Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2014, The University of Melbourne, Melbourne, Australia

Recursive Bayesian Updates for Occupancy Mapping and Surface Reconstruction

Soohwan Kim and Jonghyuk Kim
The Australian National University, Australia
{soohwan.kim, jonghyuk.kim}@anu.edu.au

Abstract

This paper proposes a new method to build two kinds of map representations, occupancy maps and surface meshes, in a single framework of Gaussian processes and update recursively using Bayesian Committee Machines. Previously, Gaussian processes were applied to robotic mapping as a batch process considering all the observations at once. However, that approach not only increases the number of training data, which is critical to the time complexity of Gaussian processes, but also is not able to update the final map with new observations. Therefore, we propose to recursively update Gaussian process maps using Bayesian Committee Machines based on the static world assumption. We demonstrate our method with a real dataset and compare the accuracy and run time with OctoMaps. Experimental results confirm that our method successfully works with a sequence of observations. Our method is slower than OctoMaps but generates more accurate occupancy maps as well as surface meshes without additional cost of computation.

1 Introduction

Understanding environments is one of the fundamental problems for intelligent robots to perform dependable tasks. For example, if an environmental map is erroneous, a mobile robot can be hit by obstacles or lose track of its poses during navigation. Also, with inaccurate object shape estimation, a robotic arm can fail to grasp unknown objects during manipulation. Therefore, accurate and reliable robotic maps are essential for robots to interact with the environments.

For localization and obstacle avoidance, occupancy maps such as occupancy grid maps [Moravec and Elfes, 1985] and OctoMaps [Wurm *et al.*, 2010] have been widely used with range sensors such as laser scanners and depth cameras. They are highly accurate with dense point clouds and even fast because each grid cell is updated for each single observation based on the independent cell assumption. However, given relatively sparse point clouds they produce poor results containing holes and discontinuities due to the strict assumption. On the other hand, surface reconstruction such as triangulation [Marton *et al.*, 2009] and implicit surfaces [Kazhdan *et al.*, 2006] has been applied to construct 3D models of unknown objects or to visualize the environments.

Recently, Gaussian processes, one of the state-of-the-art machine learning techniques for regression and classification, have been applied to build accurate and reliable robotic maps. Gaussian process occupancy maps [O'Callaghan and Ramos, 2012] generate continuous occupancy maps with uncertainties which can be further used for path planning and exploration. In addition, Gaussian process implicit surfaces [Williams and Fitzgibbon, 2006] have been applied to unknown object grasping [Dragiev *et al.*, 2011], underwater ship hull inspection [Hollinger *et al.*, 2013] and environmental mapping [Smith *et al.*, 2010]. However, since Gaussian processes suffer from high computational complexity, for large-scale environmental mapping we proposed some approximation methods [Kim and Kim, 2012; Kim and Kim, 2013a; Kim and Kim, 2013c] and further suggested a unified framework for constructing both occupancy maps and surface meshes with single Gaussian process prediction [Kim and Kim, 2013b].

However, the previous methods of robotic mapping with Gaussian processes are all batch processes which accumulate sequential observations and predict the final maps at once. That approach not only increases the number of training data, but also cannot deal with new observations to update the final maps. Therefore, in this paper we propose to recursively update Gaussian process maps using Bayesian Committee Machines [Tresp, 2000]. The authors found a similar method using Bayesian Committee Machines [Jadidi *et al.*, 2014] from the reviewer's comments. However, using Bayesian Committee Machines with Gaussian process mapping is originally proposed in our previous work for reducing the computational complexity [Kim and Kim, 2011]. More importantly, in this paper we focus on what assumption should be made to incorporate Bayesian Committee Machines with Gaussian process mapping and compare the results of different assumptions.

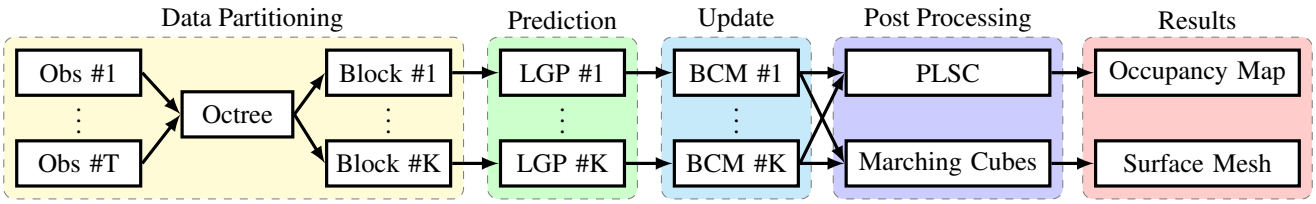
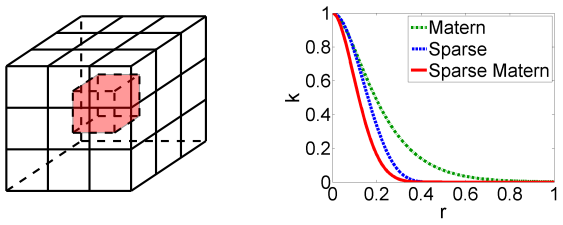


Figure 1: Flow chart of our unified framework for building and recursively updating occupancy maps and surface meshes. Obs, LGP, BCM and PLSC stand for observations, local Gaussian process, Bayesian Committee Machine and Probabilistic Least Square Classification, while T and K denote the total time steps and the number of grid blocks in the octree, respectively.



