non-informative, $1_{\mathbb{R}^3}$) and for the first pose, the previous message will be a narrow Dirac delta around the origin, to fix the trajectory to a reference. Priors can be used to fuse external information (e.g. pose constraints), but in the plain SLAM problem will be set to non-informative $1_{\mathbb{R}^3}$ distributions. Finally, it is useful to see that in the Gaussian case, these multiplications correspond to sums in the canonical representation,

$$\left(\phi(X) \prod_{i \in N(X)} m_{i\to X}(X)\right) \sim \mathcal{N}^{-1}(X;\ \eta_\phi + \sum_i \eta_{i\to X},\ \Lambda_\phi + \sum_i \Lambda_{i\to X}). \tag{5.13}$$

One the other hand, the belief over the map becomes

$$P(M, Z) \propto \phi_{\text{map}}(M) \prod_{k=1}^{T} m_{k\to\text{map}}(M), \tag{5.14}$$

where there is a prior factor and a message from each pose in the trajectory. Note that since the objective is to calculate the maximum likelihood estimator, it does not matter if the map is conditioned on the measurements (i.e. $P(M|Z)$) or they are aggregated into a joint probability (i.e. $P(M, Z)$).

These expressions calculate the statistic of each variable given the messages between them. There are three kinds of messages: between poses, from poses to the map and from the map to the different poses. The first ones are the easiest, since both nodes are Gaussian-distributed and are related by a Gaussian constraint, namely the odometry.

[†]As previously explained for the GraphSLAM methods, η_k and Λ_k are an equivalent representation for Gaussian distributions better suited for fusion. Both representations are related by $\exp(-\frac{1}{2}(\mu_k - x)^T \Sigma_k^- 1(\mu_k - x)) \propto \exp(\frac{1}{2} x^T \Lambda_k x + x^T \eta_k)$

Let $P(X_{k+1}|X_k) = \exp(\frac{1}{2}||F_k\delta X_k + G_k\delta X_{k+1} + a_k||^2_{Q_k})$ be the linearised odometry model; then the message becomes [104]

$$
\begin{aligned}
m_{k\to k+1}(X_{k+1}) &= \max_{X_k} \psi(X_k, X_{k+1}) P_{k|k+1}(X_k) && (5.15)\\
&\sim \mathcal{N}^{-1}(X_{k+1}; \\
&\qquad \eta_f^{k+1} \quad - \Lambda_f^{k+1,k}(\Lambda_f^{k,k} + \Lambda_{k|\overline{k+1}})^{-1}(\eta_f^k + \eta_{k|\overline{k+1}}), && (5.16)\\
&\qquad \Lambda_f^{k+1,k+1} \quad - \Lambda_f^{k+1,k}(\Lambda_f^{k,k} + \Lambda_{k|\overline{k+1}})^{-1}\Lambda_f^{k,k+1}),
\end{aligned}
$$

where η_f and Λ_f are defined as

$$\eta_f = \begin{pmatrix} \eta_f^k \\ \eta_f^{k+1} \end{pmatrix} = \begin{pmatrix} F_k \\ G_k \end{pmatrix} Q_k^{-1} a_k \tag{5.17}$$

$$\Lambda_f = \begin{pmatrix} \Lambda_f^{k,k} & \Lambda_f^{k,k+1} \\ \Lambda_f^{k+1,k} & \Lambda_f^{k+1,k+1} \end{pmatrix} = \begin{pmatrix} F_k \\ G_k \end{pmatrix} Q_k^{-1} \begin{pmatrix} F_k \\ G_k \end{pmatrix}^T. \tag{5.18}$$

The reverse message, $m_{k+1\to k}$ is analogous, reversing the indices. Note the use of $P_{k|k+1}(X_k)$ and its Gaussian parameters $\Lambda_{k|\overline{k+1}}$ and $\eta_{k|\overline{k+1}}$, defined in equation (5.8).

The other two messages are more nuanced, since they must deal with the RFS map. It's important to remember that the factors between poses and the map are highly nonlinear, but that they can be managed in the same way the PHD filter was proposed.

The message from the poses to the map follows the form

$$
\begin{aligned}
m_{k\to\text{map}}(M) &= \max_{X_k} \psi(X_k, M) P_{k|\overline{\text{map}}}(X_k) && (5.19)\\
&= \max_{X_k} P(Z_k|X_k, M) P_{k|\overline{\text{map}}}(X_k). && (5.20)
\end{aligned}
$$

Note that the term $P_{k|\overline{\text{map}}}(X_k)$ has been introduced, which is defined analogously to the previous $P_{k|\overline{k+1}}(X_k)$ one, i.e. it is the distribution on X_k disregarding the message from the map.

This maximization is difficult to evaluate in practice. If the pose covariance is small, the expression is dominated by the second term[‡] and the maximum will be attained close to the maximum for the second term,

$$\arg\max_{X_k} P(Z_k|X_k, M) P_{k|\overline{\text{map}}}(X_k) \approx \arg\max_{X_k} P_{k|\overline{\text{map}}}(X_k), \tag{5.21}$$

so that the original optimization can be reduced to

$$
\begin{aligned}
m_{k\to\text{map}}(M) &= \max_{X_k} P(Z_k|X_k, M) P_{k|\overline{\text{map}}}(X_k) && (5.22)\\
&\approx P(Z_k|\mu_{k|\overline{\text{map}}}, M) P_{k|\overline{\text{map}}}(\mu_{k|\overline{\text{map}}}) && (5.23)\\
&\propto P(Z_k|\mu_{k|\overline{\text{map}}}, M). && (5.24)
\end{aligned}
$$

[‡]The extreme case is to approximate the second term as a Dirac distribution so the whole distribution is zero outside this maximum. Note that if the current values are close enough to the optimum, the second term corresponds to the fusion of odometry readings from the correct past and future poses, which should be the correct pose in expectation. Also, although this is an approximation, it shows positive practical results later in the chapter.

If this is not the case, the system can still be represented as particles, and the result obtained by comparing the value at different points N_k,

$$m_{k\to\text{map}}(M) = \max_{x_i \in N_k} \{P(Z_k|x_i, M)P(x_i)\} \tag{5.25}$$

where N_k are sampled from the pose distribution or are around its mode.

From these messages, the belief over the map becomes

$$P(M, Z) \propto \phi_{\text{map}}(M) \prod_{k=1}^{T} P(Z_k|\mu_{k|\overline{\text{map}}}, M). \tag{5.26}$$

This is precisely the mapping recursion with known localization[§], which can be calculated with the PHD filter update equations. Note that the poses are not exactly in the expected location (they are not μ_k), but in the expected location disregarding the direct map input, $\mu_{k|\overline{\text{map}}}$. Note that even if the map-to-pose message is removed, the map can still affect $\mu_{k|\overline{\text{map}}}$ through the messages of the neighbourhood poses, since their own distributions do consider the message from the map.

Finally the message from the map to the poses follows the form

$$m_{\text{map}\to k}(X_k) = \max_M \psi(X_k, M) P_{\text{map}|\overline{k}}(M) \tag{5.27}$$

$$= \max_M P(Z_k|X_k, M) P_{\text{map}|\overline{k}}(M). \tag{5.28}$$

Here, the $P_{\text{map}|\overline{k}}(M)$ distribution corresponds to the message to the pose node k without considering the previous pose-to-map message from the same pose k, i.e. to obtain these messages, the map must be computed skipping the pose k.

Similarly to the previous pose cases, the distribution is not tractable and can be approximated by a fixed maximum in the second component, obtaining the message

$$m_{\text{map}\to k}(X_k) \propto P(Z_k|X_k, M_{\text{map}|\overline{k}}), \tag{5.29}$$

where $M_{\text{map}|\overline{k}}$ is the maximum likelihood estimate for the map distribution. However, as observed in Vo's work [47], RFS distributions do not have a well-defined maximum likelihood estimate, since it considers elements with different cardinalities. Indeed, by having different units of measurement, the likelihood of elements of different cardinalities can be skewed to make the maximum likelihood estimate change arbitrarily. For example, let Z be RFS distribution following

$$P(Z = \phi) = 0.2 \tag{5.30}$$

$$P(Z = \{x\}) = \frac{0.8}{2} = 0.4 \; \forall x \in [0, 2][m]. \tag{5.31}$$

It gives a 0.2 chance of being empty, otherwise it has one feature uniformly selected in the [0, 2] range (measured in metres). Here the obvious maximum likelihood is attained at any point in the [0, 2] range. But if the units are changed to centimetres,

$$P(Z = \phi) = 0.2 \tag{5.32}$$

$$P(Z = \{x\}) = \frac{0.8}{200} = 0.004 \; \forall x \in [0, 200][cm], \tag{5.33}$$

[§]Note that there are no prediction terms because there is only one static map. This will be explored further later.

the maximum changes to the empty set.

