Fast, Accurate Gaussian Process Occupancy Maps via Test-Data Octrees and Nested Bayesian Fusion

Jinkun Wang and Brendan Englot

Abstract— **We present a novel algorithm to produce descriptive online 3D occupancy maps using Gaussian processes (GPs). GP regression and classification have met with recent success in their application to robot mapping, as GPs are capable of expressing rich correlation among map cells and sensor data. However, the cubic computational complexity has limited its application to large-scale mapping and online use. In this paper we address this issue first by proposing test-data octrees, octrees within blocks of the map that prune away nodes of the same state, condensing the number of test data used in a regression, in addition to allowing fast data retrieval. We also propose a nested Bayesian committee machine which, after new sensor data is partitioned among several GP regressions, fuses the result and updates the map with greatly reduced complexity. Finally, by adjusting the range of influence of the training data and tuning a variance threshold implemented in our method's binary classification step, we are able to control the richness of inference achieved by GPs - and its tradeoff with classification accuracy. The performance of the proposed approach is evaluated with both simulated and real data, demonstrating that the method may serve both as an improved-accuracy classifier, and as a predictive tool to support autonomous navigation.**

I. INTRODUCTION

Constructing maps from range sensor measurements is one of the fundamental tasks of robotic perception. Many applications such as localization and autonomous navigation depend on reliable and accurate map representations of the environment. Besides accuracy, efficiency is another key challenge in robotic mapping. Robots equipped with volumetric laser scanners or depth cameras may generate millions of points in a single scan, which requires the efficient implementation of mapping algorithms.

Occupancy grid mapping [1], which represents the environment by grid cells of equal size, has been widely used in exploration and navigation tasks. It captures the locations of obstacles in the environment by maintaining a probability of occupancy for each cell which is updated independently and incrementally using Bayes filtering. OctoMaps [2] reduce the memory usage in occupancy grid mapping by organizing cells in an efficient data structure based on octrees. Another benefit of OctoMaps is that they can produce maps of variable resolution from the same underlying data. Despite the benefits, the above methods rely on the assumption that all grid cells are statistically independent, and sensor data is only correlated with grid cells directly intersected by range beams. As a result of this strict assumption, sparse sensor measurements will yield discontinuous occupancy maps between adjacent sensor views or scan lines, which may pose a threat for navigation tasks if path planners deem the gaps to be unoccupied. This has been highlighted previously as a problem that poses difficulties for robot exploration [3].

J. Wang and B. Englot are with the Department of Mechanical Engineering, Stevens Institute of Technology, Castle Point on Hudson, Hoboken NJ 07030, USA, {benglot, jwang92}@stevens.edu.

To overcome this limitation, Gaussian processes (GPs), a non-parametric Bayesian learning technique used for regression and classification, were introduced into occupancy mapping [4], [5]. This demonstrated that GPs could be used to achieve rich and reliable probabilistic inference about unobserved areas; gaps in the sensor data will be assigned a probability of occupancy that is correlated with neighboring areas covered by the sensor. The approach offers greater flexibility than previous efforts to capture dependency that are not suitable for online use [6] or cannot feasibly be extended to 3D mapping problems [7], [8]. However, the major drawback of GP regression is the high computational complexity of $O(N^3 + N^2M)$, where N is the number of training data and M is the number of test data. This high computational cost limits its scalability to large datasets.

Based on the property of isotropic kernel functions that correlation will decrease to zero as the distance between two points increases, local approximation methods have been proposed to reduce the size of the training data. The kd-tree, a space-partitioning data structure, has been used to search for training data near each test point [4], [9], [10]. Training data are then approximated by the subset that lies within a specified range of the test point, without sacrificing accuracy. However, local GP regression using kd-trees, which requires regression to be performed for individual test points one by one, is more time-consuming than GP regression for batch data. To improve this, the concept of extended blocks was introduced in [11], [12]. GPs are applied to the test data in individual blocks of the map using the training data in an extended block that includes neighboring blocks.

Another methodology to address scalability is based on the idea that we can perform GP regression for different sets of training data separately, after which predictions from different modules can be fused into an inclusive set of estimated test data outputs. Modular regressions can be performed over exclusively non-overlapping test regions [13], [14], but this may prove challenging in real-time mapping applications, and we will assume a test region may intersect regions where prior regressions have occurred. Fusion of multiple regressions has been achieved successfully using a Bayesian committee machine (BCM) [15]. The local GP occupancy map from a new sensor measurement is repeatedly fused