(a) Extended Block (b) Covariance, $k(r)$

Figure 2: Data partitioning with an octree. (a) All observations in the extended block are considered to predict the map of the red block in the center. (b) Comparison of covariance functions. Note that the sparse covariance function vanishes after some distance threshold.

2 Overview of Our Method

Fig. 1 shows the flow chart of our unified framework for occupancy mapping and surface reconstruction in four steps. First, we partition the world with grid blocks of an octree and divide training data (observations) and test data (query positions) into manageable subsets. Second, with the independent block assumption, we predict each block map individually using Gaussian process regression. Third, based on the static world assumption we update each block map recursively given a sequence of observations using Bayesian Committee Machines. Finally, we collect the results from all blocks and build both occupancy maps using Probabilistic Least Square Classification and surface meshes using Marching Cubes. Each step will be explained in more detail in the following sections.

3 Data Partitioning with Octrees

We first begin with data partitioning. As we will discuss later, we apply Gaussian processes to robotic mapping. However, due to the cubic computational complexity of Gaussian processes it is not directly applicable to large-scale environmental mapping. Therefore, we partition both training and test data to reduce the time complexity and enhance the scalability of Gaussian processes.

3.1 Spatial Partitioning with Octrees

An octree is an efficient data structure for partitioning a three dimensional space with a recursive tree structure of eight child nodes. As shown in Fig. 1, given a set of observations (training data) at time t we divide them into grid blocks using an octree. Note that the octree partitions the query positions (test data) as well.

Given partitioned training and test data, the simplest mapping approach would be to build each block map with its own observations. However, this may cause the discontinuity problem on the boundaries because different training data will be used for neighboring blocks. Instead, we make the training data change smoothly block by block.

3.2 Independent Block Assumption

To avoid the discontinuity problem on the boundaries, we predict each block map with the observations in the extended block (itself and neighboring blocks) as shown in Fig. 2(a).

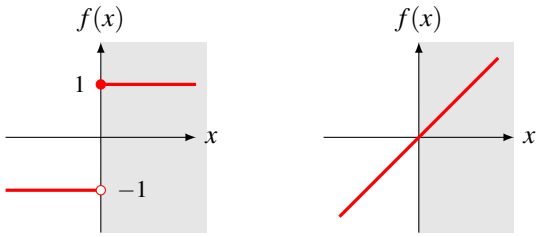
Mathematically, this can be done by assuming that the map of a grid block is independent of others given the observations in the extended block, which we call the *independent block assumption*. Thus, the whole mapping problem is factorized by sub-problems of mapping individual blocks as

$$p(\mathbf{f}_* \mid D) = \prod_{i=1}^{K} p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}^i) \ , \tag{1}$$

where $\mathbf{f}_*$ and $\mathbf{f}_*^i$ denote the whole map and the i-th block map, respectively. Similarly, D denote the whole observations of K blocks, while $\bar{\mathbf{y}}^i$ represents only those in the i-th extended block. Note that the observations in each block, $\mathbf{y}^i$ are both mutually exclusive and collectively exhaustive so that $\mathbf{y}^i \cap \mathbf{y}^j = \emptyset$, $i \neq j$ and $D = \cup_{i=1}^{K} \mathbf{y}^i$, but the extended observations, $\bar{\mathbf{y}}^i$ are overlapping each other.

This assumption is valid in that in Gaussian processes the covariance between the output values of two test positions drops exponentially as the distance grows and even vanishes at some distance when we use the sparse covariance function as shown in Fig. 2(b). More details about covariance functions will be followed in the next section.

Therefore, our method is a local approximation of Gaussian process mapping in that we individually estimate each



(a) Indicator Function (b) Signed Distance Function

Figure 3: Target functions in one dimensional space. Here we assume that it is occupied at $x \in [0, \infty)$.

block map with its own extended observations and move on the next block. This procedure looks quite similar to the overlapping sliding window approach in computer vision.

4 Gaussian Process Mapping

Now, let us focus on predicting each block map with its extended observations using a local Gaussian process. Thus for clarity, we omit the superscript i of the block index here and consider the extended observations as the whole training data.

