Online Inference of Joint Occupancy using Forward Sensor Models and Trajectory Posteriors for Deliberate Robot Navigation

Kumar Shaurya Shankar

CMU-RI-TR-20-47

August 11, 2020

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Nathan Michael, CMU, *Chair*
Martial Hebert, CMU
Michael Kaess, CMU
Ali Osman Ulusoy, Microsoft

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

Robotic navigation algorithms for real-world robots require dense and accurate probabilistic volumetric representations of the environment in order to traverse efficiently. Sensor data in a Simultaneous Localisation And Mapping (SLAM) context, however, always has associated acquisition noise and pose uncertainty, and encoding this within the map representation while still maintaining computational tractability is a key challenge in deploying these systems outside of controlled laboratory settings.

The occupancy inference problem is essentially a high dimensional search in the space of all maps. By incorporating the physics of sensor formation using forward models, it is possible to reason in terms of the likelihood of the measurements for a given map hypothesis to obtain a solution that explains the noisy observations as well as possible. However, this approach to mapping has historically been prohibitively expensive to compute in real-time. Thus, conventional robotic mapping algorithms have primarily chosen to work with limiting assumptions to maintain tractability.

In this thesis we present a framework that explicitly reasons about the conditional dependence imposed on the occupancy of voxels traversed by each ray of a depth camera as a Markov Random Field (MRF). The tight intra-and inter-ray coupling explicitly incorporates conditional dependence of the occupancy of individual voxels as opposed to making independent log-odds Bayes filter updates as conventional occupancy maps do. Visibility constraints imposed by using a forward sensor model enables simplification of the otherwise high dimensional inference. The forward model allows incorporating learnt sensor noise characteristics for accurate inference. Instead of marginalising sensor data immediately, data from camera poses is retained, and can be added, moved, or removed in an ad-hoc fashion whilst performing inference. In order to avoid prohibitive sensor data storage costs, an extension to using the framework in a submapping setting with pose graphs is presented with sensor data marginalisation deferred until as late as possible. Marginalisation is performed using succinct parametric Gaussian distribution representations. Finally, Gaussian mixture model map representations are then demonstrated to be capable of providing robust localisation in multi-hypothesis settings. All of this is made real-time feasible by the inherent parallelisability of the proposed framework and is implemented on GPUs.

Acknowledgements

I would like to start off by thanking my committee, without whom this thesis would not exist. Martial, for his continued belief in me and for teaching me how to do good research first as an intern and later as a Master's student. Michael, for his constant support and discussions through the years in impromptu meetings and on research qualifier committees that we inevitably always found ourselves on. Osman, for so willingly and enthusiastically replying to my cold email regarding difficulties in implementing ray belief propagation algorithms. He has been so encouraging and excited through the process, and I owe him a great deal for his discussions and insights into this approach. And finally, Nate, who has patiently dealt with all my idiosyncrasies over the past six years, and provided pragmatic advice, instilled rigour, and supported me in achieving my childhood dreams.

No man is an island, and I wouldn't be where I am today without the people who've been on this life-journey with me. It is not possible for me to thank every single person who has contributed to me being who I am and this thesis, but I would be amiss to not mention the singular contributions of Arun, Humphrey, and Karthik, who have been the rocks that I have been able to anchor to all these years. Their advice and affection have truly helped me become both a better person and researcher. I would also like to thank Dey, who has been a constant friend, philosopher, and guide through all the tumultuous times and a shoulder I can always lean on. Thank you, Hatem, for selflessly helping me understand the true power of Lucas-Kanade. None of this would also not have been possible without the camaraderie and the long nights of collaborative work done with RISLab members past and present. Aditya, Alex, Arjav, Cormac, Derek, Ellen, Erik, John, Kshitij, Lantao, Logan, Lauren, Micah, Mike, Moses, Vishnu, Wennie, and Xuning, it has been quite the journey! The RISLab has also given me the opportunity to mentor younger researchers which has been some of the most rewarding experience of my graduate school life. Thank you Aditya, Bo, Linan, and Siddharth, for helping me learn more as I helped you.

CMU has been an institution that has left me with cherished friendships and experiences to reminisce about for a lifetime. So many colleagues and friends have left an indelible mark on me, and it is hard to acknowledge everyone in this limited space. Thank you Nick, Chris, Emily, Wen, Rogerio, Roberto, Achal, Dhruv, Ratnesh, Rohit, Daniel, Ele, George, Matt, Ben, Shervin, Ming, Ada, Keene, Victoria, and Kate. Thank you also to my colleagues at my two homes within the RI - Smith Hall, and the Field Robotics Center.

Lastly, I'd like to thank my family who've always supported and encouraged me and have been my biggest cheerleaders. This work is dedicated to my parents, who selflessly left no stone unturned to aid me in my endeavours.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Probabilistic algorithms have served as the cornerstone on which robotics has moved from the realm of controlled environments to that of the dull, dirty, and dangerous. With the democratisation of depth sensing cameras, it is possible for mobile agents to use low cost off-the-shelf cameras for autonomous navigation and exploration. These cameras however often have poor noise characteristics that makes it imperative for navigation algorithms to explicitly reason about map uncertainty for optimal traversal.

Forward models are the way we physically instrument and compute sensor uncertainty. Fundamentally, they describe the statistical behaviour of sensor measurements given the presence of matter existing at a certain location. Mapping, then, is a problem of inverting this problem - given sensor data that has noise, what is the most likely hypothesis of the environment that explains the measurements obtained?

Consider an occupancy grid comprised of N voxels with a binary label of occupancy assigned to each voxel. Given sensor measurements the total number of states then for this representation to naïvely perform inference over is 2^N. Due to this combinatorial complexity conventional robotic mapping techniques make simplifying assumptions to make this inference feasible. Specifically, by dropping the correlations between multiple measurements of a single map element, and utilising so-called *inverse* sensor models, these techniques break down the problem to an inference over $2N$ possible states. These inverse sensor models often require environment specific parameter tuning, and because of the incorrect assumptions involved, lead to suboptimal maps.

However, clearly there is structure imposed by the forward sensor model on the map, that, if exploited, permits inference in the original high dimensional space. As shown in the seminal work by [84] this is possible to do, specifically for sonar data, and provides more accurate maps but at a prohibitive computational cost. More recently, multi-view

geometry approaches to inferring occupancy have introduced ray belief propagation for camera sensors that explicitly model intra-and-inter-ray occupancy dependency via factor graphs but that have also been expensive to evaluate in real-time [46, 87]. In concert with this different strategy that lends itself well to modern GPU computing architectures the previously intractable map occupancy inference can now be feasibly computed in real-time. This leads to our formal thesis statement, which states that

Ray belief propagation enables online inference of joint occupancy using forward sensor models and trajectory posteriors for deliberate robot navigation.

Although we limit the scope of the discussion in this thesis to using RGB-D depth camera sensors commonly utilised to operate on Micro-Aerial Vehicles, the formulation can easily be extended to other beam-based sensing paradigms.

The key contributions of this thesis are:

1. MRFMap, a volumetric inference framework that explicitly encodes forward sensor models using ray belief propagation to obtain a posterior over map occupancies and permits ad-hoc removal and addition of sensor data from any individual camera pose during inference;
2. Utilising a succinct Gaussian mixtures based parametric representation to estimate the posterior over trajectories;
3. An extension of the MRFMap framework to interoperate with this succinct parametric map representation; and
4. Real-time implementation of the above on a GPU.

A brief outline of the thesis is as follows:

Chapter 2 briefly introduces mapping in the context of full SLAM, discusses map representations based on inverse sensor models and presents a conventional forward sensor model obtained from first principles. Chapter 3 details the sensors we focus on in this thesis and the systems developed to collect and fit noise models to them. Chapter 4 presents MRFMap, our proposed framework, and details experimental evaluation on synthetic and real-world datasets. Chapter 5 presents a particle filter based framework that estimates the posterior over trajectories using a succinct parametric representation for robust localisation. Chapter 6 demonstrates the use of MRFMap in a submapping context, with a bridge to using the succinct parametric representation for dealing with increasing memory usage. Finally, Chapter 7 contains concluding remarks and discussions for future work.

Chapter 2

Background

The main focus of this thesis is presenting a volumetric occupancy inference framework that employs forward sensor models and can be used within a full SLAM context for robot autonomy. This chapter serves as a brief introduction to this context, discusses how conventional mapping algorithms fall short, and sets up the probabilistic intuitions for using forward sensor models from a first principles setting.

2.1 Mapping as a subset of the probabilistic full SLAM problem

The full SLAM problem involves representing the joint distribution of the trajectories of a robot sensor and the map representation given all the sensor measurements obtained until the current time t:

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}), \tag{2.1}$$

where $\mathbf{x}_{1:t}$ are the poses $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_t$, $\mathbf{m}$ is the map representation, and $\mathbf{z}_{1:t}$ are the sensor measurements $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_t$ corresponding to each of those poses.

It is worth noting that in the case of mobile robots, as opposed to visual SLAM, we have an additional source of information about motion from the controls applied to the robot - this is often incorporated as a probabilistic update on a one step future pose given the robot kinematic model

$$p(\mathbf{x}_t | \mathbf{u}_{t-1}, \mathbf{x}_{t-1}) = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathcal{N}(0; \sigma_{noise}), \tag{2.2}$$

where $\mathbf{u}_{t-1}$ is the control input applied for the time interval time $t-1$ and the evolution of the state is modelled as being distorted by Gaussian noise with standard deviation σ_{noise}.

Thus, the desired distribution to be obtained can then be expressed as

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}). \tag{2.3}$$

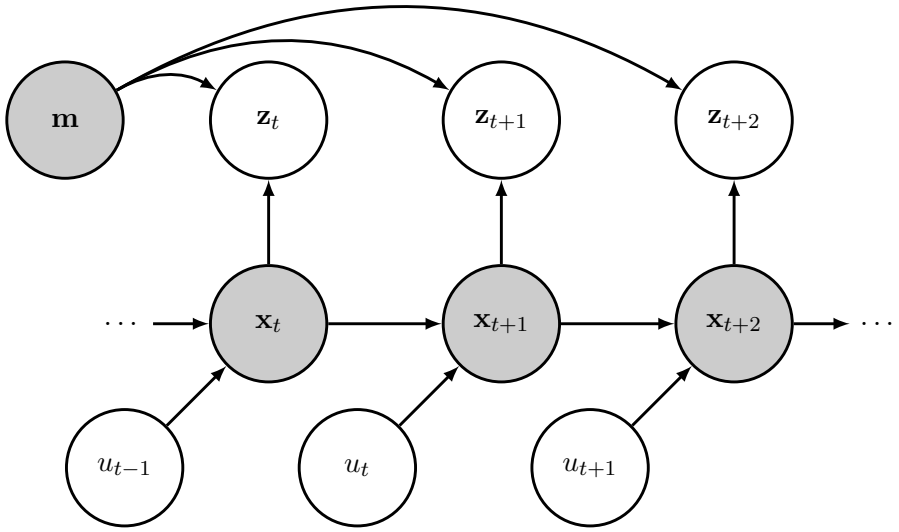


Figure 2.1: Graphical model of the full SLAM process. The map generates observations z_t for each robot pose x_t based on some subset m_i (not visualised) of the map $\mathbf{m}$. Control inputs u_{t-1} at each time step provide further information about the evolution of the state via a kinematics function. The map is constituted by a number of components that are conditionally independent given the measurements and the poses by d-separation [29].

However, obtaining this joint distribution is computationally prohibitive due to the combinatorial complexity of inference in the space of all possible map element occupancy states and trajectories, so robotic approaches have attempted to use various simplifying assumptions to make this inference tractable. For instance, by employing the Markov assumption that postulates that the past and the future states are independent given the current state, Bayes filters are employed to only use the previous state to update belief estimates of current state. These also exclude future measurements from updating the current state. Further approximations are made in terms of practical inference over these joints - multiple hypotheses are usually represented using particle filters, while smoothing and filtering approaches infer the unimodal maximum likelihood estimates of poses and cameras. The objective of this introduction to the SLAM problem is not to be comprehensive and the interested reader can purview works that have endeavoured to do so in the past [6, 12, 18].

A significant property of the full SLAM problem is that the map estimates are condi-

tionally independent given the robot path [51, 85]. This also implies that correlations in the uncertainty among different map elements arise only through robot pose uncertainty. Graphically, this is shown in Fig. 2.1. This enables us to factorise the problem as

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}), \tag{2.4}$$

which is the product of the posterior over robot poses and that of the map given the poses and the sensor measurements. Thus, we can approach the joint inference by computing the posterior over trajectories and then given a sampled trajectory estimate infer its respective map estimate separately. This thesis is primarily focussed on the framework used to obtain the latter posterior which dovetails well with approaches that compute the posterior over the trajectories. We demonstrate a system to compute the posterior over trajectories with a separate map representation and a simple system utilising the map inference framework within a full SLAM context in Chapters 5 and 6 respectively.

2.2 Conventional Volumetric Mapping Approaches and Inverse Sensor Models

In order to deal with the high-dimensional map occupancy inference problem many historical SLAM algorithms have used sparse, feature-based map representations, starting from the very first Extended Kalman Filter SLAM techniques [72] leading to modern-day sparse and semi-dense visual SLAM algorithms [22, 49]. Many more sophisticated techniques, such as using likelihood fields [11], signed distance fields [94], and parametric primitive based mapping [30, 40, 66] also exist and serve the purpose of localisation well. Although many representations exist for high fidelity surface mapping and SLAM, in this section we focus the discussion on volumetric probabilistic map representations that can be used directly within planning that explicitly represent free as well as occupied space.

In order to speed up the mapping inference, conventional mapping approaches can be considered instances of Bayes filters where two implicit assumptions are made. The first is the static world assumption, which states that the past sensor readings are conditionally independent given the map

$$p(z_t | \mathbf{z}_{1:t-1}, \mathbf{m}) = p(z_t | \mathbf{m}). \tag{2.5}$$

This is a reasonable assumption, as the map occupancy does not change in static environments. We also work on the same assumption. However, the other strong assumption that

is employed in these approaches is the independent map element assumption which states that the measurements are independent given an individual map element m^i

$$p(z_t|\mathbf{z}_{1:t-1}, m^i) = p(z_t|m^i). \tag{2.6}$$

Note that the map element could be a 3D feature, a surfel [80], a NDT-OM [65] Gaussian distribution, or an occupancy voxel. This is an incorrect assumption since sensor measurements clearly couple multiple map elements, especially with discretised representations, all of which need to be known for the conditional independence to hold. Intuitively, nearby sensor measurements share implicit information and considering them independent adds falsely confident information. However, making this assumption allows inverting this relationship to infer the occupancy of each map element independently of the others given all the sensor measurements, and thus reducing the complexity of the inference significantly [21].

Occupancy Grids [21] are the canonical representation used for robotic mapping. These involve binary labelled cells that denote occupied or unoccupied regions of the map. A Bayes filter is used to update the probability of occupancy for each cell independently. Hornung et al. [38] introduced the OctoMap framework that uses octrees as an efficient data structure to implement variable resolution occupancy grids. Variants such as Normal Distribution Transform Occupancy Maps (NDT-OM) [65] store a Gaussian density within each cell to capture a more precise measure of the occupancy distribution. Schulz et al. [68] present a real-time version of the same and add occlusion aware updates. Similarly, approaches exist that perform per cell filtering [67] that can be used to filter incoming sensor scan end points. Although the fidelity of the reconstruction in these approaches is better than equivalent occupancy grid maps, they suffer the same drawbacks with independent cells not sharing information.

Gaussian Process Occupancy Maps [55] and their variants such as [42, 92] have been employed for estimating continuous occupancy maps by casting occupancy inference as a classification problem. By exploiting implicit structure in the world correlated sparse sensor measurements are used to reason about unobserved regions of space. Additionally, these maps can also encode sensor data and pose uncertainty. However, a major drawback of these approaches is the high computational complexity and memory usage that is prohibitive for real-time operation with dense 3D data, despite recent attempts to address the same [34, 64]. There has also been work that attempts a particle filter based SLAM strategy using Hilbert maps in 2D environments [89] but the results are inferior to conventional SLAM techniques in terms of accuracy.

Recent work on combining Truncated Signed Distance Field (TSDF) and Euclidean Signed Distance Field (ESDF) [58] presents a representation that is catered for SLAM and planning. However, TSDFs are generated by using geometric weighted averaging over the raycasted endpoint estimates and are only concerned with representing surfaces. This limits their applicability in reasoning about occlusions, and specifically regions behind observed surfaces. Recently Vespa et al. [91] introduced a novel hybrid representation that combines TSDFs and occupancy grids that although doesn't combine information from multiple rays explicitly, does use an implicit forward sensor model.

These approaches, while still making the static world assumption, can be thought of as attempting to perform updates similar to the independent cell assumption in grid-based techniques by imposing local structure on the sensor endpoints where each measurement only updates a small region around it. Although practically effective, another drawback of these approaches is that they limit their applicability to environments with solid surfaces and have no way to reason about translucency or limited visibility.

All these approaches effectively marginalise out the pose information of the sensor to only use the end point samples instead of reasoning continuously over entire ray lengths and different perspectives. Intuitively, reasoning about measurements as rays instead of endpoints provides more information that they discard. Further, incorporating loop closures elegantly is not feasible for any of the above mentioned approaches because of the marginalised sensor pose information. However, they do permit incremental updates of the map that enable them to be practically efficient.

The impact of these decisions results manifests in multiple ways for these representations. In the case of grid based techniques, maps end up blurring out detail [84] or clearing out shallow grazing sensor measurements [38]. In the latter set of representations, increasingly noisy sensor data ends up blurring out the details in relation to the spatial constraints imposed which may or may not reflect the underlying geometry. Further, due to the strong underlying surface structure and high object density implicit assumptions, approaches such as TSDFs and NDT-OMs perform well in structured environments with flat, planar surfaces, but struggle in large outdoor environments with foliage, or non-regular structure [15].

We defer discussion of forward sensor model related occupancy mapping work to Chapter 4 and instead we next detail how a conventional forward sensor noise model for camera based sensors is formulated.

2.3 Forward Sensor Models for Occupancy Inference

In order to incorporate forward sensor noise models within an occupancy grid and to reason about the underlying model utilised in a wide variety of ray potential based literature, we look at deriving a simple Bayesian model of sensor readings as obtained by a depth camera. This presentation largely follows the approach presented in [33].

2.3.1 Formulation

The objective is defined as inferring the occupancy of a grid volume given camera poses and sensor measurements. To solve this problem, we note that a necessary requirement is to solve a joint distribution of a number of random variables. At the top of the causality chain is occupancy of space itself - the sensors measuring surface properties are predicated on the existence of matter itself. For the purpose of robust navigation through the environments we only need to explicitly calculate the marginal for the occupancy probability for the space, which is thus going to be the distribution in focus. All that remains then is to obtain the forward sensor model for an individual ray to calculate this occupancy probability.

We discretise the space into an occupancy grid. The occupancy of any cell is a binary variable $o \in \{0, 1\}$ signifying an empty cell or not, respectively. Given a camera location $\mathbf{x} \in \mathbb{SE}3$, the corresponding depth image pixel measurement is denoted by the random variable d. In an ideal environment the distance of the first occupied voxel along the ray would be the generating surface for the measurement. However, since the measurement could potentially originate from locations before a known occupied voxel, we model the latent unknown generating surface location by a variable η. This process is shown graphically in Fig. 2.2. Given this graphical model, the joint distribution is

$$p(o, \mathbf{x}, d, \eta, \tau) = p(\tau)p(o|\tau)p(\mathbf{x}|\tau)p(\eta|o)p(d|\mathbf{x}, \eta). \tag{2.7}$$

2.3.2 Priors

The random variable τ allows us to encode any domain specific knowledge

- $p(\tau)$ represents the prior probabilities of our parameters. Since there is no reason to favour any particular parameter value, we set it to a uniform distribution and disregard it for inference.
- $p(o|\tau)$ is the prior likelihood for occupancy. This could utilise any heuristic or guess we may have regarding occupancy (say, e.g. from CNN based long range depth

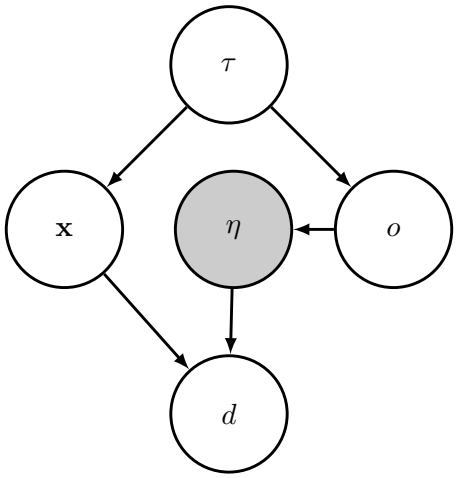


Figure 2.2: Graphical model of first principles modelling. A prior node τ encodes prior information for all the primary parameters. Given the current occupancy o, latent measurement generating surface along the ray η, and camera pose $\mathbf{x}$ we can estimate the likelihood of a given depth image pixel d being measured.

estimates). For the current example since we don't have any reason to favour a particular voxel location we set this to uniform as well.

- $p(\mathbf{x}|\tau)$ is the prior for the camera pose. This could utilise estimates say from a trajectory planner from an active camera configuration (e.g. a minimum jerk trajectory for quadrotors) given past camera locations. For the current example, we assume this pose to be known.

2.3.3 Inference

Given the joint distribution, we can estimate the marginal occupancy probability at each cell given the value of the known variables and marginalising over the latent variable

$$p(o|\mathbf{x}, d, \tau) = \frac{p(o, \mathbf{x}, d, \tau)}{p(\mathbf{x}, d, \tau)} = \frac{\sum_\eta p(o, \mathbf{x}, d, \eta, \tau)}{\sum_{\eta,o} p(o, \mathbf{x}, d, \eta, \tau)} = \frac{p(d|o, \mathbf{x})}{\sum_o p(d|o, \mathbf{x})}. \tag{2.8}$$

The term in the numerator is the forward sensor model and is discussed in the next section.

Depth Image Forward Sensor Model

The depth image sensor directly observes the distance to the occupied cell and hence in an ideal scenario there would be a direct edge from the occupancy probability to the measurement. However, following the process highlighted in [33], to account for the possibility of the sensor measurement originating from any cell along the ray we consider a latent variable η to model the generating surface distance.

In order to obtain the depth likelihood for a particular pixel value d from the depth image we need to marginalise the generating surface η. Specifically,

$$p(d|o, \mathbf{x}) = \frac{1}{n_x} \int_{d_{min}}^{d_{max}} p(\eta|o) p(d|\mathbf{x}, \eta) d\eta, \tag{2.9}$$

where d_{min} and d_{max} are the minimum and maximum ranges for the depth sensor, respectively, and n_x is the appropriate normalizing factor.

- $p(\eta|o)$ is the probability of the measurement generating surface existing at a distance η from the camera origin along the ray given the knowledge of an occupancy cell state. Accordingly we can model this corresponding to the two possible states of the given cell.

  The event that a given occupancy cell is empty provides no information regarding the location of the measurement generating surface. Thus any valid location along the ray is assigned a uniform probability $1/range$ of being the measurement generating surface, where $range = d_{max} - d_{min}$. Thus,

  $$p(\eta|o = 0) = 1/range \tag{2.10}$$

  Next, we consider the event that the occupancy of the cell o is occupied. If $o = 1$, then clearly the measurement generating surface cannot exist beyond its corresponding depth distance d_o. Note that this is valid only for sensors that follow strict visibility constraints, i.e., the sensor only observes surfaces. The measurement could still originate from a surface closer to the sensor, and so those events are assigned the same uniform probability $1/range$. In order to make the distribution sum to 1 over the domain, an appropriate Dirac delta term is specified for the depth corresponding

to the known cell d_o.

$$p(\eta|o=1) = \begin{cases} 0 & \text{if } \eta > d_o \\ \delta(1-(d_o - d_{min})/range) & \text{if } \eta = d_o \\ 1/range & \text{if } \eta < d_o \end{cases} \tag{2.11}$$

- $p(d|\mathbf{x}, \eta)$ is the depth measurement term that determines how likely a measurement is given that the originating surface exists at η. This is represented by a standard normal distribution centred around η to represent sensor measurement noise

$$p(d|\mathbf{x}, \eta) = \mathcal{N}(d|\eta, \sigma), \tag{2.12}$$

  where σ can be trained from a calibration process.

To gain an intuitive understanding of what this sensor model does, by substituting Eq. 2.9 in Eq. 2.8, we obtain the corresponding so-called *inverse* sensor model as illustrated in Fig. 2.3. The resulting sensor model clearly probabilistically spreads the measurement information along the ray taking into account the specified forward model.

2.3.4 Comparison with Conventional Inverse Sensor Models for Occupancy Grids

As mentioned in the introduction, inverse sensor models attempt to obtain the probability of matter existing for any given voxel location given a measurement. When dealing with ray based measurements, this ends up effectively spreading probability mass for voxels along a given sensor ray based on the sensor reading, independent of any other voxel or ray. These models require parameter adjustment and fine-tuning to closely approximate the inverse sensor model obtained above. Further, parameters fit for a particular environment may not work as well for others [38]. Some approaches exploit domain knowledge of stereo depth uncertainty to closely fit to the ideal model derived in the previous section [3, 35]. By accounting for the forward sensor model via the stereo depth uncertainty they incorporate the spread of the error to a higher fidelity than regular OctoMap (see Fig. 2.4), but are still limited by the choice of the simplifying independence assumption made in Eq. 2.6 to infer map occupancy. Specifically, although they can model intra-ray relationships better than vanilla occupancy grid sensor models, they still cannot model inter-ray data, a problem that we address in this thesis in Chapter 4.

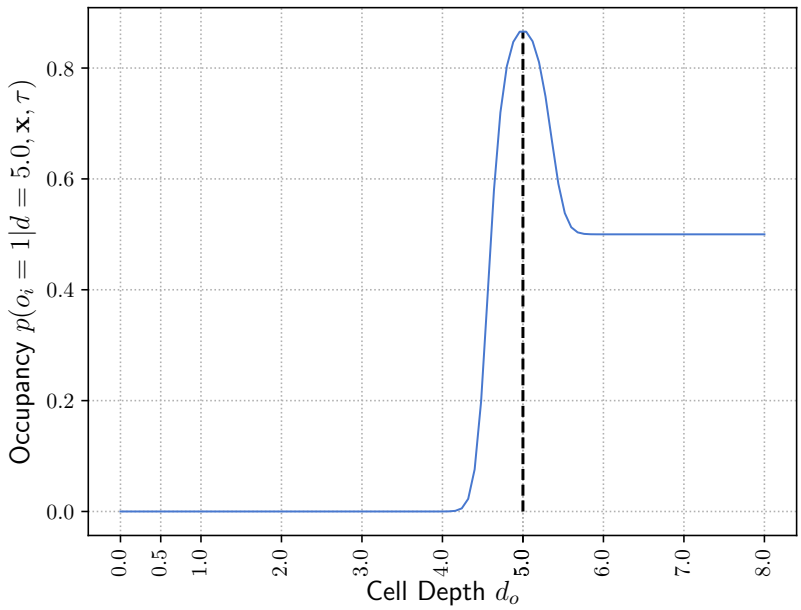


Figure 2.3: Inverse sensor model generated using the forward sensor model for a measurement of 5m with a $\sigma = 0.2$m and minimum sensor range of 0.5m. As expected, the probability rises to a maximum value at the measured sensor reading, and then falls to a completely uncertain value afterwards, since the measurement gives us no information about what exists beyond the occupied boundary.

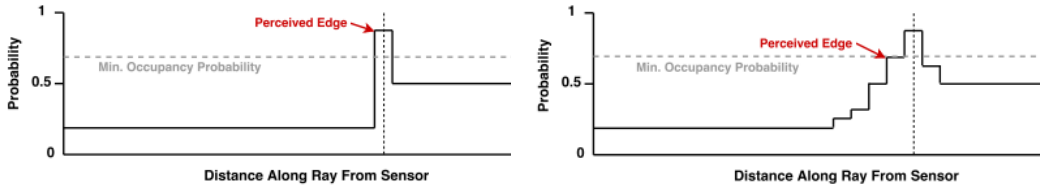


Figure 2.4: Conventional inverse sensor models have parameters for pass and hit that have to be fine tuned to get appealing results. This figure shows the resulting map occupancy after a single raycast using the standard inverse sensor model. Left: OctoMap [38] and Right: a fine tuned variant for stereo RGB sensors [35]. The perceived depth then along this ray is the edge of the first voxel above a certain occupancy probability threshold in the case of occupancy grids. These can both be seen as approximations to the continuous inverse sensor model obtained in Fig. 2.3. Figure from [58].

2.4 Conclusion

Having seen the full SLAM context and a comparison of conventional inverse sensor models with a true posterior inverse model from a first principles forward sensor model for beam based sensors, we detail the process and model we use to characterise the depth measurement term for the sensors that we use for this thesis in the next chapter. We will then incorporate this sensor characterisation and introduce inter-ray coupling within the mapping inference framework in Chapter 4.

Chapter 3

Characterising Bias and Variance Forward Models for RGB-D Cameras

In this thesis we are motivated to provide accurate occupancy inference from sensor data obtained via cheap off-the-shelf RGB-D depth sensors that are now ubiquitous and can be easily deployed on mobile robots. We first briefly highlight dominant sources of error in these sensors, describe our chosen model to fit the depth measurement term within a forward sensor model 2.12, and finally detail the experimental setup to fit this model from sensor data.

3.1 Sources of depth error for RGB-D Cameras

Commercially available off-the-shelf RGB-D cameras primarily fall under two broad categories. Structured light cameras such as the Kinect and Realsense operate on the basis of active stereo-matching that exploit near-infrared pattern projectors. These, along with passive stereo cameras form the first category of available depth sensors. The other are Time of Flight (ToF) cameras that are a lot more similar to laser rangefinders in that they have active components that measure depth using modulated emitted light radiation. Correspondingly, the two sensor types have different error characteristics.

For stereo cameras and structured light cameras, the error uncertainty is determined by transforming the error in computing disparity in pixel space to backprojected 3D space. For these cameras, the backprojected 3D point coordinates for a pixel u, v given a baseline b, focal length f, principal point coordinates c_u, c_v, and disparity measurement δ are obtained

as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{bf}{\delta}\mathbf{K}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{bf}{\delta}\begin{bmatrix} \frac{u-c_u}{f} \\ \frac{v-c_v}{f} \\ 1 \end{bmatrix}, \tag{3.1}$$

where $\boldsymbol{K} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix}$ is the pinhole camera projection matrix. Differentiating the third row gives us the relationship

$$\Delta z = \frac{bf}{\delta^2}\Delta\delta, \tag{3.2}$$

where Δz denotes the uncertainty in the backprojected depth given $\Delta\delta$, the uncertainty of the disparity measurement δ. The latter is conventionally set to 0.5 pixels to account for sub-pixel accuracy of disparity computation algorithms. Thus, the measured error uncertainty for these cameras should increase quadratically as a function of ground truth distance.

ToF sensors have systematic errors that stem primarily from the choice of modulation used in the emitted radiation, also referred to as wiggling error [27]. The magnitude of this error, however, is much smaller and is typically less than 1 cm [7]. Nevertheless, both types of sensors also suffer from non-linear radial distortion patterns that can be a function of distance as well [7]. Other factors such as angle of incidence, reflectivity, visual texture, and ambient illumination are also some of the primary contributors to sensor noise [53, 95].

3.2 Depth Measurement Model

Instead of fitting complex, camera-dependent distortion models, we choose to fit a per pixel block bias and variance model that is a function of depth. We choose to model the sensor bias and noise as a function of distance travelled along a ray. Although individually simple, computing distinct polynomials for each pixel block can capture a significant extent of sensor idiosyncrasies. We choose to directly learn the sensor bias and noise for every pixel region for all depths in the operating range of the camera. We collect measured and ground truth depths for pixel regions and fit a 2^{nd} degree polynomial as expected for RGB-D cameras. This choice has also been used in prior works [7]. The depth measurement term introduced in Eq. 2.12 for a depth measurement d_p at pixel p in the U, V pixel block is then expressed as

$$p(d_p|\mathbf{x}, \eta_p) = \mathcal{N}(d_p|f_{U,V}(\eta_p), g_{U,V}(\eta_p)), \tag{3.3}$$

where $f_{U,V}$ and $g_{U,V}$ are 2^{nd} degree polynomials fit per pixel block indexed by U and V.