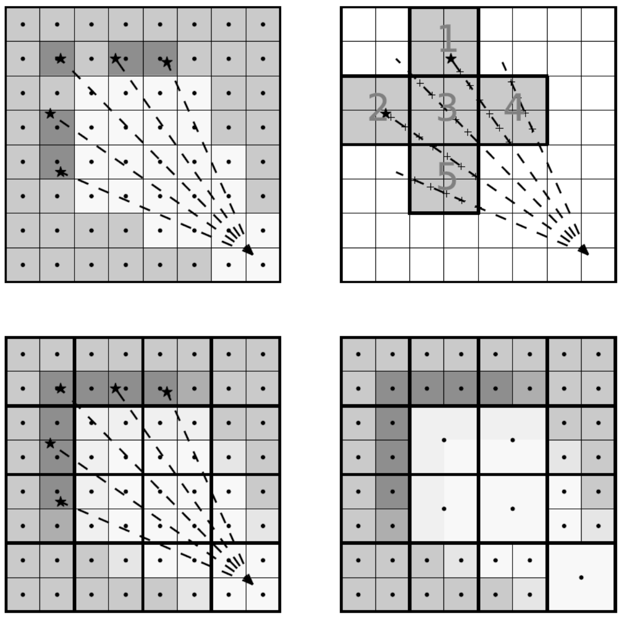

Fig. 1: 2D demonstration of our approach with an example laser scan, where dots represent test points and the stars and crosses along scan lines represent training data (hits and free points, respectively). *Top Left*: Standard occupancy grid mapping. *Top Right*: Setup for predicting the occupancy probability of Block #3. The extended block consists of 5 blocks for which separate regressions are performed using the data contained in each; the resulting predictions of Block #3 from each regression are fused via BCM. *Bottom Left*: GP occupancy mapping with test-data octrees before pruning. There are 16 depth-two octrees which are separated by thick grid lines in the map. *Bottom Right*: Final map after pruning.

with the global map using a BCM in [12] and [16]. With the help of BCMs, GP occupancy mapping is capable of supporting online 2D mapping and exploration [16].

Motivated by the use of extended blocks and the BCM in these works, we propose a nested application of the BCM in which two subsequent fusions are performed. After subdividing new sensor observations into several sets of training data and performing a GP regression for each, the first fusion step synthesizes the result, while the second fusion step merges this combined result from the newest observation into the existing occupancy map. This two-tiered utilization of BCMs attacks the cubic complexity of GP regression, reducing the size of N and enabling our method to achieve online 3D mapping with large-scale datasets. We also propose a new data structure, test-data octrees, to prune back and condense similarly-valued test data, reducing the size of M. Finally, our method provides the flexibility to control how much spatial correlation GPs can capture, by tuning the size of the neighborhood influenced by training data. This, in combination with tuning a variance threshold implemented in our method's binary classification step, affords an adjustable tradeoff between the aggressiveness of a GP occupany map's prediction capability and the accuracy of the occupancy classification that is performed. Our results will demonstrate that at the former end of the spectrum, GP occupancy maps can extrapolate successfully over sparse data, and in the latter case, the predictive capability of GP occupancy maps is more conservative, but the method exceeds the accuracy of OctoMaps as a classifier.

In Section II , we review GP occupancy mapping, including the covariance functions and binary classification functions we use with GPs. In Section III, we introduce the proposed test-data octree construct and its pruning process, employed after updating the global map with new sensor observations. In Section IV, we present the use of nested BCM updates to fuse the several constituent GP regressions performed for a new sensor observation, followed by fusion of the result with the global map. We demonstrate our method using simulated and real datasets in Section V.

II. GAUSSIAN PROCESS OCCUPANCY MAPPING

In this paper, we address the problem of generating an occupancy map from sensor observations in a static environment under the assumption that a robot's poses are known, in which case a map cell's probability of occupancy may be expressed as

$$p(m_i|z_{1:t}, x_{1:t}),$$

where m_i is map cell i, $z_{1:t}$ is the set of sensor observations, and $x_{1:t}$ is the set of robot poses. Occupancy grid mapping typically assumes all grid cells are independent, and that a cell's only dependency is on range beams that directly intersect the cell. In contrast, GP occupancy mapping collects sensor observations and the corresponding labels (*free* or *occupied*) as training data; the map cells comprise test data, which are related to the training data using covariance functions. After a regression is performed, we obtain a cell's probability of occupancy by "squashing" regression outputs into occupancy probabilities using binary classification functions. This procedure is detailed below.

A. Gaussian Process Regression

Let us consider the noisy observation model,

$$y = f(\mathbf{x}) + \epsilon, \tag{1}$$

where $\mathbf{x}$ is the input vector, y is the observed target value of the latent function value $f(\mathbf{x})$ added with noise $\epsilon \in \mathcal{N}(0, \sigma_n^2)$. A Gaussian process is defined as a distribution over functions [17],

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

with mean function $m(\mathbf{x})$ which we take as zero and covariance function $k(\mathbf{x}, \mathbf{x}')$. Given n training points $X = \{\mathbf{x}_i\}_{i=1}^n$ and observation vector $\mathbf{y}$, a Gaussian process predicts the latent values $\mathbf{f}_*$ at m test points $X_* = \{\mathbf{x}_{*i}\}_{i=1}^m$ to be

$$\mathbf{f}_*|X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)). \tag{3}$$

Here we use a compact notation to denote covariance matrices: $K = K(X, X), K_* = K(X, X_*), K_{**} = K(X_*, X_*)$.

The mean and covariance of the Gaussian process can be written as

$$\bar{\mathbf{f}}_* = K_*^T(K + \sigma_n^2 I)^{-1}\mathbf{y}, \tag{4}$$

$$cov(\mathbf{f}_*) = K_{**} - K_*^T(K + \sigma_n^2 I)^{-1}K_*. \tag{5}$$

In the case of 3D occupancy mapping, we have a set of training input data X, where $\mathbf{x}_i \in \mathbb{R}^3$ is an *occupied* point in the environment observed by the sensor or a *free* point sampled along sensor beams, and an accompanying set of training target data Y, where $\mathbf{y}_i = 1$ or -1 for *occupied* and *free* points respectively.