4.1 Target Function

Before applying Gaussian processes, we need to define the target function or the output values of observations. Fig. 3 describes two target functions. The indicator function maps occupied/empty points to $+1/-1$, while the signed distance function maps every point on the surface to zero and other points inside/outside of the surface to positive/negative distances to the surface.

The indicator function was adopted in Gaussian process occupancy maps [O'Callaghan and Ramos, 2012; Kim and Kim, 2012; Kim and Kim, 2013a] by following the concept of occupancy. However, in that case it is difficult to interpret the latent function values because it can be bigger than 1 or less than -1. Instead, we adopt the signed distance function as a target function as we did in [Kim and Kim, 2013b] By doing this, we can build two kinds of map representations, occupancy maps and surface meshes with single Gaussian process prediction. The post processing procedures for surface reconstruction and occupancy mapping will be explained in Section 6.

Consequently, given a robot pose and hit points we associate the hit points with zero values. But with only zero values, we cannot predict the signed distance field in a three dimensional space. Thus, we create empty points just before the hit points along the directions from the robot position to the hit points and associate them with negative distances.

4.2 Gaussian Process Regression

A Gaussian process regression is a Bayesian non-parametric approach to regression. It is also known as an extension of a multivariate Gaussian distribution to infinite dimensions, in other words a distribution over functions,

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

where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ denote the mean and covariance functions, respectively.

Since observations in reality are not perfect, we assume that each observation is corrupted with additive Gaussian noise,

$$y = f(\mathbf{x}) + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,\, \sigma_n^2)\ , \tag{3}$$

where σ_n^2 denotes the noise variance.

Given N noisy observations, $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = \{\mathbf{X}, \mathbf{y}\}$ and M test positions, $\mathbf{X}_* = \{\mathbf{x}_{*j}\}_{j=1}^M$, Gaussian processes assume a joint Gaussian distribution over the training outputs, $\mathbf{y}$ and the test outputs, $\mathbf{f}_* = \{f_{*j}\}_{j=1}^M$,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^{\mathrm{T}} & \mathbf{K}_{**} \end{bmatrix}\right), \tag{4}$$

where $\mathbf{K} \in \mathbb{R}^{N \times N}$, $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{K}_* \in \mathbb{R}^{N \times M}$, $[\mathbf{K}_*]_{ij} = k(\mathbf{x}_i, \mathbf{x}_{*j})$, and $\mathbf{K}_{**} \in \mathbb{R}^{M \times M}$, $[\mathbf{K}_{**}]_{ij} = k(\mathbf{x}_{*i}, \mathbf{x}_{*j})$. Note that the zero mean function is chosen here. (see [Rasmussen and Williams, 2006] for details).

Therefore, we can infer the predictive distribution of the test outputs which is also Gaussian,

$$\mathbf{f}_* \mid \mathbf{y} \sim \mathcal{N}(\mu_*,\, \Sigma_*), \tag{5}$$

where

$$\mu_* = \mathbf{K}_*^{\mathrm{T}} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}\ , \tag{6}$$

$$\Sigma_* = \mathbf{K}_{**} - \mathbf{K}_*^{\mathrm{T}} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}_*\ , \tag{7}$$

where μ_* and Σ_* denote the mean vector and the covariance matrix of the predictive Gaussian distribution.

By considering each test position independent, we can infer the mean, μ_* and variance σ_*^2 of a test output, f_* at a test position $\mathbf{x}_*$,

$$\mu_* = \mathbf{k}_*^{\mathrm{T}} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}\ , \tag{8}$$

$$\sigma_*^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{k}_*\ , \tag{9}$$

where $\mathbf{k}_* \in \mathbb{R}^N$, $[\mathbf{k}_*]_i = k(\mathbf{x}_i, \mathbf{x}_*)$, and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$.

In this paper, we call the first prediction *dependent prediction* and the second one *independent prediction*. Table 1 summaries the computational complexities of a Gaussian process and local Gaussian processes. Recognize that dependent prediction has higher computational complexity than independent one, and both of them are dramatically reduced by partitioning both training and test data. This theoretic complexity will be confirmed in the experimental results in Section 7.

4.3 Covariance Functions

A common choice for covariance functions is the squared exponential covariance function. However, it produces too smooth prediction results as it is infinitely differentiable.

Table 1: Computational complexities of Gaussian processes, where N and M denote the numbers of training and test data, respectively. For local Gaussian processes we assume that the training and test data are equally divided into K blocks.