Thus, given a set of input depth images from a RGB-D sensor, our objective is to determine per-pixel block depth bias and noise characteristics that can then be incorporated in occupancy inference.

3.3 Fitting the Depth Measurement Model

In order to fit the above mentioned sensor bias and variance polynomials, we need to obtain ground truth depth estimates for each pixel block over the entire operating range of the cameras. In order to do this we first present the systems designed to acquire the ground truth data, and then detail the process for collecting the corresponding sensor and ground truth data.

3.3.1 Acquiring Ground Truth Data

In order to obtain accurate sensor noise characteristics, we need accurate spatio-temporal ground truth measurements corresponding to the sensor data. To obtain this, we utilise

1. A graph optimisation based pipeline for computing accurate extrinsics between ground truth motion capture frame and camera optical frame;
2. A simple utility for determining the temporal lag between data streams obtained on a collecting device between the ground truth odometry and the image stream; and
3. A real-time interactive utility to collect sensor data at discrete depth bins for blocks of pixels with corresponding ground truth for evaluating per-pixel block sensor noise characteristics for use in inference.

Computing Camera-Ground Truth Extrinsics

We use GTSAM[1] for pose graph optimisation with a custom factor that solves for the extrinsics of both the chessboard frame and the camera optical camera frame with respect to their respective Vicon frames. In order to evaluate the performance of the pose graph optimisation we evaluate the extrinsics calibration in a Gazebo[2] simulation with arbitrary extrinsics of increasing magnitude to determine the basis of convergence that the calibration can accommodate. As can be seen in Fig. 3.2, the calibration process could successfully recover the correct extrinsics up to a displacement of $T_{\text{cam}}^{\text{body}}$ norm of 0.6 for $T_{\text{tag}}^{\text{world}}$ norm up to 0.05 in metric units (meters for translation and radians for orientation). The codebase is available at `https://github.com/rislab/extrinsics_calibrator`.

[1] Georgia Tech Smoothing and Mapping Library - https://gtsam.org/
[2] Gazebo Simulator - http://gazebosim.org/

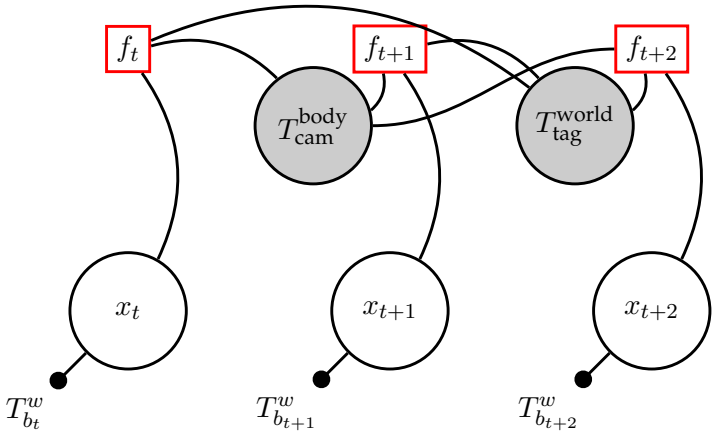


Figure 3.1: Factor graph schematic for extrinsic calibration. Multiple camera body poses x_t observe the static chessboard at pose $T_{\text{tag}}^{\text{world}}$ and generate corner pixel observations. Each of the body poses have a strong unary prior from motion capture. These pixel observations are used in custom factors f_t evaluating the error and the jacobians of the projection of the physical chessboard corners into the camera frame after being transformed by the corresponding two variables.

In line with the recommendations by the manufacturer[3], for the Realsense D435 we use the left infrared global shutter camera since it provides a global shutter image, and is pixel aligned to the stereo depth image that the ASIC provides.

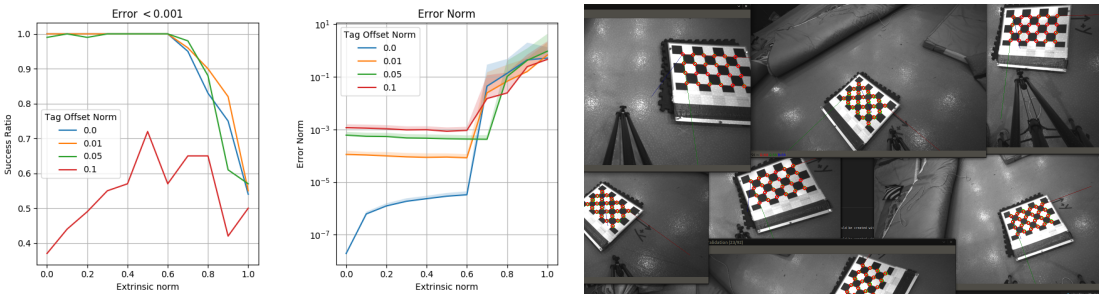


Figure 3.2: Left: Evaluating performance of extrinsics solver over 100 samples of randomised extrinsics drawn for each tag offset value. Right: Reprojected chessboard corner points through transform chain.

[3] https://dev.intelrealsense.com/docs/tuning-depth-cameras-for-best-performance

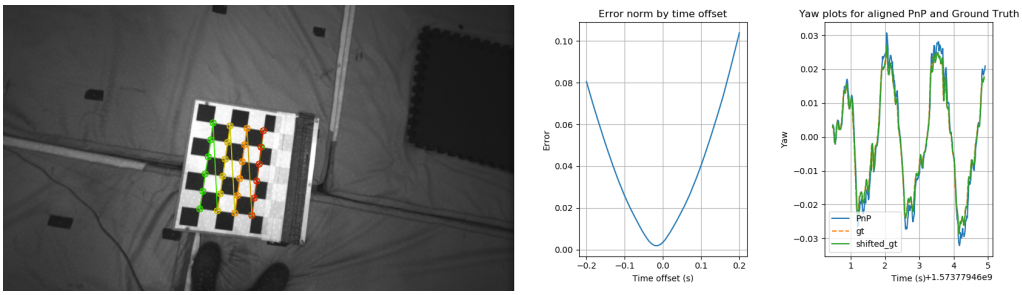


Figure 3.3: Estimating temporal lag between image stream and Vicon motion capture odometry stream. By comparing the yaw profile of PnP and Vicon poses, a simple linear interpolation search helps determine the offset to sufficient precision for alignment.

Determining Temporal Lag

Streaming image data coming via camera drivers generally has some real-world latency due to sensor characteristics and transmission delays. Similarly, Vicon motion capture data coming over the network also has some latency introduced by packet transfer. For the purpose of data collection and calibration, assuming a constant aggregate latency for both the streams, it is important to align the two data streams for correct association. To perform this, we record data from the global shutter camera at the highest framerate at the chosen resolution (848×480) at 90 Hz and perform a yaw rotation about the camera Z axis inside the motion capture arena whilst looking at a chessboard target. By comparing the yaw values from Perspective-n-Pose (PnP) and those obtained from the Vicon stream a simple linear search allows determining the temporal lag between the two streams to sufficient precision.

New Vicon Driver

The release of the latest Vicon SDK under an MIT license enables operation of the Vicon driver on ARM platforms such as the TX2, which wasn't possible earlier. Further, the SDK exposes new lower latency variants of accessing streaming pose data in the ServerPush mode. To exploit these characteristics, we have created a much simpler Vicon package to stream Vicon data at `https://github.com/pumaking/simple_vicon`.

Data Acquisition Process

The camera is held static on a tripod while a chessboard is moved around in the view frustum of the depth camera. The projected calibration target dimensions in the image are used to create a valid masked depth image. For every valid pixel in the masked depth

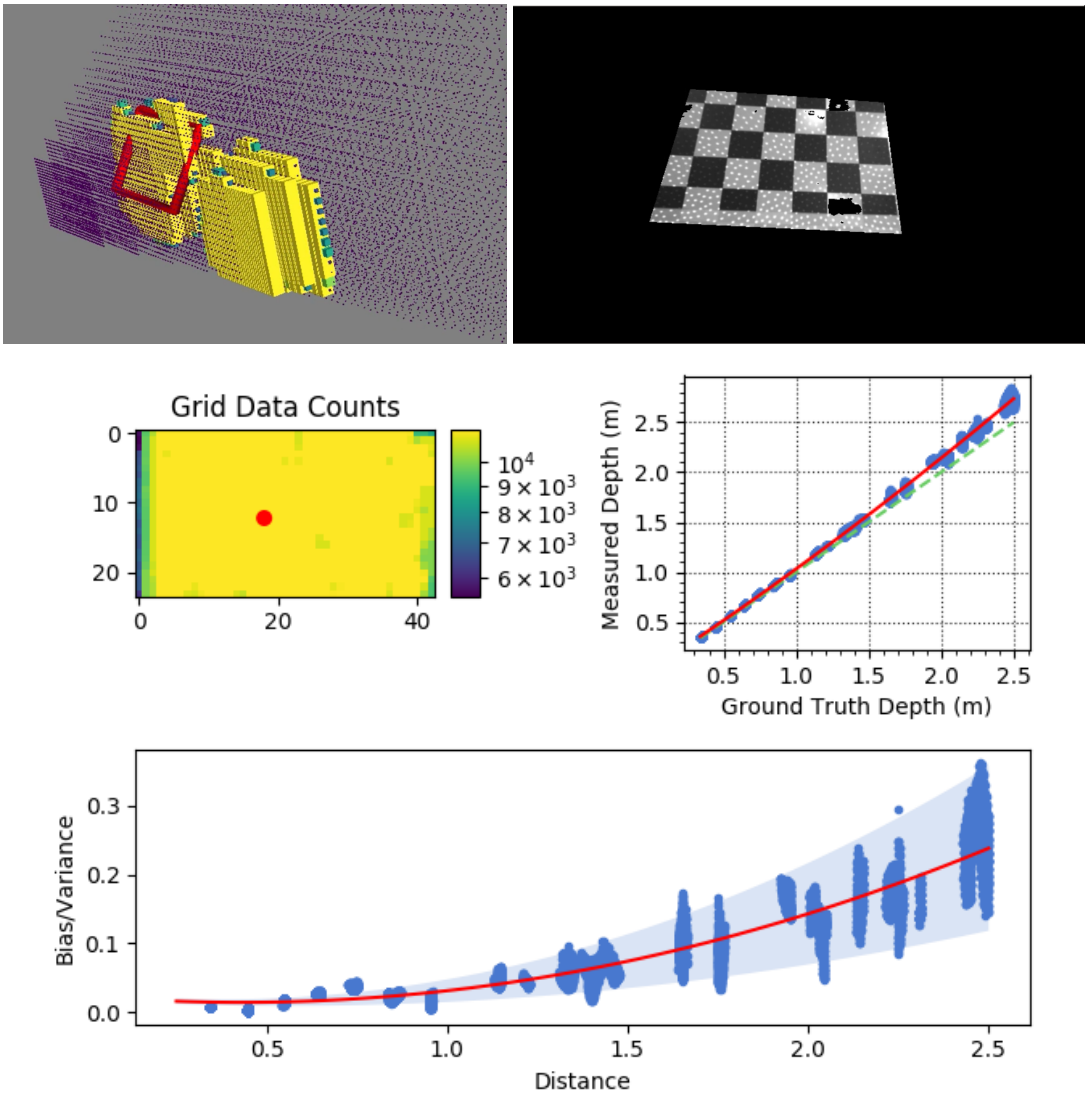


Figure 3.4: Top Left: A real-time utility that visualises the mocap calibration target location and the relevant slices of the view frustum for which sufficient data has been collected for learning the forward sensor noise model. Top Right: The segmented depth image of the chessboard using the mocap pose estimates of the board and the camera. Bottom: After collecting up to the order of 10^4 data points for each valid 20×20 pixel block, the measured depth is regressed against the ground truth depth to fit a polynomial sensor noise model and the residuals are used to regress the bias and variance. Shown for a Realsense D435 operating at 848×480 px resolution.

image, the corresponding ground truth location is determined by performing a ray to plane intersection test using the relative transform obtained from the relevant mocap frames. Thus, each masked depth image provides a large number of ground truth and measured depth values per pixel location. This data is then binned over small configurable pixel neighbourhoods that are then used to fit bias and noise models (Fig. 3.4).

3.4 Results

We collect data in two batches – for near-field and far-field data. First is in the 0.25-2.5m region, where we use a smaller chessboard, and the other is from 2.5-5m where we use a larger chessboard. In order to guide the data acquisition process we use a utility that visualises the amount of data collected for each 0.5m depth bin in real-time (see Fig. 3.4).

We also show overall meta trends of the bias and variance of the raw data compared to the ground truth. These are generated by taking the mean of the bias and standard deviation images in a central patch of pixels. The bias and standard deviation images are generated by computing the mean and the standard deviation for each 20×20 pixel block in a 0.5m window around the queried planar distance. These agree with prior work on calibrating this class of cameras [7]. The comparison for the two sensors can be seen in Fig. 3.7.

3.4.1 Kinect One

The Microsoft Kinect One is a Time of Flight (ToF) camera that can output dense RGB-D data at Full HD resolution (1920×1080) at 30 Hz. In practice however, we use it mostly used in the QHD configuration (960×540) with the depth image rectified to the downscaled colour image. The raw IR depth camera has a resolution of 512×424 px with a FoV of roughly 70.6×60 degrees, while the colour camera has a resolution of 1920×1080 px and a FoV of 84.1×53.8 degrees.

3.4.2 Realsense D435

The Intel Realsense D435 is an IR stereoscopic depth camera that can output dense RGB-D data at up to 1280×720 px at 15 Hz. We use it at the VGA resolution (640×480 px) in the high precision preset at 30 Hz. The depth camera has a FoV of roughly 87×58 degrees and the RGB camera has a FoV of 69.4×42.5 degrees.

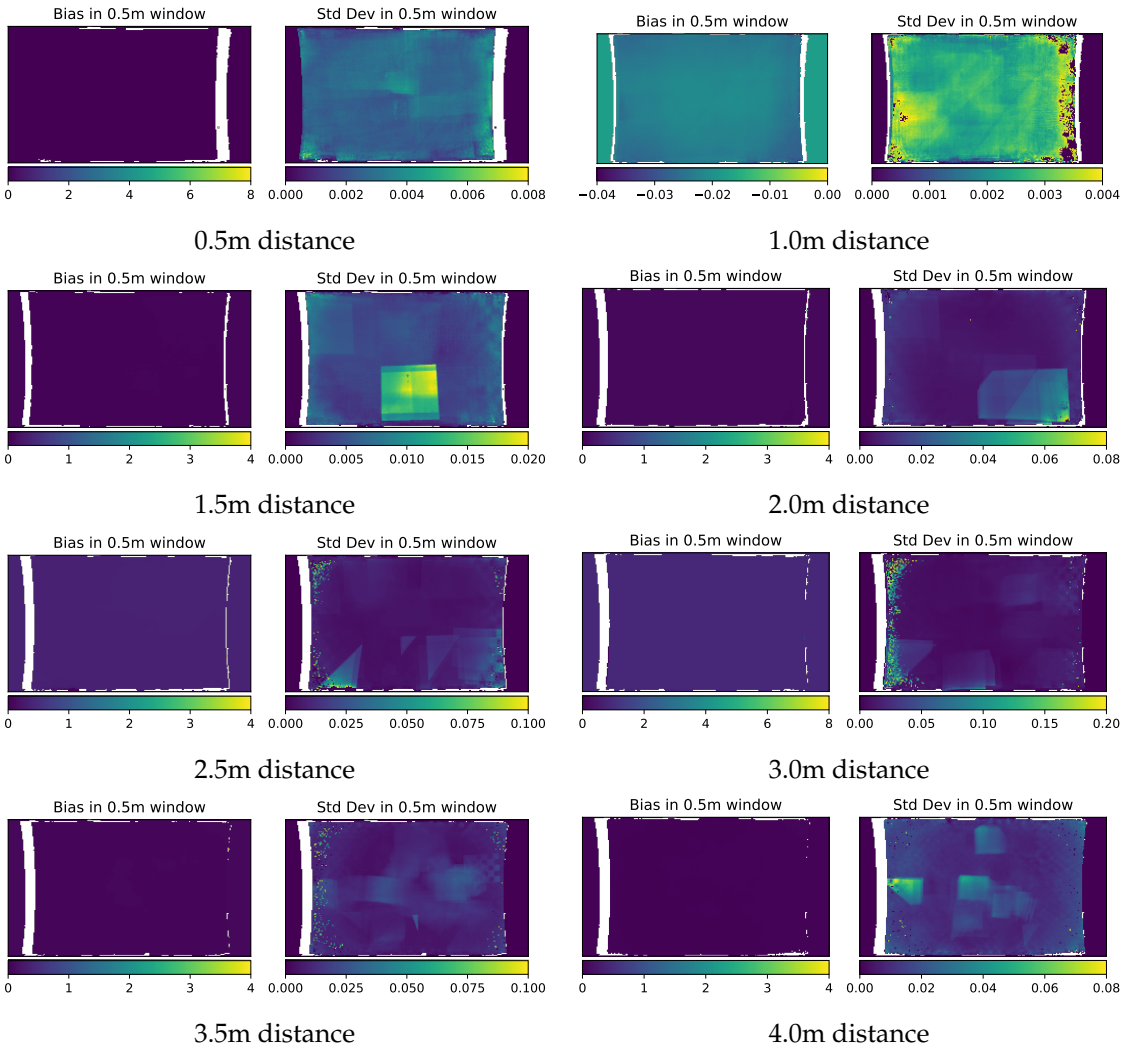


Figure 3.5: Bias and variance plots of the collected characterisation data at 0.5m increments collected over a 0.5m window around the sample planar depth for the Kinect One.

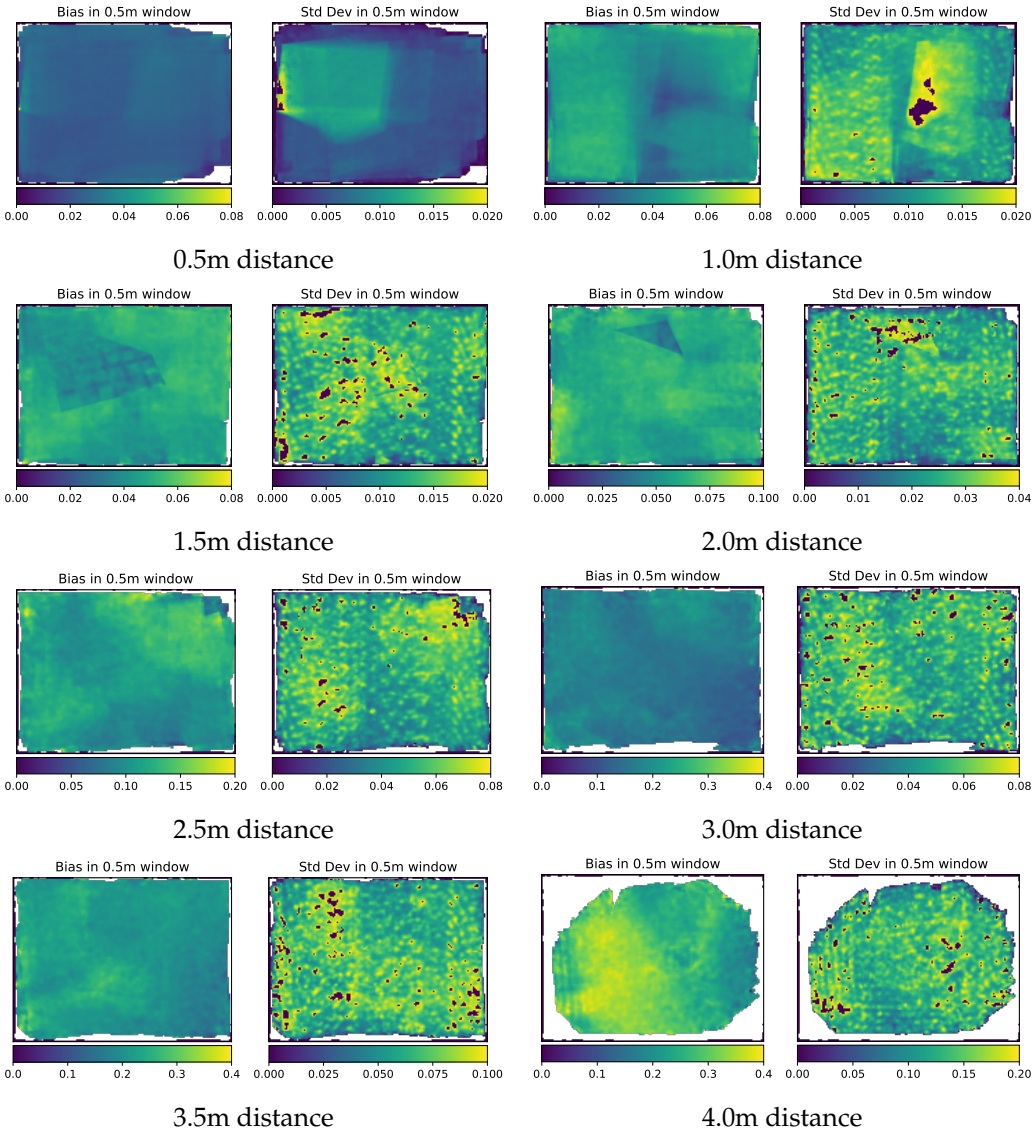


Figure 3.6: Bias and variance plots of the collected characterisation data at 0.5m increments collected over a 0.5m window around the sample planar depth for the Realsense D435.

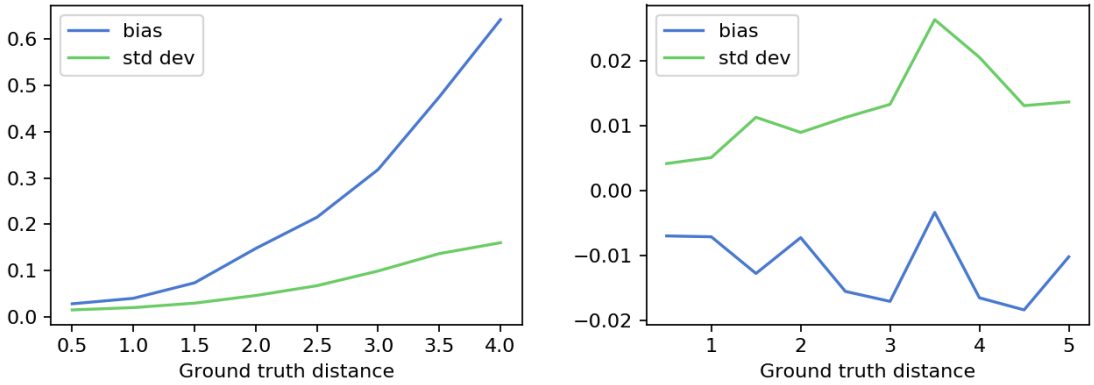


Figure 3.7: Meta level characteristics of the bias and standard deviation for the Realsense D435 and the Kinect One on the left and right, respectively. Note that the characteristics of the D435 camera follow a quadratic profile whereas those of the Kinect One are almost linear.

3.5 Conclusion

In this chapter we described the per block sensor bias and noise depth measurement model we use within the inference framework described in the next chapter for a pair of off-the-shelf RGB-D depth cameras. The meta trends of these sensors agree with prior published work on calibration for this family of sensors, and the per-pixel block capture evident asymmetric noise characteristics in the respective sensors.

Chapter 4

MRFMap: Online Probabilistic 3D Mapping using Forward Ray Sensor Models

This chapter presents the primary framework of this thesis that infers the posterior distribution over map occupancy given a trajectory hypothesis using forward sensor models as presented in the introduction. Specifically, it provides the second term in

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) \underbrace{p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t})}_{\text{This chapter}}. \tag{2.4}$$

4.1 Introduction

The seminal work by Thrun [84] argued that the occupancy mapping problem is essentially a high dimensional search in the space of all maps and should be tackled as such. By reasoning about the physics of sensor formation using forward models it is possible to reason in terms of the likelihood of the measurements for a given map hypothesis. The sensor data then directly drives a solution that explains the noisy observations as well as possible given the forward sensor model. However, this approach to mapping has historically been prohibitively expensive to compute in real-time.

We present a framework that explicitly reasons about the conditional dependence imposed on the occupancy of voxels traversed by each ray of a depth camera as a Markov Random Field (MRF). The tight intra-and inter-ray coupling explicitly incorporates conditional dependence of the occupancy of individual voxels as opposed to effectively

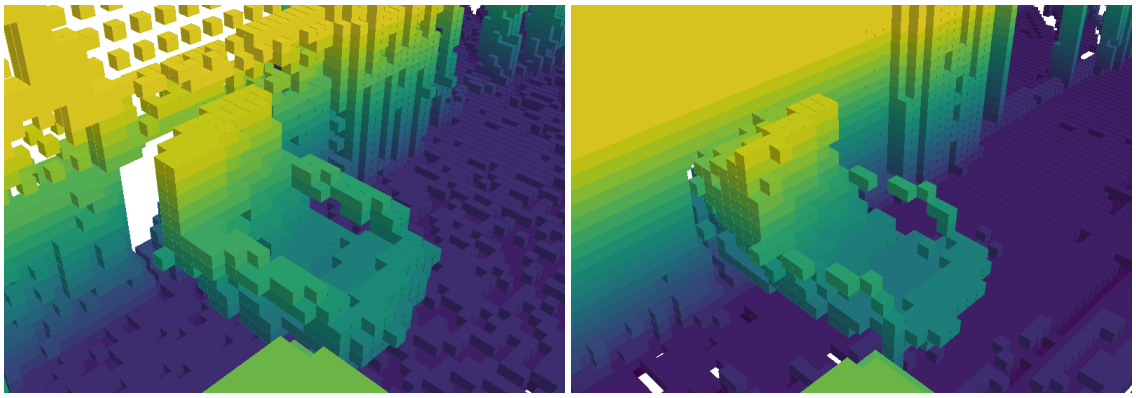


Figure 4.1: In the presence of sensor noise, or glancing rays, standard occupancy grid methods (OctoMap [38], Left) end up blurring out the map details and clearing occupied space, respectively. An MRFMap (Right) reasons about sensor ray formation and explicitly couples voxels both within each ray and between rays. Inset from Fig. 4.8 from maps inferred using the augmented ICL-NUIM dataset [13] at 0.05 m resolution. Occupied voxels coloured by height to highlight morphological differences.

marginalising out the sensor data in the form of log-odds updates as occupancy maps do. Visibility constraints imposed by using a forward sensor model enables simplification of the otherwise high dimensional inference. The forward model allows incorporating learnt sensor noise characteristics for more accurate inference. Finally, the inherent parallelisablity of both constructing the MRF and inferring it enable real-time performance on GPUs.

Our key contributions are:

- A Markov Random Field (MRF) based 3D occupancy grid framework that explicitly couples all voxels ray traced based on visibility;
- Incorporation of learnt sensor noise characteristics for improved map fidelity compared to occupancy grids;
- A principled evaluation metric for probabilistic occupancy maps; and
- An open-sourced[1] real-time GPU implementation.

4.2 Map Representations using Forward Models

We have previously discussed existing conventional volumetric probabilistic map representations that can be used directly within planning and explicitly represent free as well as occupied space in Sec. 2.2. Here we discuss mapping approaches related to our desired

[1] https://mrfmap.github.io

objective.

Confidence-rich grid mapping [1] aims to enhance occupancy grids by encoding uncertainty within the grid and coupling voxels by reasoning about forward sensor models. The Bayes filter updates turn out to be identical to the ones used in [63], and thus share similar properties as with the MRFMap framework. The key difference between their approach and ours is that our framework allows for obtaining occupancy marginals based on poses being added or removed ad-hoc rather than relying on recursive Bayes filter updates, provides varying levels of resolution fidelity, and also permits the addition of visual data for enhancing inference, if required. Factor graphs and forward sensor models were presented in [17] to tackle the occupancy inference problem with inference performed using belief propagation and dual decomposition but was prohibitive to compute in real-time. Recently Vespa et al. [91] introduced a novel hybrid representation that combines TSDFs and occupancy grids that although doesn't combine information from multiple rays explicitly, does use an implicit forward sensor model.

The theoretical foundations for performing inference using ray belief propagation trace their origins to methods that recover shape from silhouette cues [28, 33]. More recent work [46, 87, 88] use similar probabilistic foundations to come up with generative models of appearance and occupancy representations from multiple image views. However, for all these frameworks, occupancy is a latent variable that is not directly observed. Our framework, although heavily inspired by Ulusoy et al. [87, 88], utilises depth images as opposed to using grayscale camera images. Since the variable of interest is directly observed, inference is much faster and possible to do in real-time. Further, we augment the formulation to incorporate learnt sensor noise to enable higher mapping fidelity.

4.3 Theory

We focus on sensors whose measurements can be modelled as ray measurements that return a single distance. For such sensors the uncertainty in angle is negligible in comparison to the uncertainty in distance, permitting the ray approximation instead of volumetric tracing. We also assume independence of rays in order to handle the updates separately for each ray. For the purpose of this work we present utilising the ray belief propagation within a fixed size volumetric grid. However, it is important to note that the underlying spatial map parametrisation can be anything that permits easy and consistent ray traversal evaluation from multiple perspectives and indeed we demonstrate initial results with using Gaussian distributions within the framework.

We first present a brief introduction of depth ray potentials followed by the inference

procedure, and finally the extension to incorporate learnt forward sensor model uncertainty.

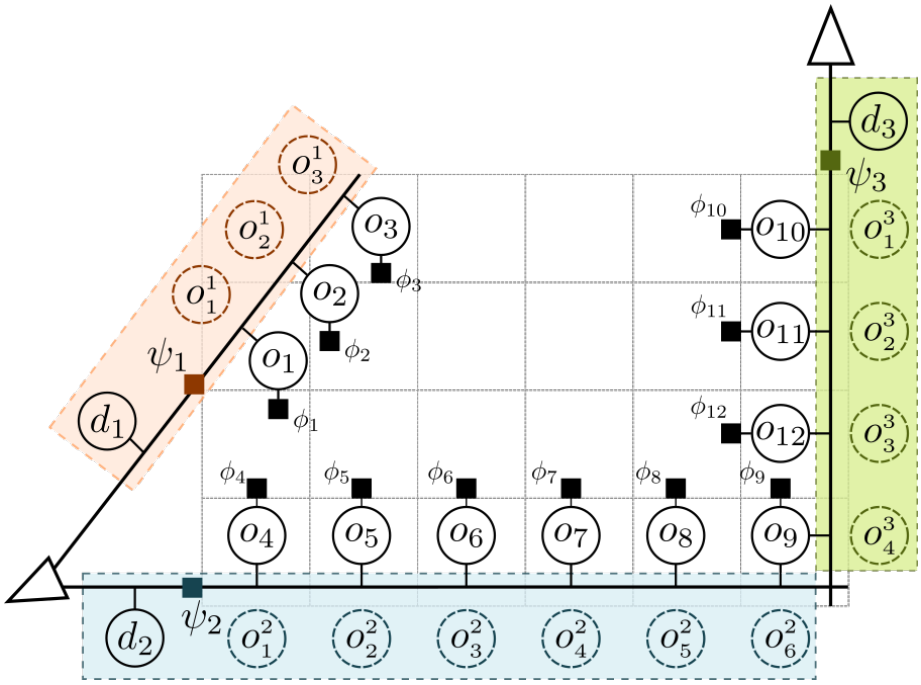


Figure 4.2: Graphical model illustration of the MRFMap framework. Each ray traversing through the voxels o_i establishes a depth potential ψ_r that connects all the occupancy nodes o_i^r it traverses through and an auxiliary depth variable d_r. Each voxel has a prior depth potential ϕ_i. Voxels connected by multiple rays (such as o_9) share information across the graph. Voxels within a shaded region are the $\mathbf{o}_r$ voxels for that particular ray.