B. Covariance Functions

Covariance functions, or kernel functions, define the similarity between a pair of points: if they are close to each other, they tend to have the same target value [17]. A common choice for GPs is to use a squared exponential covariance function [17], and a sparse covariance function was introduced in [18] that achieves comparable smoothness, while reducing to zero correleration when two points are a specified distance apart. However, such functions are often too smooth to capture the sharp variations in occupancy that typically occur in real-world mapping scenarios. To address this issue, the Matérn covariance function has been applied successfully in GP occupancy mapping [11], [16]. The Matérn covariance function [19], with its smoothness parameter set to $\nu = 3/2$, is defined as

$$k(r) = \sigma_f^2\left(1 + \frac{\sqrt{3}r}{l}\right)\exp\left(\frac{-\sqrt{3}r}{l}\right), \tag{6}$$

where $r = ||\mathbf{x} - \mathbf{x}'||$, and the two hyperparameters σ_f^2 and l are prior signal variance and characteristic length-scale. It's worth noting that the correlation drops quickly as the distance between points increases. Based on this property, in this paper we approximate Gaussian process regression by using a subset of the training data; specifically, we only utilize nearby training data to predict the value of a given test point (detailed in Section III).

C. Binary Classification Functions

In order to "squash" the output of GP regression into an occupancy probability in the range of $[0, 1]$, we adopt a logistic regression model that leverages both mean μ_i and variance σ_i^2 at every test point [16],

$$p(y = 1|\mathbf{x}_i) = \frac{1}{1 + \exp(-\gamma\omega_i)}, \tag{7}$$

where $\omega_i = \sigma_{min}^2\mu_i/\sigma_i^2$ is the weighted mean, σ_{min}^2 is the minimum variance, and γ is a positive constant. A classification is performed by combining the resulting occupancy probability from (7) and variance of the predicton from (5),

$$state = \begin{cases} \textit{free} & p < p_{free}, \sigma_i^2 < \sigma_t^2 \\ \textit{occupied} & p > p_{occupied}, \sigma_i^2 < \sigma_t^2 \\ \textit{unknown} & otherwise \end{cases} \tag{8}$$

where σ_t^2 is the variance threshold, which expresses the confidence we have in the predicted occupancy probability. The tuning of this parameter is discussed in Section IV.

III. SPATIAL PARTITIONING WITH TEST-DATA OCTREES

Our choice of data structures for occupancy mapping will have important implications for the method's time and memory complexity. By representing the environment using equally sized grid cells, we can achieve constant access and write time. However, the downside is that memory consumption grows cubically in the dimension of the environment. An alternative is to use octrees, data structures which recursively partition a space into groups of smaller, equally sized nodes such that every node in the tree has eight children. Octrees are a memory efficient approach for representing the entire 3D environment because the partitioning process need only be implemented in spaces containing observed structures, yielding a multi-resolution map. However, as a result of its tree structure, octrees suffer from an access complexity of $O(log(d))$, where d is the maximum depth of the tree.

Octrees have been applied successfully to 3D occupancy grid mapping in the OctoMap framework [2]. By casting rays in the octree according to a range sensor's observations, OctoMap incrementally updates occupancy probabilities of the cells that rays pass through. In an OctoMap, inserting or partitioning cells occurs in the vicinity of sensor beams only, which conforms to the philosophy of octrees: partitioning as needed. Unlike occupancy grid mapping, GP occupancy mapping attempts to predict the contents of many spaces where sensor data has not direcly landed, and without knowing where the boundaries in occupancy will lie, we must deploy a fine discretization everywhere. In this case, a conventional octree is inappropriate for storing map information; it offers advantages neither in space nor in time considering that all leaf nodes need to be partitioned and initialized in the beginning.

Therefore we present a new data structure called *test-data octrees*, which are pruned to lower resolution over time instead of branching to a higher resolution. When all sibling nodes within the outermost layer of an octree attain the same designation as either *free* or *occupied*, these nodes are pruned from the tree and only their parent remains. This is the case not only for representing obstacles in the map, but also for the test data used repeatedly in Gaussian process regressions from one robot measurement to the next. In instances where the octree is pruned, eight test data points, each located at the centroid of its respective grid cell, are reduced to one point located at the centroid of the parent cell. This is illustrated in a two-dimensional example at the bottom of Figure 1 using quadtrees. In addition to reducing the size of the test data, the map's memory usage can be reduced by pruning nodes with the same state (*free* or *occupied*). In Figure 1, 20 free nodes are pruned, which means that 23% of the memory in use can be freed. Among other general advantages are the fact that we can initialize octrees with pointers, and thus octree cells can be dynamically allocated as a robot explores an unknown environment, and we can retrieve test points and update their states in near-constant time if a tree has a small number of layers.