However, there are still ways to find equivalent concepts in RFS. In particular, the Joint Multi-Object estimator $\hat{\mathcal{M}}^{JoM}$ or the Marginal Multi-Object estimator $\hat{\mathcal{M}}^{MaM}$ may be used instead, which are defined as follows,

$$\hat{\mathcal{M}}^{JoM} = \arg\sup_{\mathcal{M}} \Big(P(\mathcal{M}) \frac{s^{|\mathcal{M}|}}{|\mathcal{M}|!} \Big), \tag{5.34}$$

$$\hat{m} = \arg\sup_{m} P_{|\mathcal{M}|}(m) \tag{5.35}$$

$$\hat{\mathcal{M}}^{MaM} = \arg \sup_{\mathcal{M}:|\mathcal{M}|=\hat{m}} P(\mathcal{M}). \tag{5.36}$$

In the first expression, s is an arbitrary constant with units of volume in the feature space.

The map-to-pose message is a non-linear function for any given pose, but it can be locally approximated with a Gaussian distribution using numerical methods, e.g. by gradient descent or sampling methods. Although the RFS map requires complex analysis, after $\hat{\mathcal{M}}$ has been found, the optimization over X_k is defined in a vector space, so gradient descent is easily defined.

Note that to calculate the reduced $P_{k|i}$ distributions, pose nodes (Gaussian) can make use of the canonical representation, $\mathcal{N}^{-1}(\eta_k - \eta_i, \Lambda_k - \Lambda_i)$ and for the map node, it can be calculated by just skipping the appropiate PHD update step.

This defines the update equations for the LBP method. The only thing left to do is to choose a good initialization. For this, another approach can be used first, e.g. PHD filtering. This induces an estimate for each node, which is used to send the first messages.

Dynamic maps

So far, a static map has been assumed, i.e. which does not depend on time. This lines up nicely with the typical GraphSLAM setting, but is different from the PHD SLAM filter. This filter assumes a static map (i.e. landmarks do not move around), but introduces statistics (birth and death rates) for numerical simplification. Instead of using the complete map all the time, it includes partial maps $\mathcal{M}_k$ which only contain the region that the vehicle has seen so far. This framework is more stable and robust to clutter and misdetection, e.g. if the filter has made a mistake and believes that a particular landmark exists when in fact it does not, instead of failing later on because it uses an incorrect assumption, it can kill that region of the PHD density following the death rate instead. Additionally, the SLAM system could also be used in actually dynamic maps, i.e. where landmarks can also move.

The Loopy RFS system can be extended to consider these map dynamics, be it because the landmarks actually move around or for obtaining the same robustness as in the PHD SLAM framework. The resulting graph is shown in figure (5.10). The map has been decomposed into T maps in time, each one connected to the previous one and the next one, as well as to the corresponding pose. Note that if every map-pose pair is merged into a super-node, a linear graph is formed, corresponding to the filtering case, as shown on figure (5.11).

In this dynamic setting, the messages would remain the same for inter-pose communication

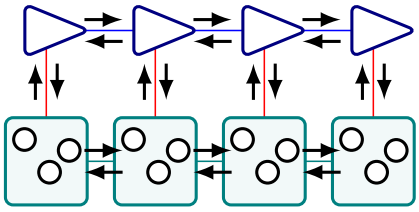


Figure 5.10: Dynamic settings for the Loopy RFS.

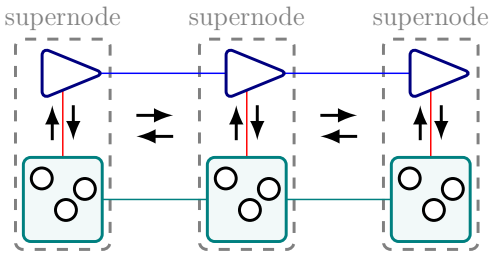


Figure 5.11: Supernodes reduce the problem to the filtering case.

and pose-map communication, since nothing has really changed in those connections. There are however two differences: an additional map-to-map message must be defined, and the pose-to-map messages are with a specific map in time. A general forward map-to-map message can be deduced from the filtering equations,

$$m_{map_k \to map_{k+1}}(M_{k+1}) = \max_{M_k} \psi(M_k, M_{k+1}) P_{\text{map}_k|\overline{\text{map}}_{k+1}}(M_k) \tag{5.37}$$

$$\approx P(M_{k+1}|M_k^*), \tag{5.38}$$

where $P_{\text{map}_k|\overline{\text{map}}_{k+1}}(M_k)$ is the reduced distribution of M_k without information from M_{k+1}. Approximating the map as a Dirac distribution, the second line's M_k^* is the JoM or MaM estimator of M_k, where this distribution has been extracted from the past and the pose (but not the future). The general backward map-to-map is similar, except that the future time is fixed instead,

$$m_{map_{k+1} \to map_k}(M_k) = \max_{M_{k+1}} \psi(M_k, M_k) P_{\text{map}_{k+1}|\overline{\text{map}_k}}(M_{k+1}) \tag{5.39}$$

$$\approx P(M_k|M_{k+1}^*). \tag{5.40}$$

This equation may be harder, since it requires an inverse model for the update distribution. Also, the M_{k+1}^* estimator must use only future information. This may be accomplished using a technique like Forward-Backward propagation.

Map-to-pose messages would only consider the map at the given time, instead of the whole map,

$$m_{\text{map} \to k}(X_k) \propto P(Z_k|X_k, M_{\text{map}_k|\overline{k}}), \tag{5.41}$$

where $P(Z_k|X_k, M_{\text{map}_k|\overline{k}})$ disregards the measurements Z_k at time k. Pose-to-map messages would only influence the map at the given time as in filtering,

$$m_{k \to \text{map}}(M) \propto P(Z_k|\mu_{k|\overline{\text{map}_k}}, M_k). \tag{5.42}$$

For practical purposes, the most complex message is the backward map-to-map, which usually must be computed using particles.

Note that the structure is very similar to parallel Forward-Backward filtering. However, the messages are local and "independent", so for example information can flow at different rates through the pose lane and the map lane as shown in figure (5.12), and they are passed around as many times as required for convergence, with the freedom of approximating distributions and changing linearisation points along the way.

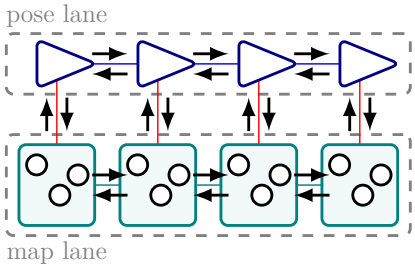


Figure 5.12: Parallel lanes allow for information to flow faster.

In the case of static maps an alternative is to, when needed, approximate all the maps in time with the last one, $\mathcal{M}_k = \mathcal{M}_T \equiv \mathcal{M}$. This is reasonable since by definition the map is static, thus the final information is the best information about the complete map and its decomposition into time-varying visited maps was for numerical simplicity, as explained earlier. As modelled, the only real difference between time steps is the visibility frustum, which can accommodate the differences in sensory inputs. Therefore, in this setting, the map may be collapsed into a single entity again, just like in the real-static case (figure (5.9)).

It may seem futile to expand the system to dynamic environments just to go back to the same static graph, but (besides being able to represent real dynamic maps) following this path justifies the use of the full filtering equations, i.e. *predict* and *correct* stages, to send pose-to-map messages. This aligns well with previous filtering approaches such as the PHD SLAM algorithm. Otherwise, the predict equation wouldn't apply, i.e. it would have to be the identity, leaving no option to model birth and death effects.

Reduction to Smoothing and Mapping

It is interesting to observe that the Loopy RFS framework is a generalization of the vector Graph-SLAM representation. By framing the vector assumptions correctly, the RFS equations reduce to Loopy SAM, whose solution itself is equivalent to those of $\sqrt{\text{SAM}}$ and iSAM2.

As has been stated before, the vector framework assumes a known data association

$$\theta_k^*(z_k^i, l_k^j) = \begin{cases} 1 & z_k^i \text{ corresponds to } l_k^j \\ 0 & \text{otherwise,} \end{cases} \tag{5.43}$$

where z_k^i are the individual measurements in the measurement set $\mathcal{Z}_k$ and l_k^j are the individual landmarks in the map $\mathcal{M}$. This entails that probability of detection $P_D = 1$ for every associated

measured landmark and $P_D = 0$ for every unassociated one, i.e.

$$P_D(X_k, m) = \begin{cases} 1 & \exists\, z_k^i \in \mathcal{Z}_k : \theta_k^*(z_k^i, m) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.44}$$

Similarly, $P_{\text{clutter}} = P_\kappa = 0$ for every associated measurement and $P_\kappa = 1$ for every unassociated one, i.e.

$$P_\kappa(z) = \begin{cases} 0 & \exists\, l_k^j \in \text{Field}(\mathcal{M}) : \theta_k^*(z, l_k^j) = 1 \\ 1 & \text{otherwise,} \end{cases} \tag{5.45}$$

where Field($\mathcal{M}$) is the space where the single elements in the set map lie. From these, it can be deduced that the cardinality of the map $|\mathcal{M}| = m^*$ is also known and the measurement likelihood sum (see equation 2.56) contains only one non-zero term. These assumptions will be labelled the "vector assumptions".

