Generative Point Cloud Modeling with Gaussian Mixture Models for Multi-Robot Exploration

Cormac OMeadhra
August, 2018
CMU-RI-TR-18-45

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

Thesis Committee:
Nathan Michael, *Chair*
Artur W. Dubrawski
Michael Kaess
Wenhao Luo

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Autonomous exploration in rich 3D environments requires the construction and maintenance of a representation derived from accumulated 3D observations. Volumetric models, which are commonly employed to enable joint reasoning about occupied and free space, scale poorly with the size of the environment. Techniques employed to mitigate this scaling include hierarchical discretization, learning local data summarizations and occupancy prediction through function approximation. However, these approaches either impose an *a priori* environment discretization, which limits resolution adaption, or else require input data sparsification or restrictive modeling assumptions to account for the computational expense of directly learning the volumetric occupancy model. This thesis proposes to overcome these limitations through the use of Gaussian Mixture Models (GMMs) to learn generative models of point cloud data. By learning the density function describing the distribution of points in a measurement, it becomes possible to reconstruct occupancy representations at arbitrary resolutions. Furthermore, the scaling issues associated with large environments can be circumvented through the use of efficient local occupancy reconstruction. Specifically, numerical and approximate analytic techniques are presented for computing occupancy from GMMs, which enables point-wise occupancy reconstruction in a low-complexity manner. The proposed approach is evaluated in two complementary scenarios: firstly, as a method for generating arbitrary resolution occupancy maps through comparison against state of the art continuous occupancy techniques and, secondly, as a compact back-end for multi-robot information theoretic exploration in large scale, communication constrained environments. Results demonstrate the efficacy of the approach as a high fidelity, low-bandwidth 3D point cloud representational framework.

Acknowledgments

I would like to thank my advisor, Prof. Nathan Michael, for all of his support and patience thus far in my graduate studies. I have greatly benefited from his guidance and the insights that he has shared with me.

For the duration of my Master's, I have been fortunate enough to work with a great group of people in RISLab. Luckily, my time in CMU has not come to a close, so I will keep this brief and thank you all as a group. Within RISLab, a special thank you goes to my collaborators: Vibhav Ganesh, for running down **many** rabbit holes with me, Kshitij Goel and Micah Corah for our one long adventure, and, of course, Wennie Tabib, for running down even more rabbit holes with me than Vib - although, we found a few good ones along the way.

To all of my great friends - thank you for adding balance to my work life. Thank you to my parents and sister for always supporting me. Finally, to Kayleigh, for being by my side even when thousands of miles away, thank you.

We gratefully acknowledge the support of DTRA grant HDTRA1-13-0026 and industry sponsors.

Contents

1 Introduction **1**

1.1 Thesis Problem 3

1.2 Contributions and Outline 4

2 Background **5**

2.1 Generative Point Cloud Modeling through Gaussian Mixture Models 5

2.1.1 Expectation Maximization 7

2.1.2 Hierarchical Expectation Maximization 8

2.1.3 Initialization Strategies 10

2.1.4 Component Pruning 11

2.1.5 Alternative Learning Strategies 13

2.2 Probabilistic Environment Modeling 14

2.3 Point Cloud Compression 16

2.4 Information Theoretic Exploration 17

3 Variable Resolution Occupancy Modeling **18**

3.1 Free Space Modeling 20

3.2 Mixtures of Gaussian Cones 20

3.2.1 2D Occupancy Grids 22

3.2.2 3D Voxel Grid 23
3.2.3 Monte Carlo Estimation 25
3.3 Occupancy Modeling 26
3.4 Results - 2D GMM Occupancy Modeling 27
3.5 Results - 3D GMM Occupancy Modeling 29

4 Distributed Mapping with Limited Communication for Multi-Robot Exploration 34

4.1 Sliding Voxel Map 36
4.1.1 Monte Carlo ray tracing 36
4.1.2 Point novelty check 37
4.1.3 Component updating 38
4.2 Distributed Mapping 38
4.3 Information Theoretic Exploration 39
4.4 Results 40

5 Conclusions and Future Work 43

5.1 Summary 43
5.2 Future Work 44

List of Tables

3.1 AUC for given memory usage of GMM occupancy model compared to the performance of BGK inference occupancy maps. The GMM is better able to reconstruct the occupancy information in the environment with 50× less data than the Bayesian Generalized Kernel inference approach for occupancy mapping. 30

List of Figures

1.1 Effect of multi-robot communication on exploration trajectories. 3

2.1 Comparison of GMM initialization strategies 12

3.1 Gaussian mixture model occupancy mapping pipeline. 19
3.2 Probability cone for a Gaussian component of the observation model. . . . 21
3.3 Error evaluation of analytic approximations to free space integrals 23
3.4 Area Under ROC Curve for 2D Occupancy Reconstruction 27
3.5 Evaluation of occupancy reconstruction in 2D 28
3.6 Area Under ROC Curve for 3D Occupancy Reconstruction 30
3.7 3D occuapancy model reconstruction for fixed memory budget 31
3.8 Effect of GMM complexity on 3D point cloud reconstruction quality 31

4.1 Multi robot exploration system diagram. 35
4.2 Sliding grid map and resampled ray trimming. 37
4.3 Simulated 3D environment 40
4.4 Entropy reduction over simulated exploration trials 41
4.5 Reduced bandwidth utilization achieved by GMM 42

5.1 Spherical polygon describing 3D angular integration domain 46

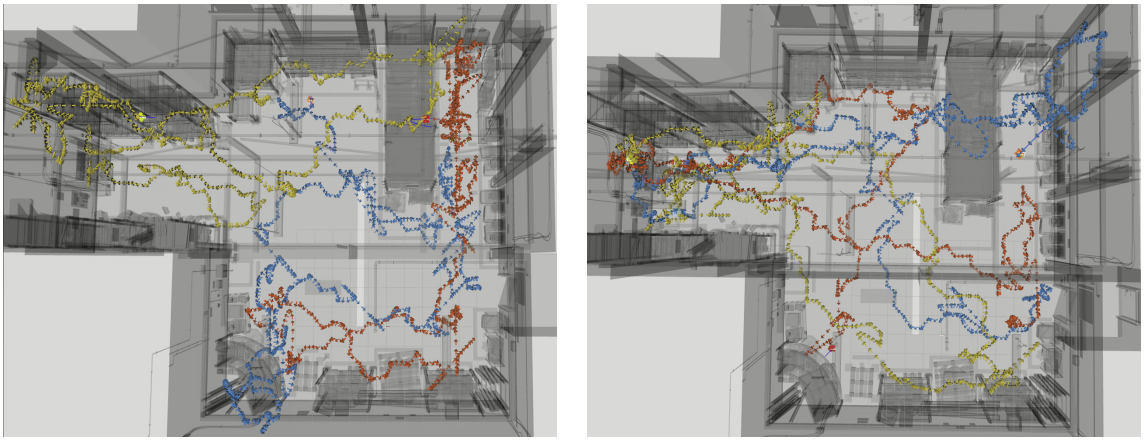
Chapter 1

Introduction

The objective of many exploration tasks include some element of time penalization. A high rate of exploration is often desirable due to explicit task requirements, e.g. search and rescue, threat localization or emergency response, or implicit system constraints such as energy budgets. Achieving a high exploration rate in rich 3D environments requires an efficient environment inference framework amenable to high sampling rates. Unfortunately, when cluttered 3D environments increase in size, the volumetric representations, such as occupancy grids, that are typically employed become computationally expensive to maintain and perform inference over. There are several strategies that can be employed to combat this scaling issue: hierarchical representation strategies can be utilized to group homogeneous regions, thus maintaining high frequency information only where necessary, local data approximations can enable coarser discretization while maintaining some measure of the data distribution, and occupancy data can be predicted through continuous function approximators which attempt to reduce complexity by capturing regularity in the volumetric model. Hierarchical representations and local data approximations both suffer from a fixed minimum representational capacity imposed by the complexity of the volumetric representation. Continuous function approximators similarly suffer from an minimum represen-

tation complexity, although, in this case it is imposed by the computational complexity of online training of high fidelity volumetric models. This thesis elects to follow an alternative strategy of utilizing a surface model which implicitly encodes volumetric information. By learning Gaussian Mixture Models (GMMs) as generative models of sensor observations, both free space and occupied space information can be encoded. Subsequent reconstruction of volumetric occupancy can be efficiently achieved through interpretation of the occupied and free space evidence as samples from a Bernoulli Distribution. Furthermore, as the GMM directly encodes sensor observations, it becomes possible to reconstruct volumetric models at arbitrary resolutions, bounded primarily by the noise level in the observation. While the representation capacity of the GMM is determined the model complexity, integration of the sensor noise into the learning procedure allows the model size to be matched to sensor data.

Beyond efficient inference frameworks, the rate of exploration can greatly benefit from the incorporation of multiple robots into the exploration team. This is especially the case in large environments, where inter-robot trajectory overlap can be significantly reduced. However, in order to achieve sufficiently high inter-robot separation, it is necessary for agents to collaborate and share information - see Fig. 1.1 for a qualitative depiction of the influence of information sharing on trajectory overlap. Unfortunately, sharing information in rich 3D environments is expensive. For example, depth images generated by RGB-D cameras have 640×480 pixels, which corresponds to 300 kB or 600 kB, depending on the encoding. When operating at a sensor rate of 30 Hz, a throughput of 9/18 MB/s per robot will almost certainly saturate any wireless network encountered during time sensitive applications. This bandwidth restriction necessitates the use of depth image or point cloud compression techniques. While many such compression techniques exist, computational complexity is typically inversely proportional to the compression level. Furthermore, typical compression techniques do not produce a representation that can be utilized in other



(a) Information sharing disabled. (b) Information sharing enabled.

Figure 1.1: Three agents explore an industrial environment. (b) When no communication occurs between agents, the trajectories followed by each agent exhibit significant overlap. (a) Conversely, communication between agents increases trajectory separation, resulting in greater efficiency in terms of time and energy expended.

components of the robotic system. Through the use of GMMs as a compact representation, significant compression levels can be achieved. Additionally, beyond use as a volumetric occupancy model as discussed previously, the GMM can be leveraged to achieve robust point cloud registration [38] and efficient collision avoidance [11]. Thus, the cost of the generating the GMM can be amortized over many components in the robotic system.

1.1 Thesis Problem

This thesis focuses on the problem of environmental modeling frameworks for efficient multi-robot exploration. Therein, this thesis tackles two sub-problems:

1. Generation of a compact occupancy representation that scales to rich, large-scale 3D environments and is amenable to efficient inference without overly restricting representation scale.

2. Achieving a consistent global environment representation between distributed mem-

bers in a multi-robot team through the use of low-bandwidth information transfer between agents.

1.2 Contributions and Outline

This this provides three distinct contributions. First, an evaluation of Gaussian mixture modeling techniques and their limitations is presented. Second, a framework is developed to leverage GMMs as a compact generative model of sensor observations, from which a volumetric representation of occupancy can be derived. As part of this framework, methodologies for evaluating the probability occupancy at individual points is presented. Finally, the GMM based occupancy framework is applied as a back end to a multi-robot exploration problem, demonstrating the ability of the approach to operate as a low bandwidth representation for distributed teams.

The structure of the thesis is as follows. Chapter 2 presents background material and relevant concepts. Specifically, extensive discussion of Gaussian Mixture Models and techniques for learning their hyperparamters is presented. Additionally, relevant concepts and the state of the art in probabilistic modeling, point cloud compression and information theoretic exploration are also introduced. Chapter 3 presents the GMM occupancy framework along with evaluation of its accuracy in both 2D and 3D. Chapter 4 details the application of the occupancy mapping system to a multi-robot exploration team and presents evaluation in a large-scale, rich 3D simulated environment. Finally, conclusions and potential directions for future work are presented in Chapter 5.

Chapter 2

Background

This section will act as primer to the topics discussed in the body of the thesis. Firstly, generative modeling of point clouds will be discussed, which will lead to a more focused discussion on the learning of Gaussian Mixture Models over point clouds and the techniques commonly employed to improve efficiency and/or accuracy of the learning process. Secondly, the topic of occupancy mapping and surface modeling will be introduced. Attention will then switch to point cloud compression techniques - both lossy and lossless. Finally, a brief background to information theoretic exploration will be provided, which will motivate the need for probabilistic occupancy representations.