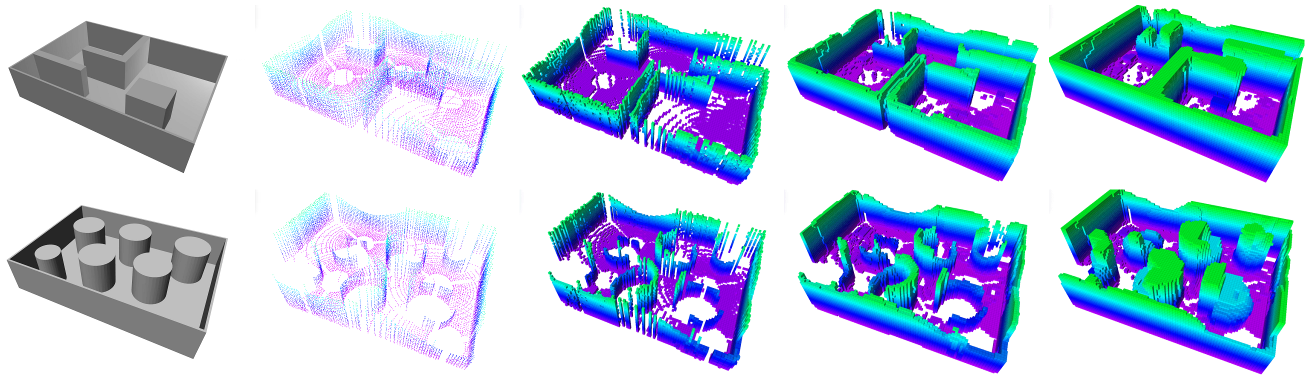

Fig. 2: Mapping results in simulated "structured" (*top row*) and "unstructured" (*bottom row*) environments. From *left* to *right*: ground truths, accumulated raw point clouds, OctoMap, conservative GP mapping, and aggressive GP mapping. All maps are generated with $0.1m$ resolution and cells with occupancy probability > 0.7 are visualized with colors indicating cell height.

IV. NESTED UPDATES WITH BCMs

Using a BCM [15], a separate GP regression may be performed over each newly arrived sensor observation, and the resulting estimates can be fused, either in batch or one new observation at a time. Suppose the entire training space D is split into K data sets such that $D_i = \{X_i, \mathbf{y}_i\}$, for $i = 1, \dots, K$. The formulation of the BCM may then be expressed as:

$$m(\mathbf{f}_*|D) = cov(\mathbf{f}_*|D)^{-1} \sum_{i=1}^{K} cov^{-1}(\mathbf{f}_*|D_i) m(\mathbf{f}_*|D_i), \tag{9}$$

$$cov^{-1}(\mathbf{f}_*|D) = -(K-1)\sigma_f^{-2} + \sum_{i=1}^{K} cov^{-1}(\mathbf{f}_*|D_i). \tag{10}$$

We will approximate the resulting covariance matrices with variance matrices, removing off-diagonal indices as proposed in a prior application of this method [12]. This prevents the potential for cubic complexity in the number of test data, while still allowing rich correlation to be expressed relating a subset of training data to every point of the test data.

In addition to leveraging the BCM to update the map one measurement at a time, as employed in [12] and [16], the occupancy information revealed by a new measurement is recovered in a modular manner, over a series of several GP regressions. First, we determine how many distinct, non-overlapping sets of test data will be updated with training data from the new measurement. In our *conservative* approach to GP occupancy mapping, in which accurate classification is the goal, the test data is drawn separately from every *block* intersected by one or more range beams. Blocks are comprised of several occupancy grid cells, and are maintained at the same resolution as the parent cells in the outermost layer of our test-data octrees. For every block of test data, the corresponding training data is comprised of all portions of the new measurement's range beams that pass through the block's *extended block*, a larger surrounding region that allows more distant training data to influence a block's test data. The notion of using an extended block to derive the training data applied to an individual block's test data was first suggested in [11].

We define an extended block to be the set of neighboring blocks with faces adjacent to the block containing the test data of interest. An example of this definition is given in Figure 1. In our *aggressive* approach to GP occupancy mapping, we perform a GP regression for the test data in every block for which range beams pass through some portion of the *extended* block. This extends the influence of a new measurement to test data in neighborhoods that were not necessarily observed by the sensor. Our aggressive approach also implements a higher variance threshold in Eq. 8 than the conservative approach.

Although the size of a given set of training data is reduced significantly by partitioning the space into blocks and confining training points from a new measurement to extended blocks, the result may often be unsuitable for real-time mapping applications (see [12] or Table I). In the results to follow in Table I, the parameterization labeled BCM-NP can be regarded as [12]'s method with different covariance and logistic functions to provide a one-to-one comparison with our proposed method. The problem can be further divided-and-conquered, however, by applying a BCM to a single block's regression, and splitting apart the training data that lies in an extended block. A separate GP can be trained for each subset of the training data, using a BCM to fuse the results and predict occupancy probabilities in each block. This second-layer BCM is formulated according to the following equations:

$$m(\mathbf{f}_*|D_i) = cov(\mathbf{f}_*|D_i)^{-1} \sum_{j=1}^{E} cov^{-1}(\mathbf{f}_*|D_{ij}) m(\mathbf{f}_*|D_{ij}),$$

$$cov^{-1}(\mathbf{f}_*|D_i) = -(E-1)\sigma_f^{-2} + \sum_{j=1}^{E} cov^{-1}(\mathbf{f}_*|D_{ij}),$$

where E is the number of segments into which an extended block's training data is partitioned (1). At this level, we do not further partition the test data because we want the