4.3.1 Ray Markov Random Field

A Markov Random Field (MRF) is a bipartite graph containing nodes corresponding to variables and factors that are connected by edges. Under mild conditions this undirected graph can be factorised over its cliques into a factor graph, which is what we use as the actual underlying formulation, but retain the MRF language for consistency with prior work [87, 88]. Consider the ray MRF formation process as shown in Fig. 4.2. Each ray from the camera sensor generates a ray potential that associates all the cells traversed by the ray. The cells are signified as nodes that can have a binary label $o_i \in \{0, 1\}$ signifying whether they are unoccupied or occupied, respectively. Similar to Ulusoy et al. [87], we specify a variable d_r associated with the ray to represent the event that the depth of the first occupied cell along the direction of the ray is d_i^r (for o_i^r, $i \in 1 \dots N_r$ in increasing distance

along a ray). However, we do not model the appearance of the voxels along the ray, since unlike their work, we have direct access to depth information. The joint distribution of this factor graph is

$$p(\mathbf{o}, \mathbf{d}) = \frac{1}{Z} \prod_{i \in \mathcal{X}} \phi_i(o_i) \prod_{r \in \mathcal{R}} \psi_r(\mathbf{o}_r, d_r), \tag{4.1}$$

where $\mathcal{X}$ is the set of all the voxels $o_i \in \{0, 1\}$ corresponding to them being empty or occupied respectively, and $\mathcal{R}$ is the set of all the rays from all the cameras viewing the scene, $\mathbf{o}_r = \{o_1^r, \ldots, o_{N_r}^r\}$ is the list of all the voxels traversed by a ray r, d_r is its corresponding depth variable, and Z is the normalisation constant. The total set of all the occupancy and depth variables are summarised as $\mathbf{o} = \{o_i \mid i \in \mathcal{X}\}$ and $\mathbf{d} = \{d_r \mid r \in \mathcal{R}\}$. ϕ_i and ψ_r are the potential factors as described as follows:

Prior Occupancy Factor

This is simply a unary factor assigning an independent Bernoulli prior γ to the voxel occupancy label for each voxel

$$\phi_i(o_i) = \gamma^{o_i}(1 - \gamma)^{1 - o_i}. \tag{4.2}$$

In our experiments we use a non-informative prior however note that these per-voxel priors can be informed from per-pixel predictive methods if desired.

Ray Depth Potential Factor

A ray depth potential creates a factor graph connecting the binary occupancy variables o_i^r for all the voxels traversed by a single ray by virtue of making a measurement. Each of these voxels has a corresponding distance from the camera origin, and thus the ray is defined to include a depth variable d_r that represents the event that the measured depth is at distance d_i^r. For this to happen all the preceding voxels ought to be empty, i.e., have the value 0, and the corresponding voxel needs to be occupied, i.e., have a value 1. This leads to the definition of the joint occupancy and depth potential factor

$$\psi_r(\mathbf{o}_r, d_r) = \begin{cases} \nu_r(d_i^r) & \text{if } d_r = \sum_{i=1}^{N_r} o_i^r \prod_{j<i} (1 - o_j^r) d_i^r \\ 0 & \text{otherwise} \end{cases}. \tag{4.3}$$

Here $\nu_r(d_i^r)$ denotes the probability of observing the measured depth measurement $\mathcal{Z}_r$ if it originated from the latent depth variable d_i^r that we model as

$$\nu_r(d_i^r) = \mathcal{N}(\mathcal{Z}_r; d_i^r, \sigma(d_i^r)). \tag{4.4}$$

This is our forward sensor model. This is based on the nature of RGB-D sensors, that are more precise compared to sonars that require different forward sensor models [84]. Similar to Basso et al. [7] we model this probability to vary in mean and noise as a function of the distance from the camera, which enables utilising learnt sensor noise characteristics for most RGB-D sensors for accurate map inference.

This ray potential measures how well the occupancy and the depth variables explain the depth measurement $\mathcal{Z}_r$. This is more apparent when Eq. 4.3 is written as

$$\psi_r(\mathbf{o}_r, d_r) = \begin{cases} \nu_r(d_1^r) & \text{if } d^r = d_1^r, o_1^r = 1 \\ \nu_r(d_2^r) & \text{if } d^r = d_2^r, o_1^r = 0, o_2 = 1 \\ \vdots & \\ \nu_r(d_N^r) & \text{if } d^r = d_N^r, o_1^r = 0, \\ & \dots, o_{N_r-1}^r = 0, o_{N_r}^r = 1 \end{cases}. \tag{4.5}$$

This sparse structure of the ray potential enables simplification of the message passing equations by reducing the inference to a linear pass along the ray instead of exponential steps as detailed in the following subsections.

4.3.2 Sum-Product Belief Propagation

Sum-Product belief propagation [45] is a common message-passing algorithm for performing approximate inference on cyclic factor graphs. By exploiting marginalisation of joint distributions using factorisation of a graph it enables computing marginal distributions very efficiently. Messages are passed back and forth between connected nodes and factors that try to influence the marginal belief of their neighbours. Although convergence is not guaranteed, in practice since the depth images are mostly consistent due to the relative accuracy of depth sensor measurements oscillations are rare and we obtain a solution quickly [44, p. 429].

The message sent from a variable node x to a factor f is the cumulative belief of all the

incoming messages from factors to the node except the factor in question

$$\mu_{x\to f}(x) = \prod_{g\in\mathcal{F}_x\setminus f} \mu_{g\to x}(x), \tag{4.6}$$

where $\mathcal{F}_x$ is the set of neighbouring factors to x. Similarly, the message sent from the factor to the node is the marginalisation of the product of the value of the factor ϕ_f with all the incoming messages from nodes other than the node in question

$$\mu_{f\to x}(x) = \sum_{\mathcal{X}_f\setminus x} \phi_f(\mathcal{X}_f) \prod_{y\in\mathcal{X}_f\setminus x} \mu_{y\to f}(y), \tag{4.7}$$

where $\mathcal{X}_f$ is the set of all neighbouring nodes of f.

Upon convergence, the estimated marginal distribution of each node is proportional to the product of all messages from adjoining factors

$$p(x) \propto \prod_{g\in\mathcal{F}_x} \mu_{g\to x}(x). \tag{4.8}$$

Similarly, the joint marginal distribution of the set of nodes belonging to one factor is proportional to the product of the factor and the messages from the nodes

$$p(\mathcal{X}_f) \propto \phi_f(\mathcal{X}_f) \prod_{\mathcal{X}_f} \mu_{x\to f}(x). \tag{4.9}$$

4.3.3 Messages to the variables

Ray Depth Potential to Depth Variable Messages

Since we are only concerned about the depth variable, we marginalise out the messages from all the occupancy nodes going to the ray depth potential. Following Eq. 4.7 we have

$$\mu_{\psi_r\to d_r}\left(d_r = d_i^r\right) = \sum_{o_1^r} \cdots \sum_{o_{N_r}^r} \psi_r\left(\mathbf{o}_r, d_r\right) \prod_{j=1}^{N_r} \mu_{o_j^r\to\psi_r}\left(o_j^r\right). \tag{4.10}$$

Naïvely evaluating this equation would involve 2^{N_r} evaluations for each ray (corresponding to the two possible event states for each o_j^r). However, we can exploit the sparse nature of the ray depth potential to recursively simplify this expression. After dropping the ray index for notational convenience and abbreviating $\mu_{o_j^r\to\psi_r}(o_j^r)$ as $\mu(o_j)$ we obtain (see

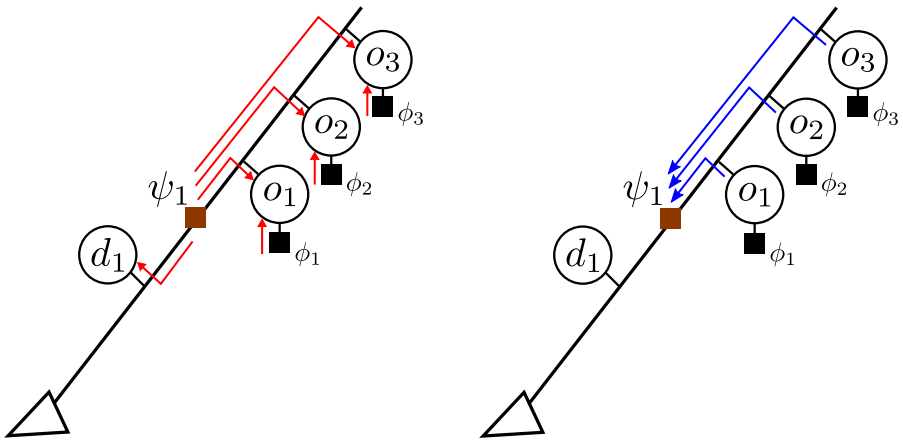


Figure 4.3: Inference scheme involves first sending messages from all the factors to the nodes, and then from the nodes to all the factors that are connected to them. This process is repeated three times in our implementation. Shown here are what the message pathways look like for a single ray potential.

Appendix B for a detailed derivation)

$$\mu_{\psi \to d}\left(d = d_i\right) = \nu(d_i)\mu(o_i = 1)\prod_{k<i}\mu(o_k = 0). \tag{4.11}$$

Depth Variable to Ray Depth Potential Messages

Since the only factor connected to the depth variable is the ray depth potential itself, the message $\mu_{d_r \to \psi_r}$ is uninformative and, following Eq. 4.6, it is set to a uniform value.

Ray Depth Potential to Occupancy Variable Messages

Similar to the depth variable messages, we marginalise out all the variables except the occupancy node in question

$$\mu_{\psi_r \to o_i^r}\left(o_i^r = 1\right) = \sum_{d_r}\sum_{\substack{o_j^r \\ j \neq i}} \mu_{d_r \to \psi_r}(d_r)\psi_r\left(\mathbf{o}_r, d_r\right)\prod_{j=1}^{N_r}\mu_{o_j^r \to \psi_r}\left(o_j^r\right). \tag{4.12}$$

After simplification using the sparse structure of Eq. 4.3 and dropping the ray indices

for convenience we obtain (see Appendix B for a detailed derivation)

$$\mu_{\psi \to o_i}(o_i = 1) = \sum_{j=1}^{i-1} \mu(o_j = 1)\, \nu(d_j) \prod_{k<j} \mu(o_k = 0) + \nu(d_i) \prod_{k<i} \mu(o_k = 0) \tag{4.13}$$

and

$$\mu_{\psi \to o_i}(o_i = 0) = \sum_{j=1}^{i-1} \mu(o_j = 1)\, \nu(d_j) \prod_{k<j} \mu(o_k = 0) + \sum_{j=i+1}^{N} \frac{\mu(o_j = 1)}{\mu(o_i = 0)} \nu(d_j) \prod_{k<j} \mu(o_k = 0)\,. \tag{4.14}$$

To provide an intuition of what these messages do, the positive outgoing message in Eq. 4.13 sends a high value only if the likelihood of all the preceding voxels being empty is high (second term) after taking into account the possibility of each preceding voxel to be occupied (first term). Similarly, the negative outgoing message in Eq. 4.14 sends a high value if any of the subsequent voxels have a high likelihood of being occupied (second term) after taking into account the possibility of each preceding voxel to be occupied (first term). The first terms in both these equations are significant since they encapsulate the concept of visibility, i.e., as we traverse the ray through regions of occlusion, we can be less certain of the information the sensor measurement provides, and eventually the outgoing messages send out uninformative uniform messages.

Occupancy Variable to Ray Depth Potential Messages

Since other rays can (and often do) pass through the same occupancy variable node, the outgoing message $\mu_{o_i^r \to \psi_r}$ is computed as per Eq. 4.6. However, if a voxel is only traversed by a single ray, the only other factor that it receives messages from is the prior factor, ϕ_i, described in Eq. 4.2.

4.3.4 Evaluation Metric for Probabilistic Occupancy Maps

In order to determine the accuracy of a given probabilistic occupancy map, especially one that has been generated from noisy data, the best map hypothesis should be one that maximises the likelihood of the sensor data. Further, it is not simply enough to look at thresholded occupancy values of a map and determine if sensor ray endpoints lie within

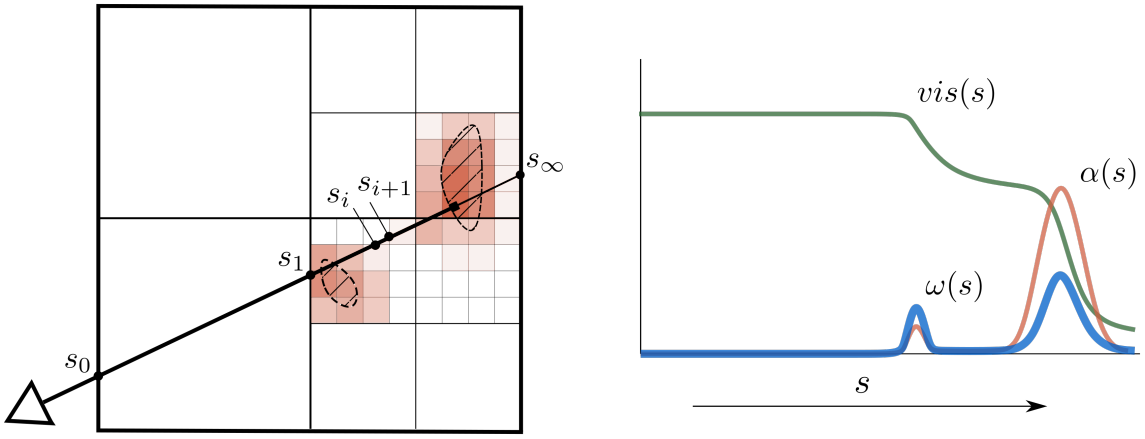


Figure 4.4: Piecewise continuous ray generating surface likelihood evaluation. For a camera ray parametrised by s, the probability of occupancy per voxel can be considered a region with a constant occlusion density $\alpha(s)$. The visibility decreases corresponding to the magnitude of $\alpha(s)$ and the length traversed within. The probability of an occluding surface existing at a distance s is $\omega(s)$, which is the product of the visibility and the map occlusion probability along the ray. Here, even though a voxel in the second region of the probabilistic map has higher probability mass of occupancy, it is obscured by a traversed region before it that reduces the likelihood of it being the generating surface for the ray depth measurement.

an occupied voxel, as done in [38]; one has to reason about visibility and occlusions by partially occupied voxels along the way. To approximate this, Stoyanov et al. [78] sampled negative free space to determine the fidelity of the reconstructed map. However, this is an ad-hoc approach and needs heuristics to determine what the region of the negative sampled distance along a ray should be. Pathak et al. [62] introduced a visibility based measure that reasoned about occlusion. However their model does not take into account the fact that rays traverse through unequal lengths within a voxel, allowing the choice of discretisation to significantly influence the likelihood. To fix this, a continuous version of the same is presented by Crispell [14] that models voxels as piecewise constant regions of occlusion density, which is what we present here in the context of probabilistic map evaluation.

As an analogue for the event that an occluding surface occurs at cell depth d_i, $\omega(s)$ is defined as the probability density function of the measurement generating surface along the ray length parametrised by a distance variable s. The probabilistic volume is reimagined as a region of occlusion density, $\alpha(s)$, that monotonically reduces the chances of the originating surface being at distance s as we move further along the ray (see Fig. 4.4).

The visibility at a distance s is defined as

$$\text{vis}(s) = e^{-\int_0^s \alpha(s')ds'} = e^{-\sum_{i=0}^{n-1} \alpha_i \ell_i + \alpha_n (s - s_n)}, \tag{4.15}$$

where α_i denote the piecewise constant occlusion density, s_n denotes the distance of the ray upon hitting the n^{th} voxel, and $l_i = s_{i+1} - s_i$ denote the lengths travelled within each voxel discretisation. This permits using voxels of differing dimensions while evaluating the accuracy. The difference between two consecutive visibility probabilities across a voxel boundary then is equivalent to an occlusion probability assigned to it

$$\Omega_i = \text{vis}_i - \text{vis}_{i+1}. \tag{4.16}$$

The likelihood of the originating surface being at a particular distance s is proportional to the rate of change in the visibility along the ray as it traverses a region. This leads to the definition

$$\begin{aligned} \omega(s) &= \frac{d}{ds}(1 - \text{vis}(s)) \\ &= \alpha(s) e^{-\int_0^s \alpha(s')ds'} \end{aligned} \tag{4.17}$$

$$\omega(s) = \alpha(s)\,\text{vis}(s). \tag{4.18}$$

Intuitively, even if there exist voxels along a ray with a higher probability of occupancy as per the map, the fact that the ray has already traversed through partially occluded regions previously reduce the probability that the measurement generating surface exists further along the ray (see Fig. 4.4). Note the similarity with the first terms in Eq. 4.13 and Eq.4.14.

We thus define a ray depth measurement as being accurately classified if the sensor measurement lies within the voxel boundaries of the voxel at which the maximum likelihood generating surface,i.e., the distance $\arg\max \omega(s)$ is found. On reaching the map boundary, if the visibility is not reduced sufficiently by the map volume, then the generating surface is determined to be accurate if the sensor measurement is greater than the distance to the map boundary, i.e., if $\mathcal{Z}_r > s_\infty$.

Taking into account that the map is going to be of a bounded volume, residual probability of visibility at the end of the ray is represented by vis_∞. In order to determine the likelihood of a ray given the probabilistic map we need to accumulate the evidence of the possibility of the occluding surface being at a distance s and the probability of s being the

true distance given the depth measurement $\mathcal{Z}_r$

$$\begin{aligned} p(\mathcal{Z}_r) &= \int_0^\infty p(s)p(\mathcal{Z}_r|s)ds \\ &= \sum_{i=0}^{S-1} \int_{s_i}^{s_{i+1}} \omega(s)p(\mathcal{Z}_r|s)ds + vis_\infty p_\infty(\mathcal{Z}_r), \end{aligned} \tag{4.19}$$

where vis_∞ is the residual visibility when a ray hits the edges of the map boundary and $p_\infty(\mathcal{Z}_r)$ is an experimentally obtained terminal probability depending on the ray measurement.

Since we model $p(\mathcal{Z}_r|s)$ as a continuous normal distribution (Eq. 4.4) although a closed form solution of the first term is feasible to compute, we make a simplifying assumption of computing its Riemann integral using the normal pdf values between the start and end point lengths of the ray within the voxel for computational reasons. The first term in Eq. 4.19 then simplifies to

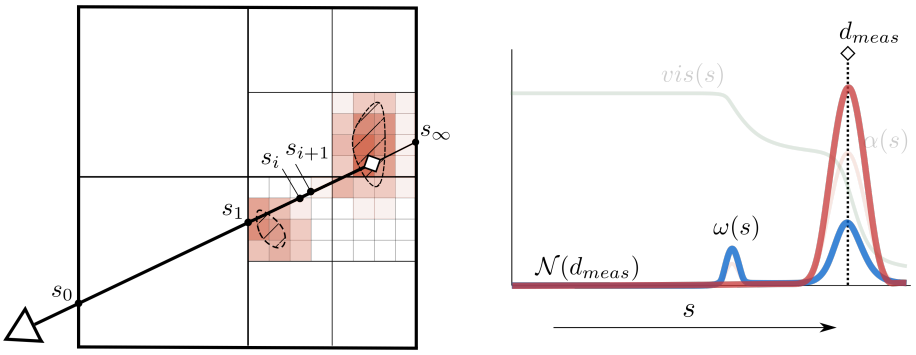
$$\int_{s_i}^{s_{i+1}} \omega(s)p(\mathcal{Z}_r|s)ds \approx \frac{\nu_r(s_{i+1}) + \nu_r(s_i)}{2}\Omega_i. \tag{4.20}$$

In order to determine $p_\infty(\mathcal{Z}_r)$, we follow the approach of Pathak et al. [62]

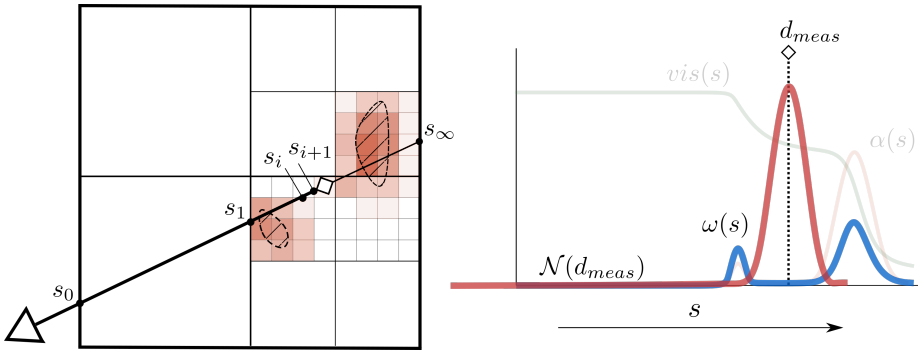
$$p_\infty(\mathcal{Z}_r) = \begin{cases} 1-\epsilon & r > r_{max} \\ \epsilon & \text{otherwise} \end{cases}, \tag{4.21}$$

where r_{max} is the maximum range of the sensor, or the intersection of the ray with the bounding volume, and ϵ is a small experimentally obtained value. Note the similarity of our likelihood to the expressions in the EM based inference for forward sensor model based mapping in [84]. Finally, due to the independence assumption of the rays, the total likelihood of an image is just the sum of the individual log likelihoods

$$p(\boldsymbol{\mathcal{Z}}) = \prod_{\mathcal{Z}_r \in \text{scan}} p(\mathcal{Z}_r). \tag{4.22}$$



(a) High likelihood measurement.



(b) Low likelihood measurement.

Figure 4.5: High likelihood vs low likelihood measurements. As a ray traverses an inferred probabilistic volume we can determine the probability of the location of the generating surface, $\omega(s)$. This can have multiple modes dependent on the map volume and the viewing perspective, as discussed earlier in Fig. 4.4. For an arbitrary pose, the likelihood of a ray measurement is the integral of the product of the probability of the measurement generating surface being at that location and the measurement probability given by Eq. 2.9 over the entire ray. This product will be high when there is significant overlap between these two probability density functions.

4.4 Implementation

4.4.1 Belief Propagation

We initialise all the occupancy variables with a Bernoulli initial distribution as in [88] with a probability of 0.1. The lower value corresponds to the observation that regions of interest are mostly empty. Lower prior probabilities aid in the initial inference steps since it permits the visibility terms to not saturate the outgoing messages. At inference time all keyframe images are ray traced and each ray atomically updates the corresponding nodes it traverses. We explicitly take into account the traversed length of each ray within a voxel to determine the contribution of the specific ray. The ray traversal is done via the standard 3DDA traversal algorithm [2]. These are then accumulated to obtain the current incoming belief and prepared for the next pass of inference. Since storing all incoming messages for each voxel is prohibitive, we make an assumption that all rays going through a voxel from a single keyframe send the same average message. Thus, for each new keyframe image we create an auxiliary buffer that stores the average outgoing message. For the purpose of this work we perform three sequential up and down passes of belief propagation between all the factors and nodes respectively which we have empirically observed to be sufficient for obtaining convergence.

4.4.2 Compact GPU representation

We extend and utilise the GVDB library [37], which represents a VDB [52]-like sparse data structure on GPUs. The smallest unit of allocated storage is a brick, which is a configurable size. In our implementation we chose it to be 8^3. Each level of the hierarchy then can contain a configurable number of bricks. Brick data is stored in contiguous space, enabling sparse data storage. The library allows for empty skip 3DDA ray traversals for regions where no data is stored. Fig. 4.4 demonstrates a pictorial 2D example of a $< 2, 2, 2 >$ configuration of the topology.

When adding new depth images, new bricks are allocated in a region around the reprojected 3D depth point cloud. This enables fast skip steps until within the neighbourhood of the ray measurement where the sensor model provides the most information. Once within an occupied brick, we traverse each voxel until a 3σ distance beyond the measured depth at which point the sensor model provides no more useful information.

4.4.3 Incorporating learnt sensor noise models

Structured light based RGB-D cameras and stereo cameras have a distinct quadratic increase in measurement uncertainty as a function of depth and often also have bias errors [7]. We choose to model the sensor bias and noise as a function of distance travelled along a ray. Other factors such as angle of incidence, reflectivity, and texture are also some of the primary contributors to sensor noise that are only partially addressed within this model and are out of scope to model in this work. Within our implementation for the forward sensor model (Eq. 4.4), at any given traversed length, we first utilise the sensor bias to predict what the mean sensor measurement would be and then evaluate the probability of a given measurement originating from this predicted measurement and distributed using the learnt noise value. Note that this is a significant difference to traditional approaches, where undistortion functions need to be computed [7] as a first step that are then provided to conventional mapping processes. In addition to incorporating it within the mapping process to reduce latency, directly using the learnt sensor noise characteristics can also easily account for other sensor or environment idiosyncrasies without doing any additional preprocessing. We choose a 20×20 pixel neighbourhood to trade-off noise model fidelity and memory usage on the GPU. Details about the noise modelling are presented in Chapter 3.

4.5 Evaluation and Results

The emphasis of our results here is that

- The MRFMap framework enables higher fidelity mapping than OctoMap [38], the standard robotic grid based volumetric occupancy framework, especially for noisy data; and
- The time taken for ray tracing and inference is much faster than CPU based approaches, especially at finer map resolutions.

Quantitatively, we measure the accuracy of the maps by utilising the metric discussed in Sec. 4.3.4. All pixels (rays) for which the distance of the maximum likelihood generating surface, s lies within a constant 1.5σ standard deviation of the measured depth are classified as accurate, and those that are not are classified as being incorrect. Sample accuracy images are shown in Fig. 4.7. The final accuracy score for a given depth image is then the sum of all pixels (rays) that are accurately explained by the map over the total number of valid depth pixels in the depth image. The accuracy score is then computed over all scans in the dataset and the mean and the standard deviation reported in the tables below.

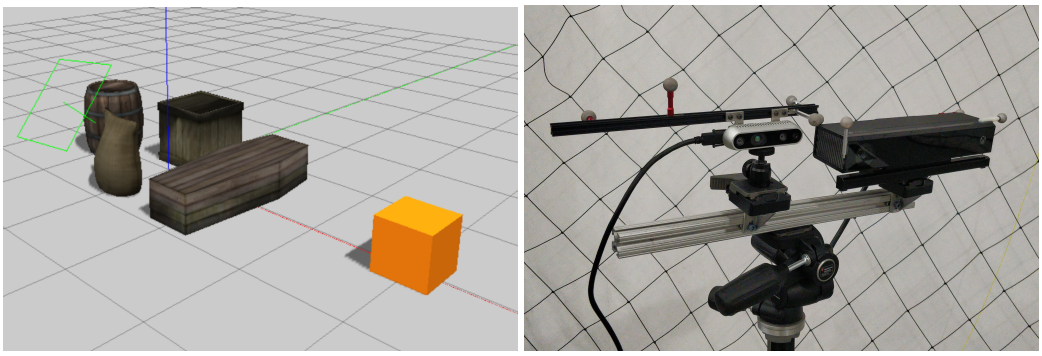


Figure 4.6: Left: We use a synthetic scene in the Gazebo simulator as viewed from 12 different simulated depth camera views of 640×480 px resolution. Right: Camera rig for a Realsense D435 and a Kinect One sensor used for acquiring the real-world dataset.

We detail three different scenarios

1. A simple simulated scene in the Gazebo[2] simulator to demonstrate the accuracy measure and ability to encode bias and noise modelling within inference;
2. A standard open access dataset with very precise simulated sensor noise for RGB-D sensors to demonstrate inference with simulated real-world noise; and
3. A real-world dataset where we use a noisy sensor (Realsense D435) and evaluate the generated map using a much more accurate sensor (Kinect One).

Demonstrating the accuracy metric

For the Gazebo simulation environment shown in Fig. 4.6 we obtain 12 ground truth ray traced images and then sample simulated depth images by utilising the learnt per 20×20 px patch bias and standard deviation noise models for the Realsense D435 at 640×480 px resolution. These simulated true noise images are used to infer the probabilistic maps and are then evaluated for accuracy in a leave one out cross-validation scheme. A sample view is shown in Fig. 4.7 and the corresponding accuracy is shown in Table 4.1. For a specific pixel, we plot the occupancy probability of the map p_{occ}, the visibility vis, and the generating surface probability $\omega(s)$. The peak of the $\omega(s)$ distribution is highlighted by a blue vertical line - thus, according to the accuracy metric the pixel would be classified as being accurate if the ground truth depth measurement at that pixel lies within 1σ of this distance. This region is shown as a purple translucent region. As can be seen in Fig. 4.6 for the given pixel the MRFMap predicted ground truth distance lies within this region, while that for the OctoMap does not. The dashed vertical line is the noisy measurement sampled

[2] http://gazebosim.org/

from the ground truth for the pixel that was used for inferring the map - highlighting the fact that incorporating the forward sensor noise and the bias within the inference allows to compensate for noisy, inflated depth measurements.

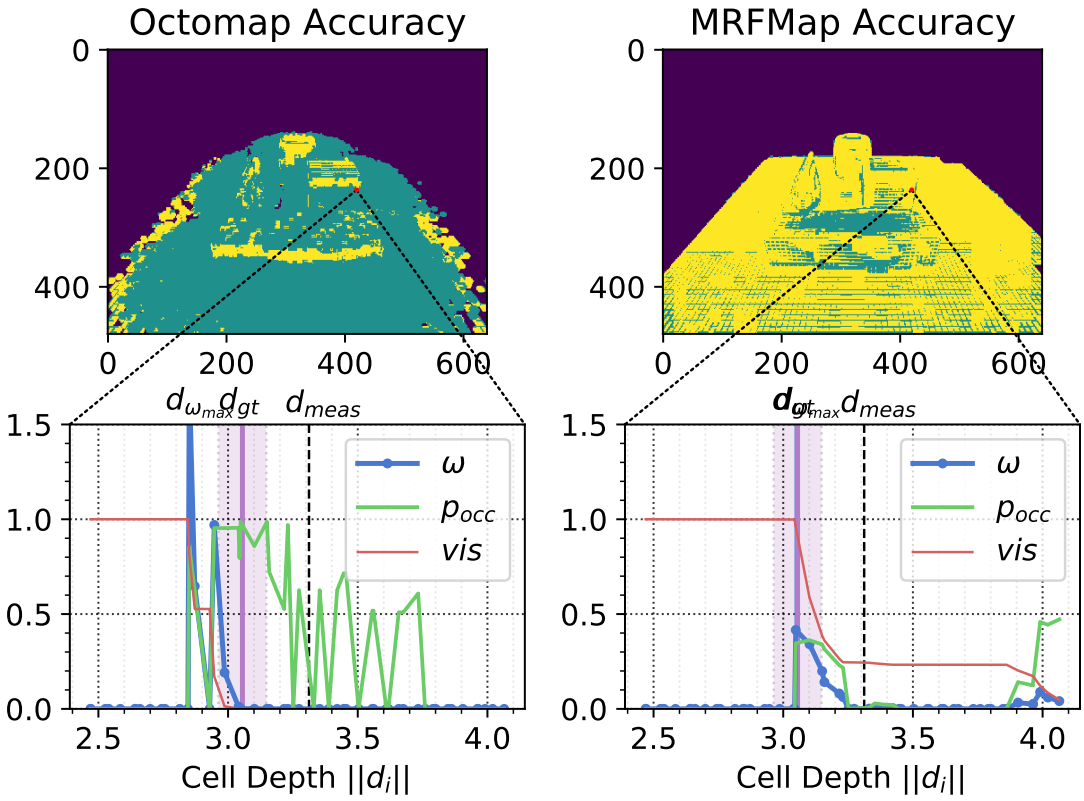


Figure 4.7: Computing accuracy with simulated learnt noise. Top: Accuracy images. Correctly predicted pixels are coloured yellow, incorrectly predicted pixels as green, and invalid pixels as purple. Insets: Plots of ray-traced $p_{occ}(s), \omega(s),$ and$vis(s)$ for the selected pixel. Purple vertical line represents the ground truth depth d_{gt}, with a corresponding 1σ band around it to determine if the maximum likelihood generating surface location is present within. Dashed vertical line is the depth measurement d_{meas} at the selected pixel in this held out sampled depth image for evaluation. Blue vertical line is the location of the maximum likelihood generating surface $d_{max} = \arg\max \omega(s)$(Sec. 4.3.4). Note how for the selected ray at the right edge of the cuboidal box, the MRFMap accurately describes the probability of occupancy along the ray (green). The ray first hits the cuboidal box, corresponding to the first peak of ω around 3.05 m and then travels through empty space till it hits the ground at a distance of 4.0 m. For the OctoMap, an incorrect over-confident map occupancy probability causes the visibility (red) to drastically drop and it incorrectly predicts the most likely generating surface.