2.1 Generative Point Cloud Modeling through Gaussian Mixture Models

We elect to model a point cloud of size N, $\mathcal{Z} = \{\boldsymbol{z}_i \in \mathbb{R}^n, i = 1, \ldots, N\}$, as a set of *i.i.d* samples drawn from a distribution $p(\boldsymbol{z})$, which is supported over some subset of $\mathbb{R}^n$. In the applications discussed in this thesis, discussion will focus on $n = 3$ and in some cases $n = 2$. The true nature of $p(\boldsymbol{z})$ is unknown to us and, accordingly, we focus on techniques

that approximate $p(\boldsymbol{z})$, without necessarily capturing its true form.

A Gaussian Mixture Model (GMM) is a mixture distribution defined as weighted sum of Gaussian distributions. Specifically, a K-component GMM, $\mathcal{G}$, is defined as

$$\mathcal{G} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.1}$$

where the component-wise hyper-parameters $\theta_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ are the prior, mean, and covariance, respectively. In order to maintain the validity of the distribution, the priors (also known as the weights) must sum to one: $\sum_k \pi_k = 1$.

While GMMs are often used as an unsupervised clustering method, they can also be leveraged as a powerful density estimator. We approximate the point cloud generating distribution $p(\boldsymbol{z})$ as a K - component GMM, denoted as $\mathcal{G}$. Accordingly, we assume the point cloud $\mathcal{Z}$ consists of N *i.i.d* samples from $\mathcal{G}$. The maximum-likelihood framework can then by applied to determine the hyper-parameters of $\mathcal{G}$. Applying the independence assumption and utilizing the GMM likelihood function Eq. (2.1), the log-likelihood function is given by

$$\ln \hat{p}(\mathcal{Z}|\boldsymbol{\theta}) = \ln \prod_{i=1}^{N} \hat{p}(z_i|\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.2}$$

Equation (2.2) is non-convex and further more, due to the non-identifiability of the Gaussian components, for each local-minima, there will be $K! - 1$ alternative parameter selections that yield the same log-likelihood, i.e. switching the parameters of components i and j will yield the same density. Of course, when performing density estimation, any of the $K!$ permutations will be equally valid, but the non-convexity of the function renders the solution sensitive to initialization - a point we will re-visit later. Another issue with Eq. (2.2) is that the sum inside of the log renders the cost function computationally ex-

pensive to optimize, since optimization variables cannot be easily isolated. The *de facto* standard approach to deal with this is the Expectation Maximization algorithm, which will be introduced next.

2.1.1 Expectation Maximization

Expectation-Maximization (EM) is an iterative procedure first introduced by Dempster *et al.* [10] that maximizes a lower bound on the log-likelihood. In the EM algorithm, we note that maximization of Eq. (2.2) in closed form would be made possible if the latent correspondence variables were known - i.e. which component generated each point. The expectation step computes the expected value of the latent correspondence between point z_i and component k, which are commonly referred to as the "responsibilities", γ_{ik}

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_k \pi_k \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{2.3}$$

Given the responsibilities, Eq. (2.2) becomes convex and can be optimized in closed form by setting the respective derivatives to zero. The M-step updates the component parameters through the following equations

$$\pi_k^{n+1} = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik} \tag{2.4}$$

$$\boldsymbol{\mu}_k^{n+1} = \frac{\sum_{i=1}^{N} \gamma_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma_{ik}} \tag{2.5}$$

$$\boldsymbol{\Sigma}_k^{n+1} = \frac{\sum_{i=1}^{N} \gamma_{ik} \boldsymbol{x}_i \boldsymbol{x}_i^T}{\sum_{n=1}^{N} \gamma_{ik}} - \boldsymbol{\mu}_k^{(n+1)} \boldsymbol{\mu}_k^{(n+1)^T} \tag{2.6}$$

The EM algorithm proceeds by iteratively performing the expectation (E) and maximization (M) steps until the log-likelihood Eq. (2.2) converges to within some pre-defined threshold. It is important to note that due to the introduction of the latent correspondence

variables, the EM algorithm is in fact maximizing a lower bound on the log likelihood. For more information on the application of the EM algorithm to GMMs, the reader is referred to Chapter 9 of [5] and the tutorial by Sridharan [35].

2.1.2 Hierarchical Expectation Maximization

A significant issue encountered when learning GMMs through the vanilla EM algorithm outlined in the previous section is the cost of computing the responsibility matrix Eq. (2.3). Due to the nature of 3D geometry, the point cloud generating function, $p(\boldsymbol{z})$, will be supported over some 2D manifold embedded in $\mathbb{R}^3$. This arises due to the fact that sensors observe surfaces and not volumes.[1] Accordingly, the responsibility matrix will have many terms that are almost zero - note that due to the infinite support of the Gaussian distribution, all entries will be non-zero given infinite numerical precision. The sparsity of the responsibility matrix can be leveraged to achieve an efficient hierarchical EM approach as proposed by Eckart *et al.* [14].

A K-component GMM is learned iteratively through a succession of smaller EM problems. For example, a $\hat{K}$-component GMM is first learned on the point cloud $\mathcal{Z}$ of size N, which produces a $\hat{K} \times N$ responsibility matrix. The point cloud is then partitioned into $\hat{K}$ chunks according to the sparsity of the responsibility matrix, γ_{ij}. Specifically, given some threshold parameter λ_p, point i is assigned to all chunks j satisfying $\gamma_{ij} > \lambda_p$. In order to prevent points being discarded, each point is always assigned to its maximum responsibility component, that is $\gamma_i^* = \operatorname{argmax}_j \gamma_{ij}$. Furthermore, in order to make λ_p more intuitive, the partitioning function can be restated as

$$\gamma_{ij} > \lambda_p \gamma_i^* \tag{2.7}$$

[1] This may not be strictly true. In the case of transparent materials, it may be possible to observe the the volume of points. This case will not be considered further.

The partitioned point cloud can be considered as $\hat{K}$ independent data sets for which $\hat{K}$ GMMs must be learned. However, in order to prevent double counting data, point-wise weighting must be introduced. Each point in the data set will be assigned a weight p_i derived from the partitioning process - this is initialized to 1 at the first layer. After the child GMM has been learned, the partitioning stage allocates a weight to each point according to the parent weight, p_i^- and the responsibility across the assigned chunks, C. The new weight is given by

$$p_{i,j}^+ = \frac{\gamma_{ij}}{\sum_{j' \in C} \gamma_{ij'}} p_i^- \tag{2.8}$$

In order to account for the new point wise weighting, the m-step of the EM algorithm must be updated. In this case, the normalizing constant N in the mixing weights Eq. (2.4) is replaced by $\sum_i p_i$.

The iterative procedure outlined above is repeated for L iterations or until some alternative stopping criteria is achieved. Eckart *et al.* propose a stopping criteria based on the ratio of eigenvalues of the covariance matrix. As stated previously, the generating distribution $p(\boldsymbol{z})$ is some 2D surface, and as a result, the Gaussian components learned will tend towards becoming degenerate along one dimension. Using this insight, expansion of the hierarchy along a given branch is terminated when a parent component satisfies the criterion

$$\frac{\min_i \lambda_{i,\boldsymbol{\Sigma}}}{\lambda_{0,\boldsymbol{\Sigma}} + \lambda_{1,\boldsymbol{\Sigma}} + \lambda_{2,\boldsymbol{\Sigma}}} < \lambda_p \tag{2.9}$$

where $\lambda_{i,\boldsymbol{\Sigma}}$ is the i-th eigenvalue of the covariance matrix $\boldsymbol{\Sigma}$.

The hierarchical EM procedure can achieve a substantial reduction in computation. For the purpose of illustration, assume that after learning a $\hat{K}$ component GMM, the partitioning step yields $\hat{K}$ equal sized chunks with some factor κ of soft assignments. This yields $\hat{K}$ problems each with data sets of size $\kappa \frac{N}{\hat{K}}$. This process is repeated for L levels, which can be determined based on the desired number of components $L = \log_{\hat{K}} K$. Thus, we see

that the computation is reduced from $O(NK)$ to $O(\kappa N\hat{K}L) = O(\kappa N\hat{K}\log(K))$, which is an exponential reduction in K. Empirical results show a dramatic performance increase with relatively little impact on modeling accuracy, see [14] for more details.

2.1.3 Initialization Strategies

The non-convexity of the log-likelihood cost function Eq. (2.2) makes the density estimation problem sensitive to initialization. In order to increase the efficiency of the algorithm, along with the accuracy of the solution it is important to consider the effect of initialization. Here we briefly compare three initialization strategies:

- **Random sampling**: The means are determined by sampling K points from the input point cloud. The covariances are isotropic with magnitude equal to the square of the maximum dimension of the point cloud bounding box. The responsibility matrix is computed using uniform weights.

- **Uniform sampling over bounding box**: The means are determined by sampling K points uniformly from the the volume defined by the bounding box. The covariances and responsibility computation is the same as the previous case.

- **k-means**: Points are hard-partitioned according to the k-means algorithm. Yielding N $1 \times K$ one-hot vectors, which are stacked to give the responsibility matrix. The Scikit-learn implementation of k-means was used for this evaluation [27].

The listed initialization strategies were evaluated on point cloud data to assess both efficiency and quality. The computational efficiency of the approach was determined by run time and number of iterations required to converge. The initialization time was recorded to account for the increased complexity of k-means.

The quality of the model was assessed using the Peak Signal to Noise Ratio (PSNR) as proposed by Eckart *et al*. [14]. The PSNR is computed by taking the inverse root-mean-square-error (RMSE) between the source point cloud and a point cloud generated by resampling from the Gaussian distribution. The error for a point in the resampled cloud is defined as the minimum squared euclidean distance to any point in the source cloud, which can be efficiently computed using a k-d tree. The RMSE is normalized with respect the length of the diagonal of the bounding box containing the source point cloud. We note that the PSNR as presented is non-symmetric, which results in no penalization for failing to model regions in the source point cloud. We amend this through the two-sided PNSR, defined as

$$PSNR2[dB] = 20\log\frac{BB_{max}}{RMSE(X,\hat{X})} + 20\log\frac{BB_{max}}{RMSE(\hat{X},X)} \tag{2.10}$$

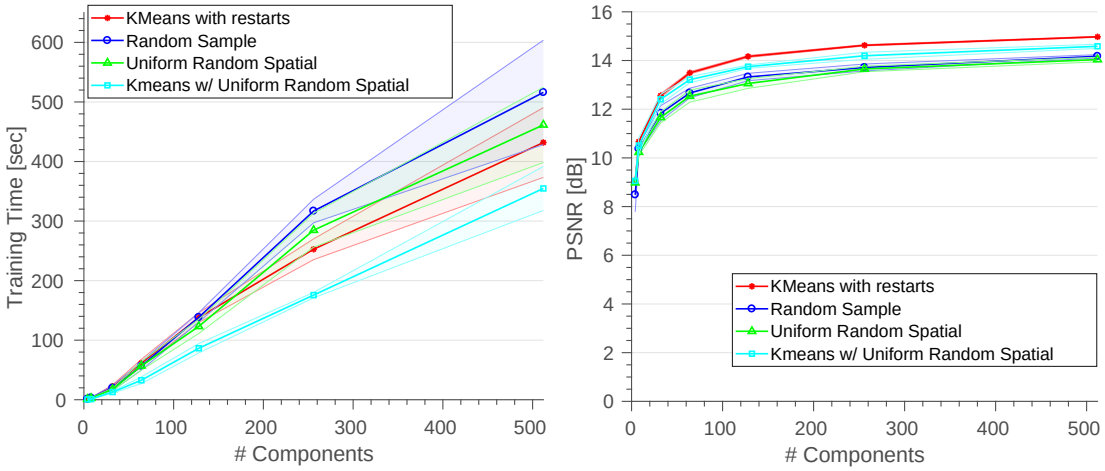
where BB_{max} is the length of the bounding box diagonal and $\hat{X}$ is the point cloud resampled from the GMM.

The results of the initialization trials are shown in Fig. 2.1.

2.1.4 Component Pruning

Component pruning strategies are typically employed to reduce the complexity of the final mixture models. The strategies employed are typically heuristic in nature, leveraging geometric insights to discard components that do not capture a sufficient portion of the observed distribution. In order to prevent data being discarded due to pruning, in the case of hierarchical EM, the pruning strategies are applied before partitioning.

The most simple pruning strategy is based on the support size of the component, which can be computed as $N_j = N\pi_j$ (or $N_j = \sum_i p_i\pi_j$ in the case of hierarchical EM). When the support - the number of points a component accounts for - is less than some threshold



(a) Training time (Initialization + EM).

(b) Peak signal to noise ratio.