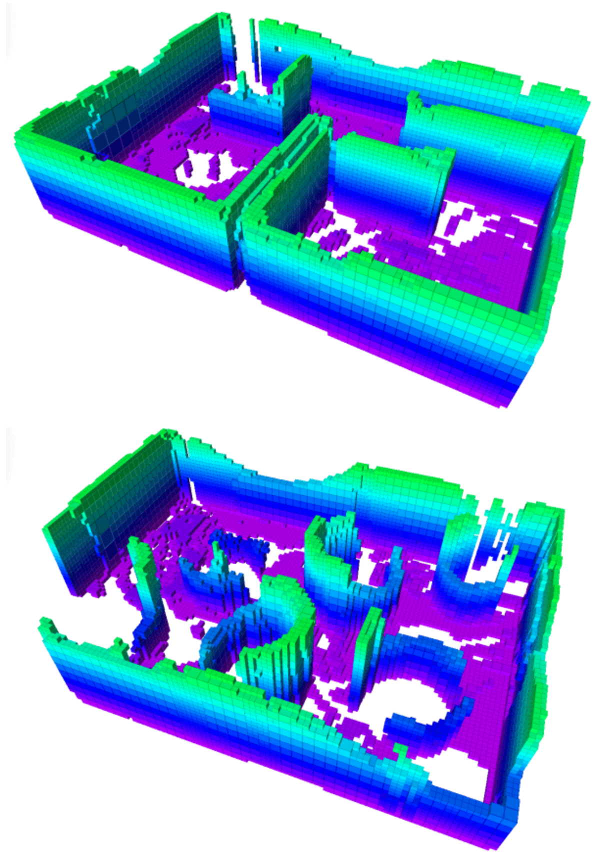
Fig. 3: The GP maps shown in the fourth column of Figure 2 are rendered above with their multi-resolution test-data octrees visible, which are used here to represent the portions of the map predicted to be occupied.

Algorithm 1 GPOctoMap-NBCM-P

```
$\mathcal{M} \leftarrow \{\mathcal{B}_i = \emptyset\}, i = 1, \dots, K$;
$\mathcal{G} \leftarrow \{g_i = \emptyset\}, i = 1, \dots, K$;
while $!MappingComplete$ do
    $\mathcal{Z}_n \leftarrow SensorHits()$;
    $\mathcal{M}_n \leftarrow \emptyset$;        # local blocks need to be updated
    for $\mathcal{B}_i \in \mathcal{M}$ do
        if $Intersect(\mathcal{B}_i, \mathcal{Z}_n)$ then
            $\mathcal{M}_n \leftarrow \mathcal{M}_n \cup \mathcal{B}_i$
        end if
    end for
    for $\mathcal{B}_i \in \mathcal{M}_n$ do  # training in blocks
        if $IsEmptyOctree(\mathcal{B}_i)$ then
            $\mathcal{B}_i \leftarrow CreateOctreeInBlock(\mathcal{B}_i)$;
        end if
        $X, \mathbf{y} \leftarrow GetTrainingPoints(\mathcal{Z}_n)$;
        $g_i \leftarrow GPRegression(X, \mathbf{y})$;
    end for
    for $\mathcal{B}_i \in \mathcal{M}_n$ do  # BCM fusion in extended blocks
        for $\mathcal{B}_e \in FindExtendedBlocks(\mathcal{B}_i)$ do
            $X_* \leftarrow GetOctreeLeafNodes(\mathcal{B}_e)$;
            $\mathbf{f}_*, \mathbf{var}_* \leftarrow GPPredict(g_e, X_*)$;
            $\mathcal{B}_i \leftarrow BCMUpdate(\mathcal{B}_e, \mathbf{f}_*, \mathbf{var}_*)$;
        end for
    end for
    for $\mathcal{B}_i \in \mathcal{M}_n$ do
        $PruneNodes(\mathcal{B}_i)$;
    end for
end while
$OccupancyProbs \leftarrow \emptyset$;
for $\mathcal{B}_i \in \mathcal{M}$ do
    if $!IsEmptyOctree(\mathcal{B}_i)$ then
        $\mathbf{x}_*, \mathbf{f}_*, \mathbf{var} \leftarrow GetOctreeLeafNodes(\mathcal{B}_i)$;
        $\{\mathbf{x}_*, \mathbf{p}\} \leftarrow BinaryClassification(\mathbf{f}_*, \mathbf{var}_*)$;
        $OccupancyProbs \leftarrow OccupancyProbs \cup \{\mathbf{x}_*, \mathbf{p}\}$;
    end if
end for
return $OccupancyProbs$;
```

diverse neighborhood of training data in the extended block to be applied to all test data. The BCM approximation, as a transductive learning method, will weigh the training data appropriately according to their proximity and relevance to the test data [20].

By performing separate regressions for test data segregated into *blocks*, whose respective training data are segregated into *extended blocks*, the computational complexity of a regression decreases from $O(N^3 + N^2M)$ to $O(N^3/K^2 + N^2M/K^2)$, where K is the number of blocks [11], [14]. By dividing each extended block's training data into E segments, and fusing the result with a BCM, the complexity is further reduced to $O(N^3/(K^2E^2) + N^2M/(K^2E))$. It should also be noted that this procedure affords the option of parallelzing the computation of the many small, separate GP regressions required to update the map with a new sensor observation. The full procedure proposed in this paper, including both the use of test-data octrees and the nested application of BCMs to the map update, is summarized in Algorithm 1.

V. EXPERIMENTAL RESULTS

In this section, we demonstrate our method using simulated and real datasets and compare the runtime with that of the OctoMap C++ library. We also give proof of the accuracy of our method using our prior knowledge of the environment boundaries in the simulated data. We implemented our algorithm in C++ using ROS [21] and OpenMP [22] and ran it on a computer with an Intel i7-4790K processor, 32GB of RAM, running 64-bit Ubuntu 14.04. The values of the hyperparameters used in GP regression and classification were $\sigma_n^2 = 0.01, l = 1.0, \sigma_f^2 = 1.0, \gamma = 100$. The hyperparameters are manually tuned, but applied consistently throughout the experiments. This is due to both the challenge of optimizing the predictive result of GP mapping with merely current observations, and the fact that the user must determine the desired aggressiveness of GP mapping's predictive capability. The size of an individual block was $0.8 \times 0.8 \times 0.8m^3$ and the octree inside it has 3 layers. E was set to 7, dividing each extended-block regression among adjacent blocks as illustrated in Figure 1.