Since the injected depth noise and bias is non-trivial, OctoMap struggles at accurately inferring the map, while MRFMap infers the map accurately, predicting the generating surface location for each pixel to be close to the ground truth depth, thus demonstrating the ability to compensate for injected bias and noise when correctly modelled.

Resolution	0.01m	0.02m	0.05m
MRFMap	0.870 ± 0.011	0.917 ± 0.022	0.843 ± 0.017
OctoMap	0.082 ± 0.010	0.118 ± 0.020	0.213 ± 0.049

Table 4.1: Accuracy values for the simulated ground truth environment using leave one out cross-validation over 12 images in a $4 \times 4 \times 3$ m^3 environment. Accuracy is determined based on the sensor likelihood metric presented in 4.3.4. Higher accuracy represents better agreement of inferred volumetric occupancy with ground truth sensor data.

Demonstration on publicly available dataset with known ground truth

For these results we use the `livingroom1` noisy depth sequence of the Augmented ICL-NUIM dataset [13] since it accurately models sensor noise characteristics of projective IR based RGB-D cameras. To select the keyframes to perform inference we use a simple geometric displacement based heuristic. A new keyframe is generated if the tangent norm of the translation or the rotation from the last keyframe pose is larger than a threshold (here 1.0 for each). To evaluate the accuracy we then use the ground truth depth images from the same trajectory. We use the Kinect noise model as reported in [7] for performing map inference, and a constant noise model for the evaluation. Accuracy results are presented in Table 4.2. As expected, at finer resolutions, the noise model can very precisely impact inference involving multiple voxels and thus provides much more accuracy than at lower voxel resolutions.

Resolution	0.01 m	0.02 m	0.05 m
MRFMap	0.945 ± 0.097	0.939 ± 0.101	0.891 ± 0.114
OctoMap	0.922 ± 0.105	0.882 ± 0.108	0.723 ± 0.113

Table 4.2: Accuracy means and standard deviations over the entire dataset for the `livingroom1` noisy depth sequence. The MRFMaps are constructed using the Kinect forward sensor noise model and evaluated on a constant noise model. Map volume is $10 \times 10 \times 5$ m^3.

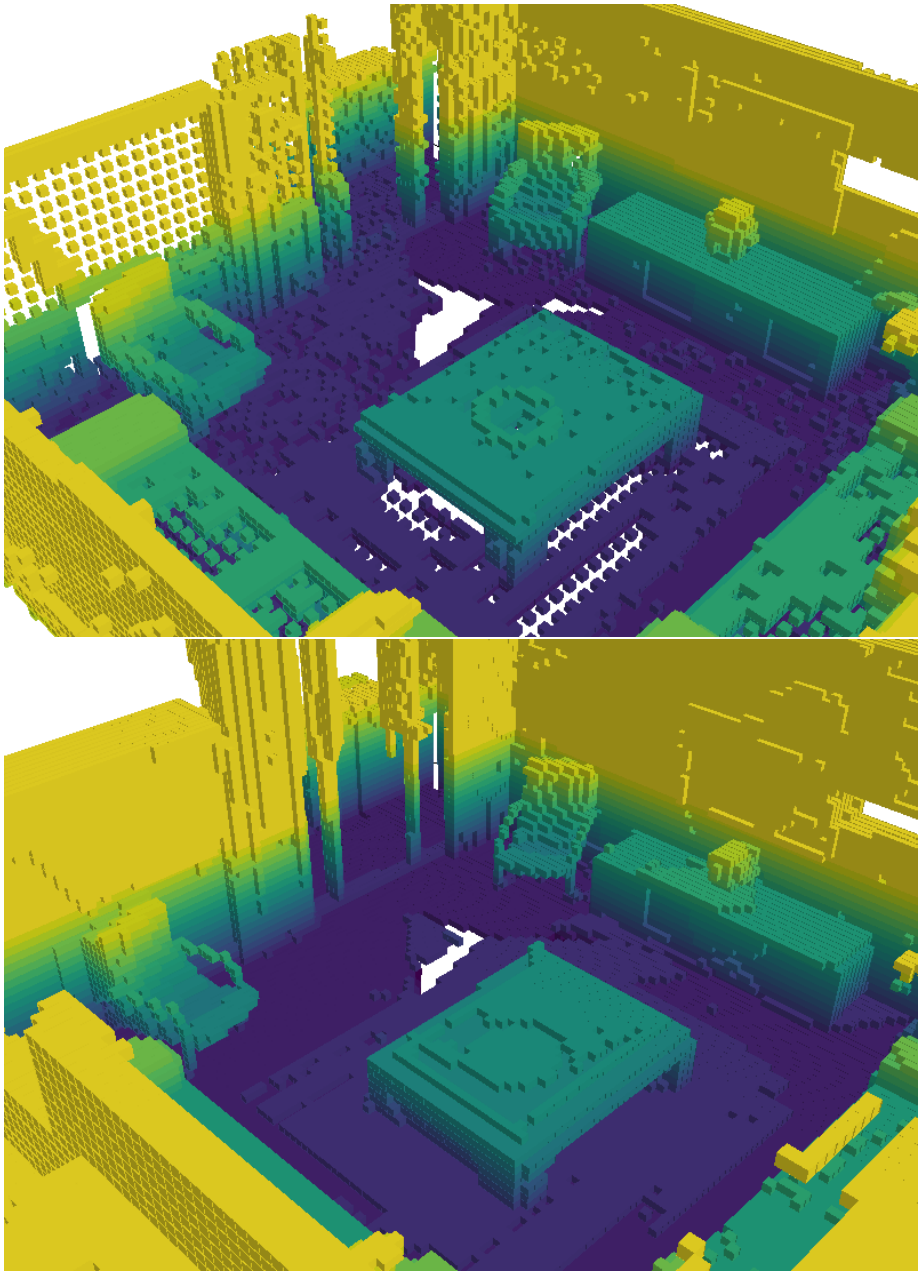


Figure 4.8: Occupied voxel map for the `livingroom1` noisy depth sequence of the Augmented ICL-NUIM dataset [13]. $10 \times 10 \times 2$ m^3 volume at 0.05 m resolution. Top: OctoMap, Bottom: MRFMap. Note the artifacts in the OctoMap due to a combination of glancing rays and sensor noise.

Real-world dataset

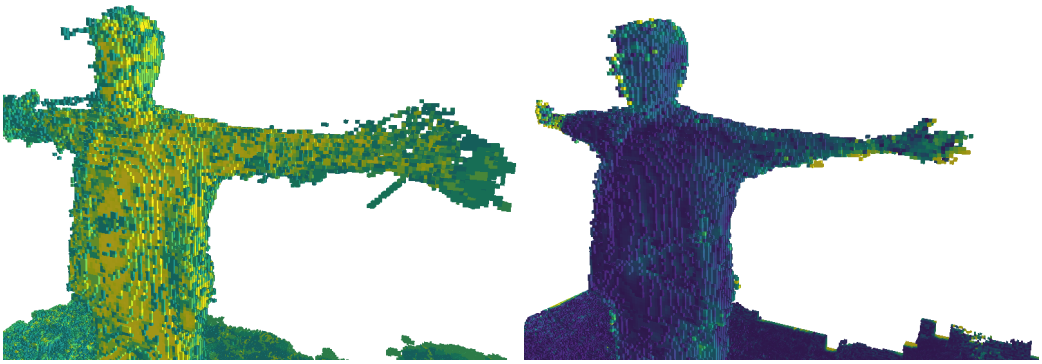
For the real-world dataset we utilise a rig shown in Fig. 4.6 in a Vicon motion capture arena and capture a human subject. Maps are built with the noisier stereo IR Realsense D435 camera and accuracy evaluated with data from the Kinect One, a Time of Flight (ToF) based camera. The latter is employed for evaluating accuracy in the absence of having ground truth depth. For inference, we use the aggregate polynomial model for bias and standard deviation demonstrated in Fig. 3.7 for all pixels. The keyframes are selected based on the same geometric heuristic with smaller thresholds (0.5 each) to account for the smaller capture volume. Low accuracy values can be attributed to errors in the inter-camera registration and the dynamic camera motion. Inferred maps are shown in Fig. 4.9 at multiple resolutions and highlight the drastically better qualitative maps obtained from the noisy sensor data.

Resolution	0.01 m	0.02 m	0.05 m
MRFMap	0.305 ± 0.185	0.335 ± 0.195	0.406 ± 0.188
OctoMap	0.309 ± 0.192	0.302 ± 0.191	0.263 ± 0.170

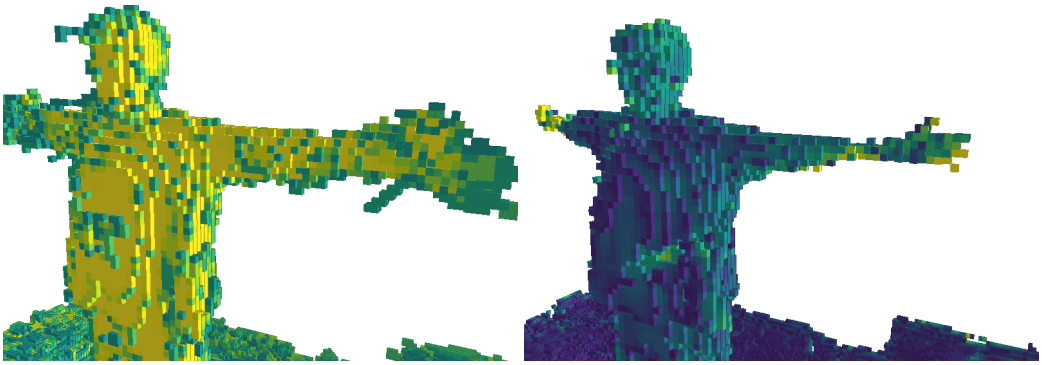
Table 4.3: Accuracy values for the real-world dataset shown in Fig. 4.9. Map volume is $3 \times 3 \times 2$ m^3.

Timing

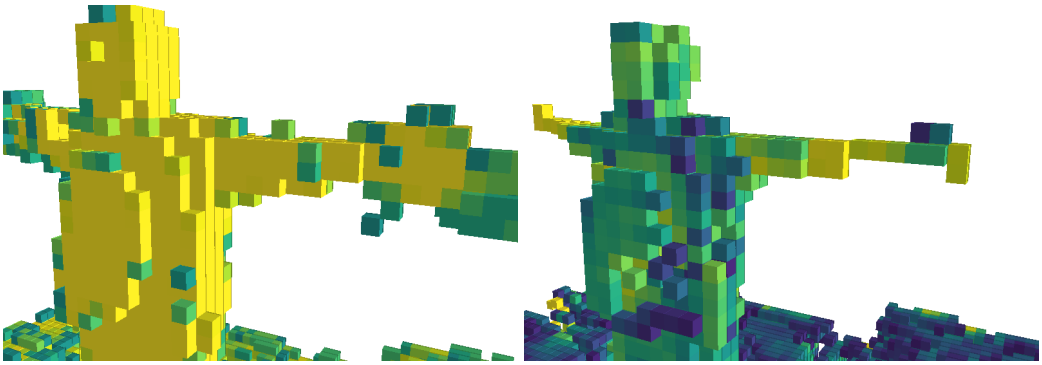
By virtue of being a framework that retains keyframe sensor data in memory, adding a new image causes the inference to iterate over all the images, and the overall time taken for inference increases monotonically. However, due to the accelerated data structure and parallelised implementation, an MRFMap is much faster at ray tracing and performing map inference than an OctoMap, especially at finer resolutions. Fig. 4.10 demonstrates that, for instance, at a resolution of 0.01 m on the `livingroom1` dataset even after adding 30 keyframes, ray tracing a new image and performing three passes of inference over all keyframes still takes two orders of magnitude less time than adding it to the OctoMap. These results were obtained on an NVIDIA RTX 2060 Super and an eight-core AMD Ryzen 3700x CPU.



(a) 0.01 m



(b) 0.02 m



(c) 0.05 m

Figure 4.9: OctoMap (left) and MRFMap (right) at various map resolutions on real-world data. The MRFMaps are constructed using Realsense D435 data at 848×480 px resolution with the aggregate forward sensor noise model (including bias) shown in Fig. 3.7 and evaluated on a constant noise model using a rigidly attached Kinect One. Note the much better reconstructed fidelity of the MRFMaps in the head and the hand region. Dark violet represents lower occupancy probability while bright yellow represents high occupancy probability.

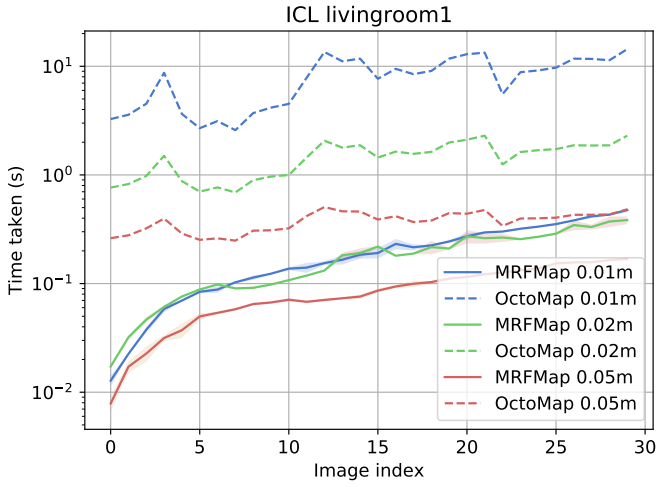


Figure 4.10: Incremental time taken for adding a new image and performing inference in an MRFMap and an OctoMap to generate Fig. 4.8

4.6 Conclusion

Through our experiments, we show that explicitly modelling intra-and inter-dependence of neighbouring voxels due to sensor ray formation as opposed to treating all voxels as being independent enable more accurate occupancy inference than conventional occupancy grid mapping frameworks. Incorporating learnt forward sensor models helps obtain even higher map fidelity, and ensures that the map is a maximum likelihood solution that best attempts to explain the sensor data.

The MRFMap framework is envisioned as an essential building block for replacing traditional grid based submapping implementations [24, 54, 73] since it enables performing on-demand inference to obtain a probabilistically accurate map without having to approximately resample the submaps when moving any subset of keyframe poses. The monotonically increasing runtime considerations can be addressed by performing inference only over images that view a region of interest, often called an active region. The increasing memory costs can be offset by marginalising out occupancy in the inactive regions. These are addressed in the subsequent Chapter 6.

Chapter 5

Localisation using Gaussian Mixture Model Maps

This chapter presents an implementation of a real-time particle filter based estimator that infers the posterior distribution over camera trajectories given a succinct parametrised map representation. Specifically, it provides the first term in the previously discussed equation

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = \underbrace{p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})}_{\text{This chapter}} p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}). \tag{2.4}$$

5.1 Introduction

For an agent lost in a known environment, much of the cost of localisation can be offset by pre-computing measures of what the sensor is expected to see; localisation can then be cast as the much simpler problem of a search through these pre-existing "hallucinated" views. However, exhaustively considering all possible views incurs a prohibitive cost that increases exponentially with both the dimensionality of the state space and the size of the environment. Further, naïve pre-rendering approaches can be susceptible to errors caused by perceptual aliasing due to slight variations in the environment appearance or by regions that are not feature rich, such as blank walls [4].

In this chapter we present a framework enabling rapid computation of sensor data likelihood via an environment representation that is both high-fidelity and memory efficient. The framework models a depth camera observation as being sampled from the environment representation with a likelihood measure that varies smoothly about the true camera pose. We thus exploit the dramatically reduced storage complexity of the representation and the local spatial regularity of a fast-to-compute likelihood function to

re-cast the pose estimation problem as that of Monte-Carlo localisation [86].

Our framework solves the problem of 6 Degree-of-Freedom pose estimation for Size, Weight, and Power (SWaP) constrained micro air vehicles operating in a known dense 3D pointcloud environment with an onboard depth camera and Inertial Measurement Unit (IMU). We assume that the vehicle pitch and roll are obtained from an attitude estimation algorithm using the IMU in order to constrain the search space to just heading and position. Our main contributions are:

- A particle filter-based localisation strategy based on a high fidelity, memory efficient environment representation enabled by a fast likelihood computation approximation; and
- Experimental evaluation of the approach on a desktop and an off-the-shelf mobile GPU system.

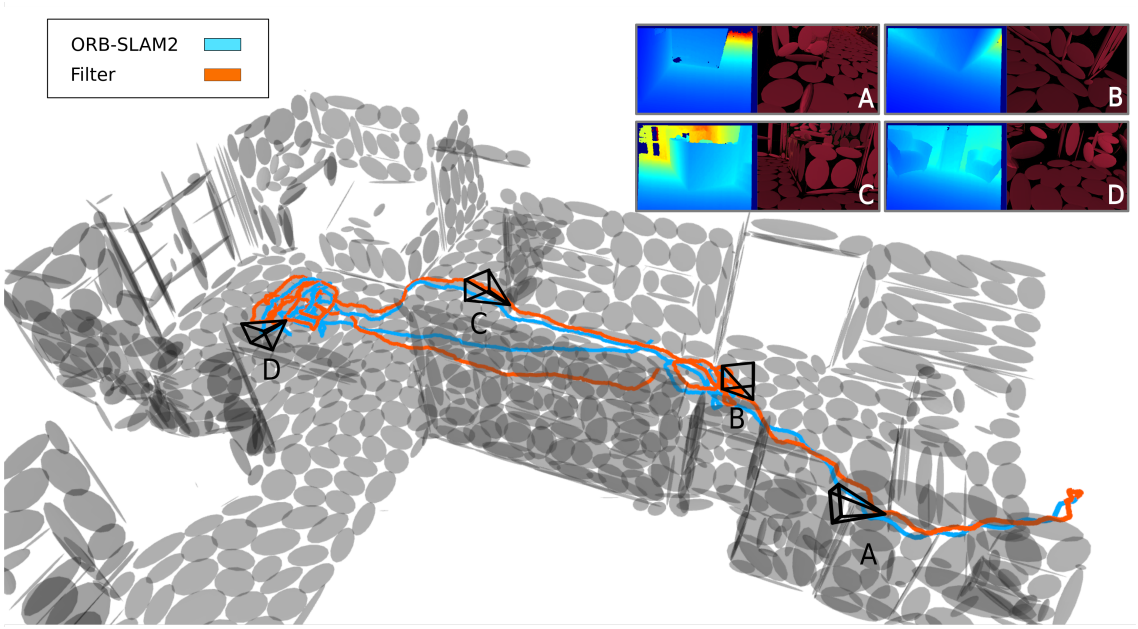


Figure 5.1: Comparison of the mean particle filter pose (orange) with that of the integrated process model trajectory (cyan) from a representative (office) dataset. The filter estimate is initialized with a uniform distribution away from the true location of the vehicle. As the camera observes more informative data the filter quickly converges to the correct pose. *Top Right*: Four views of the raw point cloud sensor data and the corresponding view of the GMM map from the mean of the particle filter estimates. The GMM components are superimposed on top of the source point cloud with their 3Σ bounds visualized as gray ellipsoids.

5.2 Related Work

Similar in spirit to our choice of representation, [25] map the world with a succinct, albeit restrictive parametrisation of using only dominant planes. Our choice of representation however, does not have any such restrictions. Likewise, [8] is similar in its restriction of representation to a structured world consisting of planes and edges and only localizes in 2D. Approaches such as NDT occupancy maps [16, 56, 77], unlike the representation used in this work, learn independent distributions over each discretised cell leading to a lower fidelity representation at the cell boundaries [19, 74]. Further, for fast pointcloud alignment they require pre-computation of the incoming data for comparison with a stored model, known as Distribution-to-Distribution (D2D) registration. The more accurate Point-to-Distribution (P2D) registration approaches, however, are not as real-time viable [16, 47], and are less so for our purpose. In [11] the authors represent each point in the reference scan of a 2D sonar scan as an isotropic Gaussian distribution and iteratively compute the 2D transformation that maximizes the likelihood of all points in the target scan. Our approach, on the other hand, represents clusters of points in the reference scan as individual anisotropic Gaussian components and is not restricted to transformations of small magnitude.

Closest in terms of our choice of implementation framework, [26] propose a particle filter based real-time RGB-D pose estimation approach and [10] present a real-time localisation approach on a fixed-winged aircraft during aggressive flights with a laser scanner and an IMU. The latter work implicitly exploits much more restricted dynamics of a fixed wing aircraft within its process model and further, both these approaches use the OctoMap [38] representation to provide correction updates within their filter estimates. Note that the memory footprint of an OctoMap is much greater than that of our choice of representation. For instance, memory consumption for an OctoMap of dataset D3 (Sec. 5.5.1) with 0.1 m resolution is 872 KB while the corresponding data usage for our Gaussian Mixture Model (GMM) map representation consisting of 1000 Gaussian components is 40 KB.

Prior works exploiting known map appearance for precise monocular pose estimation [31, 57, 61, 76] employ a textured depth map within an iterative optimization framework to compute the warp that minimizes a photometric cost function between a rendered image and the live image such as the Normalized Information Distance[61], that is robust to illumination change, or a Sum of Squared Differences cost function with an affine illumination model to tackle illumination change[57]. Both algorithms rely on initialization for tracking via a GPS prior or an ORB-based bag-of-words approach, respectively, and expensive raycasted dense textured data for refinement. Note that in contrast to the above mentioned algorithms that use RGB information, we only use depth observations and only

project a finite number of mixture components as opposed to dense pre-rendered views of the map.

5.3 Approach

Contemporary direct tracking algorithms require the projection of large numbers of dense or semi-dense points into image space to align the current sensor data to a reference model. In contrast, we employ GMMs as a succinct parameterized representation to achieve orders of magnitude computational savings via an analytic projection into image space. The consequent reduction in complexity enables projection in multiple pose hypotheses concurrently in real-time and motivates this work. This representation is then used to generate better proposal distributions for a particle filter that is provided an estimate of camera odometry.

5.3.1 Spatial GMMs as an Environment Representation for Tracking

Conventional means of representing maps such as voxel grids discretize space to encode occupancy leading to resolution dependent model fidelity and memory efficiency. An alternate approach is to represent occupancy using a GMM map that attempts to approximate the underlying distribution from which sensor measurements are sampled. This formulation is capable of representing the environment model with as high a fidelity as required that scales gracefully with model complexity when used in a hierarchical fashion [74]. Additionally, this representation provides a probabilistic uncertainty estimate of the occupancy at any sampled location. Fitting these models to data in real time is possible due to recent advances that enable efficient operation [20]. We utilise the contribution presented in [20, 74] to inform the number of Gaussian components required to pre-compute GMM maps of the environment point cloud at various fidelity levels. For the purpose of this paper, however, we limit the discussion to using only GMM maps at a certain fidelity level chosen according to Sec. 5.5.2, but note that the approach can be readily extended to a hierarchical formulation.

A spatial GMM represents the probability of matter existing at a specific position $\mathbf{P}^w$ given the model component parameters $\mathbf{\Theta} = \{\boldsymbol{\mu}_i, \mathbf{\Sigma}_i, \lambda_i\}_{i=1\ldots M}$ such that

$$p\big(\mathbf{P}^w; \mathbf{\Theta}\big) = \sum_{i}^{M} \lambda_i \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{\Sigma}_i) \tag{5.1}$$

where λ_i is the mixture weight, $\boldsymbol{\mu}_i$ the mean 3D position, and $\mathbf{\Sigma}_i$ the covariance of the i^{th}

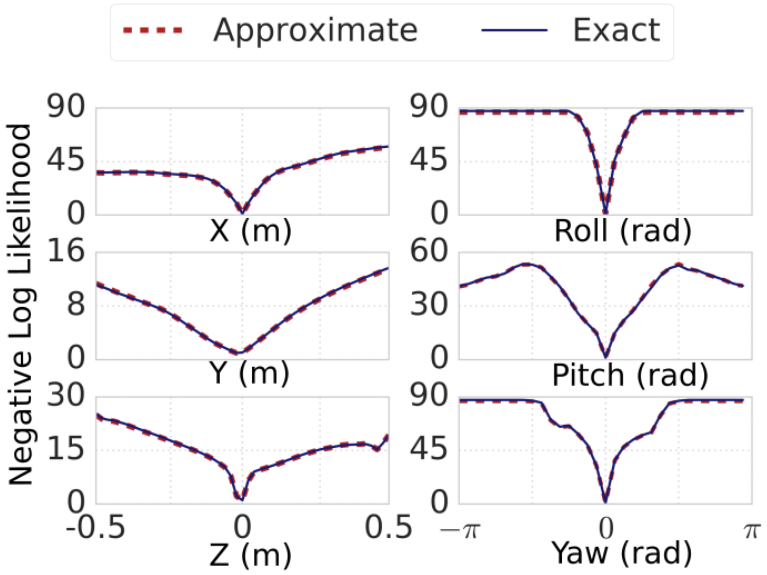


Figure 5.2: Negative log-likelihood plots of sensor data acquired from camera poses offset from a randomly chosen true pose in dataset D1 by incremental linear and rotational displacements. Utilising only the relevant components using the approximation discussed in Sec. 5.4 leads to almost identical likelihoods as when utilising all the Gaussian components present in the model.

component of the GMM respectively, with $\sum_i^M \lambda_i = 1$ and $\lambda_i > 0$.

5.3.2 Projection of a GMM Component into Image Space

In order to determine relevant mixture components of a given spatial GMM map for evaluating sensor data likelihood we first analytically project the mixture components into image space.

For a point $\mathbf{P}^w$ in the world frame, the transformed position $\mathbf{P}^c$ in a camera frame $\mathbf{T}_w^c$ is denoted as

$$\mathbf{P}^c = \mathbf{R}_w^c \mathbf{P}^w + \mathbf{t}_w^c$$

where $\mathbf{R}_w^c$ and $\mathbf{t}_w^c$ are the corresponding rotation matrix and translation vectors, respectively. Since this is a linear operation on $\mathbf{P}^w$, using Eq. 5.1 the transformed distribution of points in the camera frame for the i^{th} component is

$$p\big(\mathbf{P}^c; \boldsymbol{\Theta}_i\big) = \mathcal{N}(\mathbf{T}_w^c \boldsymbol{\mu}_i, \mathbf{R}_w^c \boldsymbol{\Sigma}_i \mathbf{R}_w^{c\,\mathrm{T}})$$

Consider a sample $\mathbf{x}_s \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a monotonic continuous nonlinear function $\mathbf{y} = f(\mathbf{x})$

(where $\mathbf{y}$ and $\mathbf{x}_s$ are in the same space). Utilising the delta method, the first order Taylor series expansion about a point $\mathbf{x}_s$ leads to

$$\mathbf{y} \sim \mathcal{N}\left(E[f(\mathbf{x})], \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_s} \boldsymbol{\Sigma} \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_s}^{\mathrm{T}}\right)$$

Under the standard pinhole projection model, a point $\mathbf{P}$ in the camera frame is projected to the image space using the operation $\pi : \mathcal{R}^3 \rightarrow \mathcal{R}^2$ defined as

$$\boldsymbol{\pi}(\mathbf{P}) = \begin{bmatrix} c_x + f\frac{\mathbf{P}_x}{\mathbf{P}_z} \\ c_y + f\frac{\mathbf{P}_y}{\mathbf{P}_z} \end{bmatrix}$$

where f is the focal length of the camera, and c_x and c_y are the principal point offsets. The derivative of the projection operation with respect to the 3D point $\mathbf{P}$ is

$$\frac{\partial \boldsymbol{\pi}}{\partial \mathbf{P}} = \begin{bmatrix} \frac{f}{\mathbf{P}_z} & 0 & -f\frac{\mathbf{P}_x}{\mathbf{P}_z^2} \\ 0 & \frac{f}{\mathbf{P}_z} & -f\frac{\mathbf{P}_y}{\mathbf{P}_z^2} \end{bmatrix}$$

Thus the projection of a 3D normal distribution component into image space is the 2D normal distribution

$$p(u, v; \boldsymbol{\Theta}_i) = \mathcal{N}\left(\boldsymbol{\pi}(\mathbf{T}_w^c \boldsymbol{\mu}_i), \left.\frac{\partial \boldsymbol{\pi}}{\partial \mathbf{P}}\right|_{\mathbf{T}_w^c \boldsymbol{\mu}_i} \mathbf{R}_w^c \boldsymbol{\Sigma}_i \mathbf{R}_w^{c\,\mathrm{T}} \left.\frac{\partial \boldsymbol{\pi}}{\partial \mathbf{P}}^{\mathrm{T}}\right|_{\mathbf{T}_w^c \boldsymbol{\mu}_i}\right) \tag{5.2}$$

where u, v are pixel coordinates in the image. Although this is only an approximation, for components far away from the camera the linearisation is accurate enough for the purpose of projection.

5.3.3 Estimating the Likelihood of a Camera Pose Hypothesis

As shown in the previous subsection, each Gaussian component can be projected into image space as a 2D Gaussian distribution. We utilise this property to determine relevant components for computing the likelihood of sensor data (Sec. 5.4). Given a scan $\mathbf{Z}_t$ of depth pixels $\{z_1, z_2, \ldots, z_k\}$ from a sensor scan and a set of 3D GMM parameters $\boldsymbol{\Theta}$, the

log likelihood of the scan being sampled from the GMM is defined as

$$l\left(\mathbf{Z}_t|\mathbf{\Theta}, \mathbf{T}_w^c\right) = \sum_i^K \ln \sum_j^M \mathbb{1}_j \lambda_j \mathcal{N}\left(\boldsymbol{\pi}^{-1}(z_i); \mathbf{T}_w^c \boldsymbol{\mu}_j, \mathbf{R}_w^c \mathbf{\Sigma}_j \mathbf{R}_w^{c\,\mathrm{T}}\right) \tag{5.3}$$

where $\mathbb{1}_i$ is a binary indicator function that signifies if the i^{th} component is used to compute the log likelihood, $\boldsymbol{\pi}^{-1}$ is the inverse projection from depth image pixel to 3D points, and K is the number of pixels in the sensor scan. This likelihood should peak at the true sensor pose and decay smoothly in the local neighbourhood, which is indeed observed as shown in Fig. 5.2.

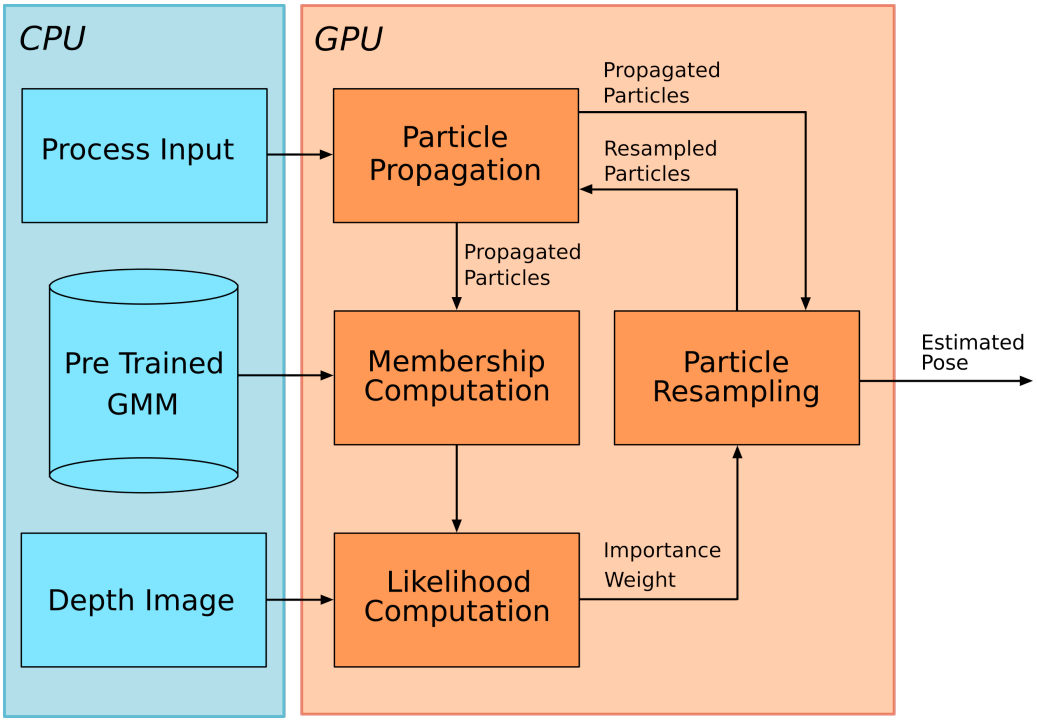


Figure 5.3: System overview. The algorithm operates on depth image streams and a source of odometry given a precomputed GMM map of the environment. For each particle, the GMM components are projected into image space using its current pose hypothesis. Relevant components are sub-selected and are then used to compute the likelihood of the depth image. The likelihood values for all the particles are used to resample a new set of particles that are then forward propagated using the process model.