Figure 2.1: Comparison of initialization strategies on the training time (a) and the resulting quality (b) of the GMM. Results indicate that while k-means initialization is more expensive, the subsequent reduced number of EM iterations required to converge gives an overall reduction in run time. The highest PSNR is seen for k-means with multiple restarts incurs a greater run-time cost, it results in a higher quality model. Although, a single run of k-means demonstrates strong performance, with a significant reduction in running time (25%). Results were generated for the numbers of components in the set $\{4, 8, 32, 64, 128, 256, 512\}$.

λ_s, then component is discarded. This thresholding strategy heavily depends on the environment of interest and can be overly aggressive in pruning components describing small features.

A more robust pruning strategy can be achieved by leveraging the sensor model of the component. Specifically, given a known range dependent point density, $\sigma(r)$, components with a support density below this value can be pruned. The $(2 - sigma)$ support density is given by $\frac{N_j}{4\pi\sqrt{\lambda_{0,\mathbf{\Sigma}}\lambda_{1,\mathbf{\Sigma}}}}$, where $\lambda_{0,\mathbf{\Sigma}}$ and $\lambda_{1,\mathbf{\Sigma}}$ are the two largest eigenvalues of the covariance $\mathbf{\Sigma}_j$. This criterion implicitly assumes that the component is planar and as such, will strictly over-estimate the point density, leading to a more conservative pruning policy than the naïve support based strategy.

2.1.5 Alternative Learning Strategies

While expectation maximization is almost certainly the most widely used approach for learning GMMs, other approaches exist and their competitiveness has been increasing in recent times. Hosseini *et al.*[18] reformulate the log-likelihood objective on the product manifold of $(\mathbb{R}^n \times \mathbb{P}^n)^K$, where P^n is the manifold of $n \times n$ positive semi-definite matrices. The reformulated cost function is optimized using first order techniques modified for optimization on Riemannian manifolds: Conjugate-Gradient, LBFGS and stochastic gradient descent. The results suggest faster convergence time than EM in some scenarios. However, their algorithms were evaluated on high-dimensional data-sets with small numbers of components (3-7), which may inhibit scaling as the dimension of the product manifold scales exponentially in the number of components.

A not yet discussed problem associated with the maximum likelihood formulation is the impact of singular components. As a component mean converges to an individual sample point, the summation inside the log of Eq. (2.2) will contain a term of the form $\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}$. Should the covariance go to towards zero (due to insufficient attractive forces from other data points), this term will go to ∞, which is desirable in terms of maximizing Eq. (2.2), but is useless in terms of density estimation. While this problem can be alleviated through component conditioning and singular component pruning, alternative strategies to MLE exist which do not suffer from this problem.

Kolouri *et al.*[21] proposed application of the Wasserstein distance to GMM learning. In order to overcome the intractability of Wasserstein distance, a 1D sliced variation is proposed, which utilizes random projections to approximate the full 3D cost function. The optimization problem then consists of minimizing the distance between the target K-component distribution and a distribution constructed from the original data set by placing Gaussian kernels over each of the data points. Once again, the results are presented for rel-

atively small mixtures (25 components), but the dimensionality is high (128-dimensional), so scaling may be impeded.

Another strategy to overcome the limitations of MLE is maximum *a posteriori* estimation through variational inference (see Chapter 10 of [5]). Variational inference can prevent the solution overfitting individual data points through a suitably defined prior. Additionally, the estimated posterior distribution gives a measure of uncertainty for the model, which can be leveraged for the purposes of exploration. The approaches described thus far have each required a pre-specified number of components (although this is somewhat overcome in the hierarchical formulation). Rasmussen overcame this problem through the formulation of the infinite GMM variational inference problem [30], which includes estimation of the number of components into the inference problem. While this sounds promising, it is not immediately clear how applicable such an approach would be to density estimation (as opposed to clustering).

2.2 Probabilistic Environment Modeling

In the robotics community, occupancy grid mapping is the dominant technique used for modeling occupancy. Occupancy grid mapping was first introduced by Elfes [16] as a tractable framework to accumulate sensor measurements and interpret their combined information probabilistically. In order to render the approach tractable, cells in the grid are assumed to be independent. This independence assumption coupled with a fixed discretization of the environment can result in the classical jagged occupancy representation. The independence assumption can be lessened through the use of ray potentials and surface priors [26]. The result of this is an occupancy grid with smoother variations, which can have the negative side effect of less defined edges. However, as with any MAP estimation, sufficiently large data samples will force the ML solution to dominate.

The many variants that have been proposed seek to overcome one or both of these limitations. OctoMap [17], which is arguably the *de facto* standard occupancy mapping framework, reduces the impact of fixed resolution through a memory efficient hierarchical structure, enabling the use of significantly higher resolution. However, OctoMap still incurs discretization errors due to fixed resolution at the lowest level. Recent work by Sun *et al.* proposes to tackle the independence assumption in grid maps through a Gaussian Markov Random Field, which decomposes a grid model into sub-grids with known inter- and intra sub-grid dependencies [37]. Once again, the underlying discretization limits the resolution of the model.

Continuous and semi-continuous approaches have been presented to mitigate against discretization errors. NDT-OM [32] builds a 3D occupancy grid map where each occupied cell contains a Gaussian Distribution to enable reasoning about the distribution of evidence in the cells. Extensions to the NDT method have proposed OctoMap-like techniques to allow for variable resolution NDT maps [15, 23]. Gaussian Processes (GP) occupancy mapping techniques first introduced by O'Callaghan and Ramos [24] learn a continuous representation of the environment using GP regression. GP occupancy maps, as first introduced, suffer from poor training and test complexity, $O(N^3)$ and $O(N^2M)$, respectively where N and M are the number of training and testing points. Several extensions have improved data efficiency, the most recent of which achieves reduced training complexity through recursive updates and reduced testing complexity through test data partitioning [13]. Recently, Hilbert Maps [29] have been proposed as an efficient method of learning occupancy models by training a classifier in a reproducing kernel Hilbert space. However, due to the sharp transitions between occupied and free space, capturing the classification boundary directly can be difficult, see Chapter 3 for more information.

2.3 Point Cloud Compression

Compressing data requires a trade off between efficiency and fidelity. An increased compression rate will typically require an increase in computation and conversely, increased computational efficiency will typical result in lower compression rates. In mobile robotics applications, real time viability on mobile computational hardware is a rigid constraint on the design of a compression system.

An octree is commonly used as the base of compression algorithms due to the compact representation and the discrete nature yielding regular bit patterns amenable to compression [33, 34]. Alternatively, depth images can be compressed directly [6] or images constructed from the 3D model [4]. Bell *et al.* utilize standard video compression techniques to achieve streaming of high-resolution 3D data at sensor frame rate (30 Hz) - although, a high powered desktop CPU and GPU were leveraged [4]. The results demonstrated using standard compression techniques are interesting given that lossy compression formats, such as JPEG, are biased towards preserving colors in a manner consistent with the human visual system. Recently, there has been increased focus on the streaming of high-resolution depth video for augmented and virtual reality applications [4, 36]. Stotko *et al.* utilize local model updates through voxel hashing to achieve a compact representation for transmission [36]. While the authors demonstrate that compression can be achieved on a low compute system, the discretization of measurements suffers from the same problems associated with vanilla occupancy mapping.

There has been increased focus on the application of deep neural networks to point clouds and training on small datasets has yielded some promising results [1]. Given a model pre-trained on a sufficiently diverse dataset, only the relatively small latent space representation of an input point cloud would need to be transmitted (since each agent would have an identical copy of the generative network). However, this approach has two limitations

that must be overcome: firstly, the need for pre-training requires the existence of a large, diverse depth image database, which is currently unavailable. Secondly, it is likely that online updating of the model would be necessary to ensure robust operation, which would require efficient training schemes and a means for synchronizing distributed models.

2.4 Information Theoretic Exploration

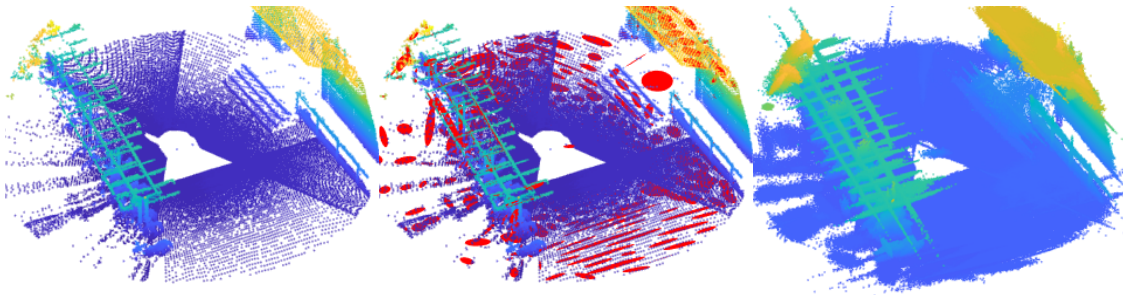
Given a probabilistic representation of the environment, it becomes possible to perform robot exploration through the optimization of an information theoretic objective. Charrow *et al.* presented a objective based on the Cauchy-Schwarz Quadratic Mutual Information (CSQMI)[8], which is derived from the Renyi's entropy - a generalization of the Shannon entropy (see [8] or Chapter 2 of [28] for more information). The CSQMI objective enables trajectories to be selected based on a measure of the information they contain about the environment being explored. Trajectories can be generated through optimization procedures that maximize the CSQMI objective [7] or, after noting that mutual information is a sub-modular function [22], through sampling based strategies.

Chapter 3

Variable Resolution Occupancy Modeling

Occupancy mapping is often applied as a basic framework for core capabilities of autonomous systems. Occupancy maps have been leveraged for many applications: localization, collision avoidance, exploration, environment reasoning, etc. Representing occupancy in a probabilistic manner enables reasoning about the uncertainty in the occupancy model that arises due to imperfect state estimation and noise in sensor measurements. Appropriate accounting for this uncertainty is useful for increased robustness and a necessity where error in the environmental model can result in catastrophic failure, e.g. planning infeasible trajectories due to occupied space being modeled as free.

Although the world we seek to model is continuous, reasoning over continuous models is difficult due to the requisite integration over non-trivial spaces, which often necessitates numerical evaluation. A commonly employed mitigation strategy is the preemptive discretization of the environment, typically through the imposition of a grid structure [15, 17, 20, 23, 24, 32], which renders evaluation of the model significantly less complex. However, *a priori* discretization results in loss of information due to the non-invertible



(a) (b) (c)

Figure 3.1: Gaussian mixture model occupancy mapping pipeline. (a) input pointcloud. (b) 850 component GMM with laser pointcloud overlaid. (c) Resampled pointcloud can be back-projected to obtain free rays and integrated into an occupancy model to derive probability of occupancy.

mapping from measurements to discrete cells, i.e. it is known that a cell contains a measurement, but it is not known where in the cell it lies. Approaches have been proposed to mitigate against this ambiguity through increased resolution made feasible by multiple levels of discretization that can be matched to the observed data [17]. Recently, there has been a growing interest in direct modeling of the continuous distributions, from which occupancy models can be derived, or in some cases, the models can be leveraged for direct evaluation [13, 29]. We propose to continue in this direction and build a compact generative model of sensor measurements using Gaussian Mixture Models (GMMs).

By leveraging GMMs as a model of observed surfaces, we build what we term as a mixture of Gaussian cones, which fully describes the continuous distribution of free space evidence contained within the measurement. In this manner, we determine the evidence of free space approximately for arbitrary shapes in 2D and polyhedra in 3D. We utilize a Bernoulli Distribution to model the opposing evidence for occupied space encoded by the GMM and the evidence for free space derived from the corresponding cones. We demonstrate the accuracy and efficiency of this approach compared to state of the art discrete and continuous methods.

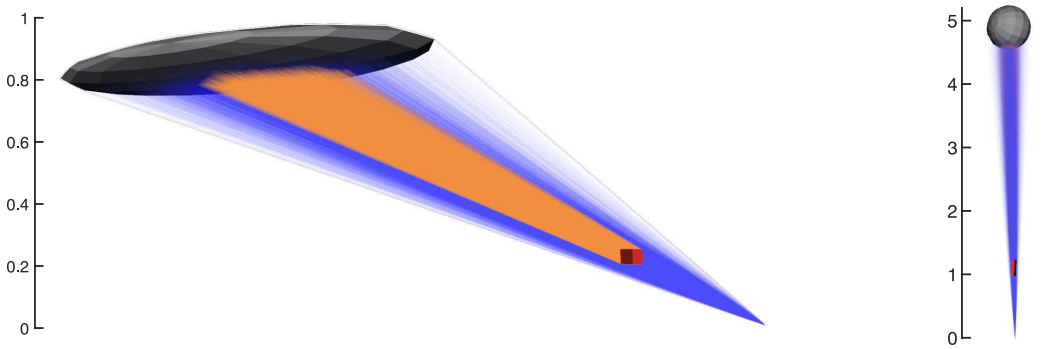
This chapter details work presented in [25].