Dataset	Size (m^3)	Scans	Points/Scan	Sampled Points/Scan	Method	Time (s)	Time/Scan (s)
Sim Structured Data	$10.0 \times 7.0 \times 2.0$	12	3500	1506	BCM-NP	6.57	0.55
					NBCM-P	0.44	0.04
					NBCM-P+	0.37	0.03
					OctoMap	0.20	0.02
FR-079	$43.8 \times 18.2 \times 3.3$	66	89445	7601	BCM-NP	724.10	10.97
					NBCM-P	10.96	0.17
					NBCM-P+	9.57	0.15
					OctoMap	6.71	0.10
FR Campus	$292.0 \times 167.0 \times 28.0$	81	247817	76009	BCM-NP	N/A	N/A
					NBCM-P	329.95	4.07
					NBCM-P+	294.62	3.64
					OctoMap	242.88	3.00
Spacebot Arena	$72.3 \times 71.4 \times 12.9$	8	296734	64950	BCM-NP	1189.42	148.68
					NBCM-P	33.06	4.13
					NBCM-P+	27.35	3.42
					OctoMap	33.65	4.21

TABLE I: Benchmarking of experimental results. Sub-sampled data are used for GP occupancy mapping whereas OctoMap uses original full-density data. In the table, "NBCM" refers to our proposed nested BCM, which includes BCMs applied within an extended block and also to sequential observations; "BCM" only applies fusion to sequential observations. Methods having "P" use test-data octrees to prune nodes with the same state; "NP" means no pruning. We also explore the "aggressive" variant of our GP mapping formulation, which is denoted as "+".

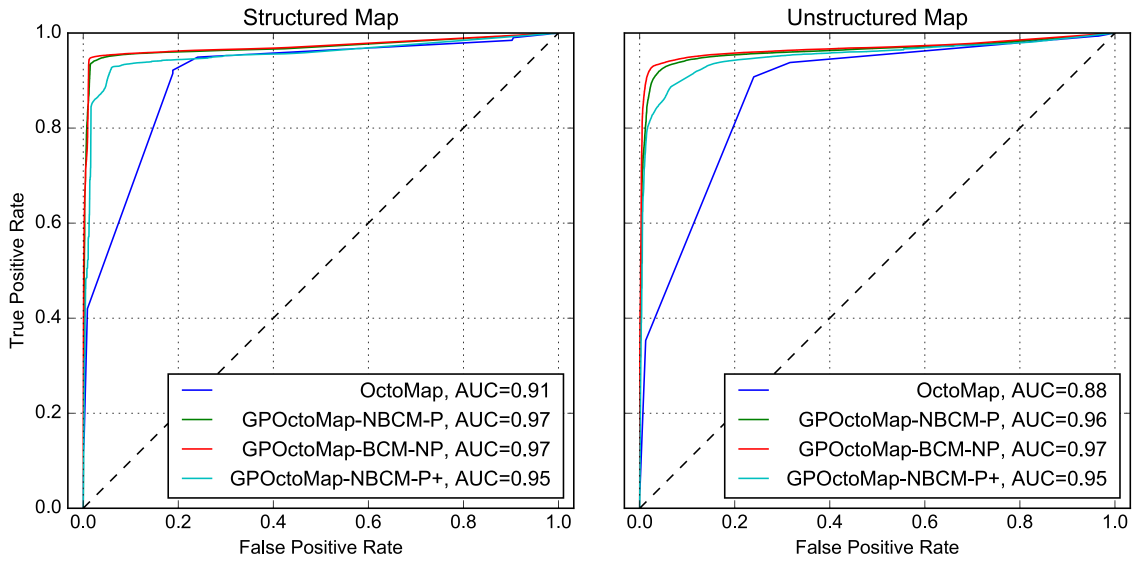

Fig. 4: Receiver operating characteristic (ROC) curves for occupancy mapping with simulated data in the "structured" and "unstructured" environments depicted in Figure 2. *Free* is considered to be a positive label. See Table I for the meanings of the labels in the plot legends.

A. *Simulated Data*

Simulated experiments were conducted in Gazebo [23] where the ground truth for both maps and robot poses were known precisely. The robot navigated to four locations in structured and unstructured example environments (first column in Fig. 2) while taking three observations at each location from a block laser scanner which had a field of view of 120° in azimuth and 60° in elevation, yielding 3500 points per scan. The sparse coverage of the laser data is intended to test the inference capability of GP occupancy mapping. In spite of that, the point clouds generated by the laser were still dense enough for training; thus we sub-sampled the raw data using VoxelGrid filtering in Point Cloud Library (PCL) [24] with resolution $0.1m$. As for OctoMaps, we used the original point cloud without sub-sampling.