The discussion above only considers the nature of the likelihood in the vicinity of the true location; in practice it is not reasonable to assume that a single viewpoint suffices

to localize the system as perceptual aliasing may arise due to a paucity of data that precludes state observability. Hence, we require a technique that permits tracking of multiple hypotheses and ensures appropriate weighting of equally likely viewpoints given the current sensor observations.

5.3.4 Inference of Posterior over Trajectories

A standard approach to tracking multiple hypotheses is a Monte Carlo filter (or particle filter). Particle filters [83] operate by continuously sampling candidate particle poses and measure the likelihood of the current sensor data having originated at the sampled pose. Based on the relative scores of the samples the particles are resampled and propagated based on a process model (often a noisy source of odometry). Convergence is generally achieved as soon as the sequence of observations made over time render alternate hypotheses inadmissible. Note that due to their inherent structure particle filters are extremely parallelisable and we exploit this in our implementation.

We use the strategy presented in Grisetti et al. [32] to use the most recent observation and the map estimate from the previous time step in the proposal distribution, especially for sensors that provide significant precision and density in measurements, such as depth cameras [5]. The final expression for the proposal distribution ends up being

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{m}, z_t, u_t) = \frac{p(z_t|\mathbf{x}_t, \mathbf{m})p(\mathbf{x}_t|\mathbf{x}_{t-1}, u_i)}{p(z_t|\mathbf{x}_{t-1}, \mathbf{m}, u_t)} \tag{5.4}$$

The former likelihood term is generally highly peaked, but can have multiple modes due to perceptual aliasing. The motion model obtained via odometry helps curtail it to regions that are actually feasible given the prior, as can be seen in Fig. 5.4. In the context of the GMM representation these two terms are estimated as follows:

- Motion Model: We assume the presence of some odometry to drive the first order Markov motion model and inject Gaussian noise into it. Note that we assume that we know the pitch and roll that can be obtained from the attitude and heading reference system onboard a robotic system to a high level of accuracy.
- Measurement Likelihood: The measurement likelihood represents a how well the sensor scan matches the GMM map at a given location. Since the negative log likelihood of the current scan $\mathbf{Z}_t$ being drawn from the GMM map is a minimum at the true location, as shown in Fig. 5.2, in practice we use the inverse of the negative log likelihood. Thus, given the current state estimate $\mathbf{T}_w^{c^{(i)}}$ of a particle i out of N

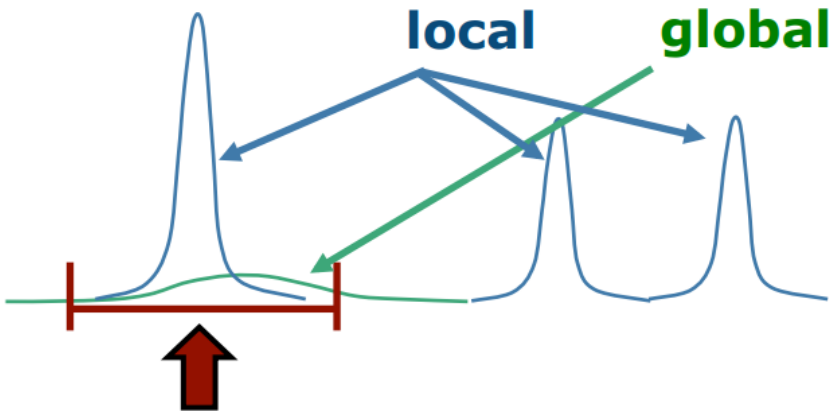


Figure 5.4: The measurement likelihood is often highly peaked for dense sensor measurements such as depth sensors, but may have multiple local minima. The distribution of future robot poses from odometry and motion models is less peaked but more unimodal. The combination of these two distributions serve to work as an effective proposal distribution. Figure from [75].

particles at time step t, the corresponding normalized importance weight is

$$w_t^{(i)} = \frac{l\left(\mathbf{Z}_t|\mathbf{\Theta}, \mathbf{T}_w^{c^{(i)}}\right)^{-1}}{\sum_j^N l\left(\mathbf{Z}_t|\mathbf{\Theta}, \mathbf{T}_w^{c^{(j)}}\right)^{-1}}. \tag{5.5}$$

Sampling Strategy

A particle filter should ideally converge to the correct hypothesis after running for a finite amount of iterations with a reduction in the filter variance signifying the confidence of the filter. At the same time, an early reduction in the filter variance may cause the filter to diverge to an incorrect hypothesis and never recover due to low variance. In order to avoid such situations, we implement the stratified sampling strategy [43] in combination with low variance sampling [86]. The particles are divided into random groups of equal weights and in each group we employ low variance sampling. This approach has low particle variance [86] and works well when the particle filter is tracking multiple hypotheses at once.

Handling Particle Deprivation

One of the most common failure modalities of a particle filter is that of particle deprivation [90]. Even with a large number of particles, the stochasticity intrinsic to a particle

filter might cause it to diverge from the correct state. We employ a modified version of Augmented MCL strategy as described in [86] where instead of adding new particles we reinitialize N_{modify} number of particles randomly selected from the original set using the parameters α_{slow} and α_{fast}. This is done since we cannot increase the number of particles once the filter is initialized because of implementation limitations. For our process model we use diagonal covariances for translation, and the final choice of parameters in all our experiments is shown in Table 5.1.

5.4 Fast Localisation

In order to perform fast localisation using the above approach it is essential to compute the likelihood of the data given a proposed pose as quickly as possible. Eq. 5.3 suggests that computing the likelihood of a scan having been sampled from the GMM map is the summation of the contribution of all the components within the GMM. However, the key insight here is that not all the components have a significant contribution to the likelihood.

The point clouds that we use in our experiments have roughly uniform coverage of points across the scene. As a consequence, all Gaussian components fit to these pointclouds end up having roughly equivalent mixture weight probabilities. This fact, in addition to the diminishing probability mass of the Gaussian distribution, permits the approximation of using only the projected components within spatial proximity of a certain pixel location for computing the likelihood of the corresponding 3D point being sampled from the map. As an added optimization step we perform this membership computation over subdivided patches of the image. These optimizations have negligible effect on the computed likelihood value of the sensor data, as demonstrated in Fig. 5.2.

We follow the following steps (graphically illustrated in Fig. 5.5) to obtain the relevant components for computing the likelihood of a depth image:

- Divide the image into 32×32 pixel patches;
- Compute the 2D projection of each Gaussian component on to the image plane of the depth sensor;
- Inflate the $3\boldsymbol{\Sigma}$-bound ellipse of the projected 2D Gaussian of each component by half the diagonal of the patch along its major and minor axis to generate ellipses $\mathcal{E}_i$; and
- For each patch, check if the center of the image patch c_p lies within or on each of the $\mathcal{E}_i$ and update the indicator variable $\mathbb{1}_{i,p}$ accordingly.

$$\mathbb{1}_{i,p} = \begin{cases} 1, \text{ if } c_p \in \mathcal{E}_i \\ 0, \text{ otherwise} \end{cases} \tag{5.6}$$

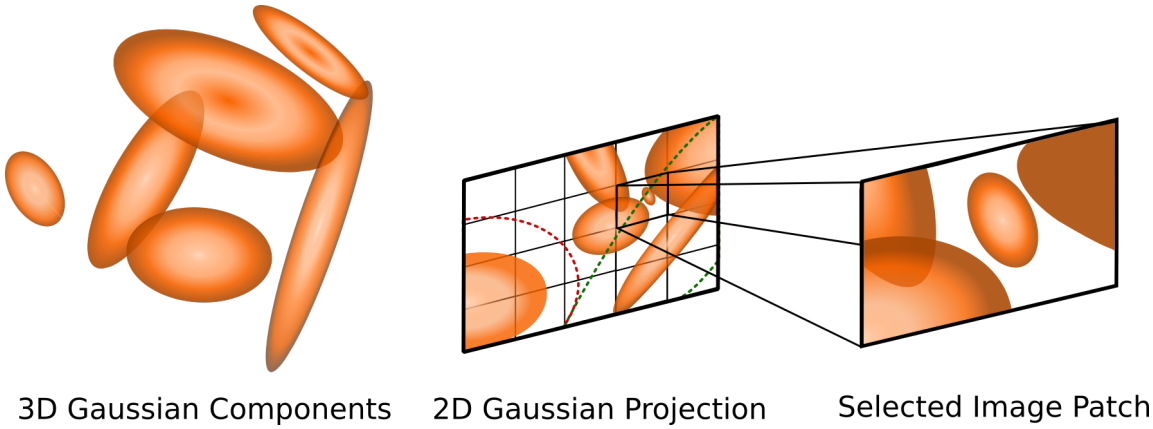


Figure 5.5: Membership computation process. 3D Gaussian components from the GMM representation of the world are projected to the image plane. The image is subdivided into multiple patches, where for a selected patch the relevant Gaussian members are determined for computing the likelihood. In order to determine the latter, we employ the heuristic described in Sec. 5.4. For instance the inflated bounds of the bottom left projected component (red) do not contain the center of the selected patch; in contrast those of the bottom right (green) do, and the component is thus selected for computing the likelihood of data within that particular patch.

Given a set of updated indicator variables $\mathbb{1}_{i,p}$ for all the Gaussian components $\mathbf{\Theta}$ and a depth image, $\mathbf{Z}_t$, the likelihood of the image can be computed as the sum of the likelihoods of all the image patches computed according to Eq. 5.3.

5.5 Results

5.5.1 Experiment Design

This section presents performance analysis of our filtering approach on a wide variety of datasets. First, we conduct a sensitivity analysis to determine the number of particles and the number of components we use in our implementation. Second, we analyse metric accuracy of the proposed filter on publicly available datasets and show that our filter output is consistent with ground truth. Third, we compare the localisation performance of our approach with a state-of-the-art RGB-D tracking algorithm (ORB-SLAM2 [49]) on the same sequences and demonstrate superior performance for localisation. Fourth, we demonstrate the ability of our approach to incorporate both different odometry algorithms and ground truth map acquisition methodologies. Finally, we analyse runtime performance

of our filter and show that its runtime is competitive both on a desktop class system and on an embedded platform, thus enabling SWaP constrained operation.

We evaluate our approach on

- D1: The (a) lounge and (b) copyroom datasets [96];
- D2: The voxblox dataset [58];
- D3: A representative dataset collected in-situ; and
- D4: The TUM Freiburg3 dataset [81] for demonstrating the ability to generalize.

In all cases we utilise a fixed number of components (Sec. 5.5.2) to first fit a GMM to the pointcloud using the scikit-learn[1] toolkit.

We employ two processing systems for evaluation: (1) A desktop with an Intel i7 CPU and an NVIDIA GTX 960 Ti GPU, and (2) An embedded NVIDIA TX2 platform.

5.5.2 Sensitivity Analysis

Particle filters can achieve increased performance with large number of particles at the cost of increased computational complexity. Conversely too few particles can lead to divergence from the true location due to an inability to represent the true underlying distribution. In order to find the appropriate number of particles that ensure precision while still being computationally feasible we compare the filter performance with various number of particles against a ground truth filter with $N = 16200$. Assuming the underlying distribution represented by the particle set to be a unimodal Gaussian (a valid assumption after convergence), we compute the variance of the KL-Divergence [36] of multiple runs of the filter output with that of the ground truth filter to determine the empirically optimal parameters to be used in our implementation. A low value of the KL-Divergence variance indicates similar performance to the ground truth filter.

The fidelity of a GMM map to the underlying distribution monotonically increases with the number of components. However, the marginal benefit (in a KL-Divergence sense) of increasing the model complexity diminishes rapidly after adding an adequate number of components [74]. In order to determine the appropriate model complexity to represent the original map concisely while enabling accurate filter performance, we perform similar experiments with the optimal number of particles obtained from the previous study, this time with varying number of Gaussian components.
We compute the optimal parameters to be $N = 1068$ and $M = 1000$ based on D3, the dataset with the largest volumetric span. This specific parameter choice is further motivated by

[1] http://scikit-learn.org/stable/modules/mixture.html

Table 5.1: Filter hyperparameters

	Process Noise σ		α_{slow}	α_{fast}
	Translation (m)	Yaw (rad)		
Desktop	0.02	0.01	0.01	0.001
TX2	0.025	0.1	0.05	0.005

implementation constraints.

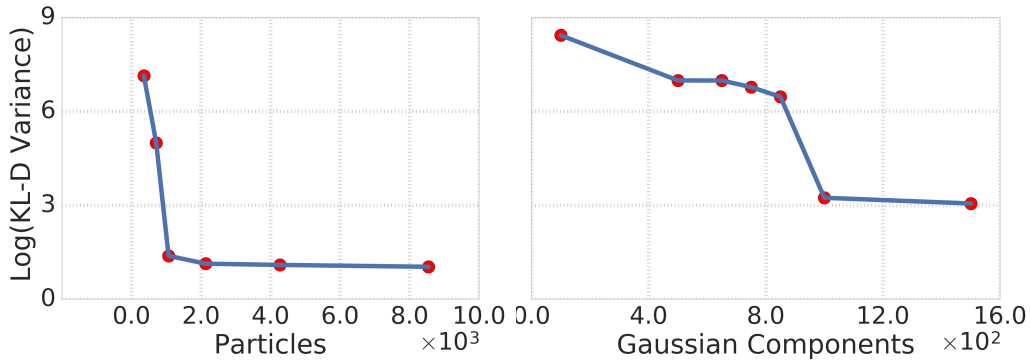


Figure 5.6: *Left*: Log of variance of KL-Divergence between the ground truth filter ($N = 16200$) and filters with reduced particle counts. The knee point implies similar performance to the ground truth filter at particles counts $N > 1000$. *Right*: A similar comparison given a ground truth map with many components ($M = 2500$) and those with a reduced number motivates the choice of $M >= 1000$. Evaluated on D3.

5.5.3 Metric Accuracy Analysis

In this subsection we discuss the localisation accuracy of our approach. As mentioned in Sec. 5.4 since we do not add new particles when the filter observes particle deprivation and instead randomly reinitialize the particles from the original set, the Root Mean Squared Error (RMSE) of the filter estimate increases when the filter observes particle deprivation. This is highlighted in the plots as vertical shaded regions. For all our evaluations we run the filter 10 times on each dataset and report the average of the mean filter estimate. We do not quantify the sensitivity of the likelihood values to the AHRS pitch and roll estimates as they are accurate enough to not cause any significant difference.

Evaluation with Ground Truth Datasets (D1, D2)

The objective of using these datasets is to demonstrate the ability of the filter to converge to the ground truth given perfect odometry. We generated a GMM map of the environments using the reconstructed point cloud and used the delta transforms between two consecutive reported sensor poses with added noise as our process model. In all these experiments, we initialized the particles from a uniform distribution over a 4 m cube and π radians yaw orientation around the known initial location. D1(a) and D1(b) contain nominal motion of the sensor, while D2 consists of very aggressive motion in all degrees of freedom.

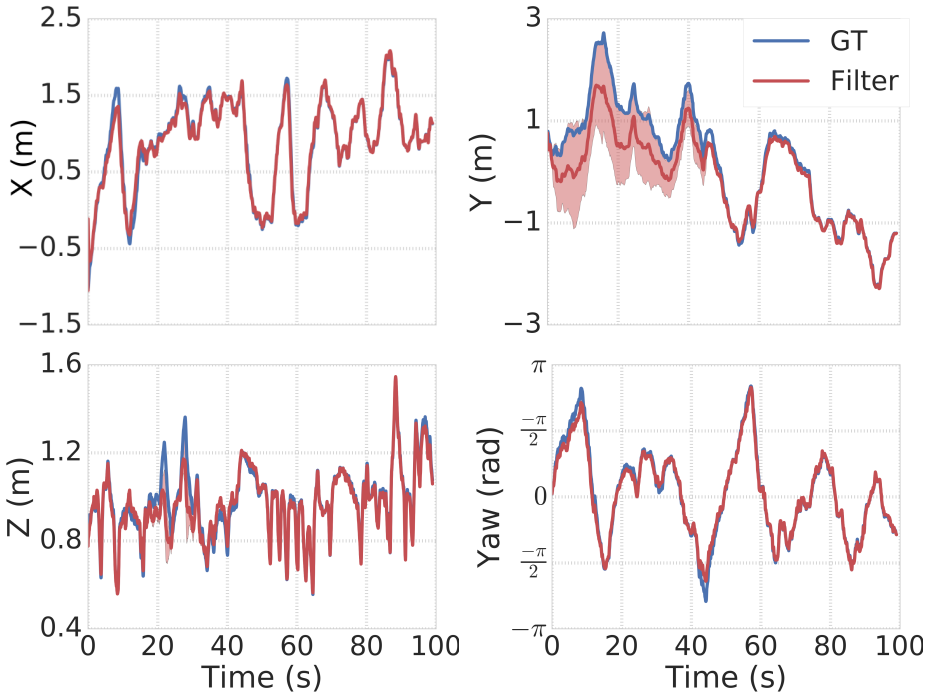


Figure 5.7: Mean trajectory (red) of the particle filter estimate of 10 trials on the D1(a) dataset compared to the process model trajectory (blue). The shaded region around the mean trajectory shows the variance of the filter estimate over multiple runs. The filter estimates have high variance in the beginning of the trajectories, but soon converge to the correct location and track the ground truth trajectory (blue).

The filter estimate converged to an incorrect hypothesis for some runs in the initial iterations due to the highly symmetric nature of the environments about the X axis, as can be seen in Fig. 5.7. The RMSE of the filter poses for these datasets is presented in Fig. 5.8.

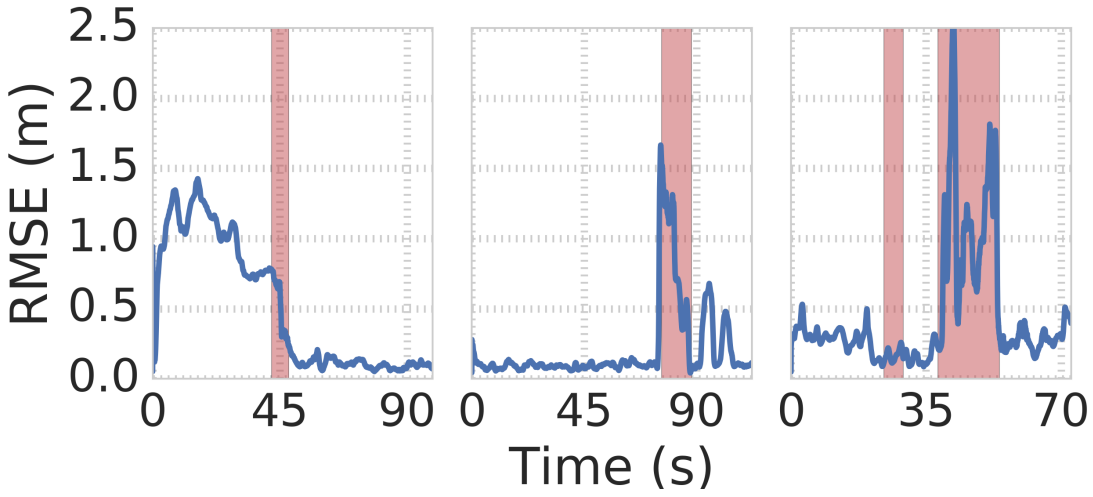


Figure 5.8: RMSE of 10 trials of the particle filter on the D1(a), D1(b), and D2 datasets respectively. The region in red indicates the time at which the particle filter observes particle deprivation and a consequential RMSE rise.

Evaluation with Representative Dataset (D3)

The objective of using this dataset is to demonstrate results on a real-world application of the filter. We no longer use ground truth odometry. Additionally, since we don't have a baseline algorithm to directly compare against, we compare the localisation performance against ORB-SLAM2 which builds its own succinct map representation. Note that ORB-SLAM2 also utilises the RGB image data in the dataset whereas we only use the depth. Finally, we also briefly contrast the performance of the filter on the same dataset.

We generate a ground truth pointcloud using a FARO Focus 3D Laser scanner [2] and use an ASUS Xtion RGB-D camera for acquiring sensor data. An IMU strapped to the camera determines the roll and pitch of the sensor. For odometry, we only use the frame to frame relative transform as opposed to the global pose output from ORB-SLAM2 as input to the process model. Note that the global ORB-SLAM2 position we compare to in Fig. 5.9 and Fig. 5.10 is using loop closure to mitigate the drift in its frame to frame estimates.

As ground truth is not available for this dataset, we report the negative log likelihood values at the mean particle filter location and the reported ORB-SLAM2 poses. We show results of two runs in this environment in Fig. 5.9: The first through a nominal path with feature rich data (as shown in detail earlier in Fig. 5.1) where the estimated positions of the sensor for the two approaches are very similar (but with worse likelihood values for ORB-SLAM2). The second run demonstrates the advantage of using particle filters

[2] https://www.faro.com/products/construction-bim-cim/faro-focus/

Table 5.2: Performance on D4 (RMSE in cm)

Our Approach			ORB-SLAM2
Process Input	mean	var (cm^2)	
ORB-SLAM2 Velocity	7.67	0.21	
Ground Truth Velocity	7.56	0.28	4.55
G-ICP Velocity	9.07	0.21	

over maximum likelihood estimators in that the former can converge to the correct result even after moving through a region of low observability. We observe that the sensor measurements register at the converged filter location better after snapping back than those for the ORB-SLAM2 estimate, as can be qualitatively seen in Fig. 5.10.

The particles in these experiments are initialized from a uniform distribution over a $4\text{m} \times 8\text{m} \times 3\text{m}$ for position and π radians in yaw.

Evaluation with TUM Dataset (D4)

To demonstrate the ability of our filter to generalize to both different odometry algorithms and datasets we compare the performance with three different odometry inputs as process models: The Generalized-ICP algorithm [69], ORB-SLAM2 frame-to-frame relative transform, and ground truth odometry. The point cloud map of the environment was created by stitching several sensor scans together using their corresponding ground truth poses. In spite of the stitched point cloud not being as well registered as that from a FARO scanner due to sensor and ground truth pose noise, the performance of the filter is similar (Table 5.2).

5.5.4 Runtime Performance Analysis

As seen in Fig. 5.12, the likelihood evaluation is the most computationally expensive operation. Execution time for this step varies with the number of Gaussian components used to compute likelihood for each image patch and therefore is dependent on the fidelity of the model.

The filter runs at an average rate of 80 Hz and 9.5 Hz on the Desktop and embedded class systems, respectively. This is comparable to the ORB-SLAM2 rates of 47 Hz and 20 Hz on the respective platforms. Initial convergence on the TX2 is slower due to the implicitly larger odometry steps. However, post convergence the metric performance is not significantly affected. As an illustrative example, the impact of the slower runtime performance on the TX2 for D1(a) is demonstrated in Fig. 5.13.

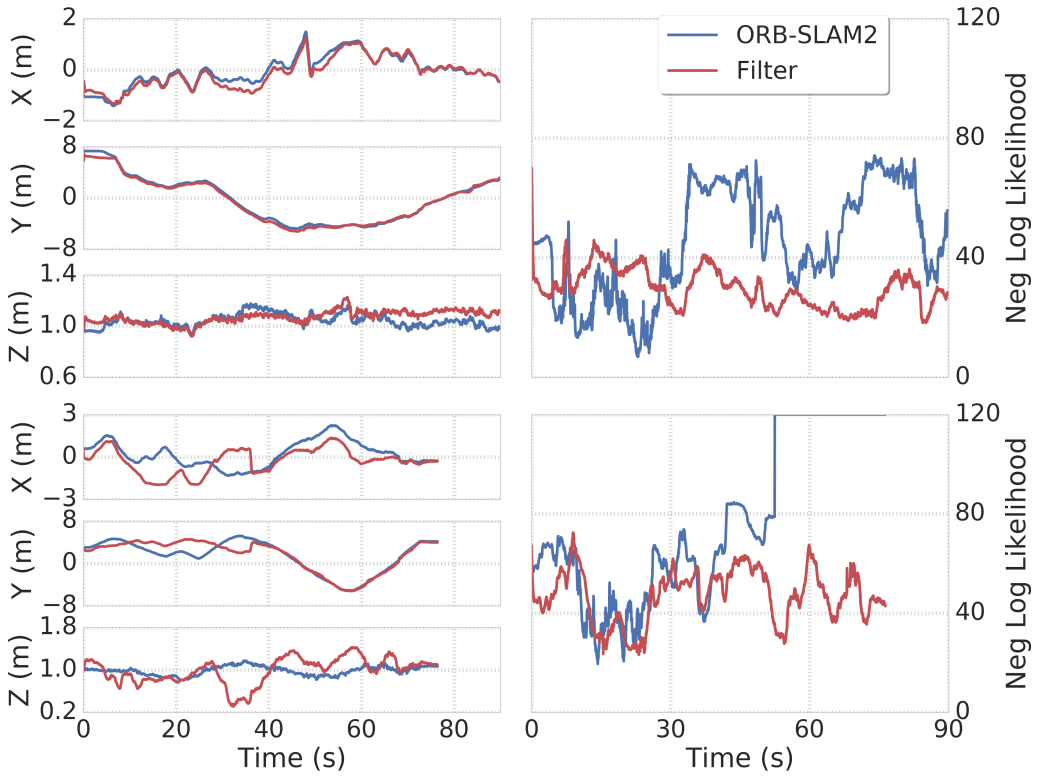


Figure 5.9: Comparison between the position and corresponding likelihood estimates for two runs from ORB-SLAM2 and our filter, respectively. *Top*: A nominal path with feature rich data, and *Bottom*: A path moving through regions of low observability. Contrast the continually increasing divergence (capped in the graph) of the ORB-SLAM2 estimate after moving through the feature poor region with the lower snapped negative likelihood values for the same locations for our filter. The corresponding poses and overlaid depth scan at approximately 55s is shown in Fig. 5.10. Due to a minimal overlap of the depth scan with the map for the ORB-SLAM2 frame, the likelihood value is very low.

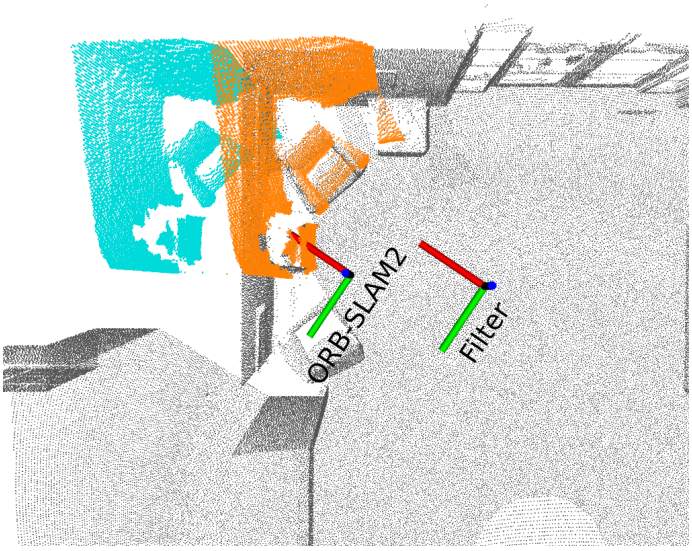


Figure 5.10: Comparison of registration of current sensor measurement at ground truth point cloud (gray) at ORB-SLAM2 pose estimation (cyan) and at the estimated filter pose (orange). The sensor measurement aligns with the ground truth point cloud in the filter estimate frame while the accumulated drift in the ORB-SLAM2 frame due to transition through a less feature rich region leads to poor alignment.

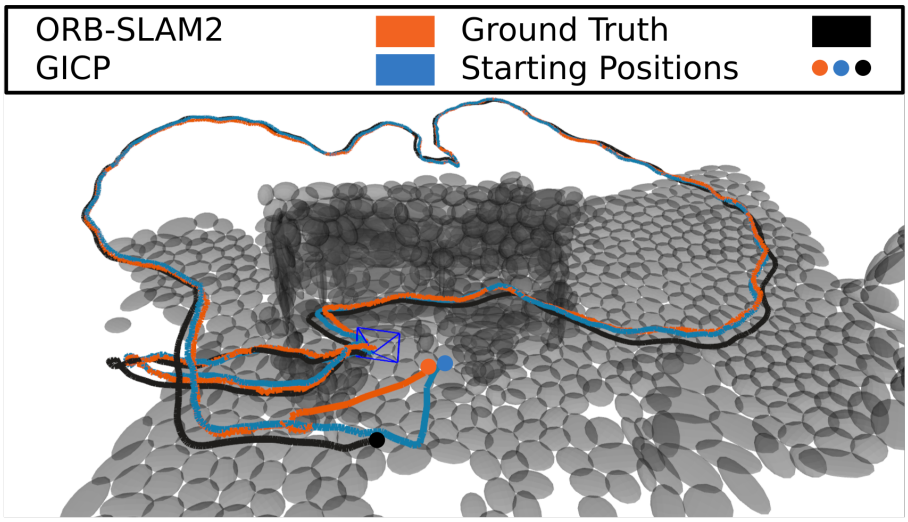


Figure 5.11: Comparison of our particle filter approach using ORB-SLAM2 frame-to-frame odometry (orange) and Generalized-ICP (cyan) as process models with ground truth pose (black) on TUM's Freiburg 3 Desk Dataset. The GMM representation of the world is created by stitching sensor scans using the ground truth pose estimates. The higher global error of our approach than that of ORB-SLAM2 can be attributed to the noisy reconstruction of the environment point cloud from the accumulated scans.

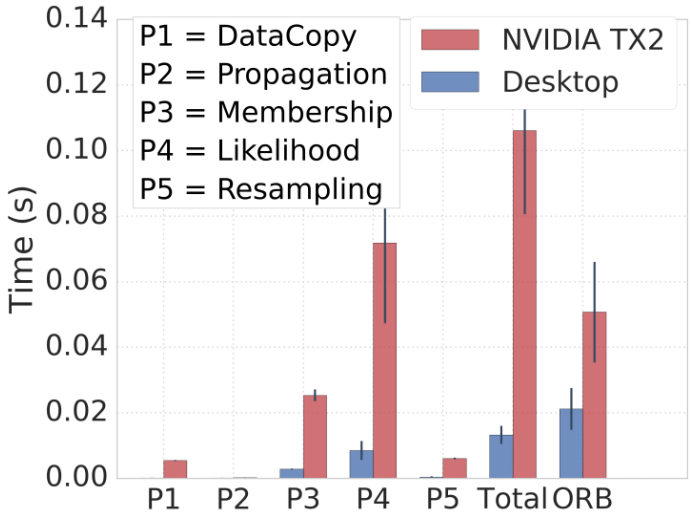


Figure 5.12: Execution time comparison for subcomponents of the algorithm for the D1(a) dataset on an Intel i7 desktop with an NVIDIA GPU and an embedded NVIDIA TX2 platform. Performance scales linearly with the number of CUDA cores. As a point of comparison ORB-SLAM2 runtime on the same dataset is faster on the embedded platform than on the desktop.