3.1 Free Space Modeling

While a GMM $\mathcal{G}$ models the observed occupied space, it ignores the information contained in the beams between the sensor and the end points. Free space can be modeled explicitly by additionally learning a free space GMM $\mathcal{F}$, whose components lie along the beams associated with a component of $\mathcal{G}$. However, this approach has two principle limitations: Firstly, each occupied point corresponds to an infinite number of free space points lying on the beam from the occupied point to the sensor location; Secondly, the free space evidence resulting from an individual point is constant along the entire beam between the sensor location and the occupied point, making Gaussian decay an inappropriate modeling choice. We can overcome both of these limitations by acknowledging the implicit modeling of free space embedded in the occupied space GMM $\mathcal{G}$.

3.2 Mixtures of Gaussian Cones

Making use of the fact that the free space is entirely defined by the beam end point, we propose to derive the free space model from the tuple $\{\mathcal{G}_t, \boldsymbol{s}_t\}_{t=1,\cdots,T}$, where $\boldsymbol{s}_t$ is the sensor pose corresponding to the observation $\mathcal{Z}_t$ on which the mixture model $\mathcal{G}_t$ was trained. The free-space beams that connect the sensor location to a given occupied component, when clustered, form an oblique cone, as shown in Fig. 3.2. The evidence of free space at a given point $\boldsymbol{x}$ is the evidence of an occupied point lying beyond $\boldsymbol{x}$ along the ray originating at the sensor location and passing through $\boldsymbol{x}$. The evidence for free-space in a region V, is given by the evidence of occupancy contained within the truncated polygonal pyramid, with apex and shape defined by the spherical polygon derived from the projection of the volume V onto a sphere centered at the sensor location. We utilize this fact to derive an analytic expression for the free-space evidence in 2D. However, due to a non-trivial region



(a) (b)

Figure 3.2: Probability cone for a Gaussian component of the observation model. (a) The 3σ gray ellipsoid shows the input space occupancy distribution. The free space distribution is shown by the blue rays. The orange rays show the evidence for free space in the red voxel. (b) The same ellipsoid and voxel after applying the isotropic transformation to enable the analytic free space computation. Note the significant stretching induced along the radial direction (in arbitrary units) due to the approximate planar nature of the input space GMM.

of integration, to the best of the authors' knowledge, direct extension of this approach to 3D is not possible (see Section 5.2 for more information). Instead, we develop two alternative techniques: integration over bivariate Gaussians through polygon projection and fast Monte Carlo ray tracing.

The free space evidence integral can be expressed as

$$Evidence(\boldsymbol{x}) = \int_V p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{3.1}$$

The nature of the volume V makes parameterization in spherical coordinates a natural choice. Before performing the coordinate transformation, it is necessary to transform the occupied Gaussian to be isotropic. This is achieved by the mapping $\boldsymbol{x}_I = \boldsymbol{S}_k^{-1/2} \boldsymbol{V}_k^T (\boldsymbol{x} - \boldsymbol{\mu}_k)$, where $\boldsymbol{V}_k \boldsymbol{S}_k \boldsymbol{V}_k^T = \boldsymbol{\Sigma}_k$ is the singular-value decomposition of the anisotropic covariance matrix.

A single Gaussian component in this new space is given by

$$p(\boldsymbol{x}_I) = \frac{1}{(2\pi)^{3/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_I - \boldsymbol{p}_k)^T (\boldsymbol{x}_I - \boldsymbol{p}_k)\right)$$

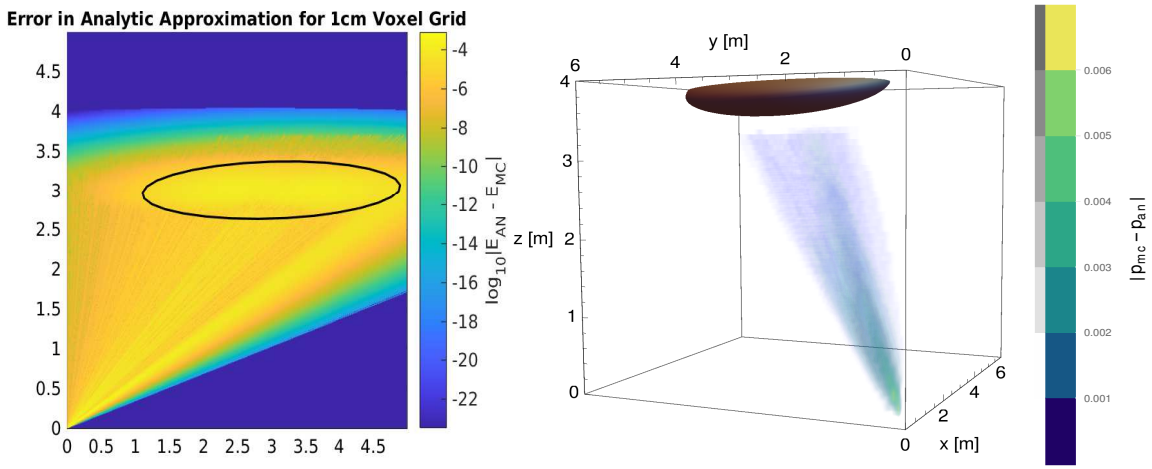
where $\boldsymbol{p}_k$ is the sensor location corresponding to component k. Finally, we apply a rotation $\boldsymbol{R}_k$, such that the component mean lies along a co-ordinate axis, i.e. in 2D, we rotate the mean onto the x-axis, such that $\boldsymbol{R}_k\boldsymbol{p}_k = \begin{bmatrix} p_{k,x} & 0 \end{bmatrix}^\top$.

3.2.1 2D Occupancy Grids

In the 2D case, we reparameterize Eq. (3.1) to polar coordinates and rotate the mean onto the x-axis

$$\begin{aligned} E(x) &= \frac{1}{2\pi}\int_V re^{-\frac{1}{2}\left\|\begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} - \begin{bmatrix} p_x \\ 0 \end{bmatrix}\right\|_2^2} \mathrm{d}r\mathrm{d}\theta \\ &= \frac{1}{2\pi}\int_V e^{-\frac{1}{2}p_x^2\sin^2\theta} re^{-\frac{1}{2}(r-p_x\cos\theta)^2}\mathrm{d}r\mathrm{d}\theta \\ &= \frac{1}{2\pi}\int_\theta e^{-\frac{1}{2}p_x^2\sin^2\theta}\int_{r'}^{\infty} re^{-\frac{1}{2}(r-p_x\cos\theta)^2}\mathrm{d}r\mathrm{d}\theta \\ &= \frac{1}{2\pi}\int_\theta e^{-\frac{1}{2}p_x^2\sin^2\theta}\Bigg[e^{-\frac{1}{2}(r'-p_x\cos\theta)^2} + \\ &\qquad \left(1-\mathrm{erf}\left(\tfrac{r'-p_x\cos\theta}{\sqrt{2}}\right)\right)\sqrt{\tfrac{\pi}{2}}p_x\cos\theta\Bigg]\mathrm{d}\theta \end{aligned} \tag{3.2}$$

The final integral has no closed form solution in general. However, we note that due to the phase offset between sine and cosine, the first term has negligible contribution except around $\theta \approx 0$. In this case the term reduces to $exp(-(1/2)(r' - p_x)^2)$, which has value of 1 for $r' \approx p_x$. However, since the mean p_x is typically very large (due to the isotropic transform significantly stretching the along beam direction, see Fig. 3.2), the contribution of this term to the integral is negligible. We can apply the same logic to approximate the



(a) Error in Eq. (3.3) for 1cm cell.

(b) Error in Eq. (3.7) for 10cm voxel.

Figure 3.3: Absolute error in the free space approximation for 2D (a) and 3D (b). The error was calculated as the difference between the Monte Carlo ray tracing estimate and the analytic value for the probability of occupancy due to a single Gaussian component observed from the origin.

term inside the error function as constant evaluated at $\theta = 0$.

$$\begin{aligned} E(x) &\approx \frac{1}{\sqrt{8\pi}} \int_{\theta} e^{-\frac{1}{2}p_x^2 \sin^2\theta} \left(1 - \text{erf}\left(\frac{r' - p_x \cos\theta}{\sqrt{2}}\right)\right) p_x \cos\theta \mathrm{d}\theta \\ &\approx \frac{1}{\sqrt{8\pi}} \left(1 - \text{erf}\left(\frac{r' - p_x}{\sqrt{2}}\right)\right) \int_{\theta} e^{-\frac{1}{2}p_x^2 \sin^2\theta} p_x \cos\theta \mathrm{d}\theta \\ &\approx \frac{1}{4} \left(1 - \text{erf}\left(\frac{r' - p_x}{\sqrt{2}}\right)\right) \left(\text{erf}\left(\frac{p_x \sin\theta_2}{\sqrt{2}}\right) - \text{erf}\left(\frac{p_x \sin\theta_1}{\sqrt{2}}\right)\right) \end{aligned} \tag{3.3}$$

Here, we determine θ_1 and θ_2 from angle centered on the sensing location that bounds the transformed cell. Additionally, the radial term r' is given by the closest radial point, which is a faithful approximation for cells outside of the 3-sigma occupied ellipse. Fig. 3.3a shows the approximation error for a range of values for r' and p_x.

3.2.2 3D Voxel Grid

Direct extension of the 2D approach to 3D is not possible, see Section 5.2. However, noticing that components are approximately planar (see Fig. 3.2a), the problem can be

converted to the integration of a bivariate Gaussian over a convex polygonal domain. The domain of integration is determined by the projection of the voxel of interest to the plane defined by the eigenvector of the Gaussian covariance corresponding to the the smallest eigenvalue. The bivariate Gaussian is defined by the remaining two dimensions of the 3D Gaussian and the isotropic transform outlined in Section 3.2 is applied. As this is an affine transform, the convexity of the projected polygon is preserved. This 2D integral has received some attention in the numerical integration literature [3, 12], however, these approaches are designed to provide significantly higher accuracy than is necessary for our application. A less complex, albeit less accurate, strategy can then be achieved through application of Green's Theorem

$$\frac{1}{2\pi}\int_P e^{-\frac{1}{2}(x^2+y^2)}\mathrm{d}x\mathrm{d}y = \frac{1}{\sqrt{8\pi}}\oint_{\partial P} e^{-\frac{1}{2}x^2}\mathrm{erf}(\frac{y(x)}{\sqrt{2}})\mathrm{d}x \tag{3.4}$$

The path of integration ∂P is defined piece-wise as the line segments defining the edges of the polygon traversed in counter clock-wise order. Unfortunately, the path integral Eq. (3.4) has no-analytic solution. This is remedied by introducing the function $e(x) = 1 - \exp\left(-\frac{2}{\pi}x(x+\sqrt{\pi})\right)$, from which a piece-wise approximation of the error function is constructed.

$$\mathrm{erf}(x) \approx \hat{e}(x) = \begin{cases} e(x) & x \geq 0 \\ -e(-x) & x < 0 \end{cases} \tag{3.5}$$

This approximation has a maximum absolute error of 0.0140, a maximum absolute relative error of 0.0166 and an average absolute error of 0.0048, which is sufficiently accurate for our application.

Each edge k of the N_P edges in ∂P is defined by the line $y = m_k x + c_k$. The path

integral Eq. (3.4) can be decomposed into a sum of integrals along the edges

$$\frac{1}{\sqrt{8\pi}} \sum_{k=1}^{N_P} \int_{x_k}^{x_{k+1}} e^{-\frac{1}{2}x^2} \hat{e}(m_k x + c_k) \mathrm{d}x \tag{3.6}$$

The solution for each edge integral, I_k, is given by

$$\frac{\pi}{\sqrt{2}} \left(\frac{e^{\frac{m_k^2 - c_k(2 s_y c_k + \sqrt{2\pi})}{d_k}}}{\sqrt{d_k}} \mathrm{erf}\left(\frac{m_k(2 s_y c_k + \sqrt{2\pi}) + s_y d_k x}{\sqrt{2\pi d_k}} \right) - s_y \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) \right)$$

where $d_k = 2m_k^2 + \pi$ and $s_y = \mathrm{sign}(y(x))$. Due to the piece-wise definition of $\hat{e}(x)$, it is necessary to evaluate I_k in a piece-wise manner. If the sign of y_k and y_{k+1} differ, then the I_k must be evaluated at the x intercept ($x_z = -\frac{c}{m}$). In general, the solution to Eq. (3.4) is given by

$$\frac{1}{\sqrt{8\pi}} \sum_{k=1}^{N_P} \left(I_k |_{x_z}^{x_{k+1}} - I_k |_{x_k}^{x_z} \right) \tag{3.7}$$

3.2.3 Monte Carlo Estimation

While the previous expression is useful for computing the free-space evidence at a point, a more efficient method can be used for computing the voxel grid representation of the environment through the use of ray-tracing [2]. Specifically, we utilize the intuition from Fig. 3.2 and sample beam end points from the occupancy model. For each sampled point, we perform ray tracing to the corresponding sensor location to determine the intersected voxels. This method naturally captures the dependence of the free space on the observations and removes the need to store two separate mixture models.