As a consequence of these limited observations, the simulated robot did not obtain a complete scan of the environment. When a laser beam swept over flat surfaces, e.g., walls and floors, gaps between beam returns increased. As a result, the raw data contained a substantial number of gaps (second column in Fig. 2). The figures in the third column of Fig. 2 depict conventional occupancy grid mapping with OctoMap; free-space gaps remain in the final result while the predicted occupied cells precisely correspond

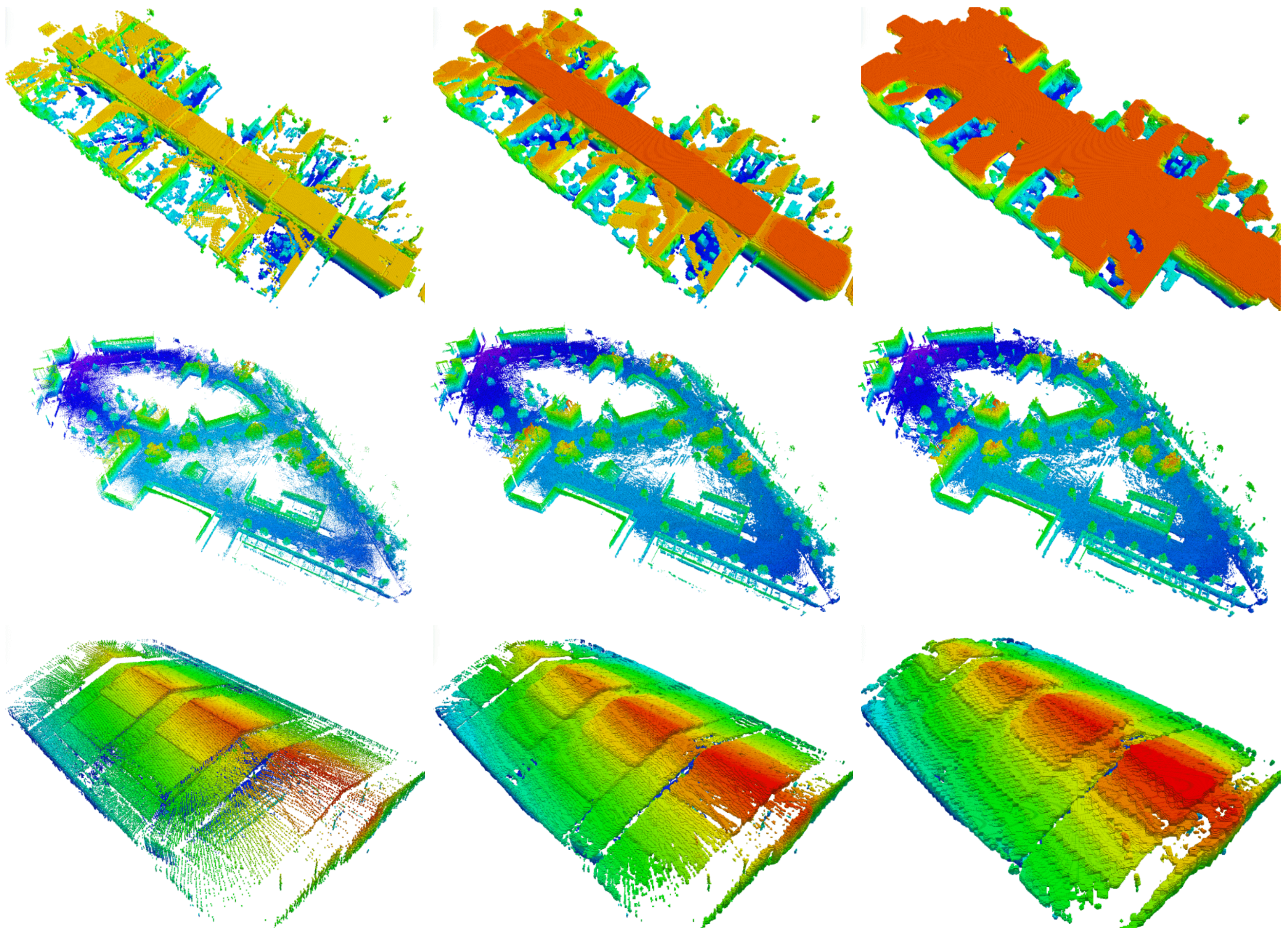

Fig. 5: 3D maps built with OctoMap (*first column*) and GPOctoMap-NBCM-P tuned 1) conservatively (*second column*) and 2) aggressively (*third column*). From *top* to *bottom*: Freiburg-079 corridor dataset ($43.7m \times 182.m \times 3.3m$), Freiburg campus dataset ($292m \times 167m \times 28m$), and Spacebot Arena dataset ($72.3m \times 71.4m \times 12.9m$). All maps are built at $0.1m$ resolution and cells are colored corresponding to cell height.

to laser hits in the raw data, which is of limited utility for further tasks. In contrast, our proposed method reasonably filled in the volumes between adjacent laser scans, which resulted in smooth surfaces (fourth column in Fig. 2). This spatial inference capability was enhanced in our "aggressive" mapping results (fifth column of Fig. 2): our method was largely successful in predicting occupancy in the regions devoid of measurement. The algorithm's test-data octrees are illustrated in Fig. 3.

The objective of our proposed method is to build GP occupancy maps online without sacrificing accuracy. With full knowledge of ground truth, the accuracy of the maps can be inspected using Receiver Operating Characteristic (ROC) curves. ROC curves in Fig. 4 show that our conservative method GPOctoMap-NBCM-P outperforms OctoMap in two ways: 1) GPOctoMap-NBCM-P has fewer incorrectly classified negatives (occupied cells), which yields a lower false positive rate in the unstructured environment; 2) while maintaining a low false positive rate, GPOctoMap-NBCM-P is able to achieve a higher true positive rate, or *recall*, which indicates that GPs can correctly predict more positives (free cells). With regards to aggressive mapping, we see an increase in the false positive rate and a decrease in the true positive rate, which can be explained by the fact that its predictions are not entirely correct. GP occupancy mapping without test-data octrees and nested BCMs (akin to the approach of [12]) marginallly outperforms our "conservative" approach, but at the cost of requiring more than an order of magnitude of additional computation time, as detailed in Table I.