Pred.	Gaussian Process	Local Gaussian Processes
Dep.	$O(N^3+N^2M+NM^2)$	$O\left((N^3+N^2M+NM^2)/K^2\right)$
Ind.	$O(N^3+N^2M)$	$O\left((N^3+N^2M)/K^2\right)$

Mátern Covariance Function

For dealing with sharp changes in object shapes, we use the Mátern covariance function with $\nu = 3/2$,

$$k(r) = \sigma_f^2 \left(1 + \frac{\sqrt{3}r}{l_m}\right) \exp\left(\frac{-\sqrt{3}r}{l_m}\right), \tag{10}$$

where $r = |\mathbf{x} - \mathbf{x}'|$, and the hyperparameters σ_f^2 and $l_m > 0$ are called the signal variance and the characteristic length-scale, respectively.

Sparse Covariance Function

Note that even for a large distance, the Mátern covariance function still has some positive value. Thus, for assuming independent blocks and applying local Gaussian processes to each block, we also use the sparse covariance function,

$$k(s) = \begin{cases} \left(\frac{2+cos(2\pi s)}{3}(1-s) + \frac{1}{2\pi}\sin(2\pi s)\right), \text{ if } s < 1\ , \\ 0 \qquad\qquad\qquad\qquad\qquad\quad , \text{ if } s \geq 1\ , \end{cases} \tag{11}$$

where the scaled distance $s = r/l_s$, and the characteristic length-scale $l_s > 0$. Recognize that the correlation between two points is now zero when their distance is greater than the threshold l_s. In order to to take advantages of both covariance functions, we use their product as a covariance function, because the positive semidefinite kernels are closed under additions and multiplications. Fig. 2(b) compares their behaviors with hyperparameters set to $\sigma_f = 1$, $l_m = 0.2$, $l_s = 0.5$.

4.4 Training Hyperparameters

In Gaussian processes, we train hyperparameters by maximizing the log marginal likelihood (or evidence),

$$\log p(\mathbf{y} \mid \mathbf{X}, \Theta) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}_n^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_n| - \frac{N}{2}\log 2\pi, \tag{12}$$

where $\mathbf{K}_n = \mathbf{K} + \sigma_n^2\mathbf{I}$ and the hyperparameters $\Theta = (\sigma_f, l_m, l_s)$ in this paper.

However, in our case we partitioned the world into blocks with an octree and applied local Gaussian processes individually. Thus, we train the hyperparameters by maximizing the sum of the log marginal likelihoods of each block,

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{K} \log p(\mathbf{y}^i \mid \mathbf{X}^i, \Theta)\ . \tag{13}$$

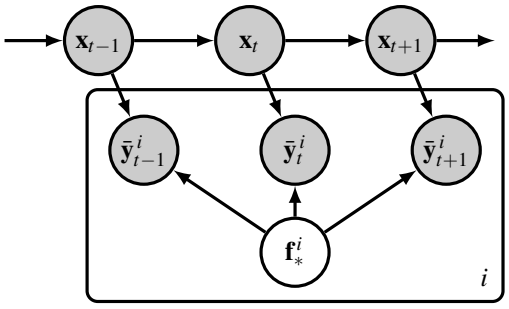


Figure 4: Graphical model for the static block assumption. $\mathbf{x}_t$ denotes the robot pose at time t, $\bar{\mathbf{y}}^i_t$ represents the extended observations for the i-th block map $\mathbf{f}^i_*$. Note that gray and white circles indicate known and unknown variables, respectively.

Note that we do not adopt hyperparameters locally to each block because it may cause the discontinuity problem on the boundaries.

5 Recursive Updates with BCM

In this section, we extend our Gaussian process mapping method to deal with sequential observations. First, we assume the static block assumption which combines the independent block assumption in the previous section and the static world assumption. Then, we propose recursive updates for Gaussian process mapping using Bayesian Committee Machines.

5.1 Static Block Assumption

Static World Assumption

Usually when we build maps for static environments, we adopt the so-called *static world assumption*, where the current observation, $\mathbf{y}_t$ and the observations up to now, $\mathbf{y}_{1:t-1}$ are assume to be independent given the whole map, $\mathbf{f}_*$,

$$p(\mathbf{y}_t, \mid \mathbf{f}_*, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t, \mid \mathbf{f}_*)\ . \tag{14}$$

Static Block Assumption

By combining the independent block assumption in Eq. 1 and the static world assumption in Eq. 14, we assume that for each block the current extended observations, $\bar{\mathbf{y}}^i_t$ and the extended observations up to now, $\bar{\mathbf{y}}^i_{1:t-1}$ are independent given the block map, $\mathbf{f}^i_*$, which we call the *static block assumption*,

$$p(\bar{\mathbf{y}}^i_t \mid \mathbf{f}^i_*, \bar{\mathbf{y}}^i_{1:t-1}) = p(\bar{\mathbf{y}}^i_t \mid \mathbf{f}^i_*)\ . \tag{15}$$

Technically, this assumption is not correct because the extended observations cover a larger space than the block map. However, it is inevitable to make the observations in each extended block independent for recursive updates because predictions are made based on the extended observations. We find that this assumption works well with real data in Section 7. Fig. 4 depicts the graphical model for the static block assumption.