3.3 Occupancy Modeling

It is possible to determine the number of observed points in any volume by solving the integral

$$N_V^H = N \sum_{k=1}^{K} \pi_k \int_V \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{3.8}$$

Since this integral cannot be computed in closed form, we evaluate Eq. (3.8) for $\mathcal{G}$ using Monte Carlo sampling. The free space evidence N_V^H is then determined by performing ray casting using the resampled points as described in Section 3.1. Alternatively, we can compute the free space evidence using the result Eq. (3.3) in 2D or Eq. (3.7) in 3D. We model the probability of occupancy as a Bernoulli distribution over every volume V in space. Specifically, we have $p(o|V) \sim \theta^{(o)}(1-\theta)^{(1-o)}$, where $o = \{0, 1\}$ is the binary occupancy variable. The maximum *a posteriori* estimate of the distribution parameters is given by

$$\theta^{MAP} = \frac{N_V^H + N_p}{N_V^H + N_V^M + 2N_p} \tag{3.9}$$

where N_p is the contribution due to the prior and is user-defined to control the sensitivity of the distribution to new observations. The prior model forces all unobserved points to have a probability of occupancy of 0.5.

It is important to note that the voxelization presented here is not the same as standard voxel grid mapping, since we merely use the voxel grid as a convenient interpretation of the underlying, continuous GMM. We could equivalently compute the occupancy in an arbitrary discretization of the space. However, this would likely preclude the use of the proposed efficient Monte Carlo ray tracing strategy and would instead require computation of the intersection between the discretized shapes and the sampled beams.

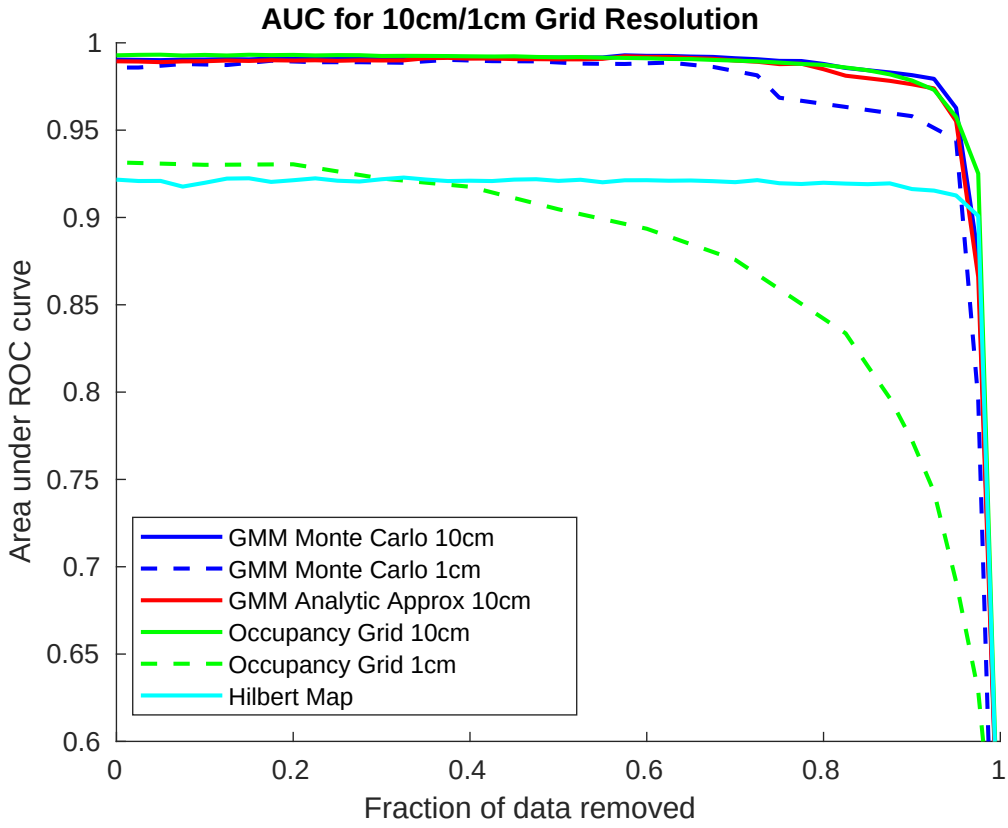


Figure 3.4: Area under the ROC curve for 10cm and 1cm resolution occupancy grids. Raw occupancy maps and both GMM variants show equivalent performance superior to that of Hilbert maps. The strong performance of standard occupancy grids can be attributed to the density of the data relative to the resolution. In the case of 1cm grids, this measurement density is no longer sufficient.

3.4 Results - 2D GMM Occupancy Modeling

The 2D occupancy model was tested on the Intel Research Lab environment from the Robotics Data Set Repository (Radish) [19]. The Monte Carlo and Analytic Approximation GMM occupancy reconstruction approaches were compared against standard occupancy grid mapping and the recently proposed Hilbert maps [29]. We evaluate the robust of the approaches to data sparsity and the accuracy of the approaches for reconstruction at high fidelity. In order to compare against Hilbert Maps, we use the code provided by the

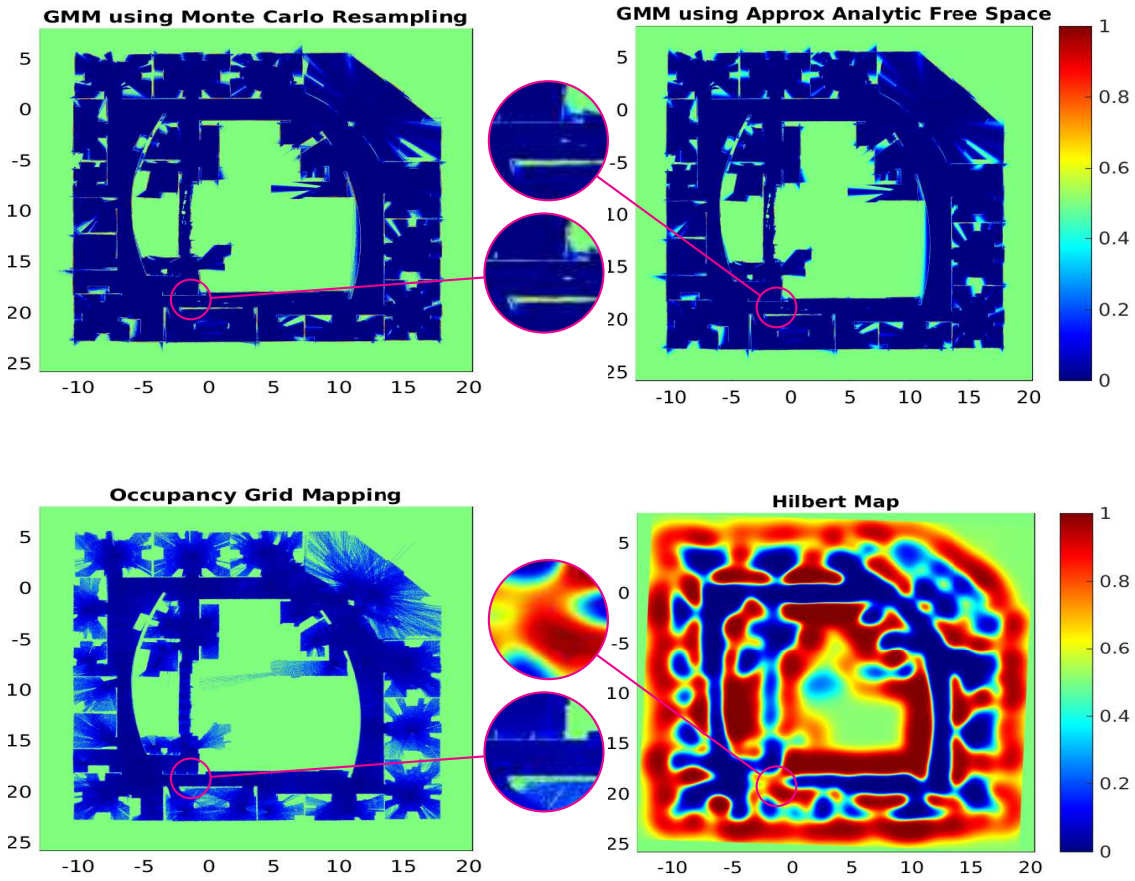


Figure 3.5: Occupancy Grid maps generated at 1cm resolution. Both GMM methods demonstrate the ability to generalize beyond the raw sensor measurement. The basic occupancy grid map (bottom right) shows inadequacy of the sensor density for this mapping resolution. The Hilbert Map generates an overly conservative estimate, resulting in incorrectly modeling free space as occupied.

authors[1] with the recommended parameter settings as in [29].

The accuracy and robustness of the proposed approach to sparsity in the observed data is shown in Fig. 3.4. We see that for 10cm resolution, the grid size is well matched to the density of the sensor and occupancy grid maps yield very strong performance, which is matched by that of the two variants of the proposed approach. As the resolution is increased to 1cm, the sparsity in the data and the independence assumption inherent in

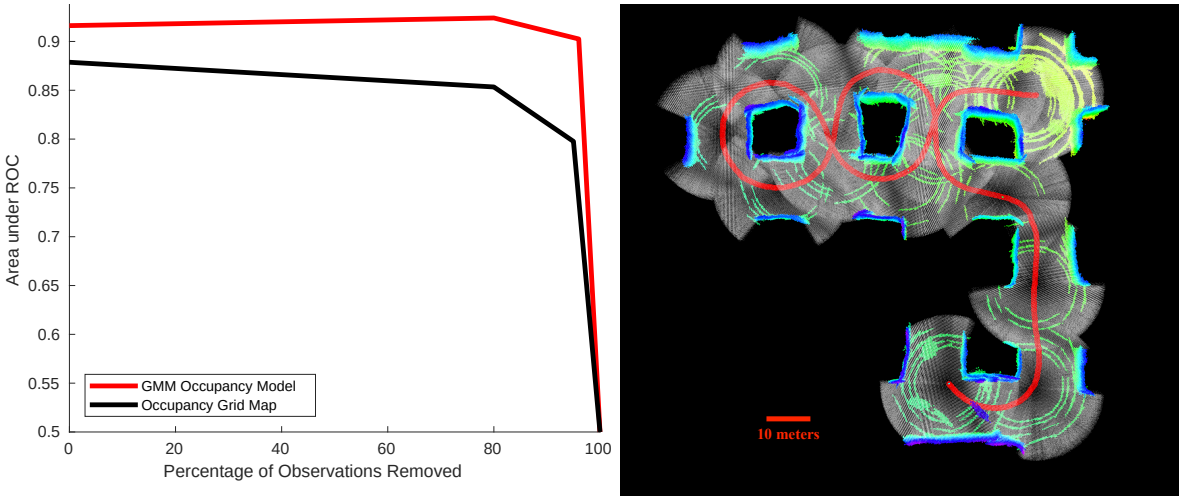
[1] `https://bitbucket.org/LionelOtt/hilbert_maps_rss2015`

occupancy grids results in reduced performance. Since the proposed analytic approach is less efficient for large grid sizes, we evaluate only the Monte Carlo strategy. We see comparable performance to the 10cm resolution grid for up to 70% data sparsity, after which point the sparse data combined with the high resolution results in regions with little evidence. However, we note that at both resolutions, the proposed approach out performs Hilbert Maps, which has a fixed intrinsic resolution determined by the kernel parameters used to learn the classifier.

The performance of the proposed approaches is compared qualitatively in Fig. 3.5. In the top row, the dense occupancy grids demonstrate the ability for the proposed approaches to reconstruct at high resolution. In the bottom left, the sparsity in standard occupancy grid mapping can be clearly observed. In the bottom right, we see the continuous occupancy distribution generated using Hilbert Maps. The magnified regions illustrate the sparsity in occupancy mapping and the conservative nature of Hilbert Maps, resulting in designation of the corridor as occupied. For the two proposed approaches, we see some free space evidence outside of the corner, which occurs due to the infinite extent of the GMM. However, we note that the occupied space is appropriately designated.

