LEARNING-AIDED 3D OCCUPANCY MAPPING FOR MOBILE ROBOTS

by

Kevin Doherty

SENIOR THESIS, B.E.

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Kevin Doherty, Candidate

ADVISORY COMMITTEE

Brendan Englot, Advisor Date

Philippos Mordohai, Reader Date

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2017

LEARNING-AIDED 3D OCCUPANCY MAPPING FOR MOBILE ROBOTS

ABSTRACT

We consider the problem of mapping an *a priori* unknown environment using a robot equipped with sparse and noisy range sensors, such as in the case of a robot navigating underwater with sonar. We present and examine novel methods leveraging recent advances in machine learning to achieve real-time, accurate mapping.

While learning-based methods have achieved success in mapping two-dimensional environments, extension to three-dimensional maps presents significant computational challenges. We propose an incremental version of Hilbert maps—a mapping framework using a logistic regression classifier trained on range sensor data—where we use multiple classifiers trained on local range data that can be merged to produce accurate maps. We also present a mapping method making use of Bayesian kernel inference with sparse kernels to demonstrate real-time mapping with accuracy comparable to alternative state-of-the-art Gaussian process regression-based methods. The model used for inference has the desirable property that it admits exact incremental updates. Finally, we have released the Learning-aided 3D Mapping Library (LA3DM) containing implementations of our Bayesian generalized kernel inference-based mapping algorithm and others at `https://github.com/RobustFieldAutonomyLab/la3dm`.

Author: Kevin Doherty

Advisor: Brendan Englot

Date: May 4, 2017

Department: Electrical Engineering

Degree: Bachelor of Engineering

Acknowledgments

I would like to thank Professor Brendan Englot, whose guidance and encouragement over the past few years made all of this work possible, and who gave me the opportunity to dive head first into robotics research, supporting me along the way. I would also like to thank my thesis reader, Professor Philippos Mordohai, whose wisdom and advice have been invaluable to me. Both Professors Englot and Mordohai have served as incredible mentors on the path to becoming a robotics researcher and as exemplary educators, and I am truly fortunate to have been able to learn from them.

I owe a great deal of thanks to all of the folks at the Robust Field Autonomy Lab, who have treated me like another graduate student throughout my time here, and always made themselves available to help me out. I would especially like to thank Jinkun Wang and Shi Bai, who have at various points fielded endless streams of questions from me about robotics and machine learning.

Finally, I would like to thank my friends for making Stevens my home for the last four years, as well as my mom, my dad, Karyn, Matt, and Ashley; without all of their love and support, none of this would be possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

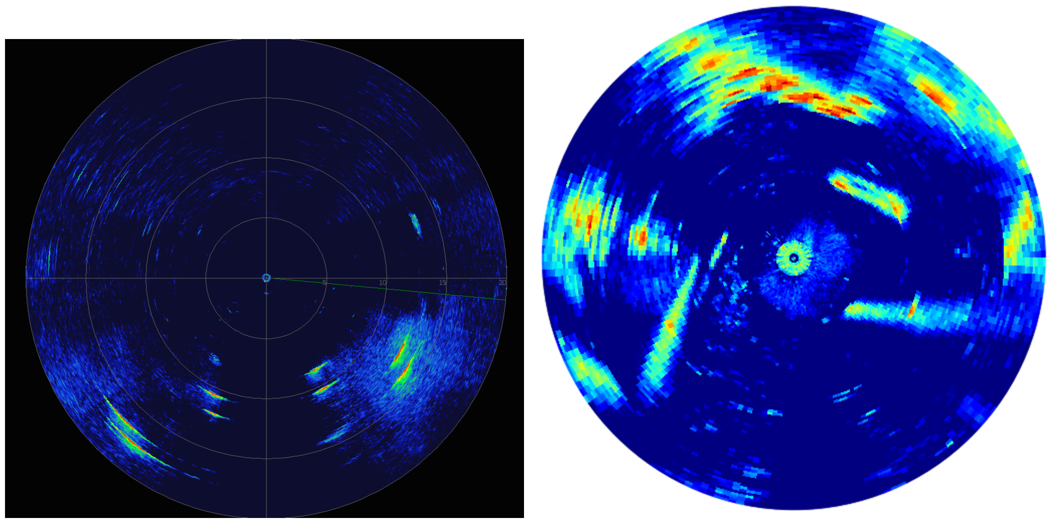
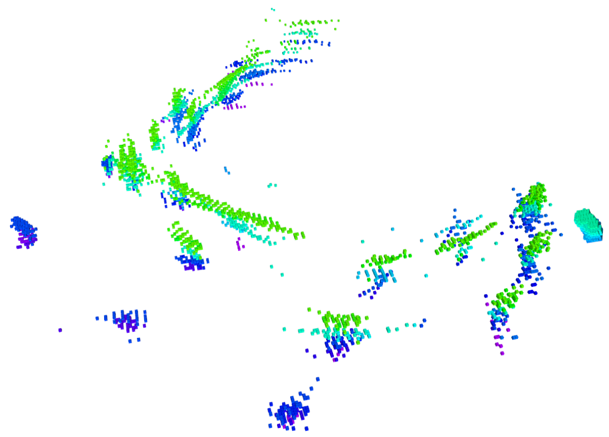


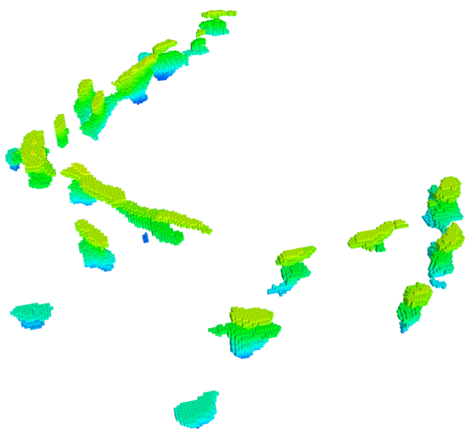
Figure 1.1: Samples of sonar data acquired from a VideoRay Pro4 ROV equipped with a Tritech Micron single-beam scanning sonar.

1.1 Motivation

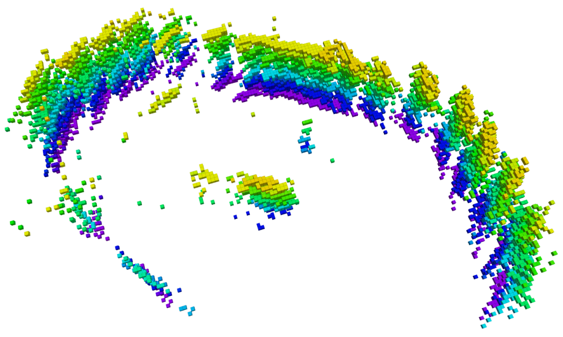
In many practical applications of robots, we require reliable planning and navigation, but available sensors provide sparse data that is often corrupted by noise. This is especially the case with underwater robots, where currents impose a challenging dynamic and stochastic navigation problem, but sensors, such as single-beam scanning sonars, may provide multiple returns suggesting different distances to obstacles and have a large beam-width, corresponding to low angular resolution, and therefore high uncertainty in the relative location of an obstacle. For reliable underwater planning and navigation, accurate maps are vital, but to obtain useful maps we must address the limitations of perception.



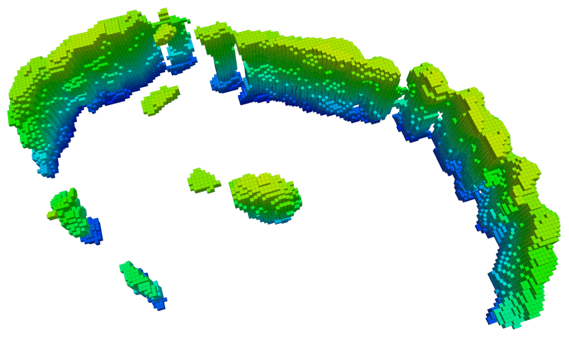
(a) Pier 84 OctoMap



(b) Pier 84 GP OctoMap



(c) USMMA OctoMap



(d) USMMA GP OctoMap

Figure 1.2: Map comparison between OctoMap, a 3D occupancy mapping framework, and GP OctoMap, an adaptation of OctoMap that uses Gaussian process regression to consider spatial correlations in the construction of the map. Colored cells represent occupied space, while empty areas represent free space, and the color of a cell varies as a function of the cell height. Figure adapted from Wang et al. [37].

Traditional occupancy grid mapping, developed by Elfes ([4], [5]), discretizes the 2D or 3D space of range data into 2D cells or 3D *voxels* respectively to form a map, then casts rays from the range sensor origin to the location of the measurement to update the probability that each cell is occupied, making the simplifying assumption that the occupancy probability of each map cell is *independent* from all others. While this assumption allows computation to be performed efficiently—explicitly avoiding consideration of correlations between every pair of grid cells of a potentially large, high-resolution grid—we lose the ability to represent existing spatial correlations. In scenarios where range sensors provide dense, high-fidelity scans, this assumption has little impact on the resulting quality of the map, but when data is limited, consideration of spatial correlations can allow us to build a denser map for planning and navigation by accurately inferring about the occupancy of voxels that have not directly intersected the beam of the sensor. It is this representation of spatial correlations that we seek to address by relaxing (or removing entirely) the independence assumption of occupancy grid mapping, and considering the mapping problem one of *learning* the map given the collected range data, making use of relationships among neighboring regions of space.

Modeling spatial correlations in occupancy maps offers several benefits over traditional occupancy grid mapping. For one, the learned model of the map *implicitly* represents the probability of occupancy over space, meaning that rather than storing occupancy probability values at every location in space we can store a potentially much smaller model instead, querying as needed. Second, and owing to the first, we can examine these spatial correlations in either a discrete framework like occupancy grid maps, or a *continuous* framework, which affords the ability to query the map at arbitrary resolution. As a motivating example, consider the sonar data displayed in Figure 1.1, collected by a VideoRay Pro4 ROV equipped with a Tritech Micron single-beam scanning sonar. Objects far away from the center, where the sensor is located, show significant distortion, and in many cases we observe stretching artifacts occurring due to the robot's change in orientation during the several seconds it takes to obtain a full 360° scan. Occupancy maps built from similar sonar data (collected at Pier 84 at Hudson River Park, New York City and the U.S. Merchant Marine Academy (USMMA) in King's Point, New York) using the OctoMap [7] framework are displayed in Figure 1.2, adapted from [37], along with the Gaussian process (GP) OctoMaps [36] built using the same data. The GP OctoMap method, a learning-aided mapping technique, provides a much denser representation that would be more useful in navigation and planning scenarios than the maps produced by OctoMap alone. These results motivate interest in learning-aided mapping techniques as a means of introducing spatial dependencies during the mapping process to build denser maps and filter noise-corrupted sensor data.

1.2 The Problem Statement

We consider the problem of mapping an *a priori* unknown environment using a sensor. We are particularly interested in scenarios where the sensor provides sparse range measurements and may be corrupted by noise. We assume that the environment we seek to map is *static* (i.e. we make the *Markov assumption*, that only the state of the robot may vary), but we do not make the assumption, as in occupancy grid mapping, that the occupancy states of all locations in space are independent. Usually we will assume that the pose of the robot is perfectly known, however we present methods for incorporating pose uncertainty (when it is known) into the resulting map estimate. Under these conditions, we seek to produce an accurate, dense map of the environment online, in real-time wherever possible.

1.3 Related Research

Some of the first contributions in learning-aided mapping were due to Thrun, who proposed learning inverse sensor models [30]—though this method did not consider dependencies between observations outside the measurement cone for a sensor—and learning in the space of all possible maps [31], which was perhaps the first effort to build a map while considering correlations between all grid cells. While the latter method fully captures dependencies between every grid cell, optimization is performed over every possible map. For finely discretized 3-dimensional grids, this method quickly becomes intractable.

In recent years there has been a growing body of literature on this topic. In 2012, Gaussian Process Occupancy Maps (GPOM) were proposed by O'Callaghan and Ramos [17]. With GPOM, a Gaussian process regression [23] is performed over sensor measurement data. The model considers correlations between all sensor measurements, and since regression is performed strictly over the measurement information, one can query the model at arbitrary resolution (i.e. the occupancy map is *continuous*). Furthermore, each prediction comes with an uncertainty estimate, which may be desirable in many planning and exploration scenarios. While GPOM enabled construction of continuous occupancy maps that considered correlations between all sensor observations, Gaussian process regression requires inversion of a kernel matrix of size $N \times N$, where N is the number of sensor observations, with a time complexity that is $\mathcal{O}(N^3)$. This inversion step makes the GPOM method difficult to implement in real-time.

Following this, Hilbert maps were proposed by Ramos and Ott [22] in 2015. Hilbert maps take advantage of regularized nonlinear logistic regression trained with stochastic gradient descent to perform fast training and inference. To introduce nonlinearity, a kernel is used, but approximated using methods like the Nyström method [39] to speed up computation. Hilbert maps perform similarly in terms of accuracy to GPOM, but by training via stochastic gradient descent, the kernel computation becomes the most computationally-expensive step (with an approximation technique this step requires a fixed, but potentially quite large, number of steps).

Kim and Kim proposed GPmap [10], a framework for Gaussian process occupancy mapping and surface reconstruction that made use of the sparse kernel from [13] for more efficient exact inference using the Gaussian process regression model, as well as the "extended block," which allows efficient retrieval of relevant training data points. Additionally, Kim and Kim, as well as Jadidi et al. [8] have proposed the use of the Bayesian committee machine for recursive updates to the Gaussian process regression model in mapping scenarios. Finally, Srivastava and Michael [29] propose learning a hierarchical mixture of Gaussians as a continuous representation of colored 3D depth data, adopting an expectation-maximization approach to learn the model and showing improvements over GPmap. Learning inverse sensor models and considering correlations among map cells as in [30] and [31], respectively, as well as GPOM and Hilbert Maps will be discussed further in Section 2.2.

1.4 Summary of Contributions

We propose two learning-aided mapping algorithms in this thesis. The first, overlapping Hilbert maps, is an approximation to the Hilbert maps method [22] where we maintain local estimators and merge their predictions, rather than maintaining a single global estimator. This method speeds up computation with limited sacrifice in terms of the accuracy of the resulting map. The second proposed method is mapping with nonparametric Bayesian kernel inference. We apply spatial interpolation directly on occupancy probabilities, formulating the problem as one of Bayesian estimation and recursively updating the Beta distributed posterior distribution (as the conjugate prior to the Bernoulli likelihood). This method affords exact recursive updates, which we show to be useful in a voxel-based approach to mapping. Additionally, the proposed method provides the prediction variance, which is not provided in the case of any of the Hilbert maps formulations. Finally, we demonstrate that the updates used in this method are advantageous when compared to the Bayesian committee machine [33] updates that are popular with Gaussian process regression-based mapping methods used in voxel grids.

1.5 Notation

Usually a range sensor measurement comprised of a distance and bearing will be denoted z, with a robot pose denoted x, and subscripts indicating the time at which these measurements and poses occurred. However, in the context of the algorithms we are considering, it will often be convenient to think about sensor measurements strictly in terms of the geometry of the resulting measurements in the global reference frame of the space we care about (for example, the coordinates of the resultant points in a 2D or 3D Euclidean space), and in this case these sensor measurements will be referred to as x (though, in these instances, measurements are considered with respect to a global reference frame, so we need not describe the pose of the robot at the time the measurements were obtained). Furthermore, m will denote a grid map with cells m_i, except in the context of kernel approximation, where we will occasionally use m to denote the number of components used to approximate a kernel. We will consistently use the notation x_* to denote a query point, which is the point in the global reference frame whose state we are asking about.

Chapter 2

Background

The following sections detail relevant preliminary content relating to traditional occupancy grid mapping, Gaussian process regression, and logistic regression and the recent use of the latter two methods in mapping.