Using the general definitions of RFS, for any given map $\mathcal{M}$ with cardinality m the joint distribution of trajectory and map will be [105]

$$P(X_{0:k}, \mathcal{M}|\mathcal{Z}_{1:k-1}) = m! P_{|\mathcal{M}|}(m) P(X_{0:k}, (m^1, m^2, ..., m^m)|Z_{1:k-1}). \tag{5.46}$$

Since the cardinality $|\mathcal{M}| = m^*$ is known, $P_{|\mathcal{M}|}$ is a Dirac delta distribution about this value. Therefore, the distribution becomes

$$P(X_{0:k}, M|\mathcal{Z}_{1:k-1}) = \begin{cases} P(X_{0:k}(m^1, m^2, ..., m^m)|Z_{1:k-1}, \theta_k) & |M| = m^* \\ 0 & |M| \neq m^*. \end{cases} \tag{5.47}$$

This is, in fact, the conditional expression for the vector case. Note that the $m!$ constant term disappears in this last expression because the distribution must integrate to one (which the vector term already does).

From RFS theory, the recurring integrating term

$$\begin{aligned} \int P(\mathcal{Z}_k, \mathcal{M}|X_{0:k}, Z_{1:k-1}) d\mathcal{M} \\ &= P(\mathcal{Z}_k, \phi|X_{0:k}, Z_{1:k-1}) \\ &+ \int P(\mathcal{Z}_k, \{m^1\}|X_{0:k}, Z_{1:k-1}) dm^1 \\ &+ \frac{1}{2!} \int\int P(\mathcal{Z}_k, \{m^1, m^2\}|X_{0:k}, Z_{1:k-1}) dm^1 dm^2 + ... \\ &+ \frac{1}{n!} \int \cdots \int P(\mathcal{Z}_k, \{m^1, m^2, ..., m^n\}|X_{0:k}, Z_{1:k-1}) dm^1 dm^2 ... dm^n + ... \end{aligned} \tag{5.48}$$

may be simplified. Since the cardinality of the map is known, only one of the terms is nonzero, when $|\mathcal{M}_k| = m^*$. Using a similar reasoning as before, it can then be reduced to the vector case formulation,

$$\begin{aligned} \int P(\mathcal{Z}_k, \mathcal{M}|X_{0:k}, Z_{1:k-1}) d\mathcal{M} \\ &= \frac{1}{m^*!} \int \cdots \int P(\mathcal{Z}_k, \{m^1, m^2, ..., m^{m^*}\}|X_{0:k}, Z_{1:k-1}) dm^1 dm^2 ... dm^{m^*} \\ &= \frac{1}{m^*!} \int \cdots \int m^* ! P_{|M_k|}(m^*) P(\mathcal{Z}_k, (m^1, m^2, ..., m^{m^*})|X_{0:k}, Z_{1:k-1}) dm^1 dm^2 ... dm^{m^*} \\ &= \int \cdots \int P(\mathcal{Z}_k, (m^1, m^2, ..., m^{m^*})|X_{0:k}, Z_{1:k-1}) dm^1 dm^2 ... dm^{m^*}, \end{aligned} \tag{5.49}$$

which is the vector integrating term.

Similarly, for the measurement likelihood term,

$$
\begin{aligned}
P(\mathcal{Z}_k|X_{0:k},\mathcal{M}) &= \sum_{\theta}\left(P(\mathcal{Z}_k^{\theta}|X_{0:k},\mathcal{M}^{\theta})\prod_{m\in\mathcal{M}^{\theta}}P_D(X_k,m)\prod_{m\in\overline{\mathcal{M}^{\theta}}}(1-P_D(X_k,m))P_{\kappa}(\overline{\mathcal{Z}_k^{\theta}})\right)\\
&= P(\mathcal{Z}_k^{\theta_k^*}|X_{0:k},\mathcal{M}^{\theta_k^*})\prod_{m\in\mathcal{M}^{\theta_k^*}}P_D(X_k,m)\prod_{m\notin\mathcal{M}^{\theta_k^*}}(1-P_D(X_k,m))P_{\kappa}(\overline{\mathcal{Z}_k^{\theta_k^*}})\\
&= P(\mathcal{Z}_k^{\theta_k^*}|X_{0:k},\mathcal{M}^{\theta_k^*})\prod_{m\in\mathcal{M}^{\theta_k^*}}1\prod_{m\notin\mathcal{M}^{\theta_k^*}}(1-0)\cdot 1\\
&= P(\mathcal{Z}_k^{\theta_k^*}|X_{0:k},\mathcal{M}^{\theta_k^*}) \qquad (5.50)\\
&= \prod_i P(z_k^i|X_{0:k},l_k^{j(i)}), \qquad (5.51)
\end{aligned}
$$

where $j(i)$ is the inverse association model, i.e. $\theta_k^*(z_k^i,l_k^{j(i)})=1$. The whole measurement likelihood term is equivalent to the vector based SLAM measurement likelihood.

So far, these are general statements about RFS and vector formulations. In the particular case of Loopy RFS and Loopy SAM, there are some additional derivations to be made. Since these systems work with messages, it will be shown that Loopy RFS messages generalize Loopy SAM messages.

Pose-to-pose messages (both forward and backward) are trivially equivalent, since they were defined exactly the same way.

Map-to-pose messages have the form

$$
\begin{aligned}
m &= \max_{\mathcal{M}} P(\mathcal{Z}_k|X_k,\mathcal{M})P(\mathcal{M})\\
&\approx P(\mathcal{Z}_k|X_k,M_{\text{map}|\bar{k}}) \qquad (5.52)\\
&\xrightarrow[assumptions]{vector} \prod_i P(z_k^{j(i)}|X_k,m^i_{\text{map}|\bar{k}}) \qquad (5.53)\\
&\propto \prod_i m_{\ell_j\to X_k}, \qquad (5.54)
\end{aligned}
$$

where the last expression is the product of all vector-based landmark-to-pose messages. Hence, map-to-pose messages are equivalent, as long as $M_{\text{map}|\bar{k}}$ corresponds to the MAP estimate of the vector landmarks.

From the definition of the MaM estimator, this can be easily checked,

$$
\begin{aligned}
\hat{m} &= \arg\sup_m P_{|\mathcal{M}|}(m) = m^* \qquad (5.55)\\
\mathcal{M}^{\hat{M}aM} &= \arg\sup_{\{m^i\}} P(\{m^1,m^2,...,m^{m^*}\}) \qquad (5.56)\\
&= \arg\sup_{(m^i)} m!P_{|\mathcal{M}|}(m)P((m^1,m^2,...,m^{m^*})) \qquad (5.57)\\
&= \arg\sup_{(m^i)} P((m^1,m^2,...,m^{m^*})) \qquad (5.58)\\
&= \prod_{i=1}^{m^*}\arg\sup_{m^i} P(m^i). \qquad (5.59)
\end{aligned}
$$

In the last line, the independence of the landmarks has been used.

Similarly, pose-to-map messages have the form

$$\begin{aligned} m &= \max_{X_k} P(\mathcal{Z}_k|X_k, \mathcal{M})P(X_k) \\ &\approx P(\mathcal{Z}_k|\mu_{k|\overline{\text{map}}}, \mathcal{M}) && (5.60) \\ &\xrightarrow[assumptions]{vector} \prod_i P(z_k^{j(i)}|\mu_{k|\overline{\text{map}}}, m^i). && (5.61) \end{aligned}$$

Thus, taking all the pose-to-map messages together, this becomes

$$\begin{aligned} m &= \prod_k \prod_i P(z_k^{j(i)}|\mu_{k|\overline{\text{map}}}, m^i) && (5.62) \\ &= \prod_{m^i} \prod_{k,j} P(z_k(m^i)^{j(m^i)}|\mu_{k|\overline{\text{map}}}, m^i) && (5.63) \\ &= \prod_{\ell_j} \prod_k m_{X_k \to \ell_j}, && (5.64) \end{aligned}$$

where the factors have been ordered by landmark instead of by pose. This, again, corresponds to the product of all the vector based pose-to-landmark messages for each landmark and pooling them together to form the map probability.

It has been shown that the messages are equivalent, but one possible difference is the dissemination scheme, i.e. how those messages are propagated through the graph. Loopy RFS, as defined here, is equivalent to using a mildly rigid scheme: at every frame, every map-to-pose message must be sent at the same time, i.e. there may not be any delay between messages from different landmarks. The constraint is similar for pose-to-map messages.

This does not preclude scheduling these frame-packets as desired, but they cannot be separated by landmarks. This shouldn't be a real problem, since the scheme is usually something like this anyway. In the rare case where more flexibility is required, the map could be decomposed into separable submaps and each submap updated independently. This independence means there's no graph connection between submaps, so every one communicates independently with the trajectory. The extreme case would be to have each landmark as a submap, which is very similar to the vector case. Naturally, in this scenario, using the measurements becomes harder since there needs to be a way to decide which submaps uses which measurement. A trivial solution would be to use data association on the submaps, which may seem paradoxical, since RFS removes the need for it, but such association would be at a larger (usually easier) scale. Nevertheless, other solutions may be derived, but since the coarse scheme should be fine, they will not be pursued here.