3.5 Results - 3D GMM Occupancy Modeling

The GMM occupancy model was tested with two datasets: a mine dataset (reconstruction shown in Fig. 3.6b) and the Freiburg campus dataset (raw pointcloud shown in Fig. 3.7a). For each pose and pointcloud, a free and occupied GMM were computed to represent the environment. 10^6 points were sampled from the GMMs, rays were computed from the start position, and the rays the probability of occupancy was updated in an occupancy grid map with 25 cm voxels. To determine the accuracy of reconstruction, the Area under ROC (AUROC) was computed for datasets with a varying percentage of observations removed. This



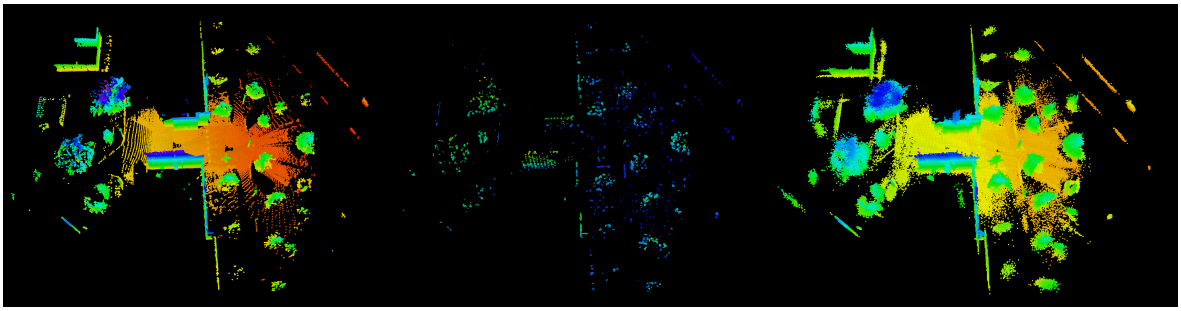
(a) (b)

Figure 3.6: (a) The effect of data sparsity on the fidelity of the map is measured by computing the area under curve for several models constructed by increasingly reducing the percentage of observations removed from the original data. (b) illustrates the GMM reconstruction of the map when training on all of the data. Free space is shown in white and the trajectory is shown in red. The occupied regions are shown with varying colors from yellow to blue according to height along the z-axis.

was compared to the accuracy of an occupancy grid map in Fig. 3.6a. The approximately continuous GMM occupancy model consistently outperforms the occupancy grid map and

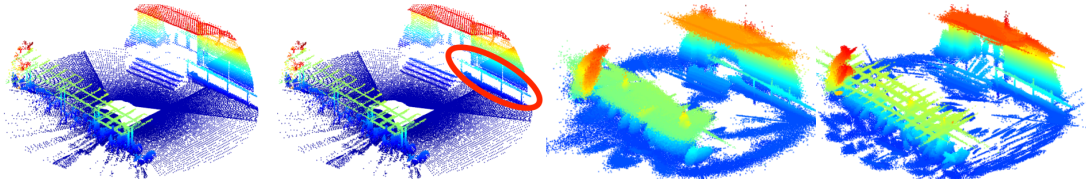
	AUC	Memory Usage (Bytes)	
GMM			Number of Components
	0.8179	40,000	1000
	0.8072	20,000	500
	0.8055	10,000	250
	0.7849	4,000	100
BGK			Number of Points
	0.7603	204,132	17,011
	0.7182	113,664	9,472
	0.6338	60,120	5,010
	0.5845	37,140	3,095
	0.5428	18,660	1,555

Table 3.1: AUC for given memory usage of GMM occupancy model compared to the performance of BGK inference occupancy maps. The GMM is better able to reconstruct the occupancy information in the environment with 50$\times$ less data than the Bayesian Generalized Kernel inference approach for occupancy mapping.



(a) (b) (c)

Figure 3.7: (a) Raw pointcloud from Freiburg campus dataset with coloring according to localization along the z-axis (total is 447,528 points and approximately 5.12MB of data assuming 32-bit floats). (b) Reconstruction of occupied regions with BGK inference for occupancy maps using 9472 points (111 KB data storage requirement). The voxels displayed are those with a probability of occupancy greater than or equal to 0.65. The AUC is 0.72. (c) Reconstruction of GMM map using 1000 components (40 KB data storage requirement). The AUC is 0.82.



(a) (b) (c) (d)

Figure 3.8: (a) A pointcloud of a courtyard and bike rack from the Freiburg campus dataset. (b) A railing is outlined in red. (c) The occupied points from the occupancy grid map derived from Monte Carlo resampling of a GMM consisting of 200 components and thresholded using a variance of 0.06. (d) The occupied points from the occupancy grid map derived from Monte Carlo resampling of a GMM consisting of 850 components and thresholded using a variance of 0.06. The AUC for (c) is 0.79 and the AUC for (d) is 0.82. For both GMMs the railing is properly reconstructed and identified as an obstacle.

as the number of points in the training dataset decreases, the GMM occupancy model is better able to infer the free and occupied spaces.

The key advantages of the GMM occupancy model over state-of-art occupancy mapping approaches like the occupancy grid map, Octomap, or Bayesian Generalized Kernel (BGK) Inference for occupancy map prediction [13] is the ability to correct for small or large changes in pose without the need to regenerate the entire map. BGK occupancy

mapping requires significantly more data storage to regenerate the map as can be seen in the results from the Freiburg Campus Dataset in Table 3.1. BGK maps must store approximately 50x more data than the GMM occupancy model to obtain the same level of accuracy. Furthermore, the timing needs for regenerating the entire map are prohibitive for real-time applications. Individual voxels cannot be meaningfully rotated and translated; however, individual components of a GMM are easily translated and rotated to correct for errors in pose [31].

The GMM infers the presence of occupied space in the neighborhood of observed points. However, the inferred occupied space may be unobservable from the sensor location due to occlusions from other surfaces. As a result, the inferred density may contribute erroneous free space evidence. For example, Fig. 3.8a depicts a laser scan from the Freiburg campus dataset that exhibits clutter and challenging surfaces to model in the form of a railing outlined in Fig. 3.8b. The inferred, but unobserved, portion of the wall contributes free space evidence to the voxels containing the railing, resulting in the probability of occupancy dropping for the railing voxels. To account for such situations, we leverage the variance of the voxels to determine the occupancy classification levels. The variance of a voxel is simply the variance of a Bernoulli random variable

$$\sigma^2 = p(x_i)(1 - p(x_i)) \tag{3.10}$$

The variance quantifies the uncertainty in the occupancy value of the cell and introduces a dead-band region in which voxels remain unclassified. The occupied/free thresholds are given by $\frac{1}{2}(1 \pm \sqrt{1 - 4\sigma^2})$. Appropriate selection of $\sigma^2 \in [0, 0.25]$ enables a threshold to be applied depending on how conservative we wish to be for a given application. Fig. 3.8c illustrates the occupied cells when a variance thresholding is applied when reconstructing a GMM with 200 components. The railing is well represented in the reconstruction and the

AUC is 0.79. Fig. 3.8d illustrates the same result with 850 GMM components and an AUC of 0.82.

Chapter 4

Distributed Mapping with Limited Communication for Multi-Robot Exploration

When performing multi-robot exploration with high density sensors such as RGB-D cameras, network utilization can quickly saturate due to the high volume of measurement data transfer. In this work, Gaussian Mixture Models (GMMs) are employed to achieve a high fidelity generative model of sensor measurements. In coming sensor measurements are modeled as GMMs and the compressed model is shared across the multi-robot team (Section 4.2). The reduced memory complexity of GMMs are further leveraged to enable exploration in large environments where the maintenance of a high fidelity occupancy is infeasible. A sliding voxel map representation (Section 4.1) is employed to overcome this challenge and is efficiently regenerated through Monte Carlo ray tracing of points resampled from the GMM. Finally, the local occupancy representation is leveraged to discard low utility points (Section 4.1.2).

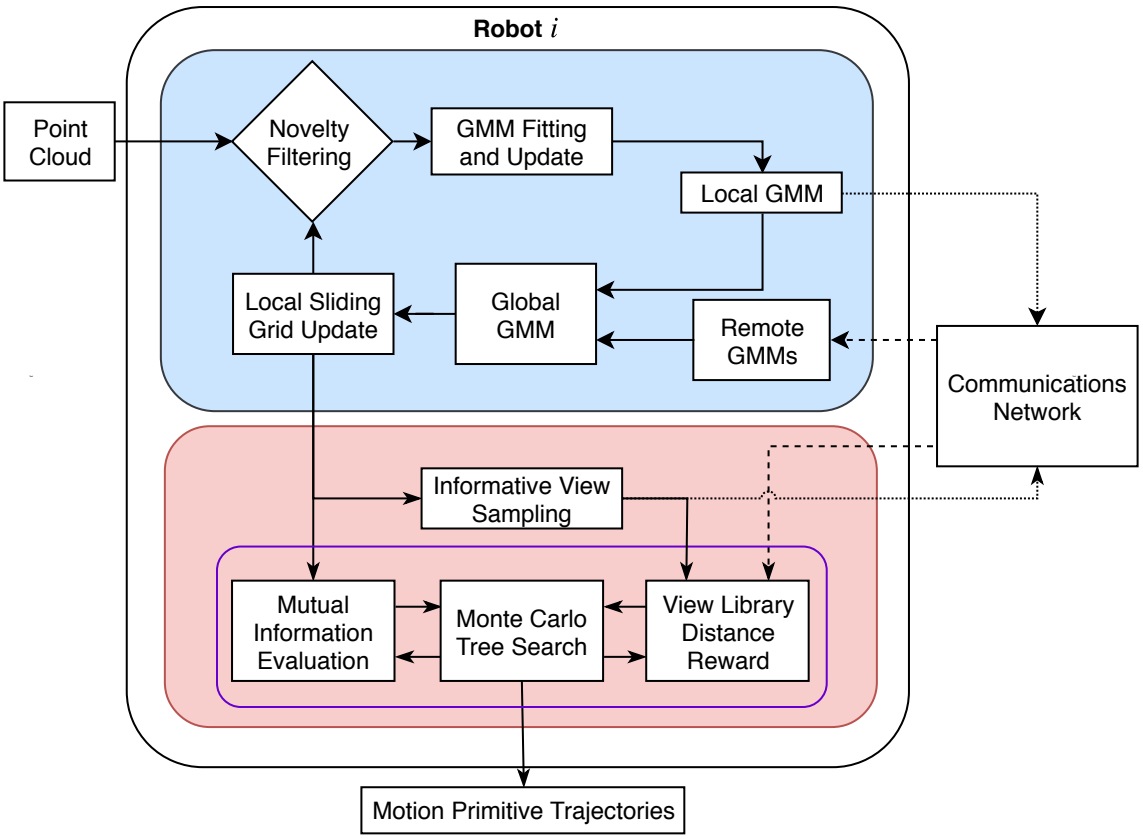


Figure 4.1: Multi-robot exploration system diagram. Solid lines represent local data flow at each member of the team. Dash lines represent information transmitted between team members over the network. The blue region depicts the GMM based mapping framework, which utilizes a local sliding occupancy grid map to identify the novel subset of each depth measurement. The novel subset is used to update and create components in the local GMM. Changes to the local GMM are transmitted across the network to other members of the team. The consistent global map is used to update the sliding occupancy representation, which tracks the agent as it moves through the environment. The local occupancy model is passed into the planner (red region), which performs information theoretic trajectory optimization using Monte Carlo Tree Search.

4.1 Sliding Voxel Map

The cost of maintaining a voxel map scales with the volume of the environment of interest. A robot centric, fixed size, sliding voxel map is employed to overcome this scaling issue. The voxel map is centered on the robot's location in world coordinates. The robot's orientation is ignored to circumvent the non-linear effect of rotation. Each time a new depth measurement is received, the sliding voxel map is updated. Based on the new map location, a local GMM, $\mathcal{G}_L$, is generated by selected the relevant subset of components from the global GMM. The local GMM is then used to update the voxel grid as detailed Sec 4.1.1. The updated map is used to identify the novel subset of the received measurement as detailed in Sec 4.1.2. The high frame rate of the depth sensors employed on exploration platforms results in small robot motion between sensor measurements. After updating the robot location, the new voxel map typically contains a substantial portion of the previous voxel map. The cost associated with memory copies is avoided by implementing the sliding map as a circular buffer, see Fig 4.2.