2.1 Mapping with Known Poses

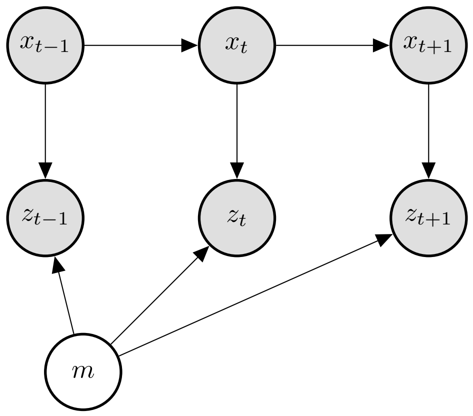


Figure 2.1: Probabilistic graphical model of mapping with known poses.

The probabilistic graphical model in Figure 2.1 succinctly describes the process of mapping with known poses. The robot at each instant in time observes its pose and takes some measurement, which is related to the current pose and the latent distribution of map occupancy. Formally, in occupancy grid mapping we are trying to estimate the posterior $p(m|x_{1:t}, z_{1:t})$, the probability of the map given all of our sensor measurements and the corresponding poses from which those measurements were taken. The classical approach by Elfes [4] to occupancy grid mapping makes the assumption that neighboring cells are independent, and estimates the map instead by estimating the occupancy probability of each cell $p(m_i|x_{1:t}, z_{1:t})$ individually.

The posterior for the map is approximated by factorization into the product of the marginals:

$$p(m|x_{1:t}, z_{1:t}) = \prod_{i=1}^{N} p(m_i|x_{1:t}, z_{1:t}), \tag{2.1}$$

which greatly simplifies the issue of estimating the map. We can now use a binary Bayes filter to estimate the probability of the state of each cell independently, and take the product over all of the cells to obtain the posterior for the map. Recursive estimation with the binary Bayes filter can be derived as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t})p(m_i|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})}, \tag{2.2}$$

which, recalling that we are making the Markov (or *static map*) assumption:

$$p(z_t|m_i, z_{1:t-1}, x_{1:t}) = p(z_t|m_i, x_t) \tag{2.3}$$

$$p(m_i|z_{1:t-1}, x_{1:t}) = p(m_i|z_{1:t-1}, x_{1:t-1}), \tag{2.4}$$

we can simplify, writing the posterior in terms of the forward sensor model:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_i, x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})}. \tag{2.5}$$

We can then apply Bayes' rule to the measurement model:

$$p(z_t|m_i, x_t) = \frac{p(m_i|z_t, x_t)p(z_t|x_t)}{p(m_i|x_t)}, \tag{2.6}$$

$$\tag{2.7}$$

acknowledging $p(m_i|x_t) = p(m_i)$, since m_i and x_t are independent:

$$p(m_i|z_{1:t}, x_{1:t}) = \frac{p(m_i|z_t, x_t)p(z_t|x_t)p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1}, x_{1:t})}, \tag{2.8}$$

and similarly write the negation as:

$$p(\neg m_i|z_{1:t}, x_{1:t}) = \frac{p(\neg m_i|z_t, x_t)p(z_t|x_t)p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t|z_{1:t-1}, x_{1:t})}. \tag{2.9}$$

Dividing (2.8) by (2.9) we can write the ratio in terms of the inverse sensor model, the previous estimate, and the prior:

$$\begin{align}
\frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} &= \frac{p(m_i|z_t, x_t)}{p(\neg m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i|z_{1:t-1}, x_{1:t-1})} \frac{p(\neg m_i)}{p(m_i)} \tag{2.10}\\
&= \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)}. \tag{2.11}
\end{align}$$

We can take the logarithm of this equation to obtain the *log-odds representation*, giving an additive update:

$$\begin{align}
l_{1:t,i} &= \log \frac{p(m_i|z_{1:t}, x_{1:t})}{p(\neg m_i|z_{1:t}, x_{1:t})} \tag{2.12}\\
&= \log \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} + \log \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} - \log \frac{p(m_i)}{1 - p(m_i)} \tag{2.13}\\
&= l_{t,i} + l_{1:t-1,i} - l_0, \tag{2.14}
\end{align}$$

where $l_{t,i}$ is obtained from our inverse sensor model (which may be a heuristic model, or obtained through learning-based methods as discussed in Section 2.2), $l_{1:t-1,i}$ is the value of the log-odds for cell i at time $t-1$, and l_0 is the log-odds representation of the prior probability assigned to m_i. The prior probability assigned to m_i is typically assumed to be 0.5. The probability of occupancy can be recovered from this representation as follows:

$$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp(l_{1:t,i})}, \tag{2.15}$$

or this can be rewritten in terms of the direct representation of probabilities in (2.11):

$$p(m_i|z_{1:t}, x_{1:t}) = \\ 1 - \left[1 + \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)}\right]^{-1}, \tag{2.16}$$

which we will find useful later, in Section 3.1.1, as a map update procedure for learning-based mapping.

From the stated equations, we can construct the algorithm for occupancy grid mapping: on each new (state, measurement) pair (x_t, z_t), traverse every cell in the grid; if the current cell is in the measurement cone of our sensor, update its log-odds representation via (2.14), otherwise leave it alone.

One issue with this model is that we do not necessarily know the inverse sensor model for a given sensor, and instead we must construct one. Heuristic methods for a range sensor are presented, for example, in [32], but may not accurately capture the characteristics of a *particular* range sensor. The more pressing issues, however, are the two fundamental assumptions in the model: the static map assumption, and the spatial independence assumption. We continue to make the static map assumption in all that follows, but we will relax the independence assumption.

2.2 Learning Occupancy Maps

Historically, learning has been explored in two contexts within the domain of occupancy grid maps. Thrun in [30] demonstrated the use of learning in a traditional occupancy grid mapping formulation by attempting to learn the inverse sensor model $p(m_i|z_t, x_t)$. This method allows us to characterize a *particular* sensor more accurately than previously used heuristic inverse sensor models, such as the one described in [32]. To learn the inverse sensor model, samples are drawn from the forward model $p(z_t|x_t, m)$ and a classifier is trained to predict the probability that m_i is occupied as a function of the measurement and pose at a point in time. This approach is straightforward, and in principle for a given type of sensor we need only train this model once, but only measurements within the measurement cone are considered.

The second way learning has been classically explored in occupancy grid mapping is in computing the full posterior $p(m|x, z)$ for the entire map, using the complete sequence of measurements and poses. Thrun [31] used an expectation-maximization (EM) approach to estimate the map in the high-dimensional space of all possible maps. The mode of the posterior, defined as the maximum of the logarithm of the posterior:

$$m^* = \underset{m}{\operatorname{argmax}} \ \log p(m|z_{1:t}, x_{1:t}), \tag{2.17}$$

is computed in practice by iteratively flipping cells in the map and keeping the change whenever it increased the likelihood of the data. This method considers the map as a whole, rather than making an independence assumption about neighboring grid cells, but requires computation over every cell in the grid, rendering it intractable for expansive, finely-discretized 3-dimensional grids.

Modern mapping methods that take advantage of machine learning are often seated somewhere between the first approach—learning the inverse sensor model—and the second approach—expectation maximization over all possible maps—but attempt to tackle a slightly different problem: that of estimating an occupancy map over a continuous space which can be queried at arbitrary resolution.

To perform this estimation tractably, we do not perform inference in the space of all possible maps (in a continuous space, this problem has infinite dimensionality). Rather, we consider the inverse sensor model problem, but instead we abstract away the sensor model itself and opt instead to consider the sensor measurement as a ray (indicating free space) which terminates when it encounters an obstacle (indicating occupied space). By extracting the 2D or 3D coordinates of free-space measurements (sampled from the ray) and occupied-space measurements (retrieved as the terminal point of the ray), we can train a model that estimates occupancy in a global reference frame over data that is continuous in nature. In these situations discretization will be a useful approximation technique, though we avoid doing EM in potentially very high-dimensional grids that would be required for a mobile robot navigating in a finely-discretized 3D map. Furthermore, it will be useful as an approximation to exercise some notion of *locality*, i.e. sensor measurements which are particularly far away from a given location are likely to have little bearing on the occupancy state of that location. The result of this is that we can adjust the region in which we consider spatial relationships, and the cell-wise independence assumption becomes a distance-based independence assumption.

2.3 Gaussian Process Occupancy Maps

Gaussian process occupancy maps (GPOM) [17], proposed by O'Callaghan and Ramos, are an attempt to compute continuous occupancy maps that consider the correlations between all of the collected data. We are primarily concerned with 3 aspects of this method: Gaussian process regression [23], the conversion of the Gaussian process regression output into a probability of occupancy, and the use of the Bayesian committee machine [33] in [11], [8], and [36] to update the Gaussian process regression model.

2.3.1 Gaussian Process Regression

Gaussian process regression can be formulated by considering the noisy observation model:

$$f(x) = x^T w, \qquad y = f(x) + \epsilon, \tag{2.18}$$

for an input vector x, where w is a vector of weights, y is an observed target value, ϵ is an additive noise term, and f is the function we aim to model. Furthermore, we assume ϵ is normally distributed with zero mean and variance σ_n^2:

$$\epsilon \sim \mathcal{N}(0,\ \sigma_n^2). \tag{2.19}$$

A Gaussian process is a distribution over functions f:

$$f(x) \sim \mathcal{GP}(m(x),\ k(x, x')), \tag{2.20}$$

with mean function $m(x)$ and covariance function $k(x, x')$). The mean function $m(x)$ is typically taken to be 0, and the covariance function, or kernel function, $k(x, x')$ is a measure of the similarity between two points x and x'. From training data $X = \{x_1, x_2, \ldots, x_N \mid x_i \in \mathbb{R}^d\}$, and corresponding observations $y = \{y_1, y_2, \ldots, y_N \mid y_i \in \mathbb{R}\}$, Gaussian process regression predicts the latent function values f_* at M query points $X_* = \{x_{*1}, x_{*2}, \ldots, x_{*M} \mid x_{*i} \in \mathbb{R}^d\}$:

$$f_* | X, y, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)), \tag{2.21}$$

where the mean and covariance of the latent function values can be computed as follows:

$$\bar{f}_* = K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} y \tag{2.22}$$

$$\text{cov}(f_*) = K(X_*, X_*) - K(X, X_*)^T (K(X, X) + \sigma_n^2 I)^{-1} K(X, X_*). \tag{2.23}$$

Here $k(X, X')$ denotes the kernel function $k(x, x')$ evaluated pairwise over the row vectors of two matrices X and X', and I is the $N \times N$ identity matrix.

One popular choice of kernel in the context of Gaussian process regression is the Matérn $\nu = 3/2$ kernel:

$$k(x, x') = \sigma_f^2 \Big(1 + \frac{\sqrt{3}\|x - x'\|}{l}\Big) \exp\Big(\frac{-\sqrt{3}\|x - x'\|}{l}\Big), \tag{2.24}$$

where σ_f^2 is a hyperparameter representing the prior signal variance and l is a hyperparameter controlling the length-scale of the kernel.

2.3.2 Gaussian Process Regression for Classification

Gaussian process regression presents a framework for computing a regression over an arbitrary nonlinear function, but the predictions need not be bounded by the observed target values. Consequently, this treatment of Gaussian process regression alone cannot compute, for example, a probability of a particular class, as in a binary classification problem. To amend this, we can train the Gaussian process regression model with $y_i \in \{-1, 1\}$ indicating the negative and positive classes, and "squash" the prediction $(\mu_{*i}, \sigma_{*i}^2)$ at the i-th query point x_{*i} with the logistic function:

$$p(y_{*i} = 1 | X, y) = \frac{1}{1 + \exp(-\gamma \omega_{*i})}, \tag{2.25}$$

where γ is a positive constant, $\omega_{*i} = \sigma_{min}^2 \mu_{*i}/\sigma_{*i}^2$, and σ_{min}^2 is the minimum predicted variance. The "squashing" function ensures that the predictions are between 0 and 1, and may be interpreted as the probability that the query point is of the positive class.

2.3.3 Bayesian Committee Machine

The Bayesian committee machine (BCM) [33] provides a means of combining Gaussian process regression models trained on different data. Suppose we have a data set comprised of individual sets of observations $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K \mid \mathcal{D}_i = \{(X_i, y_i)\}\}$, and we make the approximation:

$$p(\mathcal{D}_i|\mathcal{D}_{i-1}, f_*) \approx p(\mathcal{D}_i|f_*), \tag{2.26}$$

that is, the data sets are independent given the latent function values at the query points. With this approximation, we can recursively apply Bayes' rule to obtain an approximate predictive density:

$$\hat{p}(f_*|\mathcal{D}) \propto \frac{\prod_{i=1}^{K} p(f_*|\mathcal{D}_i)}{p(f_*)^{K-1}}. \tag{2.27}$$

When the predictive densities are normally distributed, as in the case of Gaussian process regression, we can formulate the estimated mean and covariance as follows:

$$\hat{m}(f_*|\mathcal{D}) = \widehat{\text{cov}}(f_*|\mathcal{D}) \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i)^{-1} m(f_*|\mathcal{D}_i) \tag{2.28}$$

$$\widehat{\text{cov}}(f_*|\mathcal{D})^{-1} = -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i)^{-1}, \tag{2.29}$$

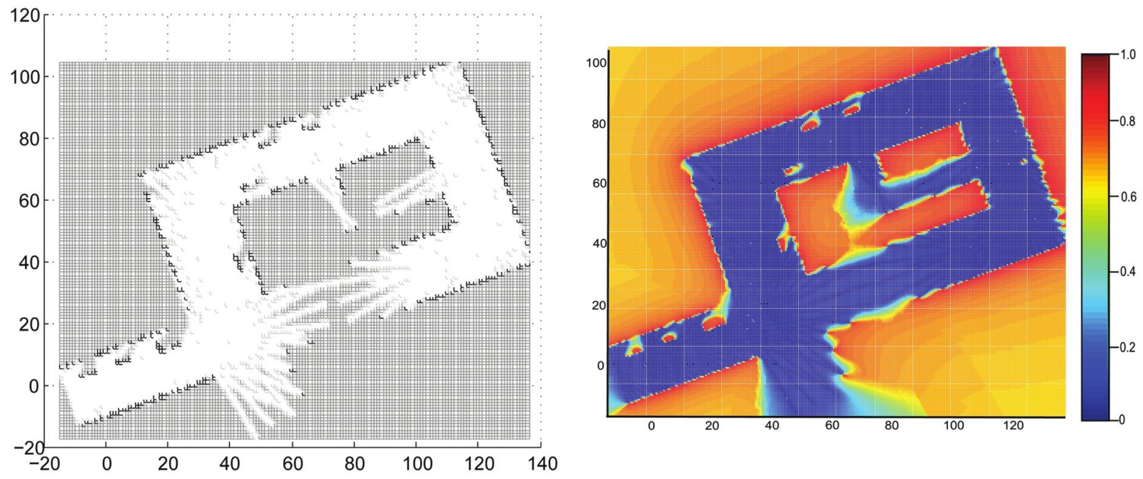


Figure 2.2: Representative demonstration of Gaussian process occupancy maps from [17]. Left: Original occupancy map; Right: Corresponding Gaussian process occupancy map. White indicates free space in the occupancy grid map, while black indicates occupied space, and grey signifies space for which the occupancy state is unknown. The Gaussian process occupancy map is colored from blue to red according to the probability of occupancy (0.0 to 1.0).

where $m(\cdot)$ and $\mathrm{cov}(\cdot)$ are the mean and covariance of the distribution over f_*, while $\hat{m}(\cdot)$ and $\widehat{\mathrm{cov}}(\cdot)$ are the BCM estimates of the mean and covariance.

2.3.4 Mapping with Gaussian Process Regression

We have now built up all of the tools that we need to perform learning-aided mapping with Gaussian process occupancy maps. The mapping problem is treated as a binary classification problem, where classes are "occupied" or "free."