2.3 Proof-of-Concept Implementation

A simple implementation of the algorithm outlined above was written, as a proof-of-concept. Special attention was put into correctness and visualization of internal computations so as to better understand the performance of the system. The implementation still requires much optimization to achieve real-time performance, but it showcases the core principles behind this approach.

For simplicity, this implementation focuses on a linear 2D case, where the odometry and measurements are trivial sums (i.e. the state update is $X_{t+1} = (x_t, y_t)^T = X_t + \Delta X_t$ and the vector-based measurement model is $Z_t = l_t - X_t$, where l_t is a landmark). It can also perform inference on other cases, but this simple case is more straightforward to interpret, e.g. covariances can be fully visualized using a 2D ellipse, unlike the 3D case where there are 6 (or 7) location and rotation components which can't be easily visualized in 3D space without projecting into a smaller space; also, in linear 2D space the information given by the measurements to the pose is straightforward (a trivial 2D-translated Gaussian), unlike the 3D case where the effective covariance is transformed by the non-unitary rank-defficient measurement Jacobian.

There are two major components in this approach: a general message passing architecture which contemplates asking for each message at the appropriate times and disseminating them to the appropriate nodes efficiently, i.e. computing values only once, caching them as necessary; and the actual message computation routines.

The dissemination scheme is contained in a single function so it may be easily changed. For the experiments, the following scheme is used: propagate messages forwards in time through the trajectory path, then send messages from the trajectory poses to the map, then from the map to the poses, propagate backwards in time through the trajectory path, then from poses to map, then from map to poses and iterate. The messages are initialized from running a PHD filter on the entire trajectory.

The forward and backward messages are straightforward as noted in the theory section and follow equation (5.15). Pose-to-map messages are computed by running independent PHD filters skipping the appropriate time steps, as indicated in the theory section; it is noted that this is inefficient and scales as $O(T^2)$, where T is the number of pose nodes, i.e. the total length of the trajectory. This becomes a bottleneck on long paths. This time complexity may be reduced in the future by using keypoints (similar to what visual SLAM systems do [54]) or by deriving a general inverse PHD step, so the full PHD filter can be run once and then only the inverse step be used once per time step (this solution would scale as $O(T)$).

The most complex message computation is map-to-pose, which corresponds to the measurement factor $P(\mathcal{Z}_k|X_k, \mathcal{M})$. This factor is complex and it is hard to approximate it with a Gaussian (so as to fuse it with the past and future messages). At first, a local maximum was sought using optimization, which was difficult because this function can be ill-behaved: Near the measurements' "ideal pose" (the pose that would exactly produce such measurements given the real landmark), the function has an approximately Gaussian behaviour (i.e. the gradient of its logarithm is a linear form, so it is easy to reach the maximum) but far from it, the function is flat, i.e. its gradient is arbitrarily close to zero because the clutter-misdetection component dominate over any possible data association, (in equation (2.56), the clutter-misdetection component corresponds to the term in the sum where every measurement is associated to clutter and every landmark is misdetected) so following the gradient doesn't help in getting closer to the solution. As more and more measurements are added, the fused "ideal pose" covariance becomes smaller and smaller, so the final landscape of the function is mostly flat with one or more sharp peaks, which is difficult to optimize.

To solve this problem, the first thing to do was to work in the log-space so the exponential decays from the Gaussians became parabolas, which span a more manageable range for numerical computation. Then, a good initialization point is required to start gradient ascent; for each

time step and each measurement-landmark association, an "ideal pose" was proposed, which was used as initialization for the gradient ascent. Each of these propositions should be in a non-flat region since at least one of the association component of equation (2.56) should be bigger than the clutter-misdetection component (in particular, the term corresponding the measurement-landmark association used to propose the pose should perfectly align by construction), i.e. there is a useful gradient to work with. Assuming reasonable levels of spatial noise, most of the $P(\mathcal{Z}_k|X_k, \mathcal{M})$ factor should be in the components found from the proposed pairs. However, for efficiency reasons it is not feasible to use all associations, so some of them are removed early if found not worth of exploring further.

After the local maxima are found, a semi-numerical approximation of the Hessian matrix is computed. To achieve this, a numerical gradient of the symbolical gradient of the factor is extracted using finite differences and a hand-crafted gradient (an automatic differentiation may also work in this case). Each maximum is then associated with a covariance corresponding to the negative inverse of the Hessian.

If the factor is approximated as the Gaussian corresponding to the highest maximum, it is effectively using its corresponding data association and using it to compute the message, just as Loopy SAM does. Note however that unlike Loopy SAM, this association would be local to one message so if the entire inference goes another way, a different maximum could be chosen, i.e. changing data association on the fly as information flows.

This was tried in a first attempt, but only worked for very simple cases where association was trivial and the initial estimates were very close to ground-truth. The problem was that in more difficult cases, the message would get stuck in a very far away maximum, which nevertheless was the best association (this messages does not consider the pose constraints with respect to the past and the future) with high certainty. Then, at fusion time, both past and future messages were relatively "uncertain" but the map message was very certain (the association could be very good, e.g. if only one measurement was made), so the final estimate was dominated by the (in reality, doubtful) map message. This artificial certainty would propagate through the graph and make the whole system unstable causing the pose estimates to oscillate, e.g. at $t = 1$, X_3 would be estimated to be around (2, 3), then at $t = 2$ it would be estimated to be around (3, 4), at $t = 3$ it would be around (2.1, 3.2), at $t = 4$ it would be around (2.9, 4.1), and so forth.

The reason for such extreme behaviour is clear: considering only one component throws away too much information and therefore the state estimate becomes unreasonable confident around an incorrect configuration, which would then be confidently contradicted in the next iteration, leading to big jumps in the state estimates. Instead, a second iteration of the system merged all components into one large component by computing the effective mean and covariance of the weighted mixture. This way if there were multiple hypotheses far apart in space, the uncertainty would be reflected in the message and the fusion would be more reasonable.

The change helped, but the system still suffered of some unstability. Another important effect was still missing: the constant clutter-misdetection component of the map-to-pose messages in equation (2.56). As noted before, the $P(\mathcal{Z}_k|X_{0:k}, \mathcal{M})$ factor has a constant component. Since the message will eventually be multiplied with the past and future messages to form the estimate for

the poses, this constant component will introduce an extra term, i.e.

$$P(X_k) \propto m_{k-1\to k}(X_k) m_{k+1\to k}(X_k) m_{\text{map}\to k}(X_k) \tag{5.65}$$
$$= \mathcal{N}_{k-1\to k} \times \mathcal{N}_{k+1\to k} \times m_{\text{map}\to k} \tag{5.66}$$
$$= \mathcal{N}_{k\pm 1} \times (\mathcal{N}_{\text{map}+C}) \tag{5.67}$$
$$= B\mathcal{N}_{k\pm 1,\text{map}} + C\mathcal{N}_{k\pm 1}, \tag{5.68}$$

where $\mathcal{N}_i$ are Gaussian messages, $\mathcal{N}_{k\pm 1}$ is the fusion of past and future messages and the map message is a Gaussian plus a constant. $\mathcal{N}_{k\pm 1,\text{map}}$ is the fusion of all three Gaussians and B and C are constants computed from the multiplications.

By not considering the constant term, i.e. $C = 0$, the relative weights B and C are disregarded. If the fused $\mathcal{N}_{k\pm 1,\text{map}}$ is small compared to the half-fused $\mathcal{N}_{k\pm 1}$, this is not acceptable (the smallness threshold is given by the constant clutter-misdetection component C). From the equation, it can be seen that this occurs when the map message is very far from the past-future fusion $\mathcal{N}_{k\pm 1}$. By considering the constant term, the system can effectively disregard unlikely map messages, reducing their capability of artificial disruption in the estimates, leading to better stability.

Additionally, since messages are independent (i.e. they may change independently from one another), the map message merging is postponed until fusion (so the latest past and future messages can be considered), i.e. the map message finally consists of a list of Gaussian functions and the clutter-misdetection constant. At fusion time every component is fused with the past and future messages and the final mixture is merged to form a single Gaussian based estimate for each pose.

Also, to improve convergence, the map-to-pose messages are associated with a temperature K, which artificially increases the uncertainty of their distribution, i.e. the final map-to-pose message is of the form $m_{\text{map}\to k}(X_k) * \mathcal{N}(0, K^2)$, where $*$ represents a convolution. This temperature reduces the effect of the map message in the final estimate. It starts at $T = 2\Sigma_{\text{measurement}}$ and decreases at each iteration following $K(t) = \frac{2\Sigma_{\text{measurement}}}{t}$. This way, the landmarks slowly move the pose estimates towards the correct distributions, allowing for incremental changes instead of jumps. After a couple of iterations, the temperature is practically zero so the rest of the algorithms is as if no temperature were introduced, so it can be seen as a good way to initialize the estimate.