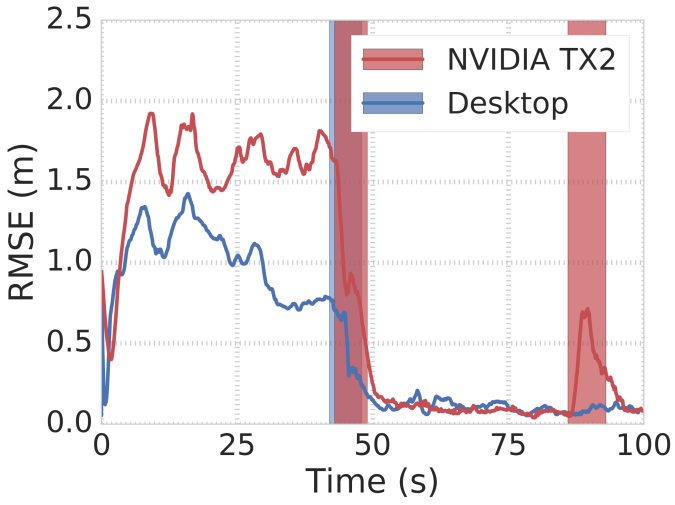


Figure 5.13: Comparison of the filter performance on the desktop with the NVIDIA TX2 on the D1(a) dataset. As the filter operates at a slower frame rate on the TX2 it initially exhibits a larger error but once the sensor observes a uniquely identifiable location, both trial sets converge to the ground truth location.

5.6 Conclusion

We present a framework to perform real-time localisation of depth sensors given a prior continuous spatial belief representation. Key to being able to do this is the ability to project a succinct representation into the image frame of the sensor to evaluate the likelihood of the data having originated from the given map for a given pose. By utilizing a fast likelihood computation approximation we can then perform robust particle filter localisation in real-time even on an embedded GPU platform.

Despite the apparent suitability of the likelihood function to an optimization based approach for more fine grained pose refinement, it is not so straightforward. For Gaussian components that are more flat (as would be expected for planar surface fits) the gradient profile can vary rapidly in the vicinity of the components leading to poor conditioning that can challenge traditional iterative Gauss-Newton descent strategies. Further, the absence of strong gradient information in spatial data as encoded by the necessarily smooth-by-construction representation hinders the application of such methods.

Chapter 6

MRFMapScape: MRFMap Submapping and Sensor Data Marginalisation with Gaussian Distributions

Having presented approaches to inferring both the posteriors of the full SLAM decomposition separately in the previous two chapters, this chapter presents an application of first steps of incorporating the MRFMap framework within a full SLAM problem. Specifically, it presents a practical approximation to

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}). \tag{2.4}$$

6.1 Introduction

There is an inherent trade-off within robotic mapping to choose between delaying the marginalisation of sensor data and bounding computational and spatial complexity. In addition to incorrectly modelled sensor noise uncertainty, even small pose errors can manifest themselves as potentially conflicting noisy sensor data measurements for a conventional occupancy grid mapping framework that assumes perfect pose information. An ideal formulation would retain all the sensor data collected up to the inference time to best maximise the joint likelihood of the sensor trajectory and map structure. This has historically not been feasible in practice and thus all conventional mapping algorithms have been designed to incorporate incremental updates.

In Chapter 4 we demonstrated that the MRFMap framework not only permits using the submapping approaches in a consistent fashion while incorporating forward sensor models, but also explicitly reasons about perspective information along each ray instead of treating sensor scans as end points. However, as presented earlier the framework does not scale with increasing data since the choice to defer inference is taken to the extreme. In contrast, convention occupancy grids and Gaussian process based mapping techniques immediately marginalise the sensor information, leading to map corruption due to sensor pose errors existing at marginalisation time.

One of the key drawbacks of using fixed sized grid based representations is the degree of redundancy and thus the memory inefficiency incorporated within. For long term robot operation it is imperative for the map representation to be able to scale in complexity. Further, it is not necessary for the map representation to have a fixed fidelity for all types of traversal. For instance, fast flight through cluttered environments only require the gross occupancy characteristics of the environment to be identified for flight concerns. Conversely, inspection tasks often require specific regions of the environment to be mapped to a greater level of specificity than others. An ideal map representation should be able to accommodate both these extremes and scale between them automatically based on time and compute budgets.

This chapter attempts to bridge the gap between the two extremes. We utilise the MRFMap framework to perform occupancy inference utilising forward sensor noise models and then marginalise older information by using the succinct parametrised mixture models that can later be used for instant likelihood evaluation. The parametric model is fit to keyframe perspectives rather than a fixed frame thus allowing for post-hoc correction in their location. Further, inference over pose posteriors is done via a pose graph.

Thus, this chapter presents two key ideas:

- Using MRFMap within submapping to provide a more accurate and probabilistic fusion of submaps within pose graph optimisation by avoiding premature marginalisation due to incremental mapping; and
- Utilising a succinct volumetric surface representation to compress occupancy information for older inactive regions to bound memory growth.

6.2 Related Work

For a mobile agent using a map it is imperative that the growth of the map as it explores the environment is not unbounded. In order to be able to reason based on past experiences and revisit old locations, it is not sufficient to only maintain local maps. Although utilising

hierarchical data structures such as OctoMap [38] help in exploiting the sparse nature of occupancy in most environments, they are not sufficient to bound the constant increase in memory. This gives rise to the idea of using a moving active volume within a larger environment that moves along with the agent and previously inferred regions cached to disk [35, 94]. This can be potentially implemented by suitably pre-caching regions of the map as determined by predictive visibility criteria.

Typical approaches to circumvent the issue of prematurely marginalising out the sensor information in the map via unoptimised, inaccurate poses have used submaps as locally consistent map representations that can be later stitched together on receiving updates to the respective keyframe poses. These updates to poses are often performed by pose graph based optimisation algorithms that take into account sources of odometry and loop closures to mitigate accumulated drift. Here we focus primarily on approaches that are used for globally consistent robotic planners.

Occupancy grid based submapping approaches such as [73] utilise updated poses from a pose graph to update the underlying octomap representation. In addition to losing fidelity due to effectively resampling from the compressed representation, the approach 'undos' previous updates and reapplies them at the new pose using log-odds updates. These updates are made possible again due to the use of independent cells approximation, but on account of working on already marginalised data are not very precise. Other recent approaches employing hybrid T-SDF and E-SDFs such as [59] utilise TSDF sub-volumes that are fused after passing a pose estimate quality check. This measure is inversely proportional to the L_2 norm of the covariance matrix of their estimated relative pose. Similar to the log-odds based approach, the fusion process involves taking two sets of previously marginalised data and aggregating them together. Neither of these class of approaches perform probabilistic updates that reason about the sensor formation. Using MRFMap within either of the two frameworks can provide significant accuracy benefits and a much more principled fusion of sensor information that can be used with frequently changing camera poses even within a local frame.

6.3 Pose Graphs as Inference over Trajectories

Recall the decomposition of the full SLAM problem in Eq. 2.4

$$p(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}). \tag{2.4}$$

MRFMap provides us an excellent means to obtain the posterior map distribution given a trajectory estimate. We now shift our focus to the posterior over trajectories. Although we presented a means of computing a posterior over trajectories in Chapter 5, here we demonstrate the incorporation of MRFMap with the common pose graph optimisation solution used within submapping.

Pose graph optimisation aims to obtain the maximum likelihood estimate of the trajectory given relative odometry measurements and loop closures between poses. Pose graphs are an instance of a unimodal Gaussian maximum likelihood estimate of this trajectory posterior, with loop closures being the only contribution from the sensor inputs $\mathbf{z}_{1:t}$. Modern day implementations utilise factor graphs to perform efficient updates to the variables and factors in question. We follow the approach of [73], where on achieving loop closure in an iSAM2 [41] framework, we add the poses of the keyframes that demonstrate a significant change in pose in the full inference window

$$\|T_{new} \ominus T_{old}\| > \epsilon, \tag{6.1}$$

where ϵ is a small threshold value, and $\ominus$ is the inverse composition operator [71]. In our experiments we choose ϵ to be 0.5 metric units.

6.4 Implementation Details

6.4.1 Active Co-visibility Based Window

Co-visibility graphs as in [50, 79] employ an inner window of point-pose constraints as in Bundle Adjustment (BA), and the number of visible features in multiple cameras to determine constraints between two poses for an outer window on the graph. In our approach we borrow the construction of the active window from [97] that splits the active window into a temporal part and a co-visible part. The temporal window ensures that the most recent keyframes are always involved in the inference, while the co-visible windows include activable keyframes that have the most overlap with the latest keyframe. The advantage of the latter is to allow incorporating data from distant perspectives. Each keyframe in the active window is utilised to do full BP inference and gets its own copy of the sparse voxel atlas. These copies of the volume are necessary in order to store the outgoing message per voxel from a given perspective. Co-visibility is determined by the number of atlas bricks that project into the camera frame. A graph keeps track of the pairwise co-visibility weight between each set of keyframes. On exiting the active window the camera is retained in the list of frames that can be activated until the maximum

limit of activable camera keyframes is reached. At this point a keyframe is chosen to be marginalised into a parametric mixture model using a distance heuristic that prefers retaining keyframes that preserve parallax [97]. The choice of when to marginalise the activable keyframe can also be made on more principled measures, such as an L_2 norm on the marginal pose covariances, as presented in [48, 73].

6.4.2 Marginalising Sensor Data using Gaussian Mixtures

For the purpose of marginalising older sensor information, we utilise Gaussian distributions fit to inferred depth images. The compressive benefits of these approaches have been highlighted in the previous Chapter 5. The rest of this section details our proposed sensor data marginalising scheme.

Generating expected depths

Given a MRFMap inference instance, for a given camera, for each ray potential we can estimate the distribution of likely depth by marginalising all the other occupancy nodes along the ray. From Eq. 4.11 we had

$$\mu_{\psi \to d}\left(d = d_i\right) = \nu(d_i)\mu(o_i = 1)\prod_{k<i}\mu(o_k = 0). \tag{4.11}$$

This depth distribution along the ray takes into account messages from other rays passing through voxels along the specific ray and thus is often multi-modal. One approach to generate the expected depth image would be to take the maximum likelihood generating surface. This can be done by operating with a L_1 norm on the distance under this depth distribution [88]. The choice of the L_1 norm ensures that the median mode of the distribution is chosen as the maximum likelihood generating surface depth. Thus, for each pixel in the keyframe we can obtain the MLE expected depth given all the other frames that also view the same set of voxels. These expected depth images can then be backprojected to 3D and Gaussian components fit to the spread of the backprojected 3D points to approximate the same depth distribution along the ray.

An alternative approach would be to sample 3D points for each ray corresponding to the depth distribution prior to backprojecting them to 3D. The resulting fit Gaussian components would potentially provide a better approximation to the depth distribution. However, since the RGB-D camera noise characteristics we utilise are so precise in comparison with map resolutions used (order of a few cm), the resulting depth distributions are not multi-modal over large cell spans, and thus the L_1 norm suffices to represent the

distribution. See appendix B.8 for a more detailed look at this phenomenon. However, for sensor models that are noisier, a sampling approach might better encapsulate potential multi-modal behaviour.

Fitting GMMs to inactive regions

For a given set of bricks deemed as being inactive, we look up their keyframe pixels and obtain the maximum likelihood generating surface depth distance as per Eq. 4.11 and then fit parametric GMMs to these virtual point clouds. At the time of marginalisation, the connectivity graph between all observed keyframes is used to select the cameras that have high overlap with the chosen camera and inference is performed on the subset. The incoming messages from the voxels from all the other camera views are then taken into account while determining the expected depth distribution. After fitting the Gaussian distribution the camera contribution to the active window is subtracted and the GPU memory for the image is deallocated. In practice due to projective geometry we do not fit the data via expectation maximisation over the span of the entire map, but do so within each brick volume as an approximation of the span of space that Gaussian components would impact an individual ray's likelihood. Thus, much like NDT-OM [65] we store a Gaussian distribution per brick, with the important distinction that the 3D points used to fit the distributions include probabilistic perspective information from multiple keyframes.

6.4.3 Extension of Ray Likelihood to Gaussian distributions

Here we demonstrate that the conditional distribution of a ray tracing through a mixture of Gaussians in 3D is a mixture of Gaussians in 1D. This property can then be used to obtain one step analytic expressions of ray measurement likelihood instead of ray tracing through the grid volume for keyframes within the active window of the MRFMap.

Consider a 3D point $\boldsymbol{p}^w = [x^w, y^w, z^w]^T$ in world space. This can be modelled to be sampled from a Gaussian mixture model, a commonly used mixture model where each component is a multivariate Gaussian [19]. If λ_i is a boolean variable that indicates whether $\boldsymbol{p}^w$ was generated by the component i, the joint distribution for k components is represented as

$$p(\boldsymbol{p}^w) = \sum_{i}^{k} p(\lambda_i)\mathcal{N}(\boldsymbol{p}^w|\boldsymbol{\mu}_i^w, \boldsymbol{\Sigma}_i^w), \tag{6.2}$$

where $\sum p(\lambda_i) = 1$.

Intuitively, a given ray from a camera will correlate the x,y,z coordinates for an arbitrary point and reduce the dimensionality of the Gaussian component along itself. Let the point

be observed from the camera frame, corresponding to a rigid body transform T_w^c. The GMM is then transformed as

$$p(\boldsymbol{p}^c) = \sum p(\lambda_i)\mathcal{N}(\boldsymbol{p}^c \mid \boldsymbol{T}_w^c \boldsymbol{\mu}_i^w, \boldsymbol{R}_w^c \boldsymbol{\Sigma}_i^w \boldsymbol{R}_w^{c\,\mathrm{T}}) \tag{6.3}$$

$$\Rightarrow p(x^c, y^c, z^c) = \sum p(\lambda_i)\mathcal{N}\left(\begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} \middle| \, \boldsymbol{T}_w^c \boldsymbol{\mu}_i^w, \boldsymbol{R}_w^c \boldsymbol{\Sigma}_i^w \boldsymbol{R}_w^{c\,\mathrm{T}} \right). \tag{6.4}$$

After dropping the camera superscripts for brevity, we can consider the partition $[x, y]$ and $[z]$, and find the conditional distribution of the z variable. Thus, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are then correspondingly partitioned as

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_{xy} \\ \mu_z \end{bmatrix}, \text{ and } \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_{xy} & \boldsymbol{\Sigma}_{xy,z} \\ \boldsymbol{\Sigma}_{z,xy} & \sigma_z^2 \end{bmatrix}, \tag{6.5}$$

where we drop the i suffix for notational brevity.

Using the law of total probability, we can express

$$p(z|x, y) = \sum p(\lambda_i|x, y)p(z|x, y, \lambda_i). \tag{6.6}$$

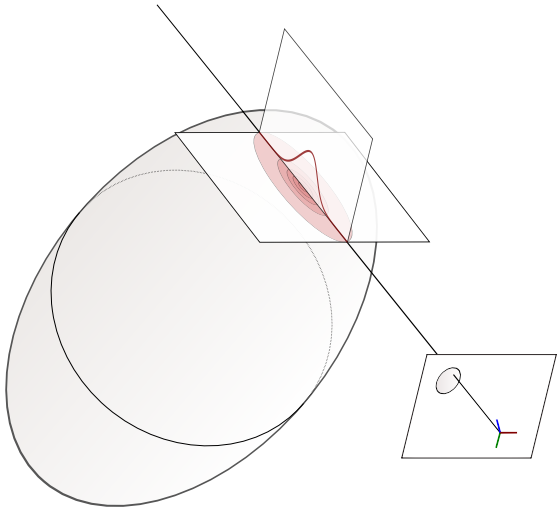


Figure 6.1: The conditional of a 3D Gaussian along a ray is also Gaussian shaped, although it is not a probability density function by itself. The projection of the 3Σ ellipsoid bounds of a 3D Gaussian in the image plane is an ellipse. See appendix for more details.

The first term, is the conditional probability of choosing each component and can be

obtained via Bayes' rule

$$p(\lambda_i|x,y) = \frac{p(\lambda_i)p(x,y|\lambda_i)}{p(x,y)} \tag{6.7}$$

$$= \frac{p(\lambda_i)p(x,y|\lambda_i)}{\sum_j p(\lambda_j)p(x,y|\lambda_j)}, \tag{6.8}$$

which can be seen as the weights of the marginalised 2-D distribution of the i^{th} Gaussian component, $p(x,y|\lambda_i) = \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix} \mid \boldsymbol{\mu}_{xy}, \boldsymbol{\Sigma}_{xy}.\right)$

The second term is the 1-D conditional distribution of the i^{th} Gaussian component with a mean and covariance

$$\mu_{z|x,y} = \mu_z + \boldsymbol{\Sigma}_{z,xy}\boldsymbol{\Sigma}_{xy,xy}^{-1}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \boldsymbol{\mu}_{xy}\right) \tag{6.9}$$

$$\sigma^2_{z|x,y} = \sigma^2_z - \boldsymbol{\Sigma}_{z,xy}\boldsymbol{\Sigma}_{xy,xy}^{-1}\boldsymbol{\Sigma}_{xy,z}. \tag{6.10}$$

In order to evaluate this conditional distribution along a ray we rotate the camera Z axis to align with the ray direction. This is equivalent to rotating the component means and covariances by an additional rotation matrix. This permits us to set x, y terms in Eq. 6.9 to be 0 instead of a parametric equation and simplifies evaluation to use pixel coordinates u, v

$$p(s) = \sum p(\lambda_i|u,v)p(s|u,v,\lambda_i). \tag{6.11}$$

This corresponding distribution is then used in the subsequent steps. The alternative would be to perform a transformation of the Gaussian variables via a perspective projection, which however no longer retains the linear Gaussian mixture model nature of the components due to the non-linear transformation involved in camera projection.

We can then utilise these GMM components instead of the probabilistic map used in Eq. 4.19 to evaluate the likelihood of a depth image measurement

$$p(\mathcal{Z}_r) = \int_0^\infty p(s)p(\mathcal{Z}_r|s)ds. \tag{4.19}$$

If we know the Gaussian components along the ray, which can be obtained by a simple projection technique, then the first term is simply the conditional of the GMM along the ray, and the second term is a Gaussian centred around the measurement $\mathcal{Z}_r$. On expanding

the integral, we have

$$p(\mathcal{Z}_r) = \int_0^\infty \sum_i^m p(\lambda_i) p(s|\lambda_i) p(\mathcal{Z}_r|s) ds + vis_\infty p_\infty(\mathcal{Z}_r) \tag{6.12}$$

$$= \sum_i^m p(\lambda_i) \int_0^\infty p(s|\lambda_i) p(\mathcal{Z}_r|s) ds + vis_\infty p_\infty(\mathcal{Z}_r), \tag{6.13}$$

where vis_∞ is the residual visibility when a ray hits the edges of the map boundary, $p_\infty(\mathcal{Z}_r)$ is an experimentally obtained terminal probability depending on the ray measurement as discussed in Sec. 5.3.3, and $p(s)$ is the 1-D conditional distribution of the GMM along the ray obtained in Eq. 6.11. Now, since both the terms within the integrand are two 1-D Gaussian distributions, and we only consider components ahead of the camera, we can obtain an approximate closed form solution of the product of the two distributions using standard properties of Gaussian distributions. We thus have

$$p(\mathcal{Z}_r) \approx \sum_i^m p(\lambda_i) \mathcal{N}\left(\mu_i | \mathcal{Z}_r, \sigma(\mathcal{Z}_r)^2 + \sigma_{z|x,y}^2\right) + vis_\infty p_\infty(\mathcal{Z}_r), \tag{6.14}$$

where we make a similar assumption as in Eq. 4.20 and assume a constant average sigma noise and bias of the measurement at the $1.5\sigma_{z|x,y}$ intervals about the 1-D conditional component mean. Further due to insignificant overlap, distant components can be safely ignored and effectively only the nearest component to the measurement remains relevant. This thus provides us an approximate closed form measure of the sensor data likelihood given marginalised Gaussian map components that is fast to evaluate and does not involve any ray tracing through the volume.

6.5 Results

This section details initial experiments and results of the proposed implementation. They serve to highlight:

- The ability of MRFMaps to defer marginalisation until required and thus incorporate pose graph updates seamlessly;
- The efficacy of the active windowing strategy in reducing the unbounded memory consumption of batch settings; additional
- Qualitative and quantitative comparison of the impact of the marginalised sensor data in terms of sensor likelihood.

6.5.1 Pose Graph Optimisation

For these preliminary results we utilise an iSAM2 based smoother within GTSAM as the pose-graph backend. This runs in parallel with the MRFMap mapping thread and asks the inference to update the camera poses for only the keyframes for which the pose changes more than a fixed geometric displacement as expressed in Eq. 6.1. As specified earlier here we use an ϵ of 0.05 metric units. We utilise the ICL-NUIM `livingroom1` dataset and for the purpose of this demonstration artificially inflate the rotational odometry by 10%. As expected, this results in a map that is severely distorted (see Fig. 6.2). We then inject a ground truth odometry edge every 500 frames out of a total of roughly 2800 frames in the dataset and demonstrate the output of the inference. Even with just five loop closure edges, the resulting map is qualitatively much better. See Fig. 6.2

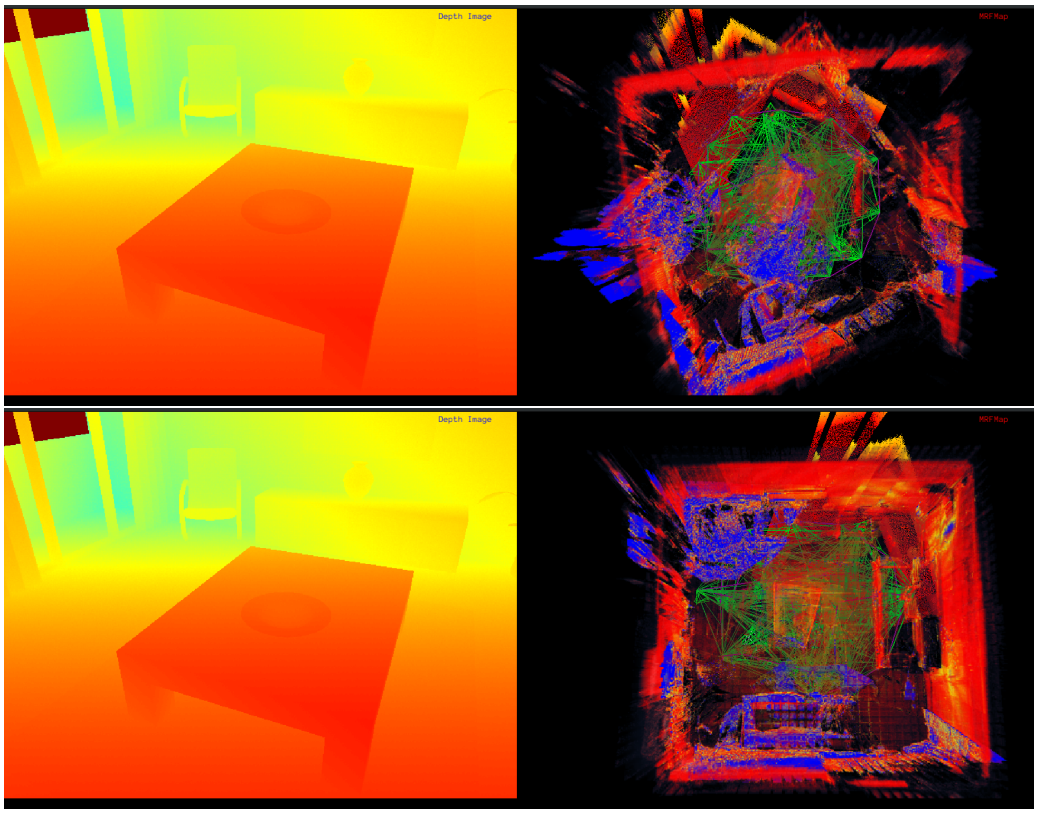


Figure 6.2: Qualitative comparison of occupancy inference with and without Pose Graph Optimisation. The odometry for the `livingroom1` dataset is artificially inflated by 10% in rotation and used to infer the MRFMap without (Top) and with (Bottom) pose graph optimisation.

6.5.2 Compute Time and Memory Usage

Since each additional keyframe within the MRFMap implementation utilises an auxiliary channel to store outgoing messages per activated voxel, the memory cost increases significantly as the map bounds and the number of keyframes increase. Marginalising older keyframes as Gaussian distributions as keyframes move out of the active window bounds the compute time and reduces the growth in memory usage substantially. The plots for running the implementation on the `livingroom1` dataset are shown in Fig. 6.3.

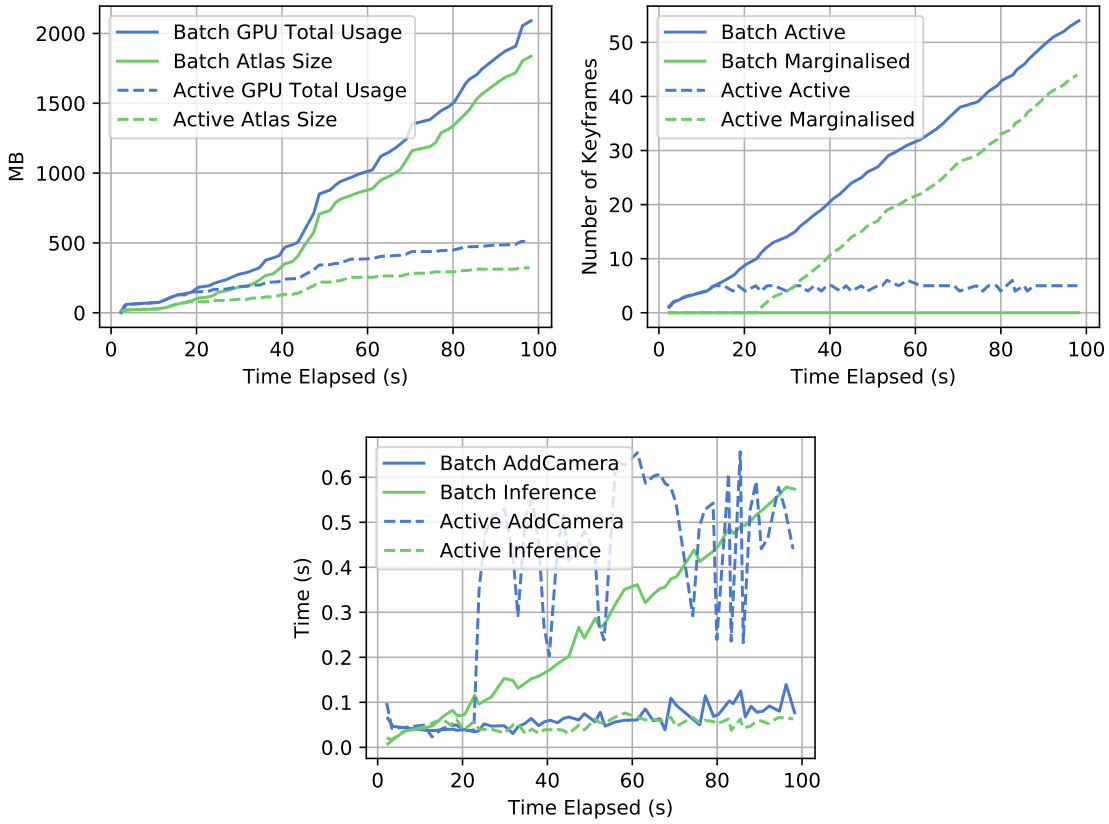


Figure 6.3: Memory usage and timing comparison. Top: Memory usage comparison between the batch mode inference and version with an active window. The active window maintains only a fixed number of keyframes for inference and the sensor data for frames moving out of the window are marginalised. Bottom: Elapsed time for the time taken to add a camera to the active window and that to perform inference of the two approaches. The marginalisation implementation is not optimised and thus the time taken to add a new keyframe is relatively high and introduces the variability.

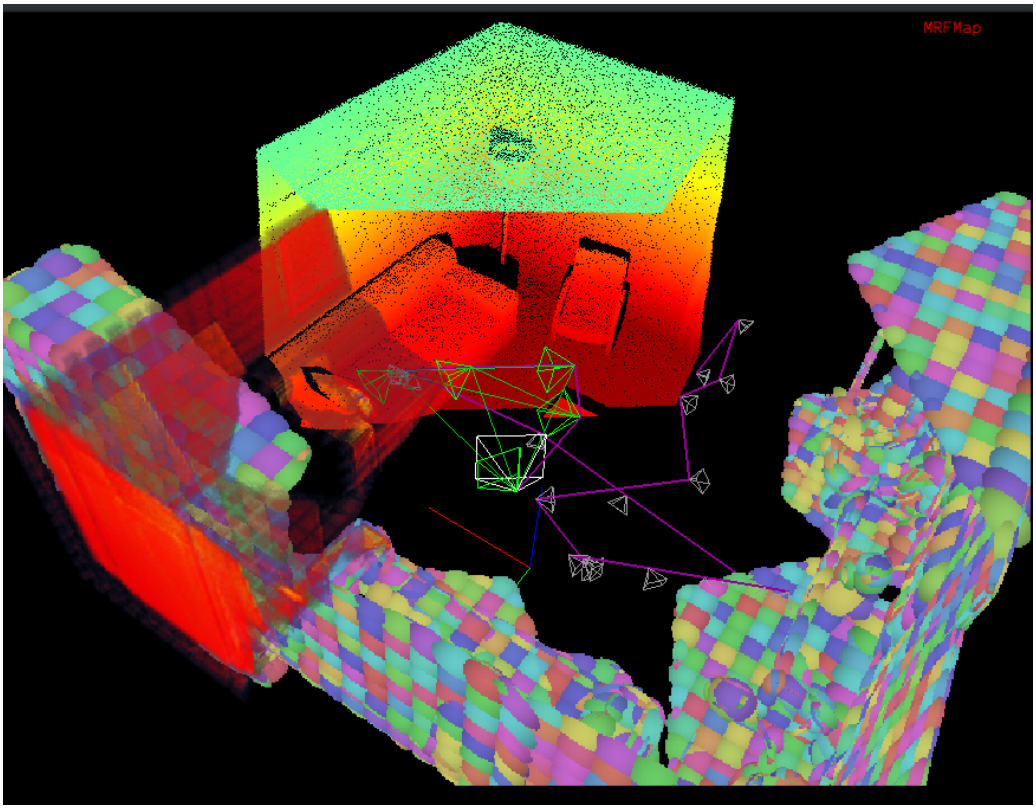


Figure 6.4: Screenshot of the active window and marginalised sensor measurements overlaid on the map with the current keyframe poses. The active keyframes are shown by green coloured frustums while the inactive keyframes are shown in gray. The current frame is shown by the large white frustum with the current backprojected depth image pointcloud rendered as well. 3Σ bounds of the marginalised Gaussian distributions fit to older keyframes are rendered with random high saturation hues. Current marginal occupancy of the MRFMap is shown in a volumetric deep brick rendering in red.

6.5.3 Expected Depth Image and Gaussian Mixture Fitting

We present a sample maximum likelihood expected depth image computed as explained in Sec. 6.4.2 and a view of the projected marginalised Gaussian distributions for the data in Fig. 6.5. Further, we evaluate the variation of the likelihood field in the vicinity of the true ground truth pose of a depth image, and show that the negative log-likelihood plots even at coarser brick discretisations show a characteristic minima close to the true pose. This suggests that the marginalised parametric representation can be used for multi-hypothesis localisation purposes as well (see Sec. 5.5.2).

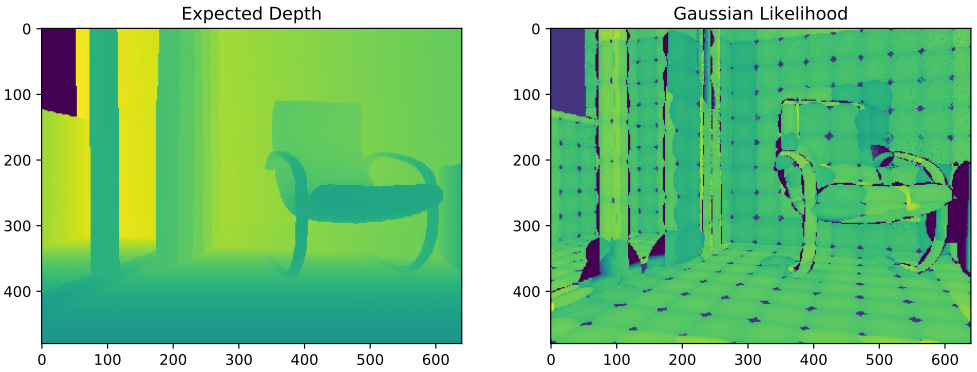


Figure 6.5: The expected depth image from a keyframe before its data is marginalised (Left) and afterwards when we fit Gaussian distributions to its expected depth image (Right).