5.2 Recursive Block Updates

From the independent block assumption, we predict each block map individually. Moreover, from Bayes' Theorem and the static block assumption in Eq. 15, each block map can be recursively updated as,

$$\begin{aligned} p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_{1:t}^i) &\propto p(\bar{\mathbf{y}}_t^i \mid \mathbf{f}_*^i, \bar{\mathbf{y}}_{1:t-1}^i)\, p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_{1:t-1}^i) && \text{(Bayes' Rule)} \\ &= p(\bar{\mathbf{y}}_t^i \mid \mathbf{f}_*^i)\, p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_{1:t-1}^i) && \text{(Static Block)} \\ &\propto \frac{p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_t^i)\, p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_{1:t-1}^i)}{p(\mathbf{f}_*^i)} \ . && \text{(Bayes' Rule)} \quad (16) \end{aligned}$$

Note that the prior distribution of the i-th block map, $p(\mathbf{f}_*^i)$ and the predictive distribution, $p(\mathbf{f}_*^i \mid \bar{\mathbf{y}}_t^i)$ at time t are multivariate Gaussian distributions from Eq. 4 and 5. Therefore, the recursive update of each block map in Eq. 16 can be represented in terms of the mean vector and the covariance matrix of each block map,

$$\xi_{1:t}^i = \xi_{1:t-1}^i + \xi_t^i \ , \tag{17}$$

$$\Lambda_{1:t}^i = \Lambda_{1:t-1}^i + \Lambda_t^i - \Lambda_0^i \ , \tag{18}$$

where $\xi = \Sigma^{-1}\mu$ and $\Lambda = \Sigma^{-1}$ denote the information vector and the information matrix, respectively. This recursive update form with independent subsets of training data is called a Bayesian Committee Machine [Tresp, 2000] in the machine learning community.

5.3 Static Cell Assumption

In the case of building large-scale environmental maps with a high resolution, the space complexity can be an issue as well as the time complexity in Table 1. Suppose that we have K blocks and each block consists of m grid test positions per each axis. Then, the space complexity of maintaining the mean vectors and the covariance matrices in a three dimensional space is $O(Km^3)$, while it is reduced to $O(Km)$ for maintaining the means and variances. Note that the number of test positions per each block is the same as $M = m^3$ for both cases.

In order to maintain the variances instead of the covariance matrix, we propose a simplified version of the static block assumption called the *static cell assumption*,

$$p(\bar{\mathbf{y}}_t^i \mid f_*^{i,j}, \bar{\mathbf{y}}_{1:t-1}^i) = p(\bar{\mathbf{y}}_t^i \mid f_*^{i,j}) \ , \tag{19}$$

where $f_*^{i,j}$ is now the j-th cell of the i-th block map.

This seems to be a very strict assumption compared with the static block assumption. However, it is still less strict than the independent cell assumption adopted in occupancy grid maps and OctoMaps,

$$p(y_t^{i,j} \mid f_*^{i,j}, y_{1:t-1}^{i,j}) = p(y_t^{i,j} \mid f_*^{i,j}) \ , \tag{20}$$

where $y_t^{i,j}$ represents a single ray observation which passes through or hits back at the j-th cell of the i-the block map. This little difference in the independence assumption makes huge difference in the final map accuracy as you will see in Section 7.

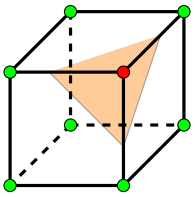


Figure 5: Concept of Marching Cubes. Here, the green vertices have positive distances, while the red vertex has a negative distance. The zero-valued points of the intersecting isosurface in red are determined by interpolating the signed distances on vertices.

5.4 Recursive Cell Updates

Now, with the static cell assumption in Eq. 19, each cell is updated recursively as,

$$\xi_{1:t}^{i,j} = \xi_{1:t-1}^i + \xi_t^{i,j} \ , \tag{21}$$

$$\lambda_{1:t}^{i,j} = \lambda_{1:t-1}^{i,j} + \lambda_t^{i,j} - \lambda_0^{i,j} \ , \tag{22}$$

where $\xi = \sigma^{-2}\mu$ and $\lambda = \sigma^2$. We will demonstrate the recursive block updates and recursive cell updates in Section 7 and show that the latter dramatically reduces run times with neglectable accuracy loss.

6 Post Processing

Now, we are given the means and variances at grid test positions which is predicted with local Gaussian processes and updated with Bayesian Committee Machines. In order to convert this result to two kinds of map representations, an occupancy map and a surface mesh, we apply Probabilistic Least Square Classification and Marching Cubes, respectively.