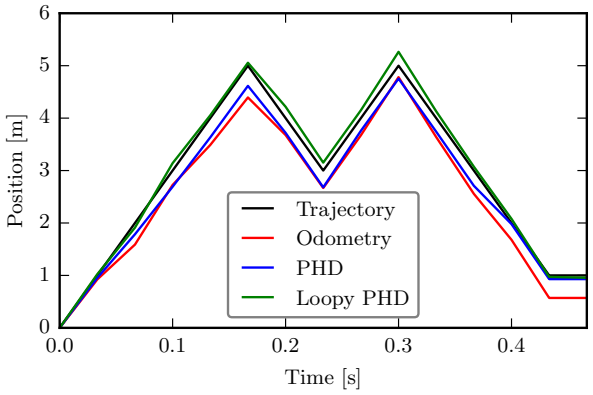
A final note for the map-to-pose messages is that when no measurements are present the message is constant, so the fused estimate for such a node is completely determined by the past and future messages.

2.4 Results

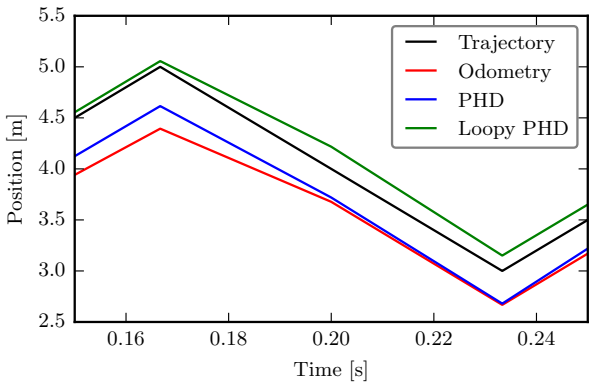
Three experiments were carried out to evaluate the loopy system performance. First, a 1D vehicle which moves around a hallway with features at both ends. The motion variance σ_x is $18\,[m^2]$, the measurement variance σ_z is $0.005[m^2]$, landmarks are missed 20% of the time, there are 2.736 clutter measurements per time step. Second, an easy 2D setting where the vehicle moves around a loop with landmarks all around it, with perfect detection statistics, no clutter and no measurement noise. Third, a more realistic setting, where the path is more complex, landmarks are missed 20% of the time, there are $N_c = 0.912$ clutter measurements per time step and the measurement noise

covariance σ_z is $0.005\,[m^2]$. In both 2D cases, the motion noise covariance σ_x is $2\,[m^2]$. For the map error in the three scenarios, the OSPA parameters are $c = 1[m]$ and $p = 1$.

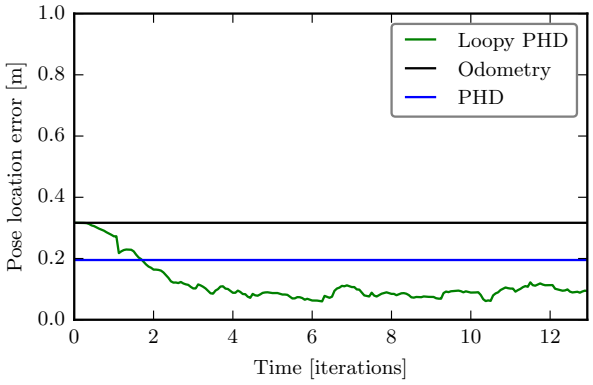
The results of the first case (one dimensional vehicle) are shown in figure (5.13). Loopy PHD manages to obtain a better estimate for the trajectory than the PHD filter with 2000 particles. In the estimate plot it can be observed that the PHD filter corrects the loop closure at the end (around 0.4 [s]) by abruptly changing its position to align it to the real trajectory, while the Loopy PHD algorithm is able to modify the entire path to more closely follow the real trajectory.



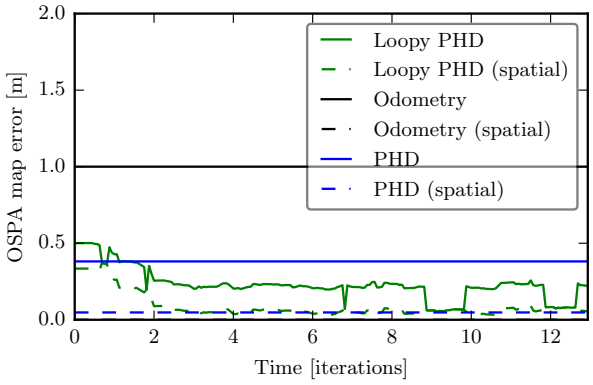
(a) Estimates.



(b) Zoomed section of the estimates.



(c) Location error.

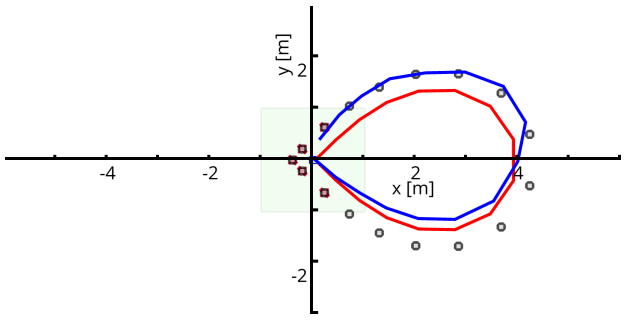


(d) Map error (OSPA, $c = 1\,[m]$ and $p = 1$). 'Spatial' corresponds to the spatial component of OSPA and disregards the difference in cardinality.

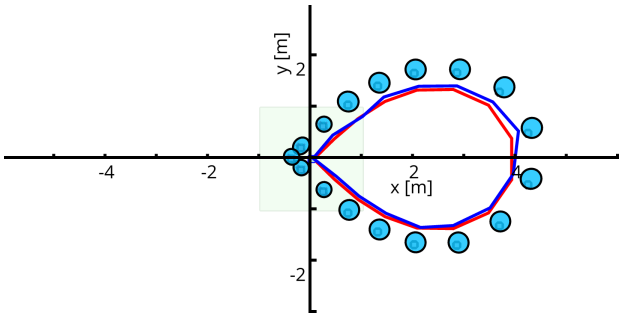
Figure 5.13: Loopy PHD performance in a 1D scenario, ($\sigma_x^2 = 18\,[m^2], \sigma_z^2 = 0.005\,[m^2], N_c = 0.912, P_D = 0.8$).

The results of the second case are shown in figure (5.14). This case tests the ability of the algorithm to find the optimum when the measurements give perfect information. The system does find the optimum in around 127 steps. This mode of behaviour is directly comparable with iSAM2 and Loopy SAM, since the perfect measurements give rise to easy data association.

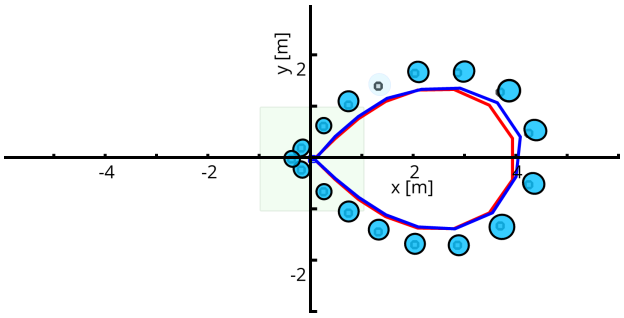
The results of the third case are shown in figure (5.15). This case reflects a realistic scenario. The system can also find the optimum, even outperforming the PHD filter with 800 particles in few iterations. This scenario is harder for the vector equivalent algorithms due to both spatial and



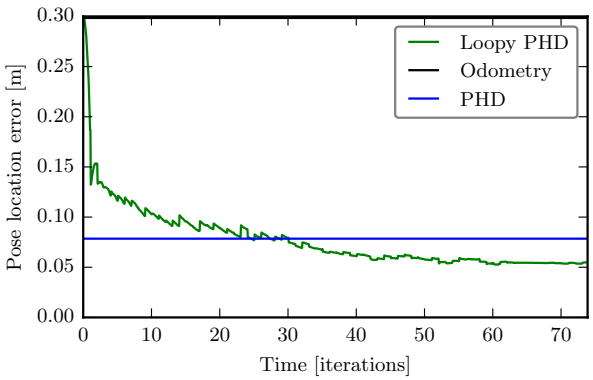
(a) Odometry dead reckon estimate.



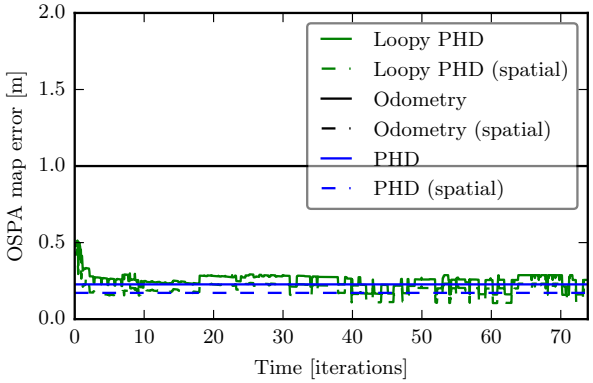
(b) PHD filter estimate. The red trajectory is the groundtruth and the blue one is the estimate.