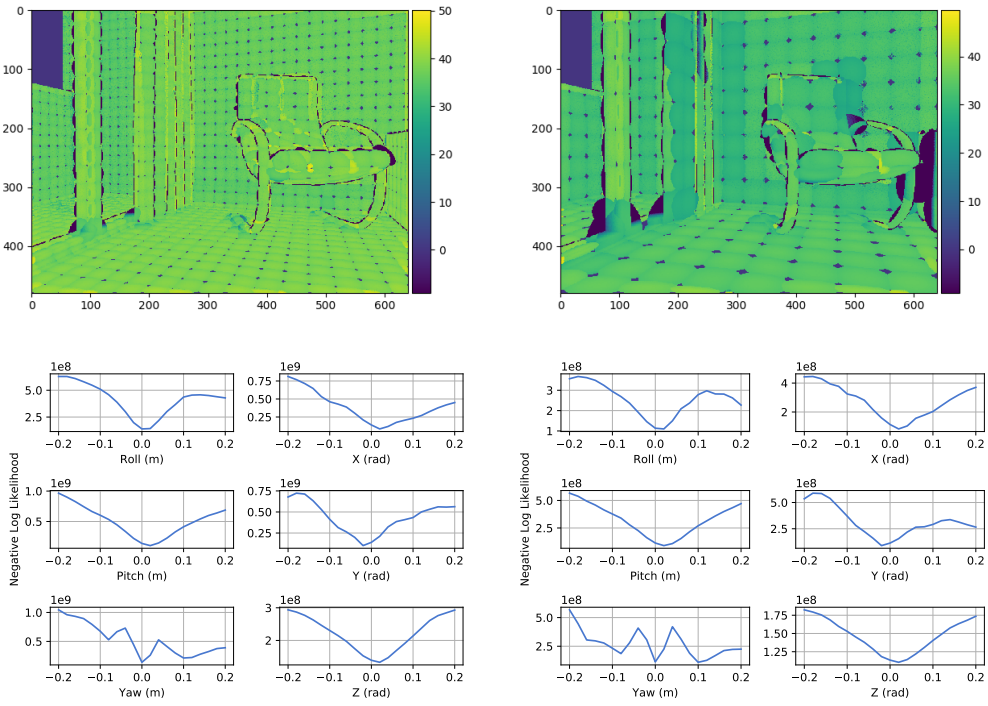


Figure 6.6: Comparison of the negative log likelihood plots of ground truth depth images in the vicinity of the ground truth camera pose using the marginalised parametric representation at voxel resolutions of $0.01\ m$ (Left) and $0.02\ m$ (Right), respectively. Note the similarity to the plots in Fig. 5.2.

6.6 Conclusion

In this chapter we demonstrated the extension of the MRFMap framework to a real-time capable system by utilising a co-visibility based active window and marginalising old inferred sensor depth information in the form of a succinct parametric representation. The representation tries to be faithful to the inferred depth distribution while still being much more sparse than a dense gridded structure. Initial results also demonstrate the viability of the succinct representation to be used for localisation as has been demonstrated previously in Chapter 5. Another interesting insight is that on account of incorporating forward sensor noise within the inference, we can obtain high fidelity results without the need of incorporating aggregated sensor data as is done with other approaches thus leading to higher data efficiency.

Chapter 7

Conclusion

Robust robotic operation requires a tightly coupled amalgamation of a series of components. State estimation is often the most critical path, and all too often perception and control are treated as independent separate blocks. However, state estimates can be informed by knowledge of the system - from high level preference of trajectories to even low level system dynamics. Similarly, control and planning algorithms can exploit map-based uncertainty to make locally deliberate decisions. Premature decisions made earlier in the pipeline may render certain later operations infeasible, and, at worse, can have catastrophic consequences.

It is thus desirable to retain as much information as possible in prior steps of the autonomous pipeline in order to make informed decisions downstream in an operating paradigm. However, this is often at odds with computational feasibility, especially on robotic agents, and thus by virtue of intelligent assumptions and approximations we endeavour to incrementally achieve robust autonomous operation. Further, there is also fundamentally a redundancy in sensor information, and there exists an information bottleneck that limits the capability of any agent to process vast amounts of information. Robotic mapping, thus, deals with the exciting challenge of retaining as much informative content from sensor data as possible while minimising redundancy.

In this thesis we presented an inference framework for estimating marginal occupancy inference using forward sensor models that runs in real-time. This is significant because to the best of our knowledge this is the first result in robotics literature that computes the inference over the joint distribution of occupancies in real-time at centimetre level volumetric resolutions. Further, by virtue of delaying sensor data marginalisation until required, the framework achieves an elegant probabilistic submapping solution that does not include discretisation errors when merging submaps in a local window. Finally, the

framework suggests exciting avenues for future work that can have potentially very high impact.

This thesis limits the scope to performing occupancy inference for a specific class of dense depth sensors. However, the framework can be used as a general system applicable to a wider variety of sensor types. For instance, with sparser depth data such as that from laser rangefinders it is often necessary to impose structural regularity across voxels to get denser maps. It is feasible to add pairwise smoothness constraints to voxels within ray belief propagation to encourage nearby voxels to take the same occupancy label [88]. Further, more complex forward sensor noise models can be incorporated within the inference framework than the simple bias and noise models that we utilise, for instance for sensors with wider beams such as sonar. Since the framework explicitly reasons about the distribution of possible generating surfaces along the ray, the framework can feasibly work with such sensor data.

In the pursuit to compress as much information as possible while retaining fidelity, this work also demonstrates the application of succinct parametrisations to reduce redundancy of representation. However, there is still a lot more data that can be compressed by utilising richer information about the environment. For instance, knowing that a span of matter is a wall should allow us to both more sparsely represent the spatial extents, and also reason about other regions connected to it. Recent developments in the field such as [9, 23] are incredibly exciting since they serve as excellent proofs of concept that a lot of the spatial and appearance information can be represented in a much smaller subspace. The challenge then lies in combining the strengths of data-driven processes and making them generalisable, interpretable, and consistent using physically correct models and principled uncertainty propagation.

7.1 Avenues for further Research

As final comments, we detail some promising avenues for further research based off the MRFMap framework.

7.1.1 Inference over Multimodal Trajectories and Occupancy

One of the most exciting future applications of the MRFMap framework would be the inference over the multimodal joint posterior over trajectories and occupancies, a problem that has so far been out of reach for 3D volumetric full SLAM. For 2D sensor data FastSLAM [82] introduced particle filters to tackle the full SLAM problem where each particle estimated its

robot trajectory as well as its own copy of the map. This system explicitly models multiple hypotheses of the world given the sensor data, and incorporates non-linearities present within state propagation. Further, the fact that the amount of computation required per incremental update is constant and each map component is updated independently of the others makes the approach amenable to GPU parallelisation. We envision being able to use MRFMap within a FastSLAM context since it allows the occupancy volume to be inferred on demand and only requires the sensor information to be retained per keyframe. The sensor information can be shared amongst all particles, thus eliminating the need of maintaining expensive concurrent multiple high-dimensional map hypotheses. This however hinges on the ability of the map representation and the likelihood measurement to be convex in the vicinity of the true pose so that most of the support of the proposal particles is drawn from a region around the local maximal scan aligned location. This is indeed the case for maps inferred with the MRFMap formulation, as is evidenced by Fig. 6.6, and is a sufficiently convincing demonstration of the ability of this formulation to work well within a particle filter implementation. Other recent backends such as multi-hypothesis Bayes tree [39] and belief-propagation based parallelised bundle adjustment [60] could also be excellent fits within such a joint inference framework.

7.1.2 Multi-resolution Parametric Map Representations

Recent work in hierarchical GMMs has shown the ability to encode depth maps in very concise representations that scale in fidelity. In this thesis, we have formulated and evaluated a particle filter based strategy for localizing within a GMM map of a fixed hierarchy in Chapter 5. We have also demonstrated a means of building up these representations with inferred depth distributions that take forward noise models into account 6. In the framework we don't specify the choice of the map representation as long as the individual components are independent given the trajectory. Thus, instead of a grid map, we can use a set of mixture components and update them per particle to build up maps that scale with time. Further, the resolution of these models can be arbitrary, and data driven techniques could be used for fitting models at only the smallest resolution that they need to be represented at, thus minimising redundancy.

7.1.3 Including Appearance Potentials and Semantic Labels

The ability to add metadata within the mapping process can enable a host of other applications, e.g., we could add a per-pixel classification score (say of rooftop damage) to each voxel to help actively map out regions of damage in an inspection scenario by focussing

compute budget on regions of the world that require it more. Another example would be utilising per-pixel class object labels to help impose semantic constraints to reconstructions of objects such as furniture in living room environments to capture difficult to reconstruct thin surfaces from multiple perspectives or under occlusion. At the high level, for potentials that we have direct depth observability over, the occupancy will converge quickly to the final value, whereas for voxels with only appearance based information, the convergence will take longer in a sort of near-far inference. This is effectively very similar to the model used in [88] in that within the MRFMap depth ray potential factor the model would have an additional random variable a that could take labels $\in 0, 1 \dots k$ for each of the occupancy locations, where k is an appropriately determined discretisation. Since the appearance of a particular cell is also a surface property, and is dependent on the first occupied cell, the ray potential would now connect appearance, occupancy, and the ray depth nodes. Note that this allows the framework to potentially not directly observe the depth label and still infer the depth as long as the rays potentials share members due to the implicit photometric consistency introduced by adding the appearance data, going truly beyond the capability of GMM-based spatial mapping, TSDFs, and Gaussian Process mapping algorithms.

7.1.4 Inferring Anisotropic Voxel Occupancy

Discretising the environment into voxels involves making an implicit assumption about the occlusion density being constant within that span. This assumption starts breaking down rapidly as voxel sizes increase. For applications on platforms with limited compute and memory it is however desirable to use as large voxels as possible. MRFMap as a framework can permit inferring different occlusion densities given different perspectives. For instance, a majority of rays going through a voxel looking head on might suggest high occlusion whereas one from the side wouldn't (as in the case of thin surfaces). The current implementation ends up averaging all outgoing messages to a voxel from one perspective to provide unimodal updates. However, voxels can store multi-modal messages that store incoming messages from different directions. Exploiting this property could lead to being able to use much larger voxels than currently possible while only nominally increasing storage costs per voxel, which could further enable additional downstream tasks that are blocked by memory constraints.

Appendices

Appendix A

Rendering Covariance Ellipsoids with GLSL

A.1 Objective

Being able to render generic Gaussian components into the camera frame can facilitate projective correspondence. In this document we discuss utilising OpenGL to project covariance ellipsoids to the image frame to do so.

A.2 Quadrics

A quadric is a 2D quadratic surface embedded in 3D, called so because of the general form

$$\boldsymbol{x}Q\boldsymbol{x}^T + P\boldsymbol{x}^t + R = 0. \tag{A.1}$$

Using homogeneous coordinates $\boldsymbol{x} = (x, y, z, 1)^{\mathrm{T}}$ this can be expressed in a convenient matrix form

$$\boldsymbol{x}^T Q\boldsymbol{x} = 0. \tag{A.2}$$

For conics, the matrix $\boldsymbol{Q}$ is symmetric and is also invariant under perspective projections

$$\boldsymbol{Q} = \begin{bmatrix} A & B & C & D \\ B & E & F & G \\ C & F & H & I \\ D & G & I & J \end{bmatrix}. \tag{A.3}$$

A.3 Ellipsoid

The general equation of an axis-aligned ellipsoid centred at the origin is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1. \tag{A.4}$$

In the form of the previous conic matrix, this can be written as

$$\boldsymbol{x}^T \begin{bmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 & 0 \\ 0 & 0 & \frac{1}{c^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \boldsymbol{x} = 0. \tag{A.5}$$

In fact, all conics can be obtained by transforming a distinct basis matrix $\boldsymbol{D}$ that is diagonal with elements $d_{ii} \in \{0, \pm 1\}$ [70]. For ellipsoids this basis is the unit sphere

$$\boldsymbol{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{A.6}$$

We can obtain an ellipsoid from this unit sphere by performing an affine transformation. Recall that the structure of the conic is invariant under perspective transformations since it is linear. For any affine transformation $\mathbb{T}$ applied to coordinates $\boldsymbol{x}$, we have

$$\begin{aligned} \boldsymbol{x}' &= \mathbb{T}\boldsymbol{x} \\ \Rightarrow \boldsymbol{x} &= \mathbb{T}^{-1}\boldsymbol{x}'. \end{aligned} \tag{A.7}$$

Substituting in equation

$$\begin{aligned} \left(\mathbb{T}^{-1}\boldsymbol{x}'\right)^T \boldsymbol{D} (\mathbb{T}^{-1}\boldsymbol{x}') &= 0 \\ \Rightarrow \boldsymbol{x}'^T \mathbb{T}^{-T} \boldsymbol{D} \mathbb{T}^{-1} \boldsymbol{x}' &= 0 \\ \Rightarrow \boldsymbol{Q} &= \mathbb{T}^{-T} \boldsymbol{D} \mathbb{T}^{-1} \end{aligned} \tag{A.8}$$

For an ellipsoid, this matrix is composed via a scaling, rotation, and translation matrix, respectively.

$$\mathbb{T} = \boldsymbol{TRS}. \tag{A.9}$$

Thus, an ellipsoid can be expressed in a parametric form as

$$\boldsymbol{x}^T(\boldsymbol{TRS})^{-T}\boldsymbol{D}(\boldsymbol{TRS})^{-1}\boldsymbol{x} = 0. \tag{A.10}$$

Deconstructing an ellipsoid into this so-called parametric form enables convenient projection operations as we shall see in Sec. A.5.

A.4 Rendering Gaussian Components

A 3D multivariate Gaussian component is characterised by a mean and an associated covariance. When fitting spatial data to a GMM using EM, a practical observation is that most of the spatial data mass coincides with the component location and span. It is thus often helpful to visualise the 3σ spatial support of the normal distribution. This is expressed as

$$\text{supp}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \approx (\boldsymbol{x} - \boldsymbol{\mu})^T\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}), \tag{A.11}$$

where $\hat{\boldsymbol{\Sigma}}$ is the scaled covariance matrix, where the support operator supp is the approximate domain for which the Gaussian component provides significant probability density.

Now, a covariance matrix can be considered an ellipsoid where the basis has been scaled by the primary axes and then rotated. Further, $(\boldsymbol{x} - \boldsymbol{\mu}) = \boldsymbol{T}\boldsymbol{x}$ is simply the operation of a translation matrix parametrised by $-\boldsymbol{\mu}$ on the vector $\boldsymbol{x}$. In order to write this span in the parametric form, we can utilise the singular value decomposition to obtain the scale and rotation matrices as

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T, \tag{A.12}$$

where $\boldsymbol{U} = [\boldsymbol{u_1}\boldsymbol{u_2}\boldsymbol{u_3}]$ is the decomposed rotation and $\boldsymbol{\Lambda} = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2)$. We can split this as

$$(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) = \boldsymbol{x}^T\boldsymbol{T}^T\boldsymbol{U}\boldsymbol{\Lambda}^{\frac{-1}{2}}\mathbb{I}_{3\times3}\boldsymbol{\Lambda}^{\frac{-T}{2}}\boldsymbol{U}^T\boldsymbol{T}\boldsymbol{x} \tag{A.13}$$

Comparing with the parametric form in Eq. A.10, and taking into account scaling the singular values for the 3σ support, the required parametrisation is

$$\mathbb{T}^{-1} = \boldsymbol{\Lambda}^{\frac{-1}{2}}\boldsymbol{U}^T\boldsymbol{T} \tag{A.14}$$

$$\Rightarrow \mathbb{T} = \begin{bmatrix} \mathbb{I}_{3\times3} & \boldsymbol{\mu} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{u_1} & \boldsymbol{u_2} & \boldsymbol{u_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3\sqrt{\sigma_1{}^2} & 0 & 0 & 0 \\ 0 & 3\sqrt{\sigma_2{}^2} & 0 & 0 \\ 0 & 0 & 3\sqrt{\sigma_3{}^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A.15}$$

A.5 OpenGL Projection

With the correct $\mathbb{T}$ one needs to only follow the technique as presented in [70] to project the ellipsoid. The summary of the process is to utilise an OpenGL fragment shader to determine if a ray from the camera space intersects the transformed ellipsoid at a billboard location determined by an axis aligned bounding box in image space obtained via the projection of the ellipsoid to the image plane (which is an ellipse). Determining the size of the bounding box helps bound the number of pixels that are discarded in the fragment shader as being outside the ellipsoid. This leads to a much more accurate projection of the ellipsoid than an approach that generates a polygonal approximation the ellipsoid in a vertex shader and then paints it in. We detail this process for completeness in the next section.

A.6 Estimating Bounding Box

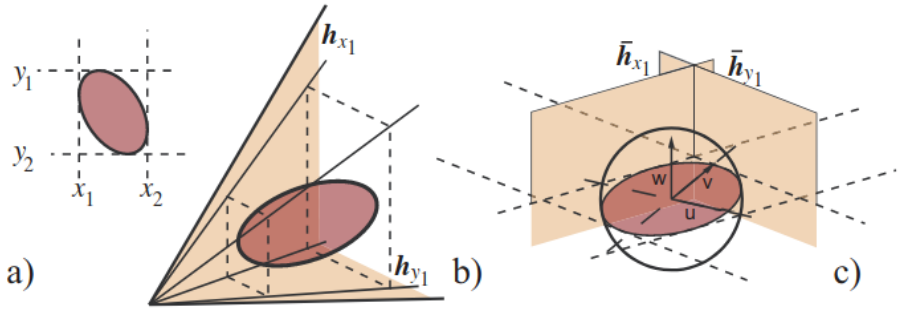


Figure A.1: Illustration of the ellipsoidal projection steps. The bounding box in image space (a) is determined by the frustum planes $\boldsymbol{h}_{x_i}$ and $\boldsymbol{h}_{y_i}$ (b). These planes, when transformed to the parameter space have to be tangent to the unit sphere (c). This property is exploited to obtain the bounding box in screen coordinates for the fragment shader. Illustration from [93].

To obtain the bounding box, the key idea we exploit is that there is a bijection between the direction of a normal and a surface point on the ellipse. Further, there's a linear mapping from the ellipsoid to the basis sphere. The mapping of the normal on the basis sphere to a point on its surface is simply the norm. Finally, we can then map this point on the surface of the sphere back to the ellipsoid, thus giving us the required point corresponding to the desired normal on the ellipsoid.

As can be seen in Fig. A.1, the axis aligned bounding boxes in camera space are simply the planes parallel to the x and y axes that are tangent to the ellipsoid. Thus, for the patch

extents we only require the points on the ellipsoid that corresponds to the four axis vectors. In particular, e.g., the rightmost point will correspond to the normal $[1, 0, 0]^T$.

Let us operate in the cuboidal clip space (thus after applying the projection matrix) to ensure that we only project ellipsoids that project into ellipses in the image plane. The mapping from a point in parameter space to a point in clip space is

$$\boldsymbol{x}_c = \mathbb{PMT}\boldsymbol{x}_p, \tag{A.16}$$

where $\mathbb{PMT}$ are the projection matrix, the model view matrix (the matrix that transforms from world space to camera space), and the parametric transformation matrix, respectively. The equation of a plane in clip space is

$$\boldsymbol{v}_c^T\boldsymbol{x}_c = 0, \tag{A.17}$$

where $\boldsymbol{v}_c = [\boldsymbol{n}_c^T, d]^T$, with $\boldsymbol{n}_c$ being the normal to the plane and d the distance from the origin. Substituting Eq.A.16

$$\boldsymbol{v}_c^T\mathbb{PMT}\boldsymbol{x}_p = 0 \tag{A.18}$$

$$((\mathbb{PMT})^T\boldsymbol{v}_c)^T\boldsymbol{x}_p = 0 \tag{A.19}$$

$$\boldsymbol{v}_p^T\boldsymbol{x}_p = 0. \tag{A.20}$$

Now each normal of a tangent plane on the basis sphere corresponds to a point on it. The equation of a tangent plane to $\boldsymbol{D}$ is

$$\boldsymbol{v}_p^T = \boldsymbol{x}_p^T\boldsymbol{D} \tag{A.21}$$

However, since we know that this point lies on the sphere

$$\begin{aligned} \boldsymbol{x}_p^T\boldsymbol{D}\boldsymbol{x}_p &= 0 \\ \Rightarrow \boldsymbol{x}_p^T\boldsymbol{D}\boldsymbol{D}^{-1}\boldsymbol{D}\boldsymbol{x}_p &= 0 \\ \Rightarrow \boldsymbol{v}_p^T\boldsymbol{D}\boldsymbol{v}_p &= 0 \end{aligned} \tag{A.22}$$

Now for $\boldsymbol{v}_c = [1, 0, 0, -x]^T$, a plane parallel to the y axis, using $\boldsymbol{v}_p = (\mathbb{PMT})^T\boldsymbol{v}_c$ above, we get

$$\boldsymbol{v}_p = \boldsymbol{r}_1 - x\boldsymbol{r}_4, \tag{A.23}$$

where $\boldsymbol{r}_i$ is the *i*-th row of the compound matrix $\mathbb{PMT}$. Substituting into Eq. A.22 yields a

quadratic equation

$$\left(\boldsymbol{r}_4^T \boldsymbol{D} \boldsymbol{r}_4\right) x^2 - 2\left(\boldsymbol{r}_1^T \boldsymbol{D} \boldsymbol{r}_4\right) x + \left(\boldsymbol{r}_1^T \boldsymbol{D} \boldsymbol{r}_1\right) = 0. \tag{A.24}$$

The two solutions correspond to the left and right borders of the bounding rectangle. Replacing $\boldsymbol{r}_1$ by $\boldsymbol{r}_2$ gives the vertical bounding borders. The centre of the bounding box is thus

$$\begin{aligned} \boldsymbol{v}_c &= \left[-\frac{2\left(\boldsymbol{r}_1^T \boldsymbol{D} \boldsymbol{r}_4\right)}{2\left(\boldsymbol{r}_4^T \boldsymbol{D} \boldsymbol{r}_4\right)}, -\frac{2\left(\boldsymbol{r}_1^T \boldsymbol{D} \boldsymbol{r}_4\right)}{2\left(\boldsymbol{r}_4^T \boldsymbol{D} \boldsymbol{r}_4\right)}, 0, 1 \right]^T \\ &= \left(\boldsymbol{r}_1^T \boldsymbol{D} \boldsymbol{r}_4, \boldsymbol{r}_2^T \boldsymbol{D} \boldsymbol{r}_4, 0, \boldsymbol{r}_4^T \boldsymbol{D} \boldsymbol{r}_4\right)^T \end{aligned} \tag{A.25}$$

The Z depth is written to the buffer later based on the ray intersection in the shader.

A.7 Fragment shader Ray-Quadric intersections

In the rasterisation process we find the roots for the ray corresponding to each fragment inside the point sprite and kill all the fragments that don't have any roots, i.e., don't lie inside the projected quadric.

A fragment in the window can be expressed as a combination of its xy coordinates and unknown depth:

$$\mathbf{x}_w = \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \mathbf{x}'_w + z_w \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \tag{A.26}$$

Now this point in window coordinate can be mapped to the parametric basis sphere as

$$\mathbf{x}_p = (\mathbb{V} \cdot \mathbb{P} \cdot \mathbb{M} \cdot \mathbb{T})^{-1} \mathbf{x}_w = \mathbf{x}'_p + z_w \mathbf{c}_3, \tag{A.27}$$

where $\mathbb{V}$ is the so-called viewport matrix

$$\mathbb{V} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & minX + \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & minY + \frac{h}{2} \\ 0 & 0 & \frac{f-n}{2.0} & \frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.28}$$

where f and n are `glDepthRange()` values that default to 1.0 and 0.0, respectively and

minX and minY are the viewport x and y coordinates that are 0 when rendering full-screen. Thus, replacing this in the basis quadric equation, we have

$$\begin{aligned} 0 &= \left(\mathbf{x}'_p + z_w \mathbf{c}_3\right)^T \mathbf{D} \left(\mathbf{x}'_p + z_w \mathbf{c}_3\right) \\ &= \left(\mathbf{c}_3^T \mathbf{D} \mathbf{c}_3\right) z_w^2 + 2 \left(\mathbf{x}_p'^T \mathbf{D} \mathbf{c}_3\right) z_w + \mathbf{x}_p^T \mathbf{D} \mathbf{x}'_p \end{aligned}. \tag{A.29}$$

The smaller of the two depths is assigned to the depthbuffer. Note that it is important to enable `GL_DEPTH_TEST` to assign the `gl_FragDepth` value and to assign a default value for the discarded fragments (we choose 1, the furthest depth).

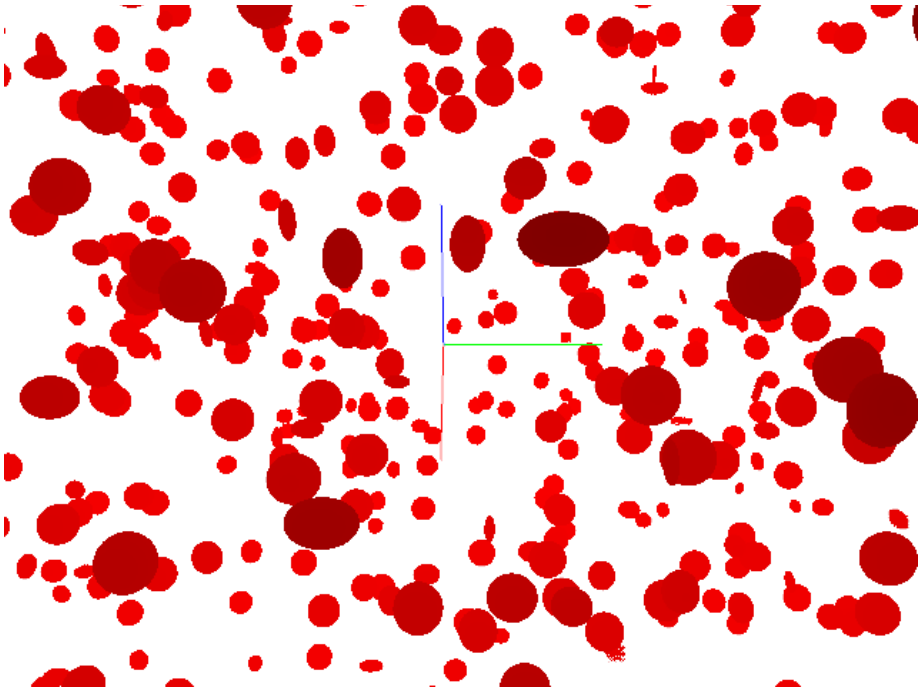


Figure A.2: Sample 1000 covariances rendered using above technique.

A.8 Inverse Depth Buffer

Conventional Z depth buffer gives too much precision to data in the near plane and loses precision for points at a further distance [1] This causes Z fighting to occur. This can be avoided by using the so-called inverse Z buffer, implementing which requires enabling the `glClipControl(GL_LOWER_LEFT, GL_ZERO_TO_ONE)`, using a floating point depth buffer, flipping the depth comparison to `glDepthFunc(GL_GREATER)`, clear-

[1] https://developer.nvidia.com/content/depth-precision-visualized

ing the depth to `glClearDepth(0.0f)`, and finally modifying the projection matrix such that z coordinates are linearly mapped from 0 to 1 accordingly whether they are at infinity or at the near plane respectively [2]. Note that the viewport matrix also changes, as per [3] to be

$$\mathbb{V} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & minX + \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & minY + \frac{h}{2} \\ 0 & 0 & f - n & n \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A.30}$$

These changes allow retaining depth precision to a very high precision for a very large viewing distance.

[2] https://nlguillemot.wordpress.com/2016/12/07/reversed-z-in-opengl/
[3] https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_clip_control.txt

Appendix B

MRFMap Message Derivation

B.1 Sum-Product Belief Propagation

Sum-Product belief propagation is a common message-passing algorithm for performing inference on factor graphs [44]. By exploiting marginalisation of joint distributions using factorisation of a graph it enables computing marginal distributions very efficiently. A factor graph is a bipartite graph containing nodes corresponding to variables and factors that are connected by edges. Messages are passed between connected nodes and factors that try to influence the marginal belief of their neighbours. The passing continues until convergence (if any) is achieved.

The message sent from a variable node x to a factor f is the cumulative belief of all the incoming messages from factors to the node except the factor in question

$$\mu_{x \to f}(x) = \prod_{g \in \mathcal{F}_x \setminus f} \mu_{g \to x}(x), \tag{B.1}$$

where $\mathcal{F}_x$ is the set of neighbouring factors to x. Similarly, the message sent from the factor to the node is the marginalisation of the product of the value of the factor ϕ_f with all the incoming messages from nodes other than the node in question

$$\mu_{f \to x}(x) = \sum_{\mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \mathcal{X}_f \setminus x} \mu_{y \to f}(y), \tag{B.2}$$

where $\mathcal{X}_f$ is the set of all neighbouring nodes of f.

Upon convergence, the estimated marginal distribution of each node is proportional to

the product of all messages from adjoining factors

$$p(x) \propto \prod_{g \in \mathcal{F}_x} \mu_{g \to x}(x) \tag{B.3}$$

Similarly, the joint marginal distribution of the set of nodes belonging to one factor is proportional to the product of the factor and the messages from the nodes

$$p(\mathcal{X}_f) \propto \phi_f(\mathcal{X}_f) \prod_{\mathcal{X}_f} \mu_{x \to f}(x) \tag{B.4}$$

B.2 Markov Random Field

Each ray in all the cameras generates a factor graph, the joint distribution of which is

$$p(\mathbf{o}, \mathbf{d}) = \frac{1}{Z} \prod_{i \in \mathcal{X}} \phi_i(o_i) \prod_{r \in \mathcal{R}} \psi_r(\mathbf{o}_r, d_r), \tag{B.5}$$

where $\mathcal{X}$ is the set of all the voxels $o_i \in \{0, 1\}$ and $\mathcal{R}$ is the set of all the rays from all the cameras viewing the scene, $\mathbf{o}_r = \{o_1^r, \ldots, o_{N_r}^r\}$ is the list of all the voxels traversed by a ray r, d_r is its corresponding depth variable, and Z is the normalisation constant. The total set of all the occupancy and depth variables are summarised as $\mathbf{o} = \{o_i \mid i \in \mathcal{X}\}$ and $\mathbf{d} = \{d_r \mid r \in \mathcal{R}\}$. ϕ_i and ψ_r are the potential factors as described as follows:

B.2.1 Prior Occupancy Factor

This is simply a unary factor assigning an independent Bernoulli prior γ to the voxel occupancy label for each voxel

$$\phi_i(o_i) = \gamma^{o_i}(1 - \gamma)^{1 - o_i}. \tag{B.6}$$

Note that these priors can be informed from predictive methods if required.

B.2.2 Ray Depth Potential Factor

A ray potential creates a factor graph connecting the binary occupancy label o_i for all voxels traversed by a single ray by virtue of making a measurement. Each of these voxels has a corresponding distance from the camera origin, and thus the ray is defined to include a depth variable d_r that represents the event that the measured depth is at distance d_i^r. For this to happen all the preceding voxels ought to be empty and the corresponding

voxel needs to be occupied. This leads to the definition of the joint occupancy and depth potential factor

$$\psi_r(\mathbf{o}_r, d_r) = \begin{cases} \nu_r(d_i^r) & \text{if } d_r = \sum_{i=1}^{N_r} o_i^r \prod_{j<i}(1-o_j^r) d_i^r \\ 0 & \text{otherwise} \end{cases}. \tag{B.7}$$

Here $\nu_r(d_i^r)$ denotes the probability of observing depth d_i^r given the measured depth value $\mathcal{D}^r$ that we model as $\nu_r(d_i^r) = \mathcal{N}(d_i^r; \mathcal{D}_r, \sigma(d_i^r))$. Note that we model this probability to vary in mean and noise as a function of the distance from the camera, which enables utilising learnt sensor noise characteristics.