It is straightforward to obtain data representing the "occupied" class—these are simply the "hit" points measured by our range sensor. To obtain "free" points, we may interpolate linearly or sample randomly along the sensor ray starting at the sensor origin and ending at the terminal "hit" point. Alternatively, we may consider for each query point only the nearest point on the sensor ray as the relevant "free" point, which is equivalent to treating the continuous line segment as a single observation. Each of these methods is not without issue, a subject that will be discussed in detail in Chapter 4.

For now, however, we will suppose we have collected a set of training observations from our range sensor $X = \{x_1, x_2, \ldots, x_N \mid x_i \in \mathbb{R}^3\}$ comprised of the 3D coordinates of our "occupied" and "free" points in some global reference frame, and corresponding labels $y = \{y_1, y_2, \ldots, y_N \mid y_i \in \{-1, 1\}\}$. For a set of arbitrary 3D query points $X_* = \{x_{*1}, x_{*2}, \ldots, x_{*M} \mid x_{*i} \in \mathbb{R}^3\}$ we can compute the probability of occupancy by first computing the predicted mean from (2.22) and covariance from (2.23) and then applying the squashing function in (2.25). When new observations are obtained we can aggregate the data collected and train a new Gaussian process regression model, or we can train a separate model and combine the estimators with the BCM, as in [11] and [36]. A representative example of Gaussian process occupancy maps from [17] is provided in Figure 2.2, where it can be observed that the Gaussian process occupancy map provides predicted occupancy probabilities for areas that are considered unknown in the corresponding occupancy grid map.

While this method enables *continuous* occupancy mapping considering spatial correlations between all sensor observations, the mean and covariance prediction equations (2.22) and (2.23) require the inversion of an $N \times N$ covariance matrix, an operation in $\mathcal{O}(N^3)$, which has motivated new methods for *real-time* online learning-aided mapping.

2.4 Hilbert Maps

Hilbert maps [22] combine several advances in machine learning to provide results comparable to Gaussian process occupancy mapping in a way that scales more suitably to the large amount of data from incoming sensors that must be processed during mapping. These advances include approximations to nonlinear kernel functions to speed up computation of kernel transformations and stochastic gradient descent for both training and updating the aforementioned classifiers in linear time.

A logistic regression [1] classifier is chosen, since its outputs naturally resemble continuous probabilities. While alternative classifiers like the Support Vector Machine (SVM) [26] can provide scores or probabilistic outputs (for example, using Platt scaling [20]), it has been demonstrated by Ramos and Ott [22] that such methods often provide results claiming high degrees of confidence in regions with relatively few training points.

The relevant components of Hilbert maps are presented in the following sections, and the adaptation of the Gaussian process occupancy mapping procedure discussed in Section 2.3.4 to Hilbert maps is straightforward: the Gaussian process regression model is replaced by logistic regression, the Matérn kernel is replaced by an approximate radial basis function kernel, and stochastic gradient descent is used to train the model, rather than a readily-available closed-form solution for the model weights.

2.4.1 Logistic Regression

In Hilbert maps, we adopt a nonlinear formulation of a logistic regression classifier in which the predicted probability of occupancy for a point x_* given a set of training points x is as follows:

$$p(y_* = 1|x_*, w) = \frac{1}{1 + \exp(w^T k(x_*, x))}, \tag{2.30}$$

where $k(\cdot)$ denotes the kernel function mapping input 3-dimensional coordinates to a high-dimensional feature space. Probability of non-occupancy for a cell $p(y_* =$

$-1|x_*, w)$ is equivalent to $1 - p(y_* = 1|x_*, w)$, since we maintain that the state of a particular cell is binary.

The major benefit of this logistic regression formulation over Gaussian process regression is that this method can be trained very quickly using Stochastic Gradient Descent (SGD) and evaluation is $\mathcal{O}(n)$ where n is the size of the test data. As a result, computation of the kernel transformation becomes the most costly part of classification.

2.4.2 Kernel Approximation

In order to represent the non-linear spatial dependency of occupancy in 3-dimensional space, we use the radial basis function (RBF) kernel. The kernel transformation is computed as:

$$k(x, x') = \exp(-\gamma \|x - x'\|^2), \tag{2.31}$$

where the parameter γ is a positive constant tuned in cross-validation. The RBF kernel maps each sample in the training data to a feature vector containing weighted pairwise distances between that sample and every other training sample. During prediction, the distance between each query point and every training point must be computed to generate a corresponding test feature vector. This pairwise distance computation becomes computationally expensive as we increase the amount of data, which presents difficulty in applying these techniques to large datasets, especially if we seek to perform real-time inference during map traversal.

For this reason, we opt to approximate the full RBF kernel. Specifically, we use the Nyström method [39] to approximate the RBF kernel with the projection of the original kernel onto a set of m inducing points $\hat{x}_{1 \ldots m}$ where we may choose the value of m. In this method we seek to find a function $\phi_{\text{Nyström}}$ such that:

$$k(x, x') \approx \phi_{\text{Nyström}}(x)^T \phi_{\text{Nyström}}(x'). \tag{2.32}$$

The resulting feature transformation of x is as follows:

$$\phi_{\text{Nystöm}}(x) = D^{-1/2} V^T (k(x, \hat{x}_1), \ldots, k(x, \hat{x}_m)), \tag{2.33}$$

where D is a diagonal matrix containing the decreasing non-negative eigenvalues of the kernel computed between all m inducing points and V is the set of its respective eigenvectors. The Nyström method was selected due to the promising results demonstrated in [22]. Kernel computation remains the most expensive step in the classification process, though this approximation offers a significant increase in speed.

2.4.3 Stochastic Gradient Descent

The general optimization problem we seek to solve in SGD is to find the weights w and bias b which minimize the regularized training error:

$$E(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha \|w\|_1, \tag{2.34}$$

where $L(y_i, f(x_i))$ is the loss function of our choosing, α is a regularization parameter, and $\|w\|_1$ is the L1-norm of the weights. For linear logistic regression, the loss function, after substitution of the hypothesis function from Equation 1 for $f(x_i)$ is as follows:

$$L(y_i, f(x_i)) = \log(\exp(-y_i(x_i^T w + b)) + 1). \tag{2.35}$$

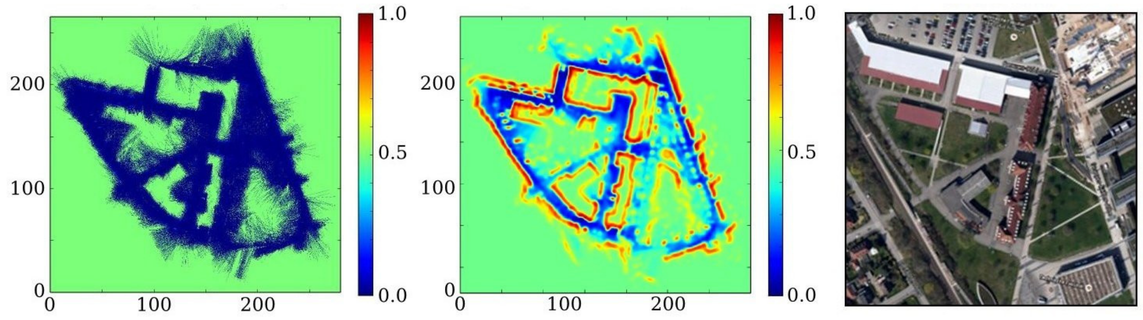


Figure 2.3: Representative image of Hilbert maps from [22]. Left to Right: Occupancy grid map, Hilbert map, aerial view of environment. Maps color varies with probability of occupancy.

To use this loss function for non-linear decision boundaries we replace x with the feature vectors from the kernel computation $k(x, x')$. We choose to regularize the L1-norm, since we are also interested in allowing feature selection to take place during learning. Ideally, features which have no significant bearing on the occupancy of a region will be eliminated, and the sparsity of the resulting weights will speed up evaluation of the model during testing.

2.4.4 Mapping with Hilbert Maps

The techniques applied in Hilbert maps have been shown to provide substantial improvements in computation time without sacrificing accuracy as compared to GPOM [22]. A representative demonstration of Hilbert maps from [22] is included in Figure 2.3. Furthermore, stochastic gradient descent provides us with a means of updating our classifier online: given a new set of data, simply perform a few more iterations of gradient descent with the recently acquired data. The kernel approximation method is quite useful in 2D, where very few inducing points are needed to accurately approximate the full kernel. In 3D, however, the number of components necessary to approximate this kernel grows substantially, making real-time computation in this framework difficult [2].

For this reason, we have proposed an approximation to Hilbert maps which we call *overlapping* Hilbert maps, where we train a separate logistic regression classifier for each new dataset collected and fuse the results. The smaller spatial extent and reduced complexity of the geometry in a single scan allows us to approximate the kernel with fewer components while achieving comparable results.

Chapter 3

Overlapping Hilbert Maps

Overlapping Hilbert maps [2] is a novel formulation of Hilbert maps in which we construct a global occupancy map by fusing local Hilbert maps. Rather than maintaining a single supervised learning model for the entire map, a new model is trained with each scan. We apply the binary Bayes filter (2.16) to merge local maps. This formulation allows Hilbert maps to be applied incrementally in real-world mapping scenarios with overlap between sensor observations. We evaluate the method on simulated and real 3D range data; these results represent the first 3D application of Hilbert maps.

While Hilbert maps offers computational advantages over Gaussian process occupancy maps, the logistic regression model used in Hilbert maps does not provide the covariance information necessary to implement fusion using a Bayesian committee machine (BCM) [33], in contrast to Gaussian process regression, where the BCM has aided online incremental map fusion [8], [11]. The original Hilbert maps formulation requires that we maintain and update a single global classifier, however it is desirable to perform scalable, incremental updates to a map as new data is gathered in real-time. Our proposed formulation of Hilbert maps allows for the use of multiple estimators by enabling local map fusion between overlapping scans.

In Section 3.1 we present the map update procedure for overlapping Hilbert maps, and in Section 3.2 we present computational results where we quantitatively compare our method to both OctoMap [7] and global Hilbert mapping on two simulated datasets and demonstrate the inference capability of the method over real data from the University of Freiburg [34].

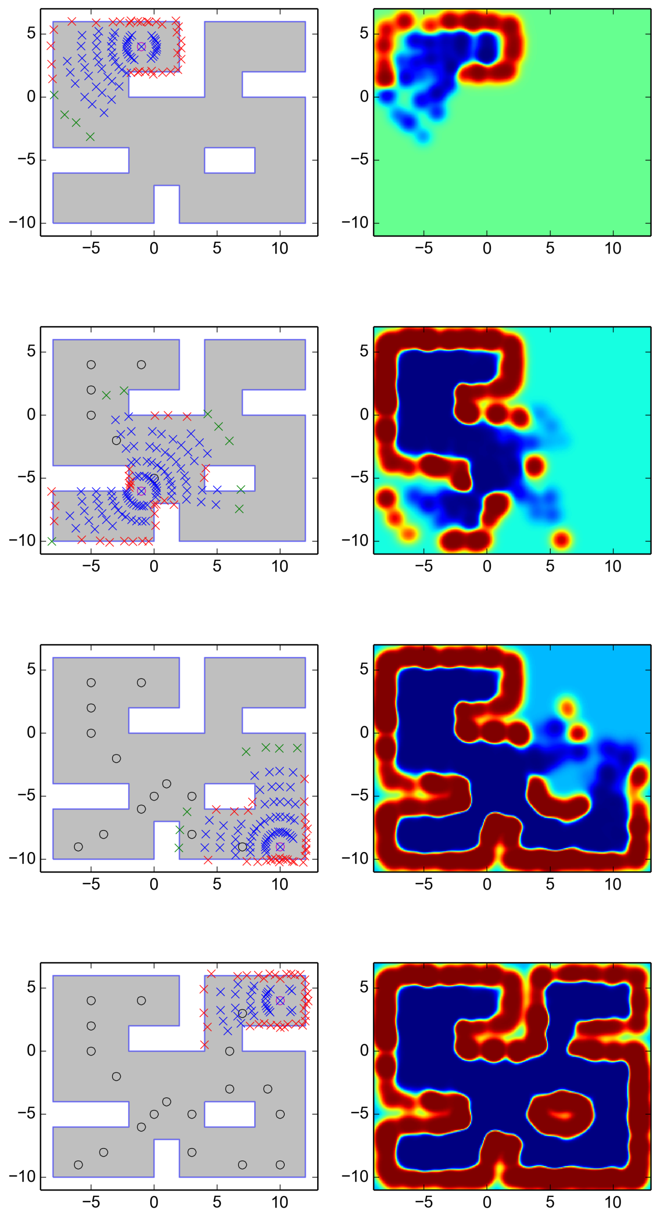


Figure 3.1: A representative 2D example of overlapping Hilbert maps in a simulated environment. Scans 1, 7, 14, and 20 of 20 are shown, with predictions performed at a 5 cm resolution. A magenta circle marks the robot's current location and black circles represent the robot's previous locations. Red markers denote hit points, blue markers denote free points, and green markers denote points assumed free where the simulated laser reached its maximum range without encountering any obstacles.

3.1 Probabilistic Local Map Fusion

Previously there have been several general approaches to constructing a global map from local maps. One approach is to develop a single classifier trained on accumulated data from across the map and predict values in postprocessing of the entire map. This is characteristic of the original methods of Gaussian process occupancy mapping. While this allows the classifier to choose informative features from the entire map, the resulting correlated occupancy map cannot be used during local planning or exploration.

Kim and Kim [11] have demonstrated overlapping map fusion for local Gaussian processes with the Bayesian committee machine (BCM) [33]. Such a method is useful in the case of Gaussian processes, where the outputs of the estimator have both predicted mean and variance information. This variance information provides insight into the uncertainty of the estimator, which can be used to make informed decisions when updating a grid cell with more than one prediction. The logistic regression estimator we use, however, does not provide variance information for its predictions.

In [22] a method has been presented to incrementally update a single global estimator given new training data. As in Gaussian process occupancy mapping, this allows a map to be produced at arbitrary resolution which very effectively captures the spatial correlation among all cells. We seek, however, to apply the methods from Hilbert maps in a way that allows us to use the inference of estimators to incrementally update maps in real-time, so they may be used for fast decision-making in the course of exploring unknown environments, as with Gaussian processes in [8].

Rather than maintaining a single classifier which we train online as the robot traverses the map, we opt instead to train a new logistic regression classifier at every new scan. This allows for real-time inference during exploration. In order for this

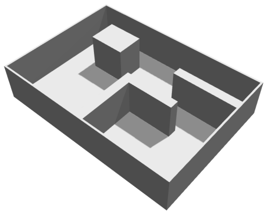
Algorithm 1 $updateMap(\mathcal{X}, \mathcal{Y})$

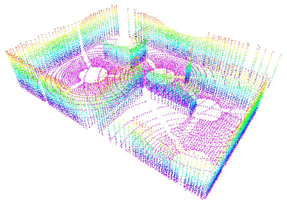
```
1: $K \leftarrow fitApproxRBF(\mathcal{X}, \mathcal{X})$;
2: $model \leftarrow train(K, \mathcal{Y})$;
3: $\mathcal{M} \leftarrow$ all cells $\in Map$ within bounding box of $\mathcal{X}$;
4: $\mathcal{X}_* \leftarrow m_i \in \mathcal{M}$ if $m_i$ in robot's perceptual field;
5: $query \leftarrow K.tranformApproxRBF(\mathcal{X}_*)$;
6: $\hat{\mathcal{Y}} \leftarrow model.predict(query)$;
7: for $\hat{y}_i \in \hat{\mathcal{Y}}$ do
8:     $p_i \leftarrow [1 + (1 - p_i)/p_i * (1 - \hat{y}_i)/\hat{y}_i]^{-1}$;
9: end for
```

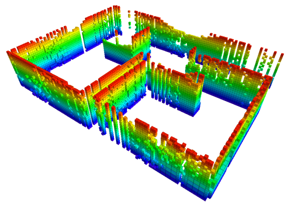
method to be effective, we must handle overlapping estimator predictions. There are several consequences of this formulation of the method, one of which is the loss of the multi-resolution property of Hilbert maps. In our implementation, a resolution must be chosen *a priori* in order to update the map cells, but we discuss in Section 3.3 a formulation of the method which does not require *a priori* discretization of space. Using one incrementally updated classifier a complete map of arbitrary resolution could be constructed *a posteriori*. However, without querying the classifiers during the exploration process, decisions regarding path-planning and navigation must be made using only the sparse sensor data. By applying the Reproducing Kernel Hilbert Spaces (RKHS) method demonstrated in [22], it is also possible to maintain the robustness of Hilbert maps to noise due to pose uncertainty, extending the applicability of the methods to noisy data.