(c) Loopy PHD estimate. White rectangles are landmarks and blue/black ellipses are the map estimate.



(d) Location error.



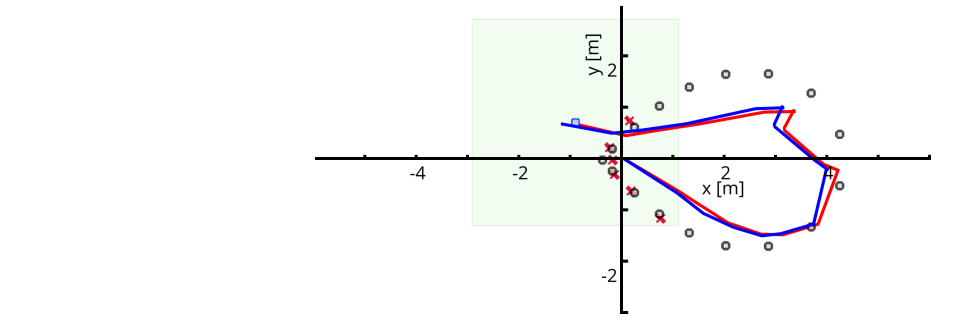
(e) Map error.

Figure 5.14: Loopy PHD performance in an easy scenario, ($\sigma_x^2 = 2\,[m^2], \sigma_z^2 = 0\,[m^2], N_c = 0, P_D = 1.0$).

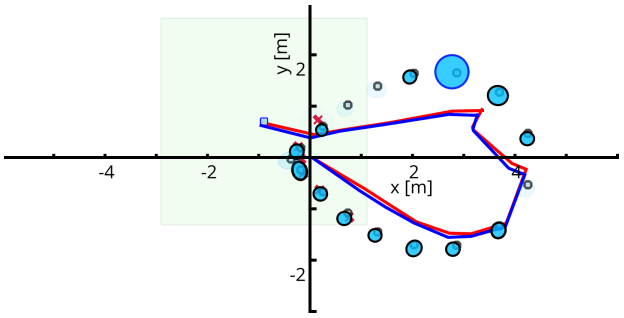
cardinality uncertainties in the measurements.

2.5 Discussion

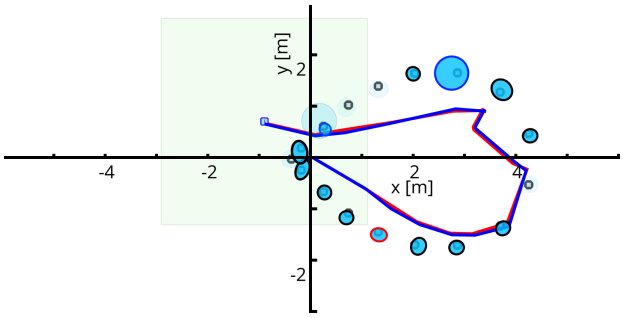
As shown in the result section, this approach has potential for solving some of the limitations of RFS filtering methods in the SLAM problem; in particular, it can close loops, moving information quickly through the graph, avoiding the expensive number of particles required for long-term inference. Compared with vector graph-based models, it does not depend on a predefined data association. Instead, it is equivalent to considering all possible data associations, and numerically focusing on those which show the biggest promise, depending on the circumstances (it can even be



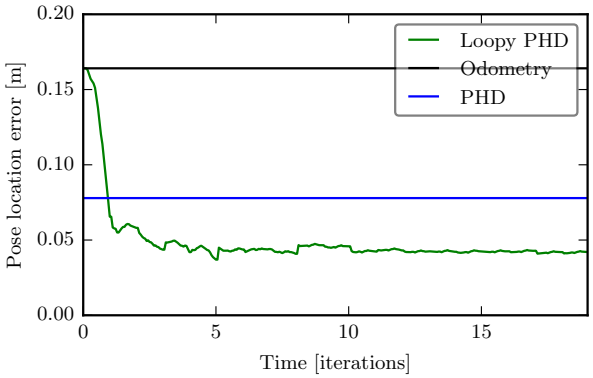
(a) Odometry dead reckon estimate.



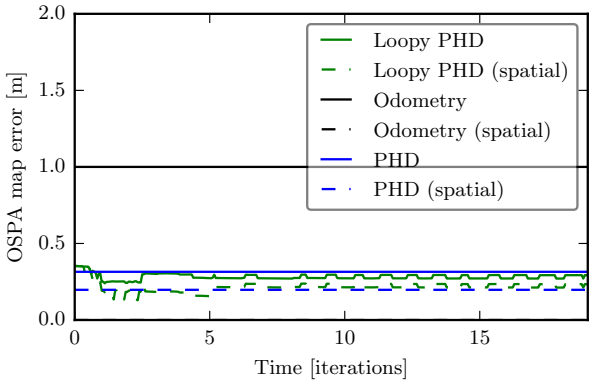
(b) PHD filter estimate.



(c) Loopy PHD estimate.



(d) Location error.



(e) Map error.

Figure 5.15: Loopy PHD performance on a realistic scenario, ($\sigma_x^2 = 2\,[m^2], \sigma_z^2 = 0.05\,[m^2], N_c = 0.96, P_D = 0.8$).

equivalent to considering multiple good associations in parallel).

It was observed that using multi-component mixtures is fundamental to modelling the map-to-pose messages satisfactorily (especially the constant clutter-misdetection threshold). Otherwise the system may not converge to any particular estimate due to overconfidence of the messages unsupported by the data, with behaviour similar to using bad data associations in a vector approach.

The current implementation has the drawback of requiring long computational times for the optimization but this is not inherent in the methodology and can be alleviated with the use of key-points or by deriving a backwards solution to the RFS filtering problem (which has been done in the literature [106], although using particles instead of the more convenient Gaussian mixtures). The methodology as presented in this thesis has $O(T^2)$ time complexity. This is due to the requirement

of removing the effect of the pose for the map-to-pose messages, which means the filtering equation must be run once per time step. It may be tempting to simplify this by using the full filtered map estimate for the map-to-pose messages, i.e. without removing the redundant message, since the difference induced by such removal should be small (one frame can only affect a small portion of the map). This would only require one filtering pass, greatly diminishing the runtime to linear time. However, this breaks the Loopy Belief Propagation theory and its properties no longer hold. In practice, trying this resulted in unstable behaviour, so other ways of reducing the time complexity must be explored.

Although the system often converges to an optimum, there are cases where it does not. In these cases, it is typical to observe the position error go up for a short period and then go down abruptly to a level close to the odometry and PHD errors, cycling between these two behaviours. These could be associated to numerical difficulties typical in optimization methods and could be reduced by using more advanced algorithms that avoid local minima such as momentum methods or more complex temperature schemes. Nevertheless, it is noted that even in these non-convergent cases, it may be possible to obtain good estimates, since the final objective of the method is to find a minimum configuration (not for such configuration to be the stable state of the iteration). So, for each visited configuration in the algorithm, the (negative of the) likelihood can be computed directly from its formula. The configuration which attains the minimum value may be a good solution (and it must be at least as good as the initialization configuration).

Finally, in previous chapters, one of the modifications to the PHD filter was to use a measurement induced birth process so as to speed up computation. However this would introduce artificial certainty in the posterior estimate of the map due to the predicted PHD being exactly in the spot where the measurement would be. To solve this issue, the introduction of the birth process was delayed by one frame, so the measurement wasn't strongly correlated with the birth process. Although this works for reducing spurious landmark estimates, it changes the dynamics of the estimator. In particular, it strongly changes the effect of skipping one frame (which is used in Loopy RFS), because the next frame will not have the appropriate birth process density. So, for the loopy approach this delay was removed and the birth covariance increased so the overconfidence gained by it was minimal. Naturally, increasing such a covariance value meant that performance-related thresholds had to adjusted and the filtering takes more time. This can be improved in the future by explicitly including a constant component in the PHD Gaussian mixtures, similarly to what was done in the Loopy PHD algorithm.

In future work, the final merging step in the fusion step could be removed, letting the fused estimate be multi-modal. This could encode richer information about multiple hypotheses and move alternate theories through the trajectory path. Of course, every computation would have to performed multiples times (one per component), so a good pruning step would be required to limit the consumption of resources.

Chapter 6

Conclusion

Autonomous systems promise to change many fields of science and many everyday life interactions, from manipulating objects in factories to driving people around safely. At the core of this technology there's the SLAM algorithm, which is able to understand the world around the navigator, allowing it to localize itself. This algorithm requires sensory inputs. Cameras are a cheap, scalable, rich input which has garnered attention throughout the years due to its highly structured visual content.

The visual SLAM problem is not new and there have been several previous approaches to solving it, which have been thoroughly reviewed. In particular, different methods' strengths and weaknesses have been reviewed and provide context around the requirements for real operations in autonomous vehicles. Nevertheless, so far none of these methods is capable of coping with the difficult visual situations that the visual features experience in extreme conditions such as nighttime, tunnels, rain, low-quality images or dynamic environments, which makes previous algorithms unsuitable for safe and robust navigation. In such scenarios, detection is not trivially established, so it becomes necessary to include clutter and misdetection statistics.