6.1 Probabilistic Least Square Classification

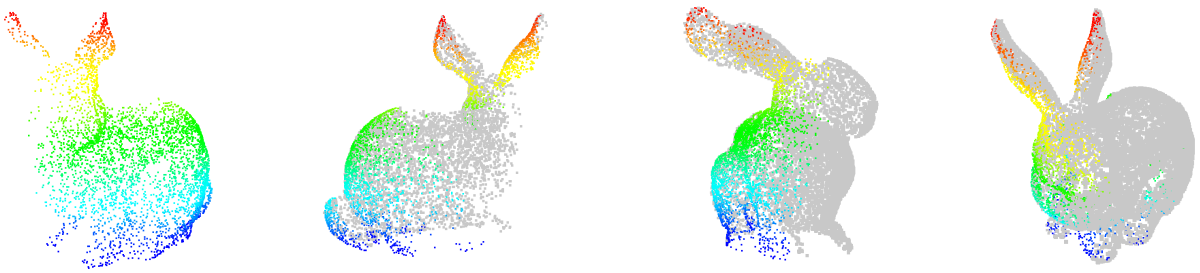
Occupancy mapping is a binary classification problem to predict the binary class probability of each test position being occupied or not. Thus, we apply Probabilistic Least Square Classification [Platt, 2000],

$$p(m_*^{i,j} = 1|D) = \Phi\left(\frac{\alpha\mu_{1:t}^{i,j} + \beta}{\sqrt{1 + \alpha^2\sigma_{1:t}^{2\,i,j}}}\right), \tag{23}$$

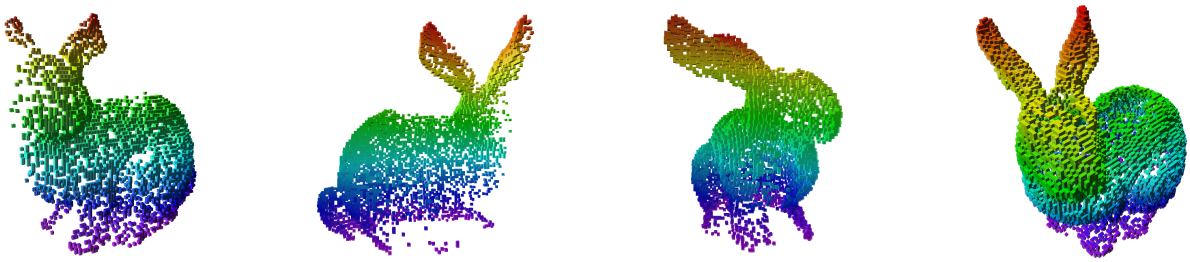
where $m_*^{i,j}$ denotes the binary random variable (1: occupied, -1: empty) for the j-th cell in the i-th block and Φ represents the cumulative Gaussian density function. The parameters α and β are optimized by maximizing the accuracy of the final occupancy maps.

6.2 Marching Cubes

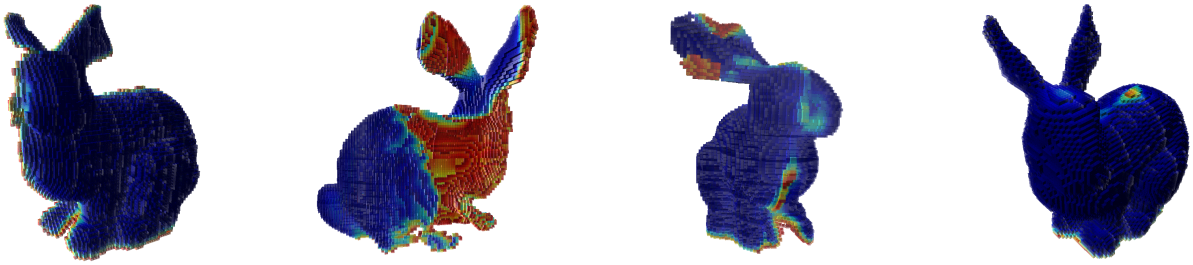
The predicted mean values at grid test positions can be viewed as a scalar field in a three dimensional space. Thus, we can reconstruct a surface by extracting the zero-valued



(a) Hit Points: Four sequential observations. New observations are color coded by height, while old ones are gray.



(b) OctoMaps: Occupied cells are color coded by height. Recognize the holes and discontinuities.



(c) GPMaps: Occupied cells are color coded by uncertainty. (red/blue color for high/low uncertainty)

Figure 6: Incremental updates of OctoMaps and our GPMaps given a sequence of observations. Recognize that GPMaps are more accurate than OctoMaps, and in GPMaps the uncertainties on the boundaries of observations and inside of the object are relatively higher.

iso-surface because by definition of the signed distance function, the outputs of the points on the surface are zero.