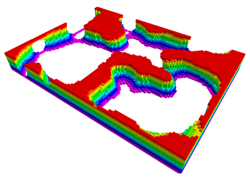
3.1.1 Map Update Algorithm

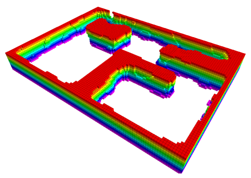
To solve the problem of merging overlapping local Hilbert maps to produce a global map, we make static map assumption, or Markov assumption stated in (2.3). Under this assumption, we consider the problem of updating the occupancy probability to be analogous to integrating the outputs of several sensors which intersect the same cell in the case of occupancy grid mapping [14] or OctoMap [7]. We generalize the sensor fusion update rule for all overlapping estimator predictions by treating each estimator as a sensor in itself, where the output of the sensor for each cell is the predicted occupancy probability for the cell. That is, we take the output of the logistic regression classifier (which we will call $\hat{y}_t$) to be the probability of occupancy for cell m_i given the data at time t.


(a) The ground truth map


(b) The raw sensor data


(c) The result of OctoMap


(d) The result of global Hilbert maps ($m = 100$)


(e) The result of global Hilbert maps ($m = 1000$)

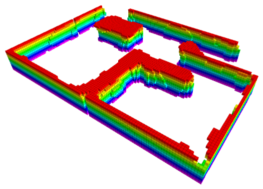

(f) The result after incrementally applying overlapping Hilbert maps

Figure 3.2: The results of 3D Hilbert maps applied to a simulated environment. Training data is provided by the raw sensor data in (b) and the model is evaluated for all cells within the perceptual field of the robot at each scan. Note that the global Hilbert map with $m = 100$ in (d) fails to effectively capture the sharp changes in occupancy probability in the structured environment, while the equivalent classifier is much more effective when updated at each new scan. We apply the update rule from Algorithm 1 to fuse overlapping local Hilbert maps and form the global map seen in (f).

With this interpretation of the classifier output, applying the update rule gives the updated occupancy probability for grid cell m_i given the current prediction $\hat{y}_t$ from the output of the logistic regression classifier as:

$$p(m_i|\hat{y}_t) = \left[1 + \frac{1 - p(m_i|\hat{y}_{1:t-1})}{p(m_i|\hat{y}_{1:t-1})} \frac{1 - p(m_i|\hat{y}_t)}{p(m_i|\hat{y}_t)} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}, \tag{3.1}$$

which is simply a rearranged form of the binary Bayes filter update in (2.16). The updated occupancy probability $p(m_i|\hat{y}_t)$ depends only on the current prediction, the previous prediction $p(m_i|\hat{y}_{1:t-1})$, and the prior occupancy probability $p(m_i)$. In practice the prior occupancy probability is assumed 0.5.

Method	AUC	Precision	Recall
OctoMap	0.92	0.320	0.097
Global Hilbert Map ($m = 100$)	0.91	0.792	0.108
Global Hilbert Map ($m = 1000$)	0.97	0.661	0.877
Overlapping Hilbert Maps	0.97	0.709	0.868

Table 3.1: Numerical comparison of the methods tested on the structured map. Precision and recall scores computed using a threshold occupancy probability of 0.7.

The sensor fusion update rule as it applies to combining estimator predictions has the property that more informative predictions have more influence on the updated occupancy probability. That is, an estimated occupancy probability close to 0.5 offers little change to the final occupancy probability, since the estimate provides very little new information. Additionally, outputs that conflict (i.e. a low occupancy probability followed by a high occupancy probability) cause the update to tend toward 0.5, so that the lack of consensus among "sensors" is reflected in the updated probability. Since there is some degree of redundancy in the data from consecutive scans, the result of this update rule is that the various classifiers trained on data from different scans may serve to improve on each others' predictions.

The complete map update algorithm is presented in Algorithm 1, including the kernel computation step, the training and querying of a logistic regression classifier,

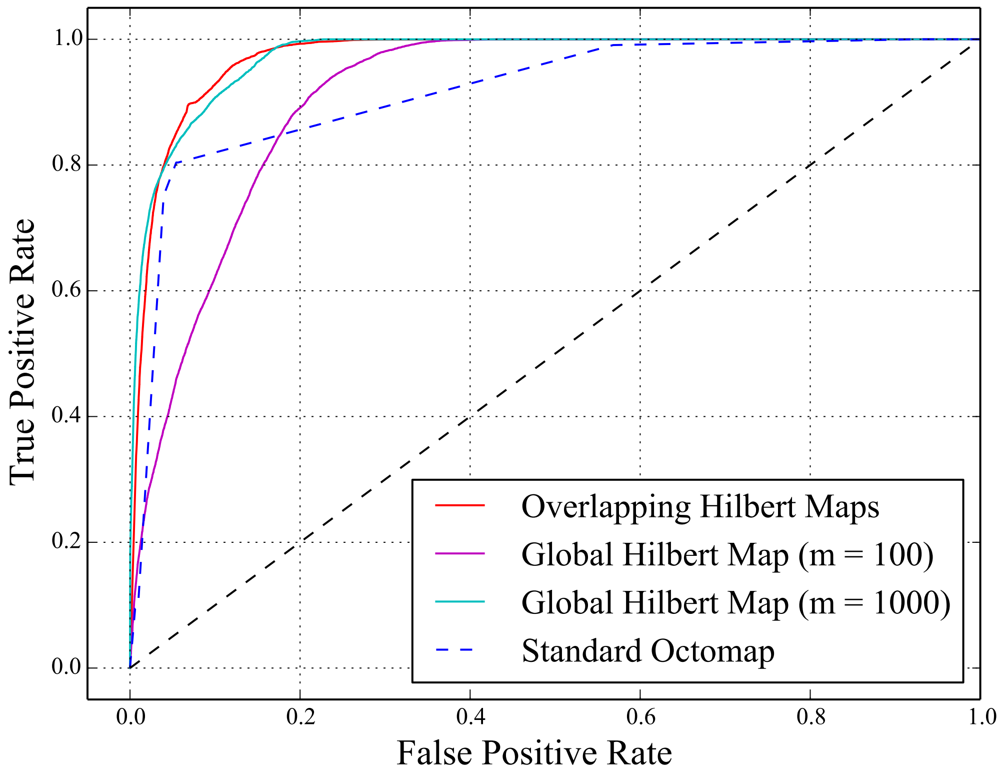
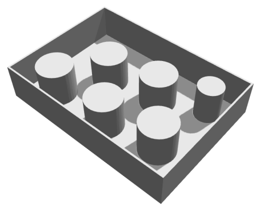
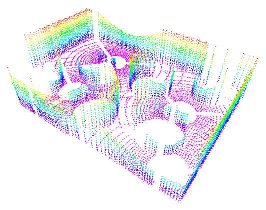


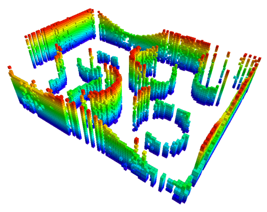
Figure 3.3: Resulting receiver operating characteristic (ROC) curves comparing incremental overlapping Hilbert maps with the same classifier trained on data across the entire map and evaluated once for each cell in the global map and standard OctoMap. These results were obtained in the structured environment depicted in Figure 3.2.

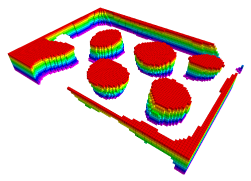
and the application of the update rule to overlapping estimator predictions. The function $fitApproxRBF(\mathcal{X}, \mathcal{X})$ computes the Nyström approximation to the RBF kernel using the provided training data. The function $transformApproxRBF(\mathcal{X}_*)$ transforms the coordinates of each cell in the set of test points $\mathcal{X}_*$ to a feature vector using the previously fit RBF kernel approximation. All $p_i \in \mathcal{P}$ denote the occupancy probabilities for the corresponding cells $m_i \in \mathcal{M}$. Map is the set of all cells of a previously selected resolution in the global map, each initialized to the prior occupancy probability 0.5. An illustrative 2D example of the incremental construction of the resulting occupancy map using Algorithm 1 is shown in Figure 3.1.

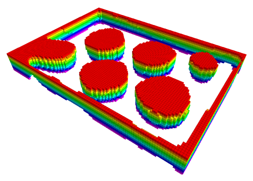
3.2 Computational Results


(a) The ground truth map


(b) The raw sensor data


(c) The result of OctoMap


(d) The result of global Hilbert maps ($m = 100$)


(e) The result of global Hilbert maps ($m = 1000$)

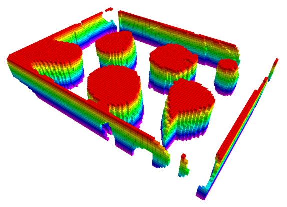

(f) The result after incrementally applying overlapping Hilbert maps

Figure 3.4: The results of 3D Hilbert mapping applied to a simulated unstructured environment. Training data is provided by the raw sensor data in (b). Standard OctoMap applied to the sensor data is shown in (c). The global Hilbert mapping model requires several minutes to compute, but generates a much more accurate, complete map. In (f) we show the results of incremental overlapping Hilbert maps, updated using the local map fusion algorithm presented in Algorithm 1.

The algorithm was evaluated in two simulated environments and one real environment. The first of the two simulations provided a structured environment, representative of an indoor setting. The second simulation provides an unstructured environment, emulating navigation of a forest. We quantitatively compare the accuracy and computation time of our method, fusing several local overlapping Hilbert maps (each with 100 Nyström components), with a formulation of Hilbert maps, which we call *global* Hilbert maps, that trains and queries a single predictor after map traversal (evaluated with 100 and 1000 Nyström components) and with OctoMap. In simulation, each method was directly compared against ground truth. The beams from the simulated laser range finder were sampled at regular intervals, providing training data for the unoccupied region, as in Figure 3.1. We then provide the visual output of inference for real laser data from the University of Freiburg. Finally, we demonstrate the applicability of this method to noisy sensor data using the RKHS method. We use $\gamma = 3.0$ for the RBF kernel except where otherwise noted. The tests for both the simulated data and real data were built using the Robot Operating System (ROS) [21]. The Gazebo Simulator [12] was used to develop the quantitative tests in the simulated environment. The logistic regression classifier and kernel approximation functions were provided by the scikit-learn [18] Python machine learning library. All computation was performed on a HP EliteBook 8570w with a 2.40 GHz Intel i7 CPU. Our goals in this section are three-fold: to show that fusing local overlapping Hilbert maps provides results comparable to the equivalent global Hilbert map, to demonstrate that Hilbert mapping can be applied in realistic 3D mapping scenarios and achieve near real-time performance, and to compare the performance of these methods with standard OctoMap [7], emphasizing that both global and incremental overlapping Hilbert maps produce denser, more accurate maps when compared to ground truth, and provide useful predictions given relatively few scans compared to OctoMap.

3.2.1 Structured Simulation

The structured environment represents a 10.0 m $\times$ 7.0 m $\times$ 2.0 m indoor setting, containing mostly right-angle edges. Some artifacts of the RBF kernel approximation can be found in the result of Hilbert mapping. For example, edges which in the ground truth map contained sharp angles are rounded in the corresponding Hilbert map. Both of the simulated environments were mapped using a laser range-finder with 120° scan sectors. On each map twelve scans were processed, consisting of a total of four 360° waypoint scans. In Figure 3.2, we present the visual results after applying the OctoMap, global Hilbert mapping with 100 and 1000 Nyström components, and incremental overlapping Hilbert maps, as well as the ground truth map and raw sensor data. The sharp boundary edge is due to the use of bounding boxes around the training data for each scan to obtain the points that will be used to query the classifier. Figure 3.3 shows the receiver operating characteristic (ROC) curves for the different methods on the structured map. The comparison of the area under the ROC curve (AUC) is provided in Table 3.1, along with the precision and recall for each method with occupancy probability threshold of 0.7 (as is common in implementations of OctoMap), which shows that OctoMap tends to misclassify occupied voxels as unoccupied voxels, while the other methods more accurately predict the occupied voxels.

We found that using only 100 Nyström components for global Hilbert mapping was not enough to effectively capture the sharp changes in occupancy probability found on this map. However, the equivalent classifier (100 components) quite effectively captured the structure of the map in comparison when evaluated for each scan and updated using our local map fusion method. When we increased the number of Nyström components to 1000 we found that we could produce a slightly more accurate map compared to ground truth than incremental overlapping Hilbert maps, but with significant additional computation time required, shown in Table 3.3.

3.2.2 Unstructured Simulation

Our unstructured simulation replicates the case of navigating a forest region, dimensioned 10.0 m × 7.0 m × 2.0 m, where obstacles are numerous and do not have well-defined angular structure. Additionally, many sections of the environment are obscured during traversal, leading to more sparsity in the training set. The walls around the environment, while adding elements of structure to the simulation, demonstrate the capability of these techniques to handle sensor data from a world with inconsistent structure. The ROC curves from the methods evaluated on the unstructured map are shown in Figure 3.5. In the case of 100 Nyström components, the area under the ROC curve for fused overlapping Hilbert maps and a single global Hilbert map were comparable. If the number of components used was increased, the AUC for the global Hilbert maps method, shown in Table 3.2, increased as well. Precision and recall scores are also provided in Table 3.2, where again we observe the low recall score of OctoMap compared to the other methods. As in the case of the structured simulation, significant additional computation time was needed for the global Hilbert map in order to achieve a marginal benefit over the fused local mapping method. Increasing the number of Nyström components for the incremental overlapping Hilbert maps method did not significantly influence results, and if the pre-determined number of Nyström components was greater than the number of training samples, standard evaluation of the full RBF kernel is preferred. It is intuitive that a larger number of components would better approximate a larger training set, as in the case of a global Hilbert map, but the number of training points per scan is relatively small compared to the data from the entire map, and thus little benefit comes from increasing the number of components for the incremental overlapping Hilbert maps method; in some cases the number of incoming samples is few enough that full evaluation of the RBF kernel is possible.

3.2.3 Real Data

We performed inference for qualitative evaluation on data from the University of Freiburg corridor OctoMap dataset. The entire map measures 43.8 m$\times$18.2 m$\times$3.3 m.