This work has presented the key points in the elaboration of a visual SLAM solver system, capable of transforming raw input from an RGB-D camera, a pre-recorded stream or a simulation into a precise estimate of a vehicle's location and an estimate of the surrounding map. For this, the RFS-based Rao-Blackwellized GM-PHD filter has been implemented in 3D space with appropriate RGB-D measurements, extracted using FAST features. By using the random finite set theory, it is robust to imperfect model statistics, i.e. misdetected features and clutter measurements.

The system has been designed and implemented with flexibility and modularization in mind, so extensions can be easily added, subsystem can be swapped and intermediate results can be observed for debugging and analysis purposes.

On one hand, simulations have been executed to understand the performance of the filtering system with respect to different parameters, such as the probability of detection, the clutter rate, the motion and measurement noise and the number of particles. On the other hand, real camera video streams have been used to validate this performance.

One of the key complications for the RFS SLAM formulation is the necessity of known statistics for the problem, which may not be trivial to estimate. However, the comparison with other

algorithms that do not require such statistics is not quite fair because these alternatives just assume a particular set of values for them. Nevertheless, the system has been shown to work well under moderate deviations from the correct values and there are works such as [75] that work on estimating them jointly, within the RFS filtering framework. The case of the probability of detection is the most difficult one in the case of visual sensors because of the inherently sparse information that cameras provide, which makes it hard to generalize from dense RGB-D maps to monocular or stereo vision. Using dense or semi-dense approaches could solve this problem.

In this work, the RFS filtering system has also been compared with the iSAM2 method from the GraphSLAM family, one of the most commonly used alternatives. It has been shown that given enough particles, the RFS approach can outperform the GraphSLAM method, since it can easily deal with uncertain associations, while iSAM2 must decide on one of them.

To cope with clutter measurements, the iSAM2 approach has been endowed with an outlier rejection mechanism, which requires a landmark to be observed a number of times before it is deemed an acceptable measurement, disregarding most clutter readings. Additionally, when using real data, both approaches use RANSAC to remove keypoints that do not coincide with the movement of the camera.

The simulations show that both algorithms perform well under reasonable assumptions, but that in difficult cases, such as high clutter, the PHD filter converges while iSAM2 frequently diverges and cannot recover. Also, given enough particles, the RB-PHD filter loop closures are better (if enough particles are used). If an approximately correctly located particle is present, it will dominate over all the others, but in the case of iSAM2 if there is too much drift there's no way to get the correct association for the revisited landmarks from the SLAM method itself so the solution becomes suboptimal. If the association is (unfairly) known it works well, but that is not realistic. In the literature, e.g. ORB-SLAM, the solution has been to match the images directly to find out when and how to close a loop, but this is very resource-intensive. The RFS alternative is also resource-intensive but to a smaller degree, it is grounded on correct mathematical foundations, and it doesn't require complex matching.

Here it is noted that the typical separation between association and SLAM is not optimal, since the implicit associations can guide the SLAM algorithm, as happens in the RFS case. Nevertheless, it is also observed that the particle-based filter is not very efficient with the particles because most of their trajectory values are very similar so the same work is performed multiple times, especially when the particle count is high. There are several ways to mitigate this in future work, such as applying the FastSLAM 2.0 [39] concept to the RFS framework, or using the alternative Single-Cluster representation from the literature [73].

Alternatively, this work has proposed to use some of the ideas from the iSAM2 approach, namely the use of graphs to represent the structure of the problem, albeit in a higher-level fashion since the problem of the data associations does not allow for fine-grained edges between poses and landmarks. Loopy Belief Propagation has been used to move the information around, making approximations where required. Just like the graph-based approach, the loopy PHD algorithm is capable of re-linearising the problem, i.e. approximating non-linear dynamics better, and efficiently moving information from the future to the estimates.

It has been shown that the final algorithm is equivalent to a smart sampling scheme which takes all of the measurement history, both past and future, into consideration. Some properties of this

algorithm have been outlined and the relationship between this RFS approach and the vector based case has been pointed out for the case of trivial data association.

In future work, this proposition can be improved in several ways. The reduced $P_{\text{map}|\bar{k}}$ probabilities (which correspond to the distribution of the map disregarding the information given by the pose node x_k) could be calculated in a faster manner by undoing the kth step instead of multiplying all the others. These would reduce the redundancy of the algorithm from $O(T^2)$ to $O(T)$, where T is the number of pose nodes in the trajectory, which is a major improvement in real usage. Also, similarly to iSAM2, not every node requires updating at every time step. In fact, as the vehicle moves forward, old nodes usually do not change much so can be skipped, except in the case of loop closures which would require to refresh the messages for all the loop, again just like iSAM2 and loopy SAM. This would reduce the complexity even further to only a constant number of updates $O(1)$ per time step almost always. Also, a GPU could be used to make the system go faster. Since the messages are almost independent from one another at each time step, their calculations can be easily parallelized.

Other filters can be used instead, such as the cardinalized PHD or the Multi-Bernoulli filters, which can be more expressive and better capture the real dynamics of the sensor.

Note that in this work, relatively clear RGB-D streams have been used. Further testing against a dataset designed to check for robustness, i.e. difficult conditions such as low-light environments, could further show the advantages of this methodology.

Following the legitimate trend of using semantic knowledge of the scene to guide mapping, the system could be endowed with additional semantic metadata. As in the case of non-semantic keypoint descriptors, these semantics are usually used in the data association step. It may seem as though RFS estimation is not a good fit for using this information, but it can be used either as a preprocessing step (i.e. removing semantically uninteresting objects or unmatched descriptors) or as part of the measurement space, where semantic space embedding could be used to define closeness between object labels. Random finite sets can help with the same troublesome conditions of unknown cardinality and association, e.g. no prior knowledge of the number of cars in front of a vehicle or associating between them when they change their relative positions.

This research has the potential to improve the reliability of visual SLAM systems and thus can help autonomous vehicles be more safe for the public. Therefore, it is imperative to continue developing these ideas.

An important but sometimes forgotten aspect of this new development in science is the immediate social impact that these technologies bring. Although in the long term we may expect the world to improve due to the new found efficiency, safety and comfort, in the short term, autonomous machines replace people in their jobs. This negatively impacts the quality of life of these employees, who need time and resources to accommodate to the new situation. Strong policies are therefore needed to deal with the transition period as gracefully as possible.

Acronyms

BA Bundle Adjustment.

BP Belief Propagation.

BRIEF Binary Robust Independent Elementary Features.

COLAMD Column Approximate Minimum Degree Permutation.

CPHD Cardinalized Probability Hypothesis Density.

CPU Central Processing Unit.

DARPA Defense Advance Research Projects Agency.

DTAM Dense Tracking and Mapping.

EAP Expected A Posteriori.

EKF Extended Kalman Filter.

FAST Features from Accelerated Segment Test.

FREAK Fast Retina Keypoint.

GM Gaussian Mixture.

GPU Graphical Processing Unit.

GUI Graphical User Interface.

HMM Hidden Markov Model.

IMU Inertial Measurement Unit.

JCBB Joint Compatibility Branch and Bound.

JIT Just In Time.

JPDAF Joint Probability Data Association Filter.

KA Known-Association.

KAZE Japanese word for *wind* (type of image features).

LBP Loopy Belief Propagation.

LSD Large-Scale Direct (SLAM).

MAP Maximum A Posteriori.

MHMC Metropolis-Hastings Montecarlo.

MHT Multi-Hypothesis Tracking.

MRF Markov Random Field.

NN Nearest Neighbours.

OMAT Optimal Mass Transfer.

ORB Oriented FAST and Rotated BRIEF.

OSPA Optimal Sub-Pattern Assignment (metric).

PGM Probabilistic Graphical Model.

PHD Probability Hypothesis Density.

PMMHMC Particle Marginal Metropolis-Hastings Montecarlo.

PTAM Parallel Tracking and Mapping.

RANSAC Random Sample Consensus.

RB Rao-Blackwellized.

RFS Random Finite Set.

RGB Red-Green-Blue.

SAM Smoothing And Mapping.

SIFT Scale-Invariant Feature Transform.

SLAM Simultaneous Localization and Mapping.

SURF Speeded Up Robust Feature.

SVD Singular Value Decomposition.

UAV Unmanned Aerial Vehicle.

Bibliography

[1] Jonathan P How, Clive Fraser, Karl C Kulling, Luca F Bertuccelli, Olivier Toupet, Luc Brunet, Abraham Bachrach, and Nicholas Roy. Increasing autonomy of uavs. *Robotics & Automation Magazine, IEEE*, 16(2):43–51, 2009.

[2] K Whitehead, BJ Moorman, and CH Hugenholtz. Brief communication: Low-cost, on-demand aerial photogrammetry for glaciological measurement. *The Cryosphere*, 7(6):1879–1884, 2013.

[3] David W Casbeer, Derek B Kingston, Randal W Beard, and Timothy W McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351–360, 2006.