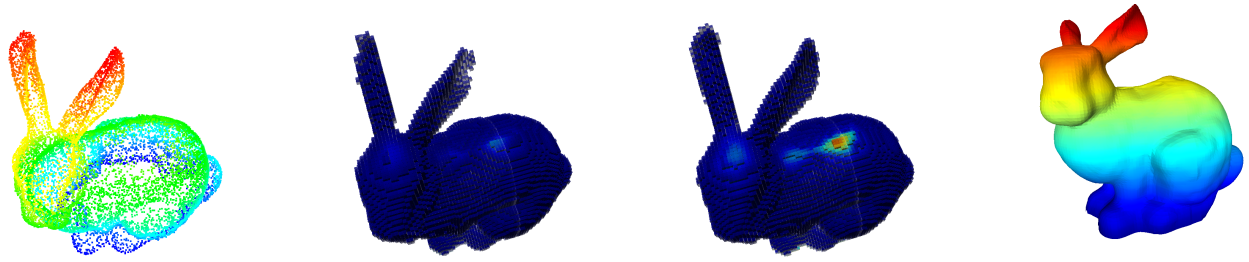
For iso-surface extraction, we apply Marching Cubes [Lorensen and Cline, 1987]. As shown in Fig. 5, given the predicted signed distances at eight cell center positions, we construct a cube and search for the intersecting surface patch by interpolating the signed distances of vertices, and move on to the next cube. Note that the iso-surfaces of occupancy grid maps or OctoMaps would be severely cracked because of their sparseness. On the other hand, our method predicts continuous signed distance fields and thus generates smooth surfaces, which is one of the benefits of using Gaussian processes.

7 Experimental Results

In this section, we demonstrate our method with a real dataset and compare the accuracy and run time with OctoMaps [Wurm *et al.*, 2010]. Our method was implemented in C++ and the source code is open to the public [Kim, 2014]. The experiments was performed on a laptop computer with an Intel Core i7-2630QM 2.0 GHz CPU and 8 GB RAM.

7.1 Sequential Dataset

For a real dataset, we use the Stanford Bunny dataset [Turk and Levoy, 1994]. It consists of four sequential observations (rotating around the z-axis) and their corresponding sensor poses in the global coordinates as shown in Fig. 6(a). Recognize that no hit points are acquired from the top of the bunny's head and back. Originally, the Stanford Bunny dataset was made for surface reconstruction so that a high resolution scanner (Cyberware 3030, FOV: 11.82°, 512×512 pixels) was used, and very dense point clouds were obtained about 74.0 cm away from the model ($16\times12\times15$ cm^3). To simulate conventional situations of robotic mapping with a laser scanner of a 0.1° angular resolution, we use randomly sampled 10%



(a) All Hit Points (b) Recursive Block Updates (c) Recursive Cell Updates (d) Surface Mesh

Figure 7: Comparison between recursive block updates and recursive cell updates. (a) All hit points, (B) Occupancy maps built with recursive block updates, (C) Occupancy maps built with recursive cell updates, and (D) Surface mesh reconstructed with recursive cell updates.

of the hit points in this paper. The numbers of sampled hit points are (1) 4,026 (2) 3,038 (3) 4,026 and (4) 3,171.

7.2 Accuracy Comparison

Given the same sequence of observations, we incrementally built OctoMaps and GPMaps (our method) with recursive cell updates. The map resolution was set to 0.2cm for both cases. For GPMaps, we trained hyperparameters to maximize the sum of the marginal likelihoods of all the blocks. (σ_f = 1.078e-1 , l_s = 6.704e-2, l_m = 5.894e-2, and σ_n = 2.616e-4) Based on them, we set the block size to 2 cm. Note that the bigger the block size is, the more training and test data is given to each block. Therefore, the block size should be carefully selected for time and space complexity.

The OctoMaps and GPMaps for sequential observations are shown in Fig. 6(b) and 6(c). Recognize the sparseness of OctoMaps compared with the dense and smooth GPMaps. In addition to the enhanced accuracy, GPMaps also provide map uncertainties (variances) with which the occupied cells are color coded. Here, uncertain cells ($\sigma_{1:t}^{i,j\,2} >$ 4e-4) were removed.

In the first view of Fig. 6(c), the boundary has relatively high uncertainty due to the lack of observations. In the second view, we can see that the inside of the object is very uncertain. This is because our training data are hit and empty points, and there exist no observations behind the surface. Thus, the inside test positions were predicted as occupied with high uncertainties. In the third view, since there exist almost no overlapping areas between the front and rear views, the occupied cells on the border line, which goes from the nose to the front feet, have high uncertainties. Finally, with the four observations, the occupancy maps were predicted with very low uncertainty except the top of the head and back. This is because originally no hit points were obtained from those areas.

We also compares the accuracy of two recursive update schemes in Fig. 7. Recognize that the occupancy maps predicted with the recursive block updates have lower uncertainties on the top of the head and back, but lost some details on the tips of both ears. Since our GPMaps predict continuous signed distance fields, we can reconstruct the iso-surface meshes with zero distances as shown in Fig. 7(d).