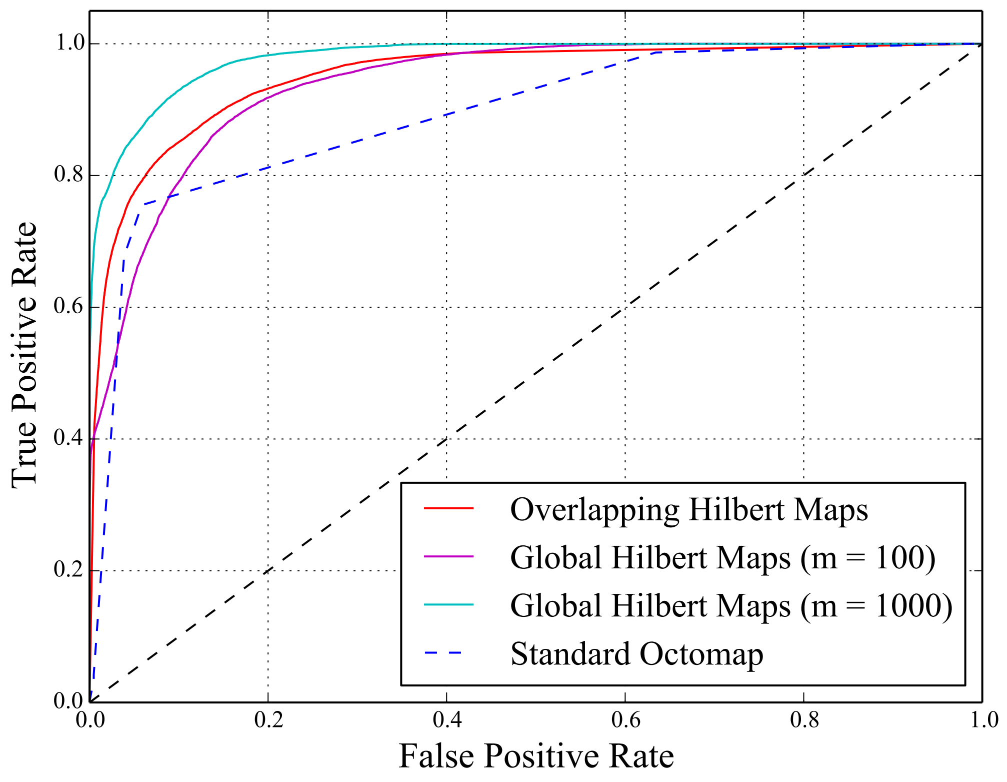


Figure 3.5: ROC curves comparing overlapping Hilbert maps with the same classifier trained on data across the entire map and evaluated once for each cell in the global map and standard OctoMap in the unstructured environment.

360° pan-tilt laser scans were sparsified to generate training sets of approximately 6% of the original data. Figure 3.6 shows a view from outside the corridor produced by incremental overlapping Hilbert maps ($m = 100$, $\gamma = 10.0$ for the kernel approximation), compared to the sparsified raw data from the range sensor. Incremental overlapping Hilbert maps produce a denser, more complete map than the map produced by the sparsified raw sensor data alone. In Figure 3.7, we show the maps from a perspective within the corridor. In both cases we see that while this method is capable of reasonable inference in a real environment, 100 components is not enough to effectively capture the sharp changes in occupancy probability in a large, structured

Method	AUC	Precision	Recall
OctoMap	0.89	0.418	0.042
Global Hilbert Map ($m = 100$)	0.94	0.999	0.248
Global Hilbert Map ($m = 1000$)	0.98	0.990	0.629
Overlapping Hilbert Maps	0.95	0.797	0.832

Table 3.2: Numerical comparison of the methods tested on the unstructured map. Precision and recall metrics computed for an occupancy threshold value of 0.7.

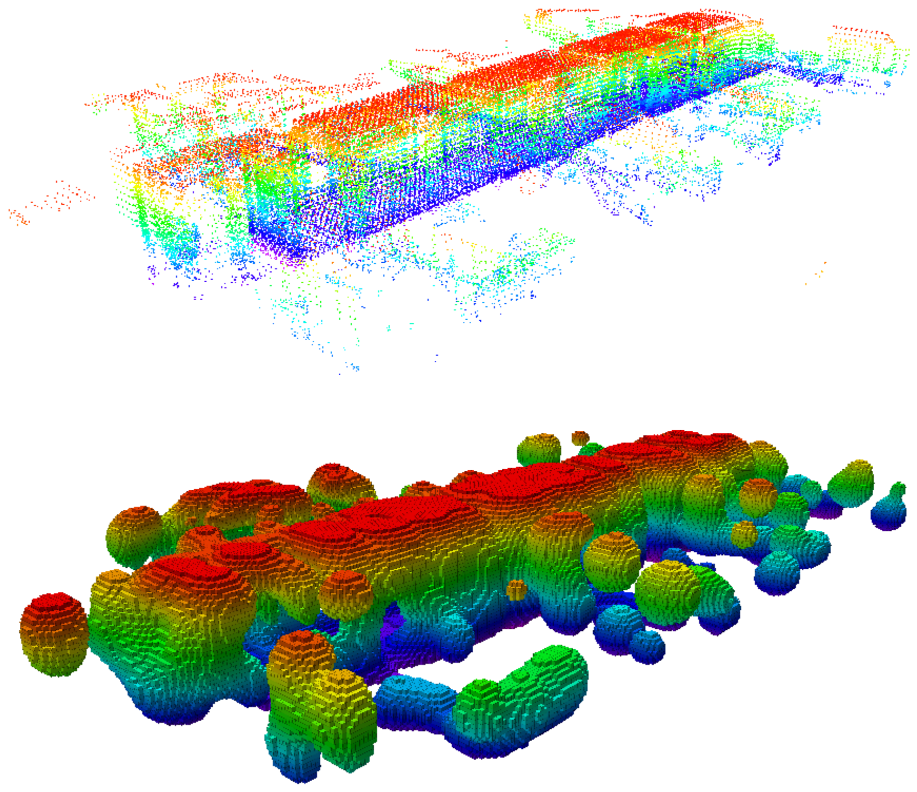


Figure 3.6: The real 3D range sensor data from the Freiburg corridor map (top) compared to the result after performing incremental overlapping Hilbert maps (bottom) viewed from outside the corridor. A threshold occupancy probability of 0.7 was used. Each map is colored by height.

Dataset	Scans	Pts/Scan	Test Pts/Scan	Method	Time/Scan	Time (s)
Structured	12	3500	29892	OHM	1.89	22.68
				GHM-100	N/A	8.70
				GHM-1000	N/A	331.64
				OctoMap	0.02	0.2
Unstructured	12	3500	29892	OHM	1.83	21.96
				GHM-100	N/A	8.52
				GHM-1000	N/A	332.82
				OctoMap	0.01	0.14
FR-079	66	4943	371170	OHM	14.6	963.6
				GHM-100	N/A	1656.95
				GHM-1000	N/A	11914.96
				OctoMap	0.1	6.7

Table 3.3: Computation times for the three maps used in testing. FR-079 is the Freiburg Corridor dataset. Comparison between overlapping Hilbert maps (OHM), global Hilbert maps (GHM) with 100 and 1000 components, and OctoMap.

environment. Similarly, when a single classifier is used, the number of components needed to accurately approximate a region also scales with the size of the region and its structural complexity, so there is a trade-off between the quality of the approximation and computational practicality in these cases. To some degree, this incremental mapping method helps to balance this trade-off, but the issue is nonetheless present when dealing with individual scans and more work here is needed.

Since these scans are much larger than the 120° sector scans used in simulation, updating the map takes much longer, as shown in Table 3.3. It should be noted however, that the time required to process each scan is dominated by the time taken to update the map after training. The mean time required only for training a new estimator using the real laser range finder data was 0.5 seconds, with the rest of the time for each scan being dominated by querying the predictor for large numbers of grid cells and updating all of queried voxels.

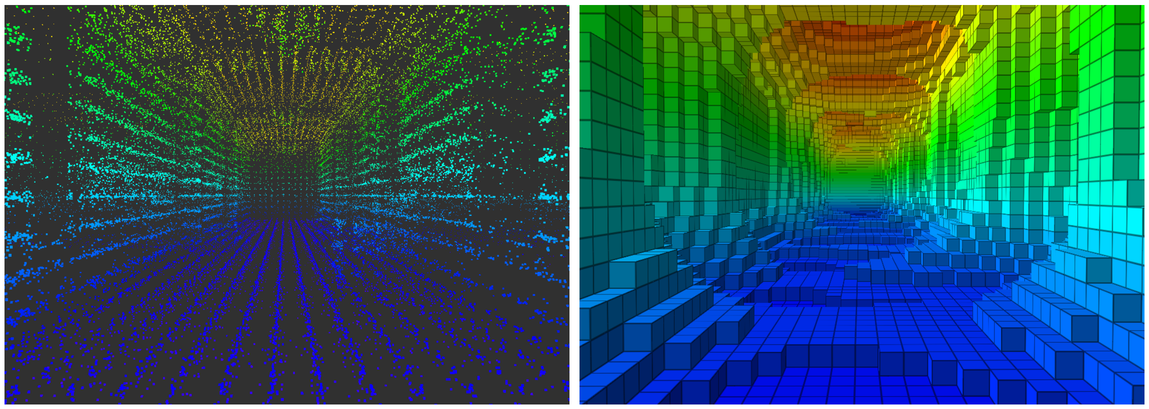


Figure 3.7: View from inside the corridor; raw data compared to the occupancy map produced using incremental overlapping Hilbert maps.

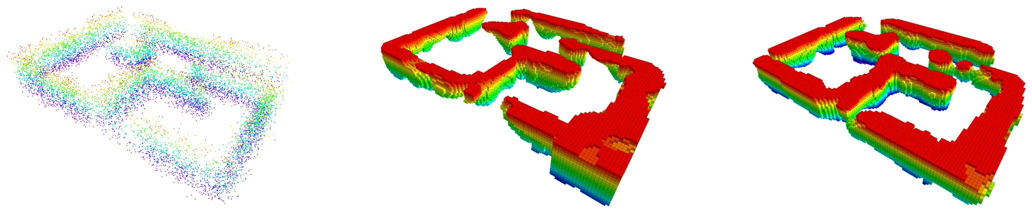


Figure 3.8: Depiction of a structured environment where the range sensor is corrupted by noise. The first image (left) shows the raw sensor data. The second (center) shows the map produced using the Nyström method, as before. The rightmost image shows the same map produced using overlapping Hilbert maps with RKHS.

3.2.4 Noise in Sensor Data

We now consider the case of sensor noise due to positional or sensor measurement uncertainty. In this demonstration, we perturb the data with Gaussian noise, $\mathcal{N}(0.0, 0.2)$ in the x- and y-direction and $\mathcal{N}(0.0, 0.02)$ in the z-direction. Figure 3.8 compares the output after traversing the map and fusing overlapping Hilbert maps with Nyström features to the output when the RKHS method is used. Using the RKHS method appears to provide a map closer to the maps produced without added noise, and shows that fusion of overlapping Hilbert maps is able to retain the robustness to noise provided by the original formulation of Hilbert maps.

3.3 Summary

We have presented a novel approximate formulation of Hilbert maps in which several classifiers are trained on local range data and fused at query time. The proposed method enables the representation of a 3D environment with fewer components required for kernel approximation, which consequently provides more computationally tractable mapping of 3D environments with Hilbert maps.

While we have been primarily concerned with a discretized form of overlapping Hilbert maps, relying on a grid of voxels to accommodate new observations, this method need not be applied in only a discrete setting. It is straightforward to adapt this approach to a continuous setting by simply storing the robot pose where each classifier was trained, and at query time retrieving some approximate set of relevant classifiers by performing a range search, which can be done efficiently with spatial data structures like k-d trees. We can then predict with each of the retrieved classifiers and fuse the results using the binary Bayes filter.

Chapter 4

Nonparametric Bayesian Inference for Occupancy Map Prediction

With overlapping Hilbert maps, we provided a means of merging several locally trained classifiers to form a global occupancy map. This representation, requiring repeated training of new classifiers, is rather cumbersome. It seems reasonable to question whether a much simpler method owing strictly to kernel computation for spatial interpolation may achieve similar results.

In this chapter, we will develop a mapping framework, referred to here as Bayesian generalized kernel occupancy maps [3], which seeks to achieve this goal, leveraging nonparametric Bayesian kernel inference to introduce local spatial dependencies. We show that this method offers several desirable properties in comparison to Gaussian process-based mapping techniques including speed of computation, the ability to be recursively updated without approximation, and consistency between batch and online inference. The method also reverts to the use of a specified prior state when insufficient relevant training data exist to predict the occupancy probability of a query point, a property which is attractive for motion planning and exploration applications with mobile robots.

Fusion of several GPs in mapping with Bayesian Committee Machines (BCMs) [33] has been found to be useful for approximating GP occupancy maps [11]. Using a divide and conquer approach, a combination of nested BCMs, the extended block feature of “GPmap” [10], and test-data octrees has enabled real-time computation of GP occupancy maps [36]. The BCM, however, is known to be an approximate update to Gaussian process regression [33]. Successive application of approximate updates may lead to unreliable prediction in long-term mapping scenarios, and can lead to poor inference in unobserved areas of the map.

For this reason, we instead opt for a simpler model which is capable of exact recursive updates. Motivated by its recent success in applications spanning safe high-speed navigation [24] to state uncertainty estimation [19], we have chosen to apply the nonparametric Bayesian generalized kernel inference method in [35] in the context of mapping. Using this method, we are able to predict the occupancy probabilities and their corresponding variances for cells of the map not directly observed by a sensor. In contrast with previous work ([10, 36]), this method explicitly reverts to a selected prior when there is insufficient training data to draw conclusions about the occupancy state of a query cell.

Models in machine learning should perform well when queried with data that is similar to training data, but may offer poor or unpredictable performance for query points which are particularly dissimilar to training data. We desire a model which will recognize when there is insufficient training data near the query point to make an accurate prediction. Rather than attempt to make increasingly inaccurate predictions as query points become far from training data, we would like the model to smoothly transition to some prior with high variance, which may consist of a prediction representing an unknown state. This recognition is desirable in a variety of scenarios, but it is particularly is attractive for avoiding situations where we may aggressively predict the contents of unoccupied space, then plan a high-speed path into areas which should instead be treated cautiously as unknown.

In summary, we provide an application of nonparametric Bayesian kernel inference to the mapping problem in order to relieve the independent cell assumption in occupancy grids, and combine this inference method with recent developments in sparse kernels and data structures for learning-aided mapping to achieve real-time viability, while retaining comparable inference accuracy to existing methods. In Section 4.1, we present related work, outlining several tradeoffs between existing map inference methods. In Section 4.2, we present our application of a recent nonparametric Bayesian local kernel inference model to the mapping problem and discuss the properties of the model and the implications of those properties with respect to the mapping application. Section 4.3 contains quantitative and qualitative evaluation of our method against previous methods.

4.1 Related Work

Several variants of Gaussian process occupancy mapping have been proposed to obtain reliable inference in unobserved regions of occupancy maps, with decreased computational cost. GPmap [10] partitions a map into "blocks" and "extended blocks", used in concert with sparse kernels to achieve offline inference with an overall complexity of $\mathcal{O}(\frac{N^3}{K^2} + \frac{N^2M}{K^2})$ where K is the number of blocks. This method was extended in [11] for online approximate updates using the BCM. By further segmenting grid blocks and making extensive use of BCM updates, the computational complexity of GP regression for occupancy mapping has been further reduced to $\mathcal{O}(\frac{N^3}{K^2E^2} + \frac{N^2M}{K^2E})$ [36] (where E represents additional training-data partitions) with computation time suitable for 3D real-time applications. This method, however, relies heavily on repeated approximate updates.

Our use of the generalized kernel inference model from [35] is motivated by its success in several other applications from safe high-speed navigation [24] to estimation of state uncertainty [19]. These applications emphasize the ability to apply prior information to the model as well as the model's predictable performance in scenarios where query data are dissimilar to training data. We are primarily interested in the model as a conservative estimator of occupancy which reverts to the prior occupancy probability for query points that lie sufficiently far from any training data.

4.2 Bayesian Generalized Kernel Occupancy Maps

In order to simplify the problem of maintaining a predictive distribution of occupancy states over potentially vast 3D maps, we make several assumptions. The proposed algorithm is intended for the case of *static* maps. In its current form, the algorithm presented is unable to support maps in which states other than that of the robot are allowed to vary. Furthermore, we assume that there is some distance l such that for all blocks, the occupancy state of a block is independent of the states of every block of distance l or further. This assumption is consistent with our choice of the sparse kernel presented in Section 4.2.2. These assumptions allow modest improvements to computation time over existing methods, but more importantly allow us to perform exact inference and updates. One important consequence of these choices is that the model is unlikely to exhibit strong predictive performance when the data is prohibitively sparse. Instead, the proposed algorithm offers a more *conservative* approach to inference-based mapping in which, in absence of sufficient training data, we revert to some prior knowledge, rather than attempting to make predictions based on limited information.