[4] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[5] Sonia Waharte and Niki Trigoni. Supporting search and rescue operations with uavs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147. IEEE, 2010.

[6] Christopher Null and Brian Caulfield. Fade to black the 1980s vision of ”lights-out” manufacturing, where robots do all the work, is a dream no more, 2003.

[7] Erico Guizzo and Evan Ackerman. How rethink robotics built its new baxter robot worker. *IEEE Spectrum*, 2012.

[8] Ben Tribelhorn and Zachary Dodds. Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1393–1399. IEEE, 2007.

[9] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[10] Todd Litman. Autonomous vehicle implementation predictions. *Victoria Transport Policy Institute 28*, 2014.

[11] Jeremie Houssineau, Emmanuel Delande, and Daniel Clark. Notes of the summer school on finite set statistics. *arXiv preprint arXiv:1308.2586*, 2013.

[12] Mark Mudge, Tom Malzbender, Alan Chalmers, Roberto Scopigno, James Davis, Oliver Wang, Prabath Gunawardane, Michael Ashley, Martin Doerr, Alberto Proenca, et al. Image-based empirical information acquisition, scientific reliability, and long-term digital preservation for the natural sciences and cultural heritage. *Eurographics Tutorials*, 2008.

[13] Jürgen Sturm, Erik Bylow, Fredrik Kahl, and Daniel Cremers. Copyme3d: Scanning and printing persons in 3d. In *Pattern Recognition*, pages 405–414. Springer, 2013.

[14] Chris Harris and Mike Stephens. A combined corner and edge detector., 1988.

[15] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[16] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[17] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[18] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005.

[19] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee, 2012.

[20] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[21] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *Computer Vision–ECCV 2012*, pages 214–227. Springer, 2012.

[22] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[23] Richard Hartley et al. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, 1997.

[24] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[25] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652. IEEE, 2004.

[26] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[27] Hernán Badino, Akiyasu Yamamoto, and Takeo Kanade. Visual odometry by multi-frame feature integration. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 222–229. IEEE, 2013.

[28] Wei Lu, Zhiyu Xiang, and Jilin Liu. High-performance visual odometry with two- stage local binocular ba and gpu. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1107–1112. IEEE, 2013.

[29] Hossein Mirabdollah and B"arbel Mertsching. Fast techniques for monocular visual odometry. In *Proceeding of 37th German Conference on Pattern Recognition (GCPR)*, pages 297–307, 2015.

[30] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *CVPR*, Columbus, Ohio, USA, 2014.

[31] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[32] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.

[33] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[34] Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge University Press, 2013.

[35] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *Journal of Fluids Engineering*, 83(1):95–108, 1961.

[36] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[37] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

[38] Juan Nieto, Jose Guivant, Eduardo Nebot, and Sebastian Thrun. Real time data association for FastSLAM. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 412–418. IEEE, 2003.

[39] Michael Montemerlo, Sebastian Thrun, Daphne Roller, and Ben Wegbreit. Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1151–1156. Morgan Kaufmann Publishers Inc., 2003.

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

[41] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[42] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[43] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, page 0278364911430419, 2011.

[44] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, 2008.

[45] John Mullane, Ba-Ngu Vo, Martin D Adams, and Ba-Tuong Vo. A random-finite-set approach to bayesian slam. *Robotics, IEEE Transactions on*, 27(2):268–282, 2011.

[46] Ronald PS Mahler. *Statistical multisource-multitarget information fusion*. Artech House, Inc., 2007.

[47] Ba Tuong Vo. *Random finite sets in multi-object filtering*. PhD thesis, University of Western Australia, 2008.

[48] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[49] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476. IEEE, 2006.

[50] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[51] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *arXiv preprint arXiv:1502.00956*, 2015.

[52] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[53] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[54] Georg Klein and David Murray. Improving the agility of keyframe-based slam. In *Computer Vision–ECCV 2008*, pages 802–815. Springer, 2008.

[55] Andrew P Gee, Denis Chekhlov, Andrew Calway, and Walterio Mayol-Cuevas. Discovering higher level structure in visual slam. *Robotics, IEEE Transactions on*, 24(5):980–990, 2008.

[56] Renato Salas-Moreno, Richard Newcombe, Hauke Strasdat, Paul Kelly, and Andrew Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.

[57] Jan Stühmer, Stefan Gumhold, and Daniel Cremers. Real-time dense geometry from a handheld camera. In *Pattern recognition*, pages 11–20. Springer, 2010.

[58] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[59] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[60] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2609–2616. IEEE, 2014.

[61] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[62] Raúl Mur Artal. Should we still do sparse-feature based slam? ICCV, 2015.

[63] Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1152–1178, 2003.

[64] Daryl J Daley and David Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*, volume 2. Springer Science & Business Media, 2007.

[65] Ba-Ngu Vo, Sumeetpal Singh, and Arnaud Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):1224–1245, 2005.

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

[67] Ba-Ngu Vo and Wing-Kin Ma. The gaussian mixture probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 54(11):4091–4104, 2006.

[68] Keith YK Leung, Felipe Inostroza, and Martin Adams. Evaluating set measurement likelihoods in random-finite-set slam. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–8. IEEE, 2014.

[69] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni. Analytic implementations of the cardinalized probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 55(7):3553–3567, 2007.

[70] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni. The cardinality balanced multi-target multi-bernoulli filter and its implementations. *Signal Processing, IEEE Transactions on*, 57(2):409–423, 2009.

[71] Denis Wolf and Gaurav S Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1301–1307. IEEE, 2004.

[72] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 842–849. IEEE, 2003.

[73] Chee Sing Lee, Daniel E Clark, and Joaquim Salvi. Slam with dynamic targets via single-cluster phd filtering. *Selected Topics in Signal Processing, IEEE Journal of*, 7(3):543–552, 2013.

[74] Feihu Zhang, Hauke Stähle, Andre Gaschler, Christian Buckl, and Alois Knoll. Single camera visual odometry based on random finite set statistics. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 559–566. IEEE, 2012.

[75] Ronald PS Mahler, Ba-Tuong Vo, and Ba-Ngu Vo. CPHD filtering with unknown clutter rate and detection profile. *Signal Processing, IEEE Transactions on*, 59(8):3497–3513, 2011.

[76] Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[77] Tuyet Vu, Ba-Ngu Vo, and Roger Evans. A particle marginal metropolis-hastings multi-target tracker. *Signal Processing, IEEE Transactions on*, 62(15):3953–3964, 2014.

[78] Ingemar J Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[79] Rainer E Burkard and Eranda Cela. *Linear assignment problems and extensions*. Springer, 1999.

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

[81] Thomas E Fortmann, Yaakov Bar-Shalom, and Mathias Scheffe. Multi-target tracking using joint probabilistic data association. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, pages 807–812. IEEE, 1980.

[82] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. The probabilistic data association filter. *Control Systems, IEEE*, 29(6):82–100, 2009.

[83] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.

[84] Katta G Murty. Letter to the editor—an algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16(3):682–687, 1968.

[85] Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization, 2010.

[86] BC Hall. An elementary introduction to groups and representations. Technical report, University of Notre Dame, 2000.

[87] Veeravalli Seshadri Varadarajan. *Lie groups, Lie algebras, and their representations*, volume 102. Springer Science & Business Media, 2013.

[88] AJ Baddeley. Errors in binary images and an lp version of the hausdorff metric. *Nieuw Archief voor Wiskunde*, 10(4):157–183, 1992.

[89] John R Hoffman and Ronald PS Mahler. Multitarget miss distance via optimal assignment. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(3):327–336, 2004.

[90] Dominic Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo. A consistent metric for performance evaluation of multi-object filters. *Signal Processing, IEEE Transactions on*, 56(8):3447–3457, 2008.

[91] Keith YK Leung, Felipe Inostroza, and Martin Adams. An improved weighting strategy for rao-blackwellized probability hypothesis density simultaneous localization and mapping. In *Control, Automation and Information Sciences (ICCAIS), 2013 International Conference on*, pages 103–110. IEEE, 2013.

[92] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[93] Bo Li, Chunhua Shen, Yuchao Dai, Anton van den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

[94] Minggang Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 716–723. IEEE, 2014.

[95] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[96] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1449–1456. IEEE, 2013.

[97] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[98] Daniel E Clark, Chee Sing Lee, and Sharad Nagappa. Single-cluster phd filtering and smoothing for slam applications. *target*, 32:34, 2012.

[99] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[100] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[101] Christopher M Bishop. Pattern recognition and machine learning. *Company New York*, 16(4):359–422, 2006.

[102] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. *AAAI*, 1982.

[103] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millenium*, 2003.

[104] Ananth Ranganathan, Michael Kaess, and Frank Dellaert. Loopy sam. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6–12, 2007.

[105] Keith YK Leung, Felipe Inostroza, and Martin Adams. A generalization of simultaneous localization and mapping with random finite sets. Submmited, 2015.

[106] Ronald PS Mahler, Ba-Tuong Vo, and Ba-Ngu Vo. Forward-backward probability hypothesis density smoothing. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(1):707–728, 2012.