4.1.1 Monte Carlo ray tracing

Conversion between the GMM and the voxel map is achieved via Monte Carlo ray tracing. In principle, the GMM provides an analytic representation of both the occupied and free space evidence in each measurement. However, in $\mathbb{R}^3$, it is not possible to extract this evidence for a given volume (e.g. a voxel) in closed form. Accordingly, Monte Carlo sampling and ray tracing is employed to extract the occupied and free space evidence from the GMM. This amounts to drawing samples $\mathcal{Z}_R$ from $\mathcal{G}_L$ and performing ray tracing with $\mathcal{Z}_R$ in the standard occupancy mapping fashion [2]. The reduced mixture model $\mathcal{G}_L$ consists of the subset of $\mathcal{G}$ relevant to the current local voxel map. The relevant subset is defined a set of components with mean or sensor location within a bounding box centered on the

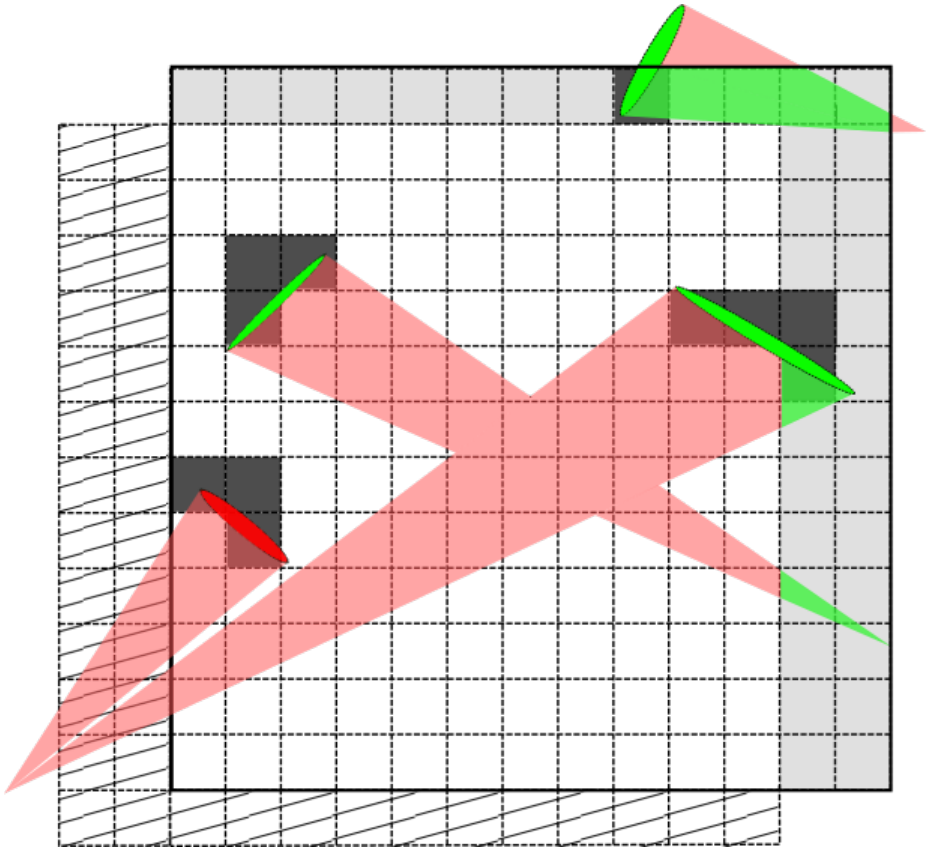


Figure 4.2: Sliding grid map and resampled ray trimming. After robot motion, the hatched region is out of bounds, its memory is reset and mapped to the new region, shown in gray. The trimmed portions of the resampled beams are shown in red and the portions used to update the map are shown in green.

robot position. The bounding box is box is set to be slightly larger than the sliding map in order to capture components that partially overlap with the map.

Due to the sliding voxel map representation, portions of resampled beams may already be contained in the map. In order to prevent double counting, beams corresponding to components already in the map are trimmed. The trimming process is detailed in Fig. 4.2.

4.1.2 Point novelty check

In typical robotic mapping applications, only a subset of each measurement will be novel. In order to increase the time and memory efficiency of the model, points below a pre-

defined novelty threshold are discarded. There exist many strategies for computing point-wise novelty [9]. In this application, the novelty of a point is defined as the entropy reduction achieved by the ray $\boldsymbol{r}$ from the sensor location to the point. The entropy reduction is computed by performing ray tracing on the local occupancy map and summing over over the cell-wise entropy delta. Any beams that intersect with previously unobserved space are considered sufficiently novel.

4.1.3 Component updating

The robustness of the occupancy map to measurement noise is increased by requiring several agreeing measurements to drive a voxel to an occupied or free state. As a result, beams that are considered novel may have been previously observed. In order to reduce the complexity of the model, the GMM is queried to determine whether existing components well model the novel beams. Points that contribute novel occupancy only can be checked directly against the components alone. Specifically, for each novel point $\boldsymbol{z}_n$ and each candidate component k, the Mahalanobis distance is computed, $d_{n,k} = (\boldsymbol{z}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{z}_n - \boldsymbol{\mu}_k)$. The distance $d_{n,k}$ is compared against a threshold d_{ll} and each valid component is updated with the fractional point $d_{n,k}^{-1} / \sum_{d_{n,i} < d_{ll}} d_{n,i}^{-1}$. In the case that the beam contributes free space entropy reduction, the valid set is to further constrained to satisfy $\|\boldsymbol{s}_k - \boldsymbol{s}\|_2 < d_s$, i.e. the current sensor location $\boldsymbol{s}$ is near the sensor location $\boldsymbol{s}_k$ from which the given component was observed.

4.2 Distributed Mapping

The principle motivation for the use of GMMs is their substantial reduction in complexity compared to raw point clouds. Leveraging this property enables a simple distributed mapping framework whereby each robot shares all novel components and all component

updates computed after each new local depth measurement is received. The result of this paradigm is that, barring communication failures, each robot maintains a complete global representation of the environment. The issue of communication failure is relegated to future work.

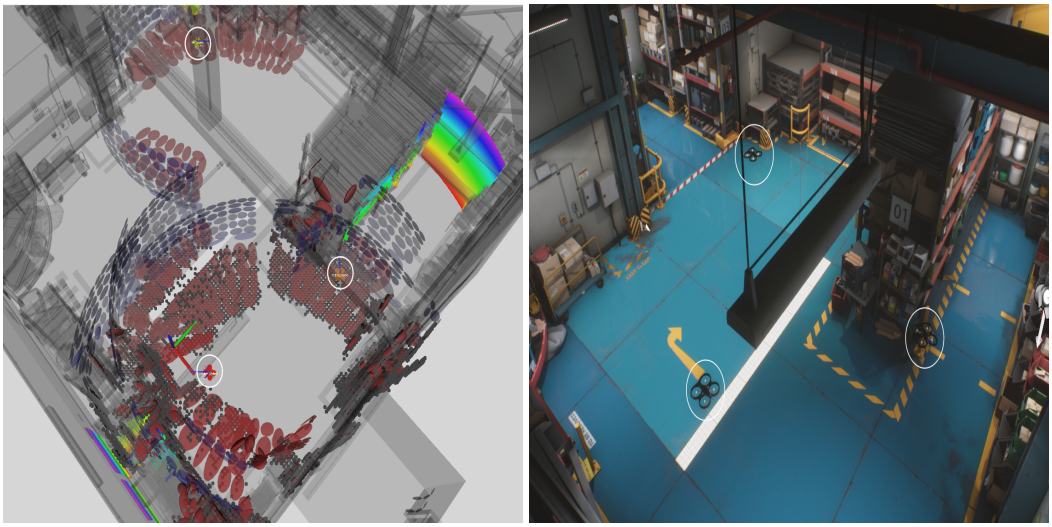
Novel data is shared in two forms. In the case of previously unseen components, the message consists of six fields:

1. **id**: Unique ID derived from the source robot ID and a local storage ID
2. **num_points**: Number of points supporting current component
3. **sensor_loc**: Position of sensor at time of observation
4. **mean**: Component mean
5. **covariance**: Size upper triangular components of the covariance matrix
6. **max_range**: Binary flag denoting support points are maximum range measurements

When signaling the update of a previously observed component, only the id and the new number of support points are shared.

4.3 Information Theoretic Exploration

The information theoretic exploration strategy used as part of the multi-robot system is not a contribution of this thesis. However, it is outlined here in brief for the sake of completeness. The exploration system optimizes a combined local and global information gain objective through Monte Carlo Tree Search. Local information gain is evaluated using the Cauchy-Schwarz Quadratic Mutual Information (CSQMI)[8] between an observation and the underlying map. Global information gain is derived from a view library that records



(a) Exploration visualization (b) Environment rendering

Figure 4.3: (a) Three robots (circled in white) perform an exploration task in (b) a complex unstructured 3D environment.

previously considered views that were predicted to be informative, but have not yet been visited. The view library is shared across all robots in the team to reduce redundancy in selected actions.

4.4 Results

The performance of the proposed exploration is evaluated quantitatively using map entropy reduction versus time and model complexity versus time. A common exploration objective is the minimization of map entropy and typically it is desirable to reduce entropy in the shortest time possible. Accordingly, the rate of entropy reduction can be used to evaluate performance. The rate of entropy reduction can be increased through inter-robot collaboration in a multi-robot team. Collaboration necessitates inter-robot communication. The complexity of the model is used to assess the amenability of the framework to information sharing and, by proxy, inter-agent collaboration.

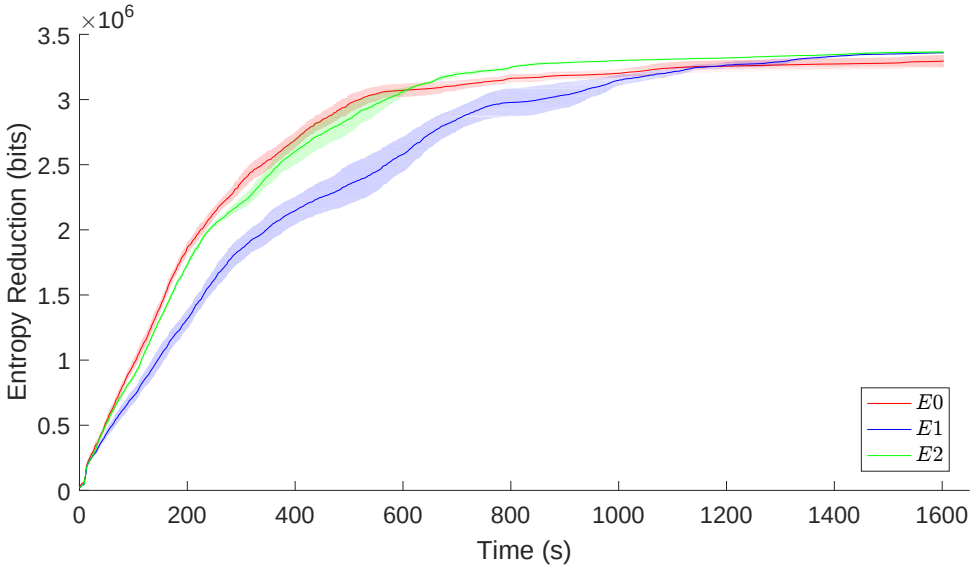


Figure 4.4: Entropy reduction achieved by the three evaluated approaches vs time. GMM based occupancy modeling achieves comparable performance to standard occupancy grid mapping.

Simulation experiments for the proposed approach were conducted in a high fidelity, rich 3D warehouse environment as shown in Fig. 4.3. Three quadrotor vehicles are used in all trials, each equipped with a forward facing depth sensor. Robots are placed at random starting positions for each trial and explore the environment for a fixed duration of 1600. Three alternative frameworks were compared and five trials were run for each framework. The evaluated frameworks were:

- **E0**: GMM derived occupancy modeling with inter-robot information sharing (informative views and GMMs).
- **E1**: Standard occupancy mapping with inter-robot information sharing (informative views and novel points).
- **E2**: Standard occupancy mapping with no inter-robot information sharing.

Entropy reduction over time shown in Fig. 4.4 demonstrates that the reduced complexity of GMM occupancy has no significant impact on the rate of exploration in the environment.