4.2.1 Bayesian Nonparametric Inference

As in the standard formulation of occupancy grid mapping, a map cell m_i is occupied with probability $p(m_i|\mathcal{D}_{1:t})$ and free with probability $p(\neg m_i|\mathcal{D}_{1:t}) = 1 - p(m_i|\mathcal{D}_{1:t})$ where $\mathcal{D}_{1:t}$ is the set of all measurements accumulated up to time t. That is, occupancy is Bernoulli distributed, with parameter $\theta = p(m_i|\mathcal{D}_{1:t})$. We seek to estimate the parameter θ of the cell centered at a query point x_*. To do so, we use the nonparametric Bayesian inference model for exponential families in [35]. With $\theta \sim Beta(\alpha, \beta)$, the predicted mean and variance of θ are as follows:

$$\mathbf{E}[\theta] = \frac{\alpha}{\alpha + \beta} \tag{4.1}$$

$$\mathbf{Var}[\theta] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}, \tag{4.2}$$

where α and β are hyperparameters. At each time step we obtain a range scan containing 3D points and their corresponding labels (i.e. occupancy probabilities) constituting the training set $(x_i, y_i) \in \mathcal{D}_t$. Locations that the sensor hits are assigned occupancy probabilities of 1.0, and free-space points are interpolated along the sensor ray and assigned occupancy probabilities of 0.0. Given this training data, we apply exact recursive updates to the hyperparameters:

$$\alpha_t = \alpha_{t-1} + \bar{y}_t \tag{4.3}$$

$$\beta_t = \beta_{t-1} + \bar{k}_t - \bar{y}_t, \tag{4.4}$$

in which $\bar{k}_t$ and $\bar{y}_t$ are the results of the following kernel computations:

$$\bar{y}_t = \sum_{i=1}^{N} k(x_i, x_*) y_i \tag{4.5}$$

$$\bar{k}_t = \sum_{i=1}^{N} k(x_i, x_*), \tag{4.6}$$

We can also solve for the log-odds representation of $\mathbf{E}[\theta]$, desirable for its numerical stability for probabilities near 0 and 1 [32], without directly computing

$\mathbf{E}[\theta]$:

$$l_t = \log \frac{\alpha_t}{\beta_t}. \tag{4.7}$$

The recursive updates in (4.3) and (4.4) rely only on the previous values of α and β and kernel computations based on the current scan. The additive nature of these updates suggests that we can generate equivalent maps offline in batch or incrementally while traversing the map. The quantities α_0 and β_0 represent prior pseudo-counts of the positive (occupied) and negative (free) classes respectively. We use α_0 and β_0 to apply a small uninformative prior over the possible values of θ, so that as the results of the kernel computation become very small, we gradually revert to the prior occupancy probability and variance. Due to the very small uninformative prior, when there is sufficient training data (i.e. $\bar{y} >> \alpha_0$ and $\bar{k} >> \alpha_0 + \beta_0$), $\mathbf{E}[\theta]$ approximates the Nadaraya-Watson estimator $\hat{m}(x_*)$ [15], [38]:

$$\hat{m}(x_*) = \frac{\sum_{i=1}^{N} k(x_i, x_*) y_i}{\sum_{i=1}^{N} k(x_i, x_*)}. \tag{4.8}$$

An extension of this method could leverage prior knowledge about the environment or sensor by making α_0 and β_0 functions of the query point x_*, as in [24].

We also take advantage of variance predictions in a similar fashion to previous work [36]. We use the following model of state for cells in the environment:

$$\text{state} = \begin{cases} \text{free}, & \text{if } p < p_{free}, \sigma^2 < \sigma_{th}^2 \\ \text{occupied}, & \text{if } p > p_{occ}, \sigma^2 < \sigma_{th}^2 \\ \text{unknown}, & \text{otherwise} \end{cases} \tag{4.9}$$

in which p corresponds to the occupancy probability, which in our case is the mean of the predictive distribution $\mathbf{E}[\theta]$, p_{free} is a threshold on the occupancy probability of cells deemed "free," and p_{occ} is a threshold on the occupancy probability of cells deemed "occupied". The variance σ^2, computed as $\mathbf{Var}[\theta]$, is thresholded by σ^2_{th} to filter out predictions with high variance as "unknown."

4.2.2 Sparse Kernel

Our choice of kernel will have important implications about the exactness of the update in (4.3) and (4.4). We opt to use the sparse kernel presented in [13],

$$k(x, x') = \begin{cases} \sigma_0 \big[\frac{2+\cos(2\pi\frac{d}{l})}{3}(1 - \frac{d}{l}) + \frac{1}{2\pi}\sin(2\pi\frac{d}{l})\big] & \text{if } d < l \\ 0 & \text{if } d \geq l \end{cases} \tag{4.10}$$

where $\sigma_0 > 0$ is a constant parameter of the kernel, $l > 0$ is the scale, and d is the distance between x and x'. By opting for a sparse kernel, we can efficiently and exactly compute $\bar{y}$ and $\bar{k}$ in $\mathcal{O}(\log N)$ time using a k-d tree. Simply by querying a k-d tree containing the training points for a scan with radius l about each query point x_*, we obtain all training points with nonzero contribution to the kernel computation in (4.10).

The overall computational complexity of the inference method is $\mathcal{O}(M \log N)$, where M is the number of test points and N is the number of training points. For other kernels, such as the radial basis function (RBF) kernel or Matérn kernel, only approximate updates can be obtained without using all of the training data. Though we use this sparse kernel, any kernel with finite support is viable, such as a polynomial approximation to the RBF kernel or the product of the sparse kernel and the Matérn kernel used in [10].

Dataset	Scans	Pts/Scan	Sampled Pts/Scan	Method	Time/Scan	Time (s)
Structured	12	3500	1506	BGK	**0.021**	**0.25**
				GPOM	0.091	1.1
				OctoMap	0.027	0.32
Unstructured	12	3500	1506	BGK	0.019	0.23
				GPOM	0.075	0.90
				OctoMap	**0.018**	**0.22**
FR-079	66	89445	7601	BGK	**0.15**	**9.6**
				GPOM	0.28	18.4
				OctoMap	0.15	10.1

Table 4.1: Computation times for the three maps used in testing. Comparison provided between Bayesian generalized kernel inference (BGK), Gaussian Process OctoMap with nested BCM (GPOM) [36], and OctoMap.

4.2.3 Test Data Octrees

We adopt the test-data octrees proposed in [36] with slight adaptations to the use of the extended block. The method proposed suggests training several separate GP regressions for a group of query points. With the kernel inference method used, we need not explicitly train at all. When a ray is cast, we sample free-space points linearly along the ray at a fixed resolution, then aggregate all "free" and "hit" point data from the extended block (i.e. all blocks within distance l from a center block) to update the predictions at the query points in the center block. Figure 4.1 shows a 2D illustration of the algorithm, as well as a depiction of occupancy grid mapping for comparison. By considering all blocks within distance l of the query block as the extended block, we are able to quickly retrieve all points with nonzero contribution to the occupancy probability of the cells in the query block which maintains the exactness of the inference.

Test-data octrees enable dynamic allocation as a robot explores, avoiding the need for large, finely-discretized grids to be initialized. Test-data octrees are pruned

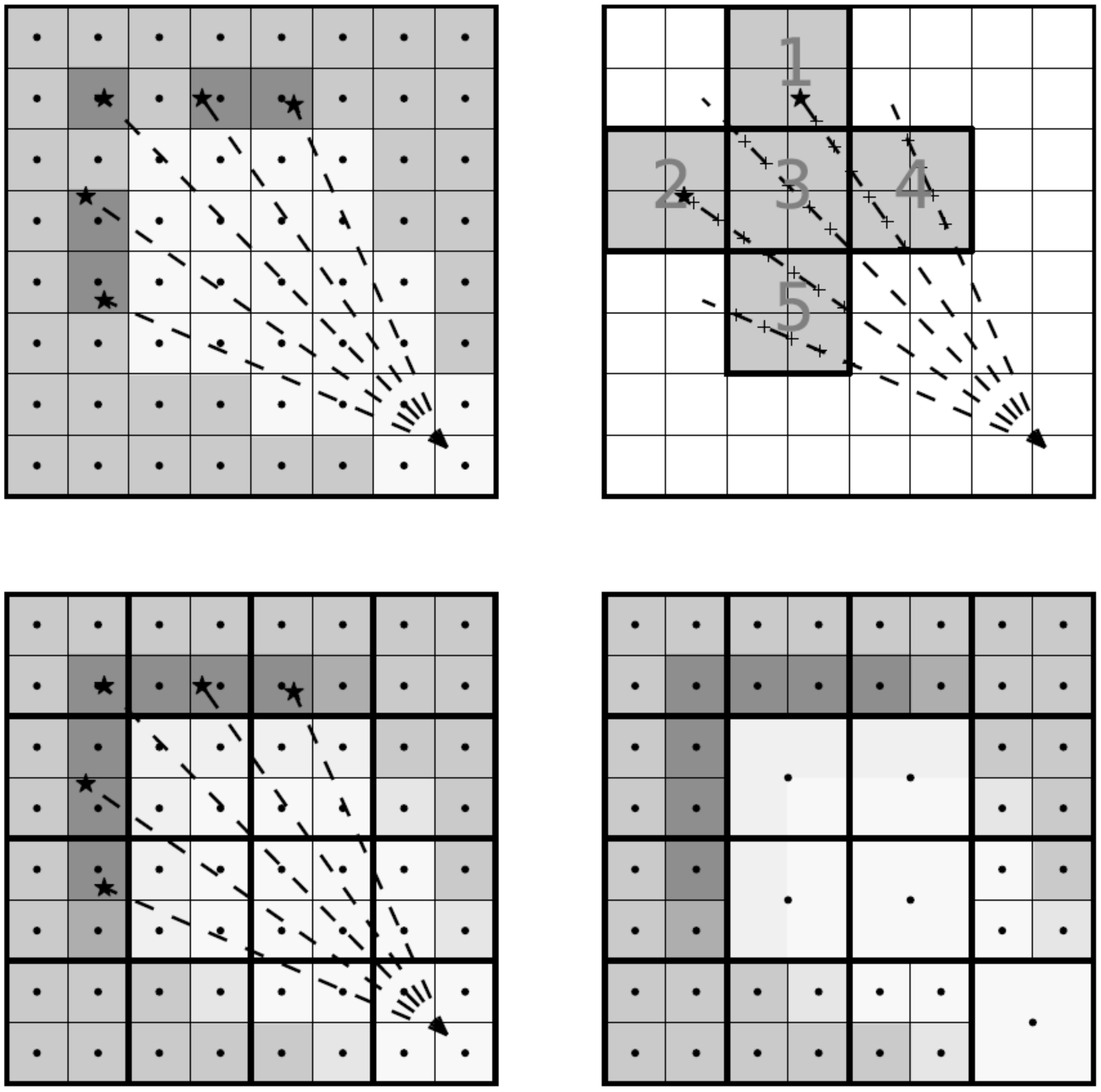


Figure 4.1: A 2D illustration of the use of test-data octrees. The top-left image depicts standard occupancy grid mapping. In the top-right we show the setup for prediction of the occupancy probability of all cells in block 3. The extended block consists of all blocks within distance l, in this case the length of two grid cells, of block 3 that contain sensor data or sampled free-space points. For each block, the data from the corresponding extended block is aggregated and inference is performed, generating the image at bottom-left. Finally, at bottom-right, neighboring cells within a block with the same occupancy state are pruned. Obtained from Wang and Englot [36].

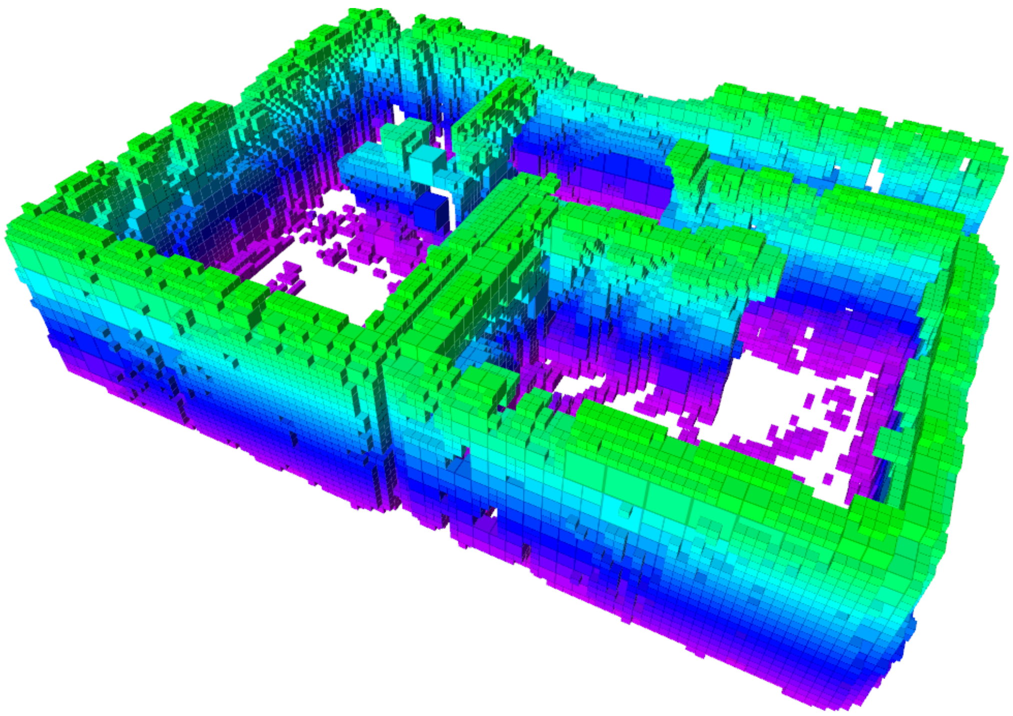


Figure 4.2: Demonstration of pruning test data octrees in a map inferred using Bayesian generalized kernel inference. The octrees have depth three, and pruned blocks are especially prevalent along the nearest wall, where high-resolution cells have been merged into lower-resolution cells. Figure 4.4 shows maps of the same environment without pruning. The 3D map is colored by height.

to lower resolution when all of the children of a particular node achieve the same state [36]. In such a situation, all of the children are removed, and the parent remains. This allows us to refine the number of query points needed for areas of space that are likely to be highly correlated, further decreasing the time needed for computation. Pruning also has the benefit of reducing the map's memory consumption. We show the effects of pruning in 3D on a map generated from simulated data in Figure 4.2.