This ray potential measures how well the occupancy and the depth variables explain the depth measurement $\mathcal{D}_r$. This is more apparent when Eq. B.7 is written as

$$\psi_r(\mathbf{o}_r, d_r) = \begin{cases} \nu_r(d_1^r) & \text{if } d^r = d_1^r, o_1^r = 1 \\ \nu_r(d_2^r) & \text{if } d^r = d_2^r, o_1^r = 0, o_2 = 1 \\ \vdots & \\ \nu_r(d_N^r) & \text{if } d^r = d_N^r, o_1^r = 0, \ldots, o_{N_r-1}^r = 0, o_{N_r}^r = 1 \end{cases} \tag{B.8}$$

This sparse structure of the ray potential enables massive simplification of the message passing equations, as shown next.

B.3 Message Passing Derivation

B.4 Ray Depth Potential to Depth Variable Messages

Since we're only concerned about the depth variable, we marginalise out the messages from all the occupancy nodes going to the ray depth potential. Following B.2 we have

$$\mu_{\psi_r \to d_r}\left(d_r = d_i^r\right) = \sum_{o_1^r} \cdots \sum_{o_{N_r}^r} \psi_r\left(\mathbf{o}_r, d_r\right) \prod_{j=1}^{N_r} \mu_{o_j^r \to \psi_r}\left(o_j^r\right). \tag{B.9}$$

Naïvely evaluating this equation is not feasible. However, we can exploit the sparse diagonal nature of the ray depth potential to recursively simplify this expression. After dropping the ray index for notational convenience and abbreviating $\mu_{o_j^r \to \psi_r}(o_j^r)$ as $\mu(o_j)$

we have

$$\mu_{\psi\to d}\left(d=d_i\right)=\mu\left(o_1=1\right)\overbrace{\left[\sum_{o_2}\cdots\sum_{o_N}\psi\left(o_1=1,o_2,\ldots,o_N,d=d_i\right)\prod_{j=2}^{N}\mu\left(o_j\right)\right]}^{\triangle}$$
$$+\,\mu\left(o_1=0\right)\underbrace{\left[\sum_{o_2}\cdots\sum_{o_N}\psi\left(o_1=0,o_2,\ldots,o_N,d=d_i\right)\prod_{j=2}^{N}\mu\left(o_j\right)\right]}_{\square} \tag{B.10}$$

From Eq. B.8, for the top expression $\triangle$ the ray potential term $\psi\left(o_1=1,o_2,\ldots,o_N,d=d_i\right)$ evaluates to $\nu(d_1)$ if $i=1$ and 0 otherwise. Since it only depends on d_1 it can be brought out of the summation as follows:

$$\triangle=\nu(d_1)\underbrace{\sum_{o_2}\cdots\sum_{o_N}\prod_{j=2}^{N}\mu\left(o_j\right)}_{\text{evaluates to 1.}}. \tag{B.11}$$

Assuming that all the incoming messages μ are normalised such that they sum to 1, the terms highlighted with the underbrace evaluate to 1. We maintain this normalisation in our implementation.

Assuming that $i\neq 1$, the bottom expression $\square$ can be recursively expanded similar to this step. Each such expansion brings in a term of the form $\prod_{k<j}\mu(o_k=0)\mu(o_j=1)\nu(d_j)$, until we reach the ith term Thus

$$\mu_{\psi\to d}\left(d=d_i\right)=\prod_{k<i}\mu(o_k=0)$$
$$\Bigg[\mu(o_i=1)\sum_{o_{i+1}}\cdots\sum_{o_N}\overbrace{\psi\left(o_1=0,o_2=0,\ldots,o_i=1,o_{i+1},\ldots o_N,d=d_i\right)}^{\text{evaluates to }\nu(d_i)}\prod_{j=i+1}^{N}\mu\left(o_j\right)$$
$$+\,\mu(o_i=0)\sum_{o_{i+1}}\cdots\sum_{o_N}\underbrace{\psi\left(o_1=0,o_2=0,\ldots,o_i=0,o_{i+1},\ldots o_N,d=d_i\right)}_{\text{evaluates to }0}\prod_{j=i+1}^{N}\mu\left(o_j\right)\Bigg] \tag{B.12}$$

where the first term evaluates to $\nu(d_i)$, and the next term evaluates to 0, giving us

$$\mu_{\psi\to d}\left(d=d_i\right)=\prod_{k<i}\mu(o_k=0)\mu(o_i=1)\nu(d_i)\underbrace{\sum_{o_{i+1}}\cdots\sum_{o_N}\prod_{j=i+1}^{N}\mu\left(o_j\right)}_{\text{evaluates to 1}}$$

$$\Rightarrow \mu_{\psi\to d}\left(d=d_i\right)=\nu(d_i)\mu(o_i=1)\prod_{k<i}\mu(o_k=0). \tag{B.13}$$

B.5 Depth Variable to Ray Depth Potential Messages

The message from the depth variable to the depth ray potential is irrelevant since the only factor connected to the variable is the potential itself.

B.6 Ray Depth Potential to Occupancy Variable Messages

Similar to the depth variable messages, we marginalise out all the variables except the node in question

$$\mu_{\psi_r\to o_i^r}\left(o_i^r=1\right)=\sum_{d_r}\sum_{\substack{o_j^r\\ j\neq i}}\mu_{d_r\to\psi_r}(d_r)\psi_r\left(\mathbf{o}_r,d_r\right)\prod_{j=1}^{N_r}\mu_{o_j^r\to\psi_r}\left(o_j^r\right). \tag{B.14}$$

After dropping the ray indices, we have

$$\mu_{\psi\to o_i}\left(o_i=1\right)=\underbrace{\sum_{d=d_1}^{d_N}\mu(d)}_{\text{evaluates to 1}}\sum_{o_1}\cdots\sum_{o_{i-1}}\sum_{o_{i+1}}\cdots\sum_{o_N}\psi\left(o_1,\ldots,o_i=1,\ldots,o_N,d\right)\prod_{\substack{j=1\\ j\neq i}}^{N}\mu\left(o_j\right), \tag{B.15}$$

where we use the shorthand $\mu(d)=\mu_{d_r\to\psi_r}(d_r)$, and $\mu(o)=\mu_{o_j^r\to\psi_r}(o_j^r)$. Note that as mentioned above, $\mu(d)$ sends a uniform message to the potential since it has no other factor connected to it. Thus the outermost summation evaluates to 1.

$$\mu_{\psi\to o_i}\left(o_i=1\right)=\sum_{o_1}\cdots\sum_{o_{i-1}}\sum_{o_{i+1}}\cdots\sum_{o_N}\psi\left(o_1,\ldots,o_i=1,\ldots,o_N,d_j\right)\prod_{\substack{j=1\\ j\neq i}}^{N}\mu\left(o_j\right) \tag{B.16}$$

We intend to simplify this expression in a similar manner to the previous derivation. Breaking apart into two terms, we have

$$\mu_{\psi\to o_i}(o_i=1)= \mu(o_1=1)\Bigg[\sum_{o_2}\cdots\sum_{o_{i-1}}\sum_{o_{i+1}}\cdots\sum_{o_N}\overbrace{\psi(o_1=1,\ldots,o_i=1,\ldots,o_N,d)}^{\text{evaluates to }\nu(d_1)}\prod_{\substack{j=2\\j\neq i}}^{N}\mu(o_j)\Bigg] + \mu(o_1=0)\Bigg[\sum_{o_2}\cdots\sum_{o_{i-1}}\sum_{o_{i+1}}\cdots\sum_{o_N}\psi(o_1=0,\ldots,o_i=1,\ldots,o_N,d)\prod_{\substack{j=2\\j\neq i}}^{N}\mu(o_j)\Bigg] \tag{B.17}$$

Similar to the previous strategy, we can keep breaking it up till o_{i-1}. At that point we have

$$\mu_{\psi\to o_i}(o_i=1)=\sum_{j}^{i-1}\mu(o_j=1)\nu(d_j)\prod_{k<j}\mu(o_k=0)+\prod_{k<i}\mu(o_k=0) \Bigg[\sum_{o_{i+1}}\cdots\sum_{o_N}\underbrace{\psi(o_1=0,o_2=0,\ldots,o_{i-1}=0,o_i=1,\ldots,o_N,d)}_{\text{evaluates to }\nu(d_i)}\prod_{j=i+1}^{N}\mu(o_j)\Bigg]. \tag{B.18}$$

Thus, we have

$$\mu_{\psi\to o_i}(o_i=1)=\sum_{j=1}^{i-1}\mu(o_j=1)\nu(d_j)\prod_{k<j}\mu(o_k=0)+\nu(d_i)\prod_{k<i}\mu(o_k=0) \tag{B.19}$$

For the negative case we get to the same point as Eq. B.18, except with $o_i=0$

$$\mu_{\psi\to o_i}(o_i=0)=\sum_{j}^{i-1}\mu(o_j=1)\nu(d_j)\prod_{k<j}\mu(o_k=0)+\prod_{k<i}\mu(o_k=0) \Bigg[\sum_{o_{i+1}}\cdots\sum_{o_N}\psi(o_1=0,o_2=0,\ldots,o_{i-1}=0,o_i=0,\ldots,o_N,d)\prod_{j=i+1}^{N}\mu(o_j)\Bigg]. \tag{B.20}$$

Observing that starting from o_{i+1} it is the same form of expansion, we then simplify and

get

$$\mu_{\psi\to o_i}(o_i=0)=\sum_{j=1}^{i-1}\mu(o_j=1)\,\nu(d_j)\prod_{k<j}\mu(o_k=0)+\sum_{j=i+1}^{N}\mu(o_j=1)\,\nu(d_j)\prod_{\substack{k<j\\k\neq i}}\mu(o_k=0)\,, \tag{B.21}$$

Which for convenience can also be written as

$$\begin{aligned}\mu_{\psi\to o_i}(o_i=0)=&\sum_{j=1}^{i-1}\mu(o_j=1)\,\nu(d_j)\prod_{k<j}\mu(o_k=0)+\\&\frac{1}{\mu(o_i=0)}\sum_{j=i+1}^{N}\mu(o_j=1)\,\nu(d_j)\prod_{k<j}\mu(o_k=0)\,.\end{aligned} \tag{B.22}$$

B.7 Occupancy Variable to Ray Depth Potential Messages

Since other rays can (and often do) pass through the same occupancy variable node, the outgoing message $\mu_{o_i^r\to\psi_r}$ is computed as per Eq. B.1.

B.8 Depth Distribution along a Ray

Here we illustrate how the message passing based inference interacts between two rays. The objective is to demonstrate how the resulting evaluated depth distribution along a ray is impacted by other rays passing through nearby voxels. Recall the expression for the depth distribution along a ray as shown in Eq. B.13

$$\mu_{\psi\to d}(d=d_i)=\nu(d_i)\mu(o_i=1)\underbrace{\prod_{k<i}\mu(o_k=0)}_{\text{visibility of } i^{\text{th}}\text{voxel}}\,. \tag{B.13}$$

Consider a single ray with a measurement falling within the displayed voxel in an empty map, as shown in Fig. B.1. All voxels in this map have a small uniform prior probability factor ϕ_i attached. For the first pass of messages sent from the ray depth factor to the nodes (Eq. B.2) the only outgoing messages from the voxel to the ray depth potential factor are the small uniform prior probability messages $\mu_{o_i\to\psi}=\mu_{\phi_i\to o_i}$ since the prior factor is the only other connected factor to each voxel. Correspondingly, the visibility term monotonically decreases slowly. Corresponding to the visibility term and the forward

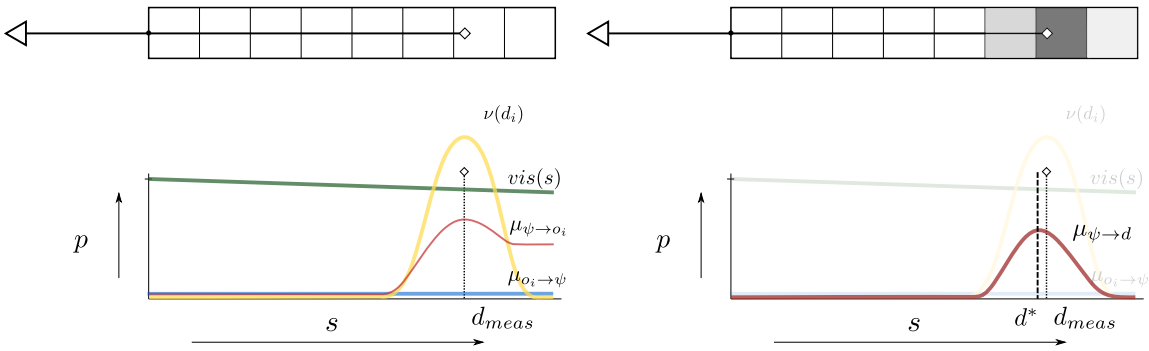


Figure B.1: Left: First pass of messages sent from the ray depth potential factor to the nodes. The incoming messages from each voxel along the ray $\mu_{o_i \to \psi}$ are shown in blue. The corresponding visibility term is displayed in green. The measurement distribution ν corresponding to the measured depth is displayed in yellow. The resulting outgoing message from the ray depth potential to each of the voxels is displayed in red. At the end of this first pass, each voxel now has two incoming messages - one from the prior factor, and one from the ray depth potential factor. Right: Marginal occupancy of voxels along the map are shown at the top. Resulting depth distribution is shown in red. The L_1 expected depth for this depth distribution is denoted by d^* and very nearly coincides with the actual measured depth d_{meas}. Plots are not discretised for illustration purposes.

model ν the outgoing message $\mu_{\psi \to o_i}$ is sent to each voxel. At the end of this first pass, each voxel now has two incoming messages - one from the prior factor, and one from the ray depth potential factor. Right: Nodes along the ray receive occupied messages according to the continuous inverse sensor model as shown in Fig. 2.3. This results in a occupancy distribution as shown on the top, and a depth distribution as defined by Eq. **??** as shown in the bottom plot. The depth distribution (red) then closely follows the forward sensor noise model, shifted by virtue of the probability mass of the area under the prior probability. This can be seen as an analogue to setting a uniform low probability of occupancy for any voxel ahead of the measurement as discussed in Sec. 2.3. The L_1 expected depth for this depth distribution is denoted by d^* and very nearly coincides with the actual measured depth.

Now consider another ray terminating close to the previous ray that sends a high outgoing message to the voxel right before the occupied voxel shown in Fig. B.2. The corresponding outgoing message from that ray would now be at odds with what the original ray suggested for this voxel. As expected, then the visibility and hence the outgoing message of the original ray is impacted by this additional information. For the

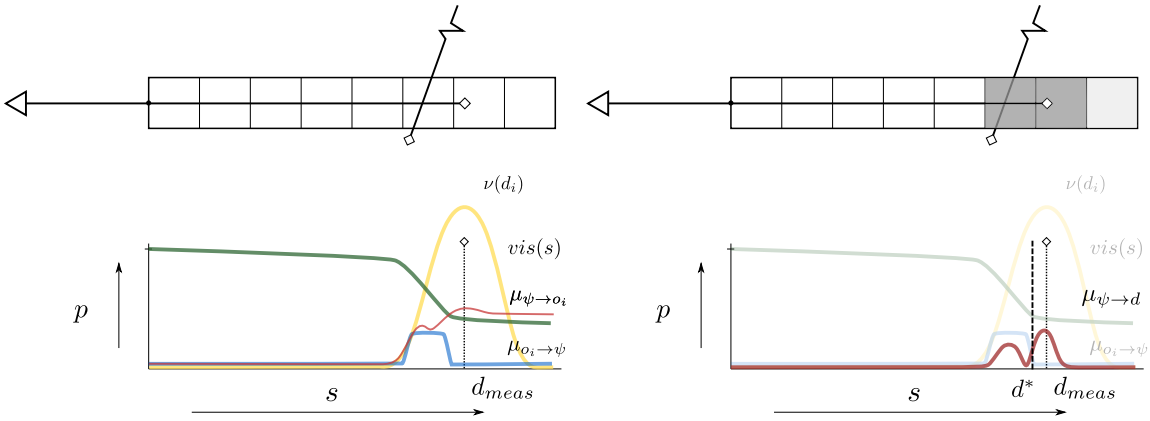


Figure B.2: Now consider another ray terminating close to the previous ray that sends a high outgoing message to the voxel right before the occupied voxel in Fig. B.1. Left: Corresponding to the incoming message from the intersecting ray the voxel sends a higher occupied belief (blue bump) to the original ray. This causes a drop in visibility (green), that, in concert with the sensor measurement probability (yellow) impacts the outgoing message (red). Right: The resulting occupancy distribution is now more spread out between two the voxels (top). The resulting depth distribution is also split into two modes corresponding to Eq. B.13.

preceding voxel that the new ray passes through since the set of incoming messages are the product of the prior and the message sent by the intersecting ray (Eq. B.1), there is a drop in visibility, and a corresponding increase in the probability of the generating surface being at that voxel. Further, the probability of the generating surface close to the actual measurement of the ray correspondingly decreases by virtue of the loss in visibility. Thus, depending on how confident either ray is the relative size of the modes would change. The choice of the L_1 maximum likelihood estimator used for marginalisation biases the MLE generating surface to prefer lying within one of the two modes.

Bibliography

[1] Ali-Akbar Agha-Mohammadi, Eric Heiden, Karol Hausman, and Gaurav Sukhatme. Confidence-rich Grid Mapping. *The Intl. Journal of Robotics Research*, 38(12-13):1352–1374, 2019. 4.2

[2] John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Proc. of Eurographics*, volume 87, pages 3–10, 1987. 4.4.1

[3] Franz Andert. Drawing stereo disparity images into occupancy grids: Measurement model and fast implementation. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 5191–5197. IEEE, 2009. 2.3.4

[4] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words. *IEEE Transactions on Robotics*, 24(5), 2008. 5.1

[5] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002. 5.3.4

[6] Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006. 2.1

[7] Filippo Basso, Emanuele Menegatti, and Alberto Pretto. Robust Intrinsic And Extrinsic Calibration Of RGB-D Cameras. *IEEE Transactions on Robotics*, (99):01–18, 2018. 3.1, 3.2, 3.4, 4.3.1, 4.4.3, 4.5

[8] Joydeep Biswas and Manuela Veloso. Depth Camera Based Indoor Mobile Robot Localization and Navigation. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2012. 5.2

[9] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. CodeSLAM-Learning a Compact, Optimisable Representation for Dense Visual SLAM. *arXiv preprint arXiv:1804.00874*, 2018. 7

[10] Adam Bry, Abraham Bachrach, and Nicholas Roy. State Estimation for Aggressive Flight in GPS-Denied Environments Using Onboard Sensing. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2012. 5.2

[11] Antoni Burguera, Yolanda González, and Gabriel Oliver. The Likelihood Field Approach to Sonar Scan Matching. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2008. 2.2, 5.2

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. 2.1

[13] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust Reconstruction of Indoor Scenes. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4.1, 4.5, 4.8

[14] Daniel E. Crispell. *A Continuous Probabilistic Scene Model for Aerial Imagery*. PhD thesis, Brown University, 2010. 4.3.4

[15] Arun Das and Steven L Waslander. Scan Registration using Segmented Region Growing NDT. *The International Journal of Robotics Research*, 33(13):1645–1663, 2014. 2.2

[16] Arun Das, James Servos, and Steven L Waslander. 3D Scan Registration Using the Normal Distributions Transform with Ground Segmentation and Point Cloud Clustering. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2013. 5.2

[17] Vikas Dhiman, Abhijit Kundu, Frank Dellaert, and Jason J Corso. Modern MAP Inference Methods for Accurate and Fast Occupancy Grid Mapping on Higher Order Factor Graphs. In *Proc. of Intl. Conf. on Robotics and Automation (ICRA)*, pages 2037–2044. IEEE, 2014. 4.2

[18] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation magazine*, 13(2):99–110, 2006. 2.1

[19] Ben Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. MLMD: Maximum Likelihood Mixture Decoupling for Fast and Accurate Point Cloud Registration. In *Proc. of IEEE Intl. Conf. on 3D Vision*, 2015. 5.2, 6.4.3

[20] Benjamin Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Accelerated Generative Models for 3D Point Cloud Data. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 5.3.1

[21] Alberto Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989. 2.2

[22] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014. 2.2

[23] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. 7

[24] Carlos Estrada, José Neira, and Juan D. Tardós. Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005. 4.6

[25] Maurice F Fallon, Hordur Johannsson, and John J Leonard. Efficient Scene Simulation for Robust Monte Carlo Localization using an RGB-D Camera. In *Proc. of IEEE Intl.*

Conf. on Robotics and Automation, 2012. 5.2

[26] Zheng Fang and Sebastian Scherer. Real-time Onboard 6DoF Localization of an Indoor MAV in Degraded Visual Environments Using a RGB-D Camera. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2015. 5.2

[27] Sergi Foix, Guillem Alenya, and Carme Torras. Lock-in Time-of-Flight (ToF) Cameras: A Survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011. 3.1

[28] Pau Gargallo, Peter Sturm, and Sergi Pujades. An Occupancy-Depth Generative Model of Multi-view Images. In *Proc. of Asian Conf. on Computer Vision*, pages 373–383. Springer, 2007. 4.2

[29] Dan Geiger, Thomas Verma, and Judea Pearl. Identifying Independence in Bayesian Networks. *Networks*, 20(5):507–534, 1990. 2.1

[30] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuñiga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments. *IEEE Transactions on Robotics*, 35:734–746, 2019. 2.2

[31] Tiago Gonçalves and Andrew I Comport. Real-time Direct Tracking of Color Images in the Presence of Illumination Variation. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2011. 5.2

[32] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007. 5.3.4

[33] Li Guan, Jean-Sébastien Franco, and Marc Pollefeys. 3D Object Reconstruction with Heterogeneous Sensor Data. In *Proc. of Intl. Symposium on 3D Data Processing, Visualization and Transmission*, 2008. 2.3, 2.3.3, 4.2

[34] Siwei Guo and Nikolay A. Atanasov. Information Filter Occupancy Mapping using Decomposable Radial Kernels. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. 2.2

[35] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous Visual Mapping and Exploration With a Micro Aerial Vehicle. *Journal of Field Robotics*, 31(4):654–675, 2014. 2.3.4, 2.4, 6.2

[36] John R Hershey and Peder A Olsen. Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, volume 4, 2007. 5.5.2

[37] Rama K. Hoetzlein. GVDB: Raytracing Sparse Voxel Database Structures on the GPU. In *Proc. of High Performance Graphics*, pages 109–117, 2016. 4.4.2

[38] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. 2.2, 2.3.4, 2.4, 4.1, 4.3.4, 4.5, 5.2, 6.2

[39] Ming Hsiao and Michael Kaess. MH-iSAM2: Multi-hypothesis iSAM using Bayes Tree and Hypo-tree. In *Intl. Conf. on Robotics and Automation*, pages 1274–1280. IEEE, 2019. 7.1.1

[40] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar SLAM. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5110–5117, 2017. 2.2

[41] 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*, 31(2):216–235, 2012. 6.3

[42] Soohwan Kim and Jonghyuk Kim. Building Occupancy Maps with a Mixture of Gaussian processes. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4756–4761. IEEE, 2012. 2.2

[43] Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1), 1996. 5.3.4

[44] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009. 4.3.2, B.1

[45] Frank R. Kschischang, Brendan J. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001. 4.3.2

[46] Shubao Liu and David B. Cooper. Ray Markov Random Fields for Image-Based 3D Modeling: Model and Efficient Inference. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1530–1537, 2010. 1, 4.2

[47] Martin Magnusson, Narunas Vaskevicius, Todor Stoyanov, Kaustubh Pathak, and Andreas Birk. Beyond points: Evaluating Recent 3D Scan-Matching Algorithms. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2015. 5.2

[48] Alexander Millane, Zachary Taylor, Helen Oleynikova, Juan Nieto, Roland Siegwart, and César Cadena. C-blox: A Scalable and Consistent TSDF-based Dense Mapping Approach. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 995–1002. IEEE, 2018. 6.4.1

[49] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5), 2017. 2.2, 5.5.1

[50] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE transactions on robotics*, 31(5): 1147–1163, 2015. 6.4.1

[51] Kevin P Murphy. Bayesian Map Learning in Dynamic Environments. In *Advances in Neural Information Processing Systems*, pages 1015–1021, 2000. 2.1

[52] Ken Museth. VDB: High-resolution Sparse Volumes with Dynamic Topology. *ACM Transactions on Graphics (TOG)*, 32(3):27, 2013. 4.4.2

[53] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530. IEEE, 2012. 3.1

[54] Kai Ni, Drew Steedly, and Frank Dellaert. Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM. In *Proc. 2007 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1678–1685, 2007. 4.6

[55] Simon T. O'Callaghan and Fabio T. Ramos. Gaussian Process Occupancy Maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012. 2.2

[56] Shuji Oishi, Yongjin Jeong, Ryo Kurazume, Yumi Iwashita, and Tsutomu Hasegawa. ND Voxel Localization Using Large-Scale 3D Environmental Map and RGB-D Camera. In *Proc. of IEEE Intl. Conf. on Robotics and Biomimetics*, 2013. 5.2

[57] Kyel Ok, W Nicholas Greene, and Nicholas Roy. Simultaneous Tracking and Rendering: Real-time Monocular Localization for MAVs. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2016. 5.2

[58] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. IEEE, 2017. 2.2, 2.4, 5.5.1

[59] Helen Oleynikova, Christian Lanegger, Zachary Taylor, Michael Pantic, Alexander Millane, Roland Siegwart, and Juan Nieto. An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. *Journal of Field Robotics*, 37(4):642–666, 2020. 6.2

[60] Joseph Ortiz, Mark Pupilli, Stefan Leutenegger, and Andrew J Davison. Bundle Adjustment on a Graph Processor. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 2416–2425, 2020. 7.1.1

[61] Geoffrey Pascoe, Will Maddern, and Paul Newman. Robust Direct Visual Localisation using Normalised Information Distance. In *Proc. of British Machine Vision Conf.*, 2015. 5.2

[62] Kaustubh Pathak, Andreas Birk, Jann Poppinga, and Soren Schwertfeger. 3D Forward Sensor Modeling and Application to Occupancy Grid based Sensor Fusion. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2059–2064, 2007. 4.3.4, 4.3.4

[63] Thomas Pollard and Joseph L. Mundy. Change Detection in a 3-d World. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007. 4.2

[64] Fabio Ramos and Lionel Ott. Hilbert Maps: Scalable Continuous Occupancy Mapping with Stochastic Gradient Descent. *The Intl. Journal of Robotics Research*, 35(14):1717–1730, 2016. 2.2

[65] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, Juha Ala-Luhtala, and Achim J. Lilienthal. Normal Distributions Transform Occupancy Maps: Application to Large-

Scale Online 3D Mapping. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2233–2238, 2013. 2.2, 6.4.2

[66] Renato F. Salas-Moreno, Ben Glocker, Paul H. J. Kelly, and Andrew J. Davison. Dense Planar SLAM. *IEEE Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 157–164, 2014. 2.2

[67] Marcelo Saval-Calvo, Luis Medina-Valdés, José María Castillo-Secilla, Sergio Cuenca-Asensi, Antonio Martínez-Álvarez, and Jorge Villagrá. A Review of the Bayesian Occupancy Filter. *Sensors*, 17(2):344, 2017. 2.2

[68] Cornelia Schulz, Richard Hanten, and Andreas Zell. Efficient Map Representations for Multi-Dimensional Normal Distributions Transforms. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2679–2686, 2018. 2.2

[69] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems*, volume 2, 2009. 5.5.3

[70] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus H Gross. GPU-Based Ray-Casting of Quadratic Surfaces. In *SPBG*, pages 59–65. Citeseer, 2006. A.3, A.5

[71] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990. 6.3

[72] Randall C Smith and Peter Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The Intl. Journal of Robotics Research*, 5(4):56–68, 1986. 2.2

[73] Paloma Sodhi, Bing-Jui Ho, and Michael Kaess. Online and Consistent Occupancy Grid Mapping for Planning in Unknown Environments. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 7879–7886, 2019. 4.6, 6.2, 6.3, 6.4.1

[74] Shobhit Srivastava and Nathan Michael. Approximate Continuous Belief Distributions for Precise Autonomous Inspection. In *Proc. of IEEE Intl. Symposium on Safety, Security, and Rescue Robotics*, 2016. 5.2, 5.3.1, 5.5.2

[75] Cyrill Stachniss. Grid-based FastSLAM, 2013. URL `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam13-gridfastslam.pdf`. 5.4

[76] Alexander D Stewart and Paul Newman. LAPS-Localisation using Appearance of Prior Structure: 6-DoF Monocular Camera Localisation using Prior Pointclouds. In *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2012. 5.2

[77] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The Intl. Journal of Robotics Research*, 31(12), 2012. 5.2

[78] Todor Stoyanov, Martin Magnusson, and Achim J. Lilienthal. Comparative Evaluation of the Consistency of Three-dimensional Spatial Representations used in Autonomous Robot Navigation. *Journal of Field Robotics*, 30(2):216–236, 2013. 4.3.4

[79] Hauke Strasdat, Andrew J. Davison, J.M. Martìnez Montiel, and Kurt Konolige.

Double Window Optimisation for Constant Time Visual SLAM. In *Proc. of Intl. Conf. on Computer Vision*, pages 2352–2359. IEEE, 2011. 6.4.1

[80] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1): 137–147, 2014. 2.2

[81] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, October 2012. 5.5.1

[82] Sebastian Thrun. A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots. *The International Journal of Robotics Research*, 20(5):335–363, 2001. 7.1.1

[83] Sebastian Thrun. Particle Filters in Robotics. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002. 5.3.4

[84] Sebastian Thrun. Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous robots*, 15(2):111–127, 2003. 1, 2.2, 4.1, 4.3.1, 4.3.4

[85] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial intelligence*, 128(1-2), 2001. 2.1

[86] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 5.1, 5.3.4, 5.3.4

[87] Ali O. Ulusoy, Andreas Geiger, and Michael J. Black. Towards Probabilistic Volumetric Reconstruction using Ray Potentials. In *Proc. of Intl. Conf. on 3D Vision (3DV)*, pages 10–18, 2015. 1, 4.2, 4.3.1

[88] Ali O. Ulusoy, Michael J. Black, and Andreas Geiger. Patches, Planes and Probabilities: A Non-Local Prior for Volumetric 3D Reconstruction. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3280–3289, 2016. 4.2, 4.3.1, 4.4.1, 6.4.2, 7, 7.1.3

[89] G Vallicrosa and P Ridao. H-SLAM: Rao-Blackwellized Particle Filter SLAM Using Hilbert Maps. *Sensors (Basel, Switzerland)*, 18(5), 2018. 2.2

[90] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The Unscented Particle Filter. In *Advances in neural information processing systems*, 2001. 5.3.4

[91] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul H.J. Kelly, and Stefan Leutenegger. Efficient Octree-based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, 2018. 2.2, 4.2

[92] Jinkun Wang and Brendan Englot. Fast, Accurate Gaussian Process Occupancy Maps via Test-Data Octrees and Nested Bayesian Fusion. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1003–1010, 2016. 2.2

[93] T. Weyrich, S. Heinzle, T. Aila, D. B. Fasnacht, S. Oetiker, M. Botsch, C. Flaig, S. Mall, K. Rohrer, N. Felber, et al. A Hardware Architecture for Surface Splatting. In *ACM*

Transactions on Graphics (TOG), volume 26, page 90. ACM, 2007. A.1

[94] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially Extended KinectFusion. In *Proc. of RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012. 2.2, 6.2

[95] Thiemo Wiedemeyer. IAI Kinect2. `https://github.com/code-iai/iai_kinect2`, 2014 – 2015. Accessed June 12, 2015. 3.1

[96] Qian-Yi Zhou and Vladlen Koltun. Dense Scene Reconstruction with Points of Interest. *ACM Transactions on Graphics (TOG)*, 32(4). 5.5.1

[97] Jon Zubizarreta, Iker Aguinaga, and J. M. M. Montiel. Direct Sparse Mapping. *IEEE Transactions on Robotics*, 2020. 6.4.1