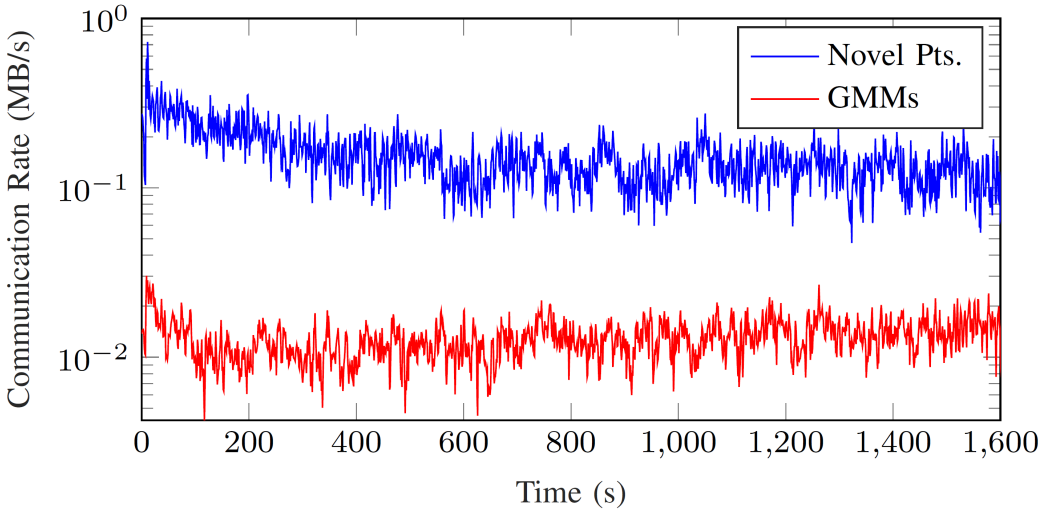


Figure 4.5: Reduction in information transferred between simulated agents communicating over UDP on a local network as illustrated by a reduced bandwidth utilization. The GMM framework demonstrates substantial reduction in memory complexity compared to the transmission of uncompressed novel portion of the observed point cloud.

Furthermore, Fig. 4.5 demonstrate the substantial reduction in network utilization when observations are shared using GMMs versus the novel components of the observed point clouds. The novel sub-cloud is computed as detailed in Section 4.1.2.

Chapter 5

Conclusions and Future Work

5.1 Summary

This thesis presented a framework for leveraging compact generative models of point cloud observations to achieve volumetric occupancy models, which can be utilized as a low-bandwidth back end for multi-robot exploration. Chapter 3 presented the variable resolution occupancy mapping framework, which reconstructs occupancy grids from Gaussian Mixture Models. Techniques were presented to compute occupancy for individual regions based on approximations to analytic integrals over Gaussian cones. Additionally, Monte Carlo based integration is presented to enable efficient reconstruction of large occupancy regions using standard ray tracing techniques. The GMM based occupancy reconstruction framework showed favorable performance compared to both 2D and 3D occupancy grid mapping in terms of memory complexity and accuracy. Furthermore, results also demonstrated increased performance over continuous mapping techniques - Hilbert Maps in 2D and Bayesian generalized kernel inference occupancy mapping in 3D.

In Chapter 4, the GMM occupancy modeling framework was applied to multi-robot exploration as a low-bandwidth back end. In order to enable operation in large scale

environments, a sliding voxel map representation was presented. Efficient updates were achieved through selection of the relevant Gaussian components and bounding box based ray trimming to enable partial map updates. The mapping back end was evaluated on a simulated three robot team exploring a large 3D warehouse environment. Results demonstrate that GMM occupancy mapping achieves a comparable entropy reduction rate to standard occupancy grid mapping but with an order of magnitude reduction in cumulative shared information.

5.2 Future Work

Future work will focus on increased incorporation of the sensor model into the GMM learning framework to achieve increased fidelity in the generated model. Through appropriate reasoning about the observation model, it may be possible to explicitly optimize for near-planar Gaussians. Through such a strategy, it would become possible to extend to 3D the projection based integral approximation detailed in Section 3.2.2. Additionally, improved approximation strategies for the 3D free space integral will be explored along with development of highly parallel evaluation on a GPU. Through this line of work, it may become possible to achieve efficiency beyond that of the Monte Carlo approach.

Appendix 3D Voxel Grids - Angular Integration

Following a similar approach to the 2D case, we arrive at an integral over the angular coordinates θ and ϕ. Applying the standard transformation to spherical coordinates yields

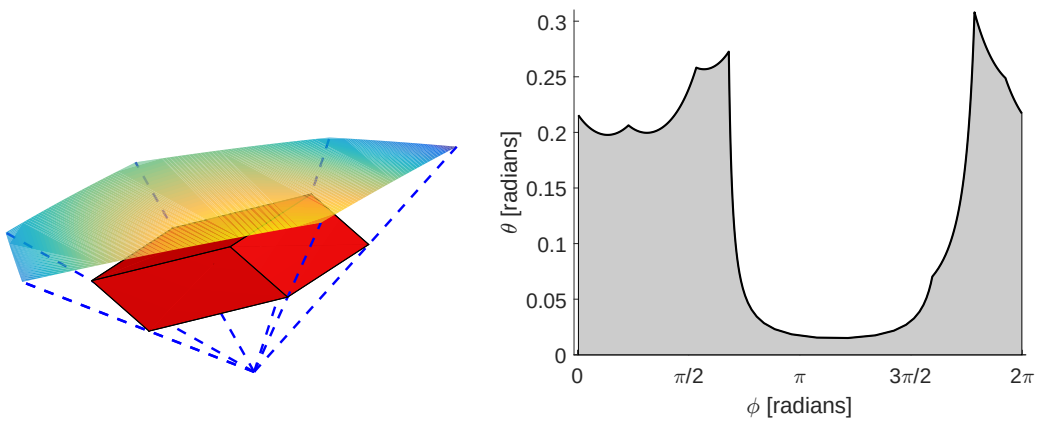
$$p(r,\theta,\phi) = \frac{1}{(2\pi)^{3/2}} e^{-\frac{1}{2}\left\| \begin{bmatrix} r\cos\phi\sin\theta \\ r\sin\phi\sin\theta \\ r\cos\theta \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ p_z \end{bmatrix} \right\|_2^2} \tag{5.1}$$

$$= \frac{1}{(2\pi)^{3/2}} e^{-\frac{1}{2}p_z^2\sin^2\theta} e^{-\frac{1}{2}(r-p_z\cos\theta)^2} \tag{5.2}$$

The volume integral can be specified in spherical coordinates as

$$Evidence(\boldsymbol{x}) = \int_V p(r,\theta,\phi) r^2 \sin\theta \mathrm{d}r\mathrm{d}\theta\mathrm{d}\phi \tag{5.3}$$

Integrating out the radial dimension (from r', the distance to the closest point of interest) allows the angular distribution to be specified in terms of the complementary error function



(a) (b)

Figure 5.1: Spherical polygon describing the angular region of integration projected onto a sphere. (b) The corresponding angular region of interaction is significantly more complex.

erfc(x). For brevity, we denote $s(\theta) = \sin\theta$ and $c(\theta) = \cos\theta$.

$$
\begin{aligned}
E(\boldsymbol{x}) &= \frac{1}{(2\pi)^{\frac{3}{2}}} \int_{\phi,\theta} e^{-\frac{1}{2}p_z^2 s^2(\theta)} s(\theta) \int_{r'}^{\infty} r^2 e^{-\frac{1}{2}(r - p_z c(\theta))^2} \mathrm{d}r\mathrm{d}\theta\mathrm{d}\phi \\
&= \frac{1}{(2\pi)^{\frac{3}{2}}} \int_{\phi,\theta} e^{\frac{1}{2}\mu_z^2 s^2(\theta)} s(\theta) \Bigg[e^{-\frac{1}{2}(r' - p_z c(\theta))^2} (r' + p_z c(\theta)) \\
&\qquad + \sqrt{\frac{\pi}{2}} \left(1 + p_z^2 c^2(\theta)\right) \mathrm{erfc}\left(\frac{r' - p_z s(\theta)}{\sqrt{2}} \right) \Bigg] \mathrm{d}\theta\mathrm{d}\phi
\end{aligned}
$$

When considering a the evidence for free space in a voxel, the domain of integration for ϕ, θ is a spherical (or geodesic) polygon defined by the shadow cast by V as viewed from the sensor (see Fig. 5.1). However, it is difficult to integrate over this region in closed form, which necessitates resorting to Monte Carlo estimation or the polygon projection technique described in Section 3.2.2.

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. 2018.

[2] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[3] N. Balakrishnan and Chin-Diew Lai. *Continuous Bivariate Distributions*. Springer, New York, 2009.

[4] Tyler Bell, Jan P Allebach, and Song Zhang. Holostream: High-accuracy, high-speed 3d range video encoding and streaming across standard wireless networks. *Electronic Imaging*, 2018(18):1–6, 2018.

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007.

[6] Arnaud Bletterer, Frédéric Payan, Marc Antonini, and Anis Meftah. Point cloud compression using depth maps. *Electronic Imaging*, 2016(21):3DIPM–396, 2016.

[7] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, volume 6, 2015.

[8] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4791–4798. IEEE, 2015.

[9] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3d object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[11] Aditya Dhawale, Xuning Yang, and Nathan Michael. Reactive collision avoidance using real-time local gaussian mixture model maps. page to appear, 2018.

[12] AR Didonato, MP Jarnagin, Jr, and RK Hageman. Computation of the integral of the bivariate normal distribution over convex polygons. *SIAM Journal on Scientific and Statistical Computing*, 1(2):179–186, 1980.

[13] Kevin Doherty, Jinkun Wang, and Brendan Englot. Bayesian generalized kernel inference for occupancy map prediction. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3118–3124. IEEE, 2017.

[14] Benjamin Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Accelerated generative models for 3d point cloud data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5497–5505, 2016.

[15] Erik Einhorn and Horst-Michael Gross. Generic ndt mapping in dynamic environ-

ments and its application for lifelong slam. *Robotics and Autonomous Systems*, 69: 28–39, 2015.

[16] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, (6):46–57, 1989.

[17] 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.

[18] Reshad Hosseini and Suvrit Sra. Matrix manifold optimization for gaussian mixtures. In *Advances in Neural Information Processing Systems*, pages 910–918, 2015.

[19] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003. URL `http://radish.sourceforge.net/`.

[20] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Warped gaussian processes occupancy mapping with uncertain inputs. *IEEE Robotics and Automation Letters*, 2(2):680–687, 2017.

[21] Soheil Kolouri, Gustavo K Rohde, and Heiko Hoffmann. Sliced wasserstein distance for learning gaussian mixture models. *arXiv preprint arXiv:1711.05376*, 2017.

[22] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[23] Martin Magnusson. The Three-Dimensional Normal-Distributions Transform an Efficient Representation for Registration, Surface Analysis, and Loop Detection. 2013. URL `https://pdfs.semanticscholar.org/120a/f9c331f9159229d0246b1ebb5ae2592cdf6a.pdf`.

[24] Simon T OCallaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[25] Cormac OMeadhra, Wennie Tabib, and Nathan Michael. Variable resolution occupancy mapping using gaussian mixture models. Submitted for review to the IEEE Robotics and Automation Letters (RA-L), 2018.

[26] Ali Osman Ulusoy, Michael J Black, and Andreas Geiger. Patches, planes and probabilities: A non-local prior for volumetric 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3280–3289, 2016.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Jose C Principe, Dongxin Xu, and John Fisher. Information theoretic learning. *Unsupervised adaptive filtering*, 1:265–319, 2000.

[29] Fabio Ramos and Lionel Ott. Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35 (14):1717–1730, 2016.

[30] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

[31] George Rigas, Christophoros Nikou, Yorgos Goletsis, and Dimitrios I Fotiadis. Hierarchical similarity transformations between gaussian mixtures. *IEEE transactions on neural networks and learning systems*, 24(11):1824–1835, 2013.

[32] 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 *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2233–2238. IEEE, 2013.

[33] Gustavo Sandri, Ricado L de Queiroz, and Philip A Chou. Comments on" compression of 3d point clouds using a region-adaptive hierarchical transform". *arXiv preprint arXiv:1805.09146*, 2018.

[34] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *Spbg*, 6:111–120, 2006.

[35] Ramesh Sridharan. Gaussian mixture models and the em algorithm. *Avilable in: http://people.csail.mit.edu/rameshvs/content/gmm-em.pdf*, 2014.

[36] Patrick Stotko, Stefan Krumpen, Matthias B Hullin, Michael Weinmann, and Reinhard Klein. Slamcast: Large-scale, real-time 3d reconstruction and streaming for immersive multi-client live telepresence. *arXiv preprint arXiv:1805.03709*, 2018.

[37] Liye Sun, Teresa Vidal-Calleja, and Jaime Valls Miro. Coupling conditionally independent submaps for large-scale 2.5 d mapping with gaussian markov random fields. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3131–3137. IEEE, 2017.

[38] Wennie Tabib, Cormac OMeadhra, and Nathan Michael. On-manifold gmm registration. *Robotics and Automation Letters*, page to appear, 2018.