4.3 Computational Results

We evaluated the inference method on two synthetic datasets representing "structured" and "unstructured" environments, as well as the corridor dataset from the Uni-

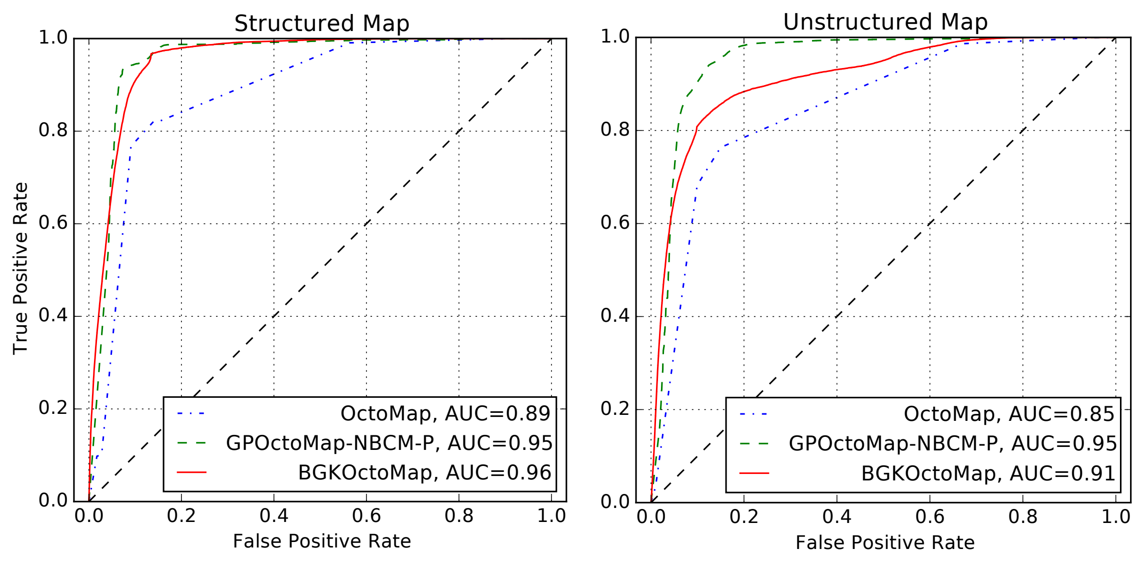
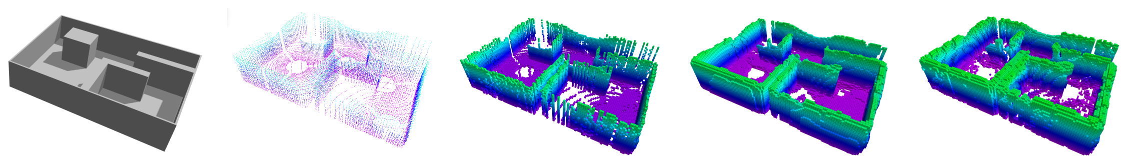


Figure 4.3: Receiver operating characteristic curves for the 3 evaluated methods on the Structured and Unstructured maps.

versity of Freiburg [34]. The mapping algorithm is our own C++ implementation[1], and we compare our method quantitatively to GPOctoMap with test-data octrees [36]. We also compare our method to OctoMap [7], which provides an efficient multi-resolution occupancy grid. We use the Robot Operating System (ROS) [21] as well as the Point Cloud Library (PCL) [25] in our tests. The synthetic examples were made using the Gazebo simulator [12]. We apply the parameters $\alpha_0 = \beta_0 = 0.001$ enforce a weak uninformative prior on grid cells, and the remaining parameters $\sigma_0 = 10.0$ and $l = 0.3$ were hand-tuned on one synthetic dataset and applied consistently throughout. Generally the desired locality of inference will depend on the resolution of the range sensor being used. Free space samples are taken linearly along each ray at 0.5m resolution. All computations were performed on an HP EliteBook 8570w with a 2.40 GHz Intel i7 CPU.

In this section, we refer to the GPOctoMap implementation with nested BCM

[1]The code for this section is implemented in LA3DM, a C++ library for learning-aided 3D mapping, available at `https://github.com/RobustFieldAutonomyLab/la3dm`.



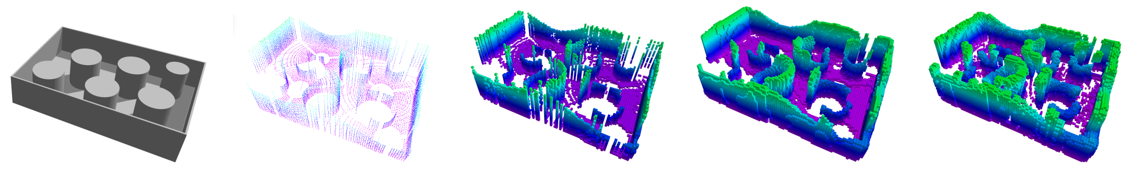
(a) The Gazebo model (b) Raw sensor data (c) OctoMap (d) GPOctoMap (e) Our method, BGKOctoMap

Figure 4.4: Structured environment simulation. We show (a) the Gazebo simulation model for the environment, (b) the simulated raw sensor data, (c) the map produced by OctoMap, (d) the map produced by a prior method, GPOctoMap, and (e) the result of applying our proposed method, BGKOctoMap.

updates as GPOctoMap-NBCM and GPOctoMap-NBCM-P when we apply pruning [36], while we refer to our method as Bayesian generalized kernel OctoMap or BGKOctoMap. We show that our approach offers comparable performance to GPOctoMap-NBCM-P when there is sufficient data. A comparison of computation time is given in Table 4.1 where GPOctoMap-NBCM-P is abbreviated as GPOM. Generally the proposed method achieves map inference in time comparable to OctoMap, which suggests applicability to real-time tasks. In maps with sparser coverage, such as the "unstructured" map, our more conservative method exhibits decreased predictive performance, since more of the query points are far from training data. We additionally provide an experimental demonstration of a robot performing station-keeping in the simulated "structured" environment while scanning one region repetitively. We show that our method is reliable over many scans of the same area, whereas GPOctoMap predictions gradually become overly aggressive, over-predicting occupancy.

4.3.1 Simulated Data

The simulated environments each span $10.0 \times 7.0 \times 2.0$ meters. The structured simulation used is significantly more open than the unstructured simulation, allowing for more complete sensor coverage. Qualitatively, we observe in Figure 4.4 that BGKOc-

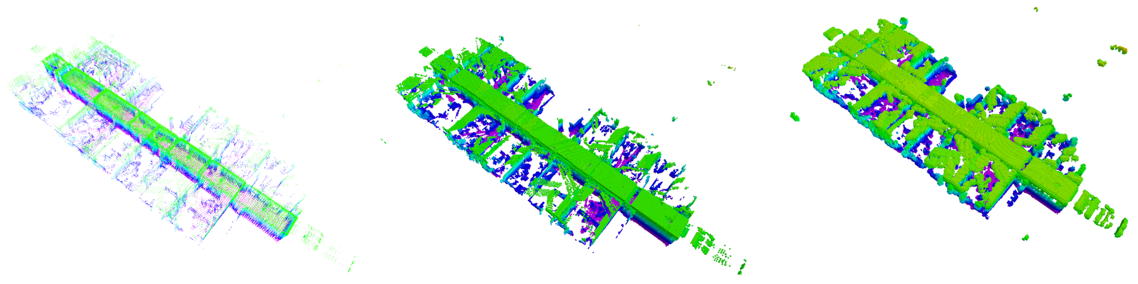


(a) The Gazebo model (b) Raw sensor data (c) OctoMap (d) GPOctoMap (e) Our method, BGKOctoMap

Figure 4.5: Unstructured environment simulation, showing (a) the simulated environment in Gazebo, (b) the simulated raw sensor data, (c) the result after applying OctoMap, (d) the map produced by GPOctoMap, and (e) our Bayesian generalized kernel mapping method, BGKOctoMap.

toMap fills in several walls where OctoMap does not, drawing useful conclusions about regions such as the far corner. In Figure 4.5 we again show that BGKOctoMap closes many of the gaps in the OctoMap. The receiver operating characteristic (ROC) curves for the structured map in Figure 4.3 show comparable performance between BGKOctoMap and GPOctoMap when even though data is sparse, overall coverage of the map is good. On the other hand, the ROC curve for the “unstructured” map shows that the limited sensor coverage affects the performance of our method, while GPOctoMap-NBCM-P performs well in both cases. The ROC curve plots the true positive rate versus the false positive rate. Here we use the predicted occupancy probabilities and compare to ground truth occupancy probabilities of 1.0 for an occupied cell or 0.0, so that the comparison of inference accuracy is independent of our choice of thresholds for the state model in (4.9). Along the curve, the probability threshold we use to choose the positive or negative class varies from 1.0 to 0.0. This can be seen as a plot of predictive performance as we change the occupancy probability threshold. The area under the curve (AUC) is also provided in each case for comparison of inference accuracy.

In the case of the “unstructured” map, BGKOctoMap often reverts to a prior occupancy probability of 0.5 with high variance in regions where data is particularly



(a) Raw sensor data (b) OctoMap (c) Our method, BGKOctoMap

Figure 4.6: BGKOctoMap applied to the Freiburg FR-079 corridor dataset [34].

sparse. In some ways this is a desirable attribute of the model, since it captures the uncertainty inherent in prediction with limited data. On the other hand, GPOctoMap is capable of producing accurate predictions even when training data is far away from query points. In both cases, GPOctoMap and OctoMap achieve better coverage of the floor than BGKOctoMap. This is an artifact of the 2.5D nature of the simulated environment coupled with our naïve representation of free space used for inference. Interpolation of too many free-space samples along the sensor ray artificially increases support for the free-space class in the more open regions of the environment, causing many areas of the floor to be misclassified as free or unknown, not passing the occupancy and variance thresholds in (4.9).

4.3.2 Real Data

The Freiburg corridor pointcloud dataset [34] has dimensions $43.8 \times 18.2 \times 3.3$ meters. It represents a substantially more expansive environment than the simulated data, and accordingly, the data requires more computation time. Since the pointcloud is dense and we are primarily concerned with the application of our algorithms to sparse data, we downsample the 89445 points per scan on average to a resolution of 0.1m, amounting to 7601 points per scan, which provides an artificially sparsified dataset. In

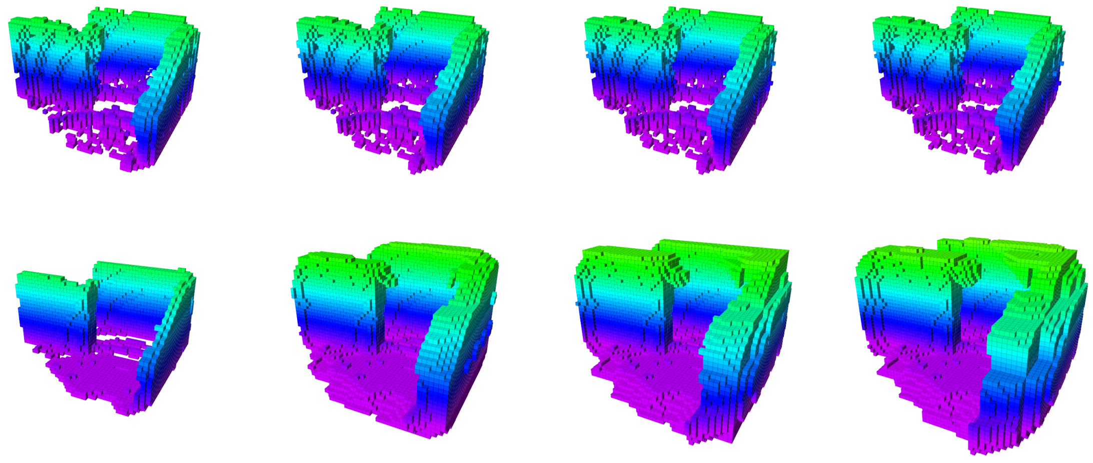


Figure 4.7: In this simulated station-keeping demonstration, we show the results of updating both BGKOctoMap (Top) and GPOctoMap-NBCM (Bottom) after the introduction of 1, 15, 30, and 60 scans containing the same data (Left to Right).

this case, with substantially more data spanning a large environment, BGKOctoMap continues to perform comparably to OctoMap in computation time required. The map produced by BGKOctoMap from the Freiburg corridor pointclouds is shown in Figure 4.6.

4.3.3 Comparison of Long-Term Behavior

Here we demonstrate the stable long-term performance of the proposed mapping algorithm, one of its most desirable and useful features. We provide a station keeping scenario in which a robot repeatedly scans a single location. Using the structured environment simulation, we repeatedly input the same pointcloud to both BGKOctoMap and GPOctoMap-NBCM. The effects of this demonstration are provided in Figure 4.7, where we show the output of each method after 1, 15, 30, and 60 scans. We observe that while our method does experience some slight change due to the contribution from the new data (particularly in areas where $\alpha_0 + \beta_0 \approx \bar{k}$ after one scan), the change is mild in comparison to that of GPOctoMap. Repeated application of the BCM update approximations cause GPOctoMap to gradually predict that the walls and floor of the map are thicker, even though we update it with the same pointcloud.

The BCM update for Gaussian process regression does not perform well in this scenario because of the approximation:

$$p(\mathcal{D}_i|\mathcal{D}_{i-1}, f_q) \approx p(\mathcal{D}_i|f_q), \tag{4.11}$$

where $\mathcal{D}_i$ is comprised of a single set of training points and corresponding outputs. In our case this is a single range scan with "occupied" hit points and "free" points interpolated along the sensor ray. The vector f_q consists of the unknown response variables corresponding to a set of query inputs. The assumption made in the BCM update is that the two datasets used to train separate Gaussian process regression models are conditionally independent given the response variables to the query input. We blatantly violate this assumption in this experiment; by using repeat observations, we create a situation that generates highly correlated observations. To see how this happens in practice, we can evaluate the BCM update to the mean in (2.28) and covariance in (2.29) in this scenario. Letting $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K \mid \forall_{i>1}\mathcal{D}_i = \mathcal{D}_1\}$, we can obtain the following expression for the inverse covariance:

$$\begin{aligned}
\widehat{\text{cov}}(f_*|\mathcal{D})^{-1} &= -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i)^{-1} && (4.12)\\
&= -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_1)^{-1} && (4.13)\\
&= -(K-1)\sigma_f^{-2} + K\text{cov}(f_*|\mathcal{D}_1)^{-1}. && (4.14)
\end{aligned}$$

Here we observe that the inverse covariance of K estimators trained on the same data is K times the covariance of one of the estimators. That is, each time we repeat this estimation procedure, the predicted covariance goes *down*. In principle, there is nothing wrong with this, and we would observe a similar result for the Bayesian nonparametric kernel inference method we have presented. When we examine the predicted mean, we obtain the following:

$$\begin{aligned}\hat{m}(f_*|\mathcal{D}) &= \widehat{\text{cov}}(f_*|\mathcal{D}) \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_i) m(f_*|\mathcal{D}_i) && (4.15)\\ &= \widehat{\text{cov}}(f_*|\mathcal{D}) \sum_{i=1}^{K} \text{cov}(f_*|\mathcal{D}_1) m(f_*|\mathcal{D}_1) && (4.16)\\ &= K\widehat{\text{cov}}(f_*|\mathcal{D})\text{cov}(f_*|\mathcal{D}_1) m(f_*|\mathcal{D}_1). && (4.17)\end{aligned}$$

Finally, we examine the result of "squashing" in (2.25) by dividing the expression for the mean in (4.17) by the expression for the covariance in (4.14):

$$\begin{aligned}\frac{\mu_*}{\sigma_*^2} &= \frac{K\sigma_*^2\sigma_1^2\mu_1}{\sigma_*^2} && (4.18)\\ &= K\sigma_1^2\mu_1, && (4.19)\end{aligned}$$

where, for consistency with (2.25) we have written the mean as μ_*. We also approximate the covariance $\widehat{\text{cov}}(f_*|\mathcal{D})$ as a variance, denoted σ_*^2, and denote the mean and variance of the estimator trained on $\mathcal{D}_1$ as μ_1 and σ_1^2, respectively. The output of the "squashing" function then becomes:

$$p(y_* = 1|X, y) = \frac{1}{1 + \exp(-\gamma\omega_*)}, \ \gamma > 0 \tag{4.20}$$

$$= \frac{1}{1 + \exp(-\gamma \frac{\sigma^2_{min}\mu_*}{\sigma^2_*})} \tag{4.21}$$

$$= \frac{1}{1 + \exp(-K\gamma\sigma^2_{min}\sigma^2_1\mu_1)}. \tag{4.22}$$

In the limit of an infinite number of scans, this becomes:

$$\lim_{K\to\infty} p(y_* = 1|X, y) = \begin{cases} 1, & \text{if } \mu_1 > 0 \\ 0.5, & \text{if } \mu_1 = 0 \\ 0, & \text{if } \mu_1 < 0 \end{cases} \tag{4.23}$$

so we observe that the predicted probability will be driven toward 1 or 0 if the predicted mean is nonzero, else it will remain at 0.5. It is this process that causes the artifacts observed in Figure 4.7. The nonparametric Bayesian inference method differs in that our estimation operates directly on probabilities. By performing this update to the variance in (4.2), we can easily find that the variance decreases with each identical dataset. However, the equivalent update to the mean (4.1) for K datasets with our proposed formulation is as follows:

$$\mathbf{E}[\theta] = \frac{\alpha}{\alpha + \beta} \tag{4.24}$$

$$= \frac{\alpha_0 + \sum_{j=1}^{K}\sum_{i=1}^{N} k(x_i, x_*)y_i}{\alpha_0 + \beta_0 + \sum_{j=1}^{K}\sum_{i=1}^{N} k(x_i, x_*)} \tag{4.25}$$

$$= \frac{\alpha_0 + K\sum_{i=1}^{N} k(x_i, x_*)y_i}{\alpha_0 + \beta_0 + K\sum_{i=1}^{N} k(x_i, x_*)}, \tag{4.26}$$

which, when there is sufficient training data, can be approximated by the Nadaraya-Watson estimator in (4.8), denoting the estimate of the mean after K updates as $\hat{m}(x_*)_K$ and after one scan as $\hat{m}(x_*)_1$, as:

$$\hat{m}(x_*)_K = \frac{K \sum_{i=1}^{N} k(x_i, x_*) y_i}{K \sum_{i=1}^{N} k(x_i, x_*)} \tag{4.27}$$

$$= \frac{\sum_{i=1}^{N} k(x_i, x_*) y_i}{\sum_{i=1}^{N} k(x_i, x_*)} \tag{4.28}$$

$$= \hat{m}(x_*)_1, \tag{4.29}$$

so the estimate of the mean remains unchanged whenever this approximation holds.