Table 2: Comparison of run times. Recognize that OctoMaps are much faster than GPMaps (our method), and the independent prediction and update of GPMaps is much faster than the dependent ones.

Seq	Octo Maps (sec)	GPMaps (dep.)		GPMaps (indep.)	
		Pred. (min)	Update (min)	Pred. (min)	Update (sec)
#1	1.186	11.50	10.10	1.648	0.078
#2	0.764	9.55	9.14	1.249	0.062
#3	1.061	11.31	11.96	1.688	0.062
#4	1.014	9.38	9.64	1.282	0.047
Total	4.025	41.74	40.84	5.867	0.250

7.3 Speed Comparison

Table 2 compares the run times between different mapping methods. As expected, OctoMaps are much faster than GPMaps. It is because each cell is updated for each single observation based on the independent cell assumption. The independent prediction and update for GPMaps dramatically reduces the run times, which confirms the theoretical time complex in Table 1.

8 Conclusions

In this paper, we proposed a new method to build and recursively update occupancy maps and surface meshes given a sequence of observations using Gaussian processes and Bayesian Committee Machines. With the independent block assumption, we partitioned both training and test data into grid blocks using an octree and applied local Gaussian processes to each block. Moreover, with the static block/cell

assumption, we recursively updated the maps for each block/cell. With those assumption, the time and space complexity of our method was dramatically reduced. Experimental results showed that our method successfully dealt with sequential observations. Our method was slower than OctoMap but generated more accurate occupancy maps with map uncertainties. Moreover, our method also produced surface meshes without additional cost of computation.

However, our recursive update scheme, Bayesian Committee Machines, is based on the static world assumption. Therefore, our method is not suitable for dynamic environments. To address this problem, we need to consider the transitional probability of the map as time passes, which is included in our future work.

References

[Dragiev *et al.*, 2011] S. Dragiev, M. Toussaint, and M. Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2845–2850, 2011.

[Hollinger *et al.*, 2013] G.A. Hollinger, B. Englot, F. Hover, U. Mitra, and G.S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *Int'l Journal of Robotics Research*, 32(1):3–18, 2013.

[Jadidi *et al.*, 2014] Maani Ghaffari Jadidi, Jaime Valls Miró, Rafael Valencia, and Juan Andrade-Cetto. Exploration on continuous Gaussian process frontier maps. In *Proceedings of the the IEEE International Conference on Robotics and Automation*, pages 6077–6082. IEEE, 2014.

[Kazhdan *et al.*, 2006] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of Eurographics Symposium on Geometry processing*, pages 61–70, 2006.

[Kim and Kim, 2011] Soohwan Kim and Jonghyuk Kim. Towards large-scale occupancy map building using Dirichlet and Gaussian processes. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2011.

[Kim and Kim, 2012] Soohwan Kim and Jonghyuk Kim. Building occupancy maps with a mixture of Gaussian processes. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4756–4761, 2012.

[Kim and Kim, 2013a] Soohwan Kim and Jonghyuk Kim. Continuous occupancy maps using overlapping local Gaussian processes. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4709–4714. IEEE, 2013.

[Kim and Kim, 2013b] Soohwan Kim and Jonghyuk Kim. GPmap: A unified framework for robotic mapping based on sparse Gaussian processes. In *Proceedings of International Conference on Field and Service Robot*, 2013.

[Kim and Kim, 2013c] Soohwan Kim and Jonghyuk Kim. Occupancy mapping and surface reconstruction using local gaussian processes with kinect sensors. *Cybernetics, IEEE Transactions on*, 43(5):1335–1346, 2013.

[Kim, 2014] Soohwan Kim. Gpmap++: Gaussian processes for robotic mapping in C++. `https://github.com/kimsoohwan/GPMap`, 2014.

[Lorensen and Cline, 1987] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 163–169, 1987.

[Marton *et al.*, 2009] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3218–3223. IEEE, 2009.

[Moravec and Elfes, 1985] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121, 1985.

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

[Platt, 2000] John C. Platt. Probabilities for SV Machines. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

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

[Smith *et al.*, 2010] Mike Smith, Ingmar Posner, and Paul Newman. Efficient non-parametric surface representations using active sampling for push broom laser data. In *Proceedings of Robotics: Science and Systems*, 2010.

[Tresp, 2000] V. Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[Turk and Levoy, 1994] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, pages 311–318. ACM, 1994.

[Williams and Fitzgibbon, 2006] O. Williams and A. Fitzgibbon. Gaussian process implicit surfaces. In *Proceedings of the Workshop on Gaussian Processes in Practice*, 2006.

[Wurm *et al.*, 2010] Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA workshop on best practice in 3D perception and modeling for mobile manipulation*, 2010.