B. Real Data

In this section, we demonstrate our method using three large-scale real datasets (Fig. 5): Freiburg-079 Corridor[1], Freiburg Campus[1], and SpaceBot Arena[2]. Although the obtained point clouds were dense enough to produce comprehensive occupancy maps in some areas, there was sparse coverage of other areas, e.g., the outside of the corridor in the Freiburg 079 dataset, the ceiling of the spacebot arena

[1]http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/

[2]http://www.ais.uni-bonn.de/mav_mapping/

dataset, and the ground in the Freiburg campus dataset. GPOctoMap-NBCM-P improved upon OctoMap's coverage by successfully inferring the contents of gaps to a large extent. It's worth noting that our "aggressive" GP mapping is able to predict most of the ceiling in the FR-079 dataset. The richness of inference signals at least two potential applications of GPOctoMap-NBCM-P. First, it is able to build a map with improved quality using only a subset of sensor measurements, which means a cheap and low-resolution sensor can be used for mapping. Second, the time spent on scanning can be reduced without the need to cover the whole space. In addition to being superior in runtime to prior variants of GP occupancy mapping (represented by the BCM-NP parameterization), the runtime of GPOctoMap-NBCM-P is also comparable to OctoMap's, as shown in Table I. In the spacebot arena dataset, GPOctoMap-NBCM-P even exceeds OctoMap in runtime because there exists a large volume of free space, which dramatically reduces the number of test data through octree pruning.

VI. CONCLUSION

In this paper, we presented an improved formulation of Gaussian process occupancy mapping using nested BCMs and test-data octrees. Equipped with these time-saving approximations, the computational complexity of GPs has been reduced to a level where it can be used for real-time 3D mapping, while retaining high-quality performance as a classifier. By extending the range of influence of our training data and increasing our classifier's variance threshold, we can also accomplish mapping tasks in an aggressive way, where occupancy predictions over sparse data may be useful in guiding high-performance autonomous navigation. The inference richness, accuracy and efficiency of our method are validated with results in simulated and real environments.

However, the usage of test-data octrees can be considered a trade of space in exchange for time; for real-world datasets, our method will typically consume 10 times the memory of OctoMap. Another limitation is that the nodes pruned from the octree can't be partitioned again if their occupancy states change at a later time. Alternative machine learning algorithms continue to be proposed for occupancy mapping, and may offer keys to improved performance in this and other categories. For example, Hilbert maps [25], using a logistic regression classifier, can express correlation within ocupancy maps while using approximate kernels with inexpensive computation.

ACKNOWLEDGMENTS

The authors would like to thank Fanfei Chen for his assistance with our simulated mapping comparison and Kevin Doherty for helpful conversations on supervised learning. This research has been supported in part by the National Science Foundation, grant number IIS-1551391.

REFERENCES

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

[2] 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.

[3] 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.

[4] S. O'Callaghan, F.T. Ramos, and H. Durrant-whyte, "Contextual occupancy maps using Gaussian processes," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1054-1060, 2009.

[5] S.T. O'Callaghan and F.T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31(1), pp. 42-62, 2012.

[6] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15(2), pp. 111-127, 2003.

[7] M. Veeck and W. Burgard, "Learning polyline maps from range scan data acquired with mobile robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1065-1070, 2004.

[8] M. Paskin and S. Thrun, "Robotic mapping with polygonal random fields," *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 450-458, 2005.

[9] Y. Shen, A. Y. Ng, and M. Seeger, "Fast Gaussian process regression using kd-trees," *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Scholkopf, and J. Platt, Eds., Cambridge, MA: MIT Press, pp. 1225-1232, 2006.

[10] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair, "Gaussian process modeling of large scale terrain," *Journal of Field Robotics*, vol. 26(10), pp. 812-840, 2009.

[11] 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*, 14 pp., 2013.

[12] S. Kim and J. Kim, "Recursive bayesian updates for occupancy mapping and surface reconstruction," *Proceedings of the Australasian Conference on Robotics and Automation*, 8 pp., 2014.

[13] S. Kim and J. Kim, "Occupancy mapping and surface reconstruction using local Gaussian processes with kinect sensors," *IEEE Transactions on Cybernetics*, vol. 43(5), pp. 1335-1346, 2013.

[14] S. Kim and J. Kim, "Continuous occupancy maps using overlapping local Gaussian processes," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4709-4714, 2013.

[15] V. Tresp, "A Bayesian committee machine," *Neural Computation*, vol. 12(11), pp. 2719-2741, 2000.

[16] 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.

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

[18] A. Melkumyan and F. Ramos, "A sparse covariance function for exact Gaussian process inference in large datasets," *Proceedings of the IJCAI International Joint Conference on Artificial Intelligence*, pp. 1936-1942, 2009.

[19] M. Stein, *Interpolation of spatial data: some theory for kriging*, New York: Springer Verlag, 1999.

[20] A. Schwaighofer and V. Tresp, "Transductive and inductive methods for approximate Gaussian process regression," *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds., Cambridge, MA: MIT Press, pp. 977-984, 2003.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T.B. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, 6 pp., 2009.

[22] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5(1), pp. 46-55, 1998.

[23] 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.

[24] 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.

[25] F. Ramos and L. Ott, "Hilbert maps : scalable continuous occupancy mapping with stochastic gradient descent," *Proceedings of Robotics: Science and Systems*, 9 pp., 2015.