We observe that by estimating the expected value of the parameter θ, we explicitly enable confident predictions that are neither 0 or 1. If a substantial amount of data is collected, all in roughly the same proximity to a query point x_*, our estimate will be 0.5 with very low variance. That is, we know for sure that we have no idea what is in x_*, and it will take even more data to convince us otherwise. We posit that the consequences of the differences between these two methods are relevant, since it is often a priority in simultaneous localization and mapping scenarios to seek out such correlated observations for loop closures. With the BCM update, those observations may produce errors in the map.

4.4 Preliminary Experiments using Point-to-Line Distance

We remarked in the previous section that the free-space representation we use poorly represents the unoccupied space. In this work we sample linearly along sensor rays at a fixed resolution. Clearly, this can have significant effects on the nonparametric inference model we use, since we can skew the model toward low occupancy probabilities by sampling free-space points very densely along a range beam. Other works

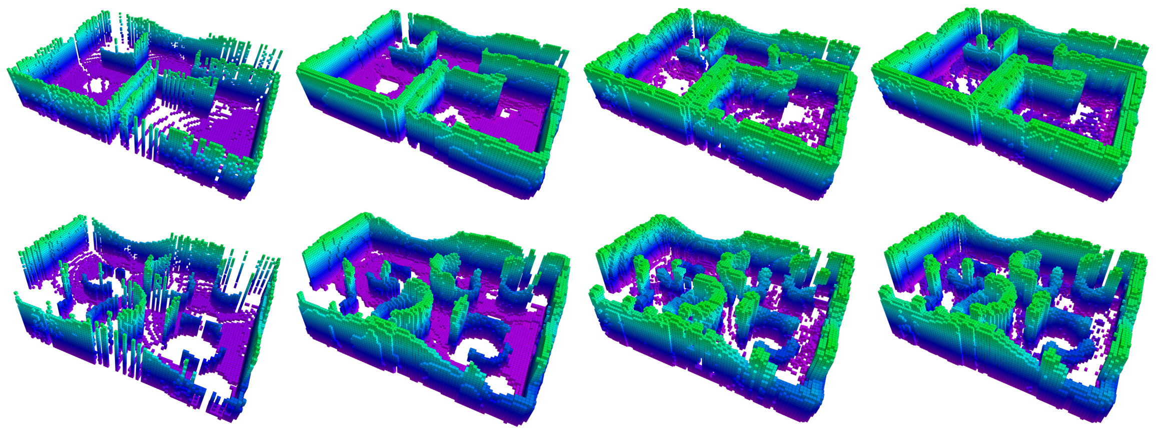


Figure 4.8: Preliminary results using point-to-line distance for free-space representation. Left to Right: OctoMap, GPOctoMap, BGKOctoMap, BGKOctoMap using point-to-line distance. Top: Structured environment; Bottom: Unstructured environment.

have spanned a wide spectrum of parameterization choices, from representing a sensor ray with the single free point on the beam closest to the test point of interest [17] to employing free-space output values weighed by the entire continuous length of a range beam [16].

We posit that the free-space representation proposed by O'Callaghan and Ramos [17] is more appropriate for this method than a sampling-based approach. The primary adaptation to BGKOctoMap is the adoption of the "point-to-line" distance function. Here, we treat each sensor ray as a single observation and compute the shortest distance between a query point and the sensor ray. Equivalently this can be thought of as retrieving the nearest single free point on the sensor ray to the query point in question, as in [17]:

$$x_{free} = \begin{cases} P, & \text{if } d < 0 \\ P + d\frac{PQ}{|PQ|}, & \text{if } 0 \leq d \leq 1 \\ Q, & \text{if } d > 1 \end{cases} \tag{4.30}$$

$$d = \frac{PQ \cdot Px_*}{|PQ|}. \tag{4.31}$$

That is, for a sensor ray PQ, we compute the projection of the vector from the sensor origin P to the query point x_* onto PQ as d. If d is negative, the nearest point on the ray to the query point is P, if it is between 0 and 1, the nearest point to x_* is a distance d along PQ, and if d is greater than 1, the nearest point to x_* is Q. This is simply the shortest distance between the continuous line segment representing the sensor ray and the query point x_*.

Preliminary qualitative experimental validation of this method is provided in Figure 4.8. We observe that this method obtains a more accurate map of occupied space than BGKOctoMap, and better approximates the results of GPOctoMap. Furthermore, we avoid biasing the estimator toward free-space by oversampling.

4.5 Summary

We have proposed a novel inference-based occupancy mapping algorithm, BGKOctoMap, that leverages Bayesian kernel inference, sparse kernels, and test-data octrees. We have demonstrated that the method provides accurate predictions in areas where there is sufficient training data, and smoothly transitions to a prior occupancy probability with high variance when there is not. The method also shows promise for applications where multiple scans of the same areas are captured, since it integrates new data more reliably than existing inference-based mapping methods, and updates can be performed exactly. For the same reason, this method is also useful in situations where we may want to update a single map exactly over multiple sessions.

Chapter 5
Discussion and Conclusion

In the following sections we will discuss compelling areas for future work and summarize our contributions thus far.

5.1 Avenues for Future Work

The two primary assumptions we have made in this work provide very interesting areas for future work: the static map assumption, and the relaxed independence assumption. Adaptation of these methods to dynamic maps would greatly enhance the applicability of these methods. Senanayake et al. [27] have investigated the use of Hilbert maps in spatio-temporally varying environments. The relaxed independence assumption we have made (i.e. that spatial correlation is 0 after a specified distance) may be reasonable in many circumstances; it satisfies intuition that data that is particularly far away from an area in question is unlikely to be highly correlated with the area we want to know about. The main issue we take with this assumption is that we have defined *a priori* a fixed distance for which we make the independence assumption. In practical scenarios, it may not be known beforehand what a reasonable value for this distance is. There are a few ways to mitigate this. For example, we may simply choose a large value for this distance, ensuring that we consider a spatial region that encompasses all potentially correlated areas. We may also avoid this assumption altogether, in which case the exact recursive updates used in BGKOctoMap still apply, but we must query every cell of the map whenever we acquire new data. Finally, we may choose one or more methods to automatically learn an appropriate value for this distance; we might perform cross-validation, or use a method like automatic relevance determination (as was used by Guizilini and Ramos [6]) to allow this value to vary for different regions of the map.

Hilbert maps are a recent innovation in robotic mapping, and there are many possible directions for future research with this technique. We hope this research moves toward real-time inference over occupancy maps. One concern with Hilbert maps as opposed to Gaussian process occupancy mapping is that Hilbert maps require parameter tuning, specifically of γ for the RBF kernel approximation and the regularization parameter α from the SGD formulation, to achieve robust performance. In principle, parameters can be tuned beforehand on similar maps. If a reserved set of sensor data from a scan is used as a cross-validation set, it is possible to perform grid search to estimate reasonable parameters automatically. Grid search is parallelizable, allowing many parameter configurations to be evaluated simultaneously. A major performance consideration for these methods is storing and querying large numbers of grid cells at fine resolution. It may be effective to apply multi-resolution techniques like OctoMap to significantly reduce the memory needed in the map update step. Updating map cells can also be done in parallel and with GPU programming, which can potentially offer a significant speed up over sequentially updating cells.

While we apply the proposed local map fusion method specifically to Hilbert maps, in principle it can also be applied to the output of Gaussian process regression for fusion without the use of a BCM. Whether this technique would improve upon the results of Gaussian process occupancy mapping has yet to be determined. Another potentially promising area for future work in this local map fusion technique would be to limit the query area by filtering outliers from the training data. Further, a rigorous evaluation of kernel approximation techniques, particularly the sparse kernel approximation provided in [22] in 3D mapping using Hilbert maps would be interesting. Finally, it may be worthwhile to investigate the possibility of aggregating data prior to training and classification, i.e. submap fusion. For example, we train and test on 120° sector scans in our simulations, but it may be possible to increase the accuracy of estimators by aggregating data from several scans, then performing inference and fusion using the information from the larger submap. The resulting map could still be built quickly in steps, but the map construction would be less incremental, whereas one of our primary goals in this evaluation was to perform inference that assimilates each new individual scan.

With regard to BGKOctoMap, we have implicitly assumed certainty of pose knowledge in these experiments. This assumption can be relaxed somewhat by incorporating the use of the "expected kernel" used by Jadidi et al. [9] and formulated independently using reproducing kernel Hilbert spaces by Ramos and Ott [22]. Extension of this method with the "expected kernel" would enable the use of this inference technique in mapping scenarios where there is known pose uncertainty, as we examined in our evaluation of overlapping Hilbert maps.

We do not make full use of all of the capabilities of the nonparametric Bayesian inference model used. It is possible with this method to incorporate more informed prior knowledge in a principled way by making α and β functions of the query point, as in [24]. Here we simply use a small uninformative prior to maintain an "unknown" state ($p = 0.5$) in areas with very little training data. The use of the sparse Matérn kernel in [10] may also benefit this method, since exact inference could still be performed, but the ability of the Matérn kernel to capture sharp changes in occupancy probability could improve predictive performance.

While we provide preliminary results on the use of point-to-line distance for BGKOctoMap, a more rigorous evaluation of this method including a quantitative comparison with previous research are warranted for further investigation.

5.2 Conclusion

Learning-aided occupancy mapping is a promising research area with substantial room for improvement. Thus far, the proposed methods have not been deployed on real robotic systems for online applications. Our proposed formulation of Hilbert maps has substantially decreased the computational burden of the method, and may make this method viable for real-time updates to occupancy maps constructed from logistic regression classifiers. We have demonstrated real-time performance with Bayesian generalized kernel inference, and we seek to develop this method further; exploring its use with real data acquired from underwater vehicles like the VideoRay, and examining ways to better plan paths or perform exploration in maps constructed using this method. Learning-aided occupancy mapping has provided a tractable way to reason about spatial correlations between measurements, and we hope that ultimately this work will provide practical benefit in the areas of robot navigation and exploration where sensor data is sparse or noisy, like the applications of underwater robotics that motivated our research in this area.

Bibliography

[1] C.M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[2] K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3D Hilbert maps," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1011-1018, 2016.

[3] K. Doherty, J. Wang, and B. Englot, "Bayesian Generalized Kernel Inference for Occupancy Map Prediction," *Proceedings of the IEEE International Conference on Robotics and Automation*, Accepted, To Appear, May 2017.

[4] A. Elfes, "Sonar-based real-world mapping and navigation." *IEEE Journal on Robotics and Automation*, vol. 3(3), pp. 249-265, 1987.

[5] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22(6), pp. 46-57, 1989.

[6] V. Guizilini and F. Ramos, "Large-scale 3D scene reconstruction with Hilbert Maps." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3247-3254, 2016.

[7] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34(3), pp. 189-206, 2013.

[8] M. G. Jadidi, J. V. Miro, R. Valencia, and J. Andrade-Cetto, "Exploration on continuous Gaussian process frontier maps," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 6077-6082, 2014.

[9] M. G. Jadidi, J. V. Miro and G. Dissanayake, "Warped Gaussian Processes Occupancy Mapping With Uncertain Inputs," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 680-687, April 2017.

[10] S. Kim and J. Kim, "GPmap: A unified framework for robotic mapping based on sparse Gaussian processes," *Proceedings of the 9th International Conference on Field and Service Robotics*, 2013.

[11] S. Kim and J. Kim,"Recursive Bayesian Updates for Occupancy Mapping and Surface Reconstruction," *Proceedings of the Australasian Conference on Robotics and Automation*, 2014.

[12] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2149-2154, 2004.

[13] A. Melkumyan and F. Ramos, "A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets," *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*, vol. 9, pp. 1936-1942, 2009.

[14] H.P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 116-121, 1985.

[15] E.A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9(1), pp. 141-142, 1964.

[16] S. O'Callaghan and F. Ramos, "Continuous occupancy mapping with integral kernels," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1494-1500, 2011.

[17] S. O'Callaghan and F. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31(1), pp. 42-62, 2012.

[18] 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." *The Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[19] V. Peretroukhin, W. Vega-Brown, N. Roy, and J. Kelly, "PROBE-GK: Predictive robust estimation using generalized kernels," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 817-824, 2016.

[20] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." *Advances in large margin classifiers*, 10(3) pp. 61-74, 1999.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, 2009.

[22] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," *Proceedings of Robotics: Science and Systems*, 2015.

[23] C.E. Rasmussen and C.K.I. Williams, *Gaussian Processes for Machine Learning*, Cambridge, MA: The MIT Press, 2006.

[24] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian Learning for Safe High-Speed Navigation in Unknown Environments," *Proceedings of the 17th International Symposium on Robotics Research*, 2015.

[25] R.B. Rusu and S. Cousins, "3D is here: Point cloud library (pcl)," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1-4, 2011.

[26] B. Scholkopf and A.J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, 2001.

[27] R. Senanayake, L. Ott, S. O'Callaghan, F.T. Ramos, "Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments," *Advances in Neural Information Processing Systems*, 2016.

[28] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 9-15, 2012.

[29] S. Srivastava and N. Michael, "Approximate continuous belief distributions for precise autonomous inspection," *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 74-80, 2016.

[30] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence* 99.1 (1998): 21-71.

[31] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots* 15.2 (2003): 111-127.

[32] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, Cambridge, MA: The MIT Press, 2005.

[33] V. Tresp, “A Bayesian Committee Machine, *Neural Computation*, vol. 12(11), pp. 2719-2741, 2000.

[34] University of Freiburg OctoMap 3D Scan Dataset http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/

[35] W.R. Vega-Brown, M. Doniec, and N.G. Roy, “Nonparametric Bayesian inference on multivariate exponential families,” *Advances in Neural Information Processing Systems*, pp. 2546-2554, 2014.

[36] J. Wang and B. Englot, “Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion,” *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1003-1010, 2016.

[37] J. Wang, S. Bai, and B. Englot “Underwater Localization and 3D Mapping of Submerged Structures with a Single-Beam Scanning Sonar,” *Proceedings of the IEEE International Conference on Robotics and Automation*, Accepted, To Appear, May 2017.

[38] G.S. Watson, “Smooth regression analysis,” *Sankhya: The Indian Journal of Statistics, Series A*, vol. 26(4), pp. 359-372, 1964.

[39] C. K. I. Williams and M. Seeger, “Using the Nyström method to speed up kernel machines,” *Proceedings of Neural Information Processing Systems*, 2000.