Hierarchical Gaussian Processes for Robust and Accurate Map Building

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

Abstract
This paper proposes a new method for building occupancy maps and surface meshes using hierarchical Gaussian processes. Previously, we found that a Gaussian process, one of the state-of-the-art machine learning techniques for regression and classification, could serve a unified framework for occupancy mapping and surface reconstruction. Particularly, due to its high computational complexity, we partitioned both training and test data into manageable subsets and applied local Gaussian processes. However, when the local regions do not include any observations, this approach makes no predictions and thus generates blanks in the map. Therefore, in order to deal with missing data and make it more robust, we combine a global Gaussian process with local Gaussian processes. The global Gaussian process covers the overall trend of the whole observations, while the local Gaussian processes adapt to precise local observations. We demonstrate our method with a real dataset and compare accuracy and speed with OctoMaps and our previous method, GPmaps. Experimental results show that our method is relatively slow than the previous work but produces more robust and accurate robotic maps.

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 intelligent robots to interact with the environment.

For localization and obstacle avoidance, occupancy maps such as occupancy grid maps [Moravec and Elfes, 1985] and OctoMaps [Hornung et al., 2013] 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 every 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 methods such as triangulation [Marton et al., 2009] and implicit surfaces [Kazhdan et al., 2006] have been applied to construct 3D models of unknown objects or to visualize 3D 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. Gaussian process implicit surfaces 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, we proposed global and local approximations [Kim and Kim, 2011; Kim and Kim, 2012; Kim and Kim, 2013a; Kim and Kim, 2013c] for large-scale environmental mapping and further suggested a unified framework for constructing both occupancy maps and surface meshes with single Gaussian process prediction [Kim and Kim, 2013b].

However, because local Gaussian processes only focus on the local regions, they make no or poor predictions when there exist no or less observations in those regions. This problem does not arise when a single and global Gaussian process is applied, because it generates continuous and smooth predictions. Therefore, in order to build more robust and accurate robotic maps, we propose hierarchical Gaussian processes which combines a global Gaussian process with local Gaussian processes. Particularly, we impose a Gaussian process prior on the mean function for sampled global training data. We demonstrate our method with a real dataset and compare accuracy and speed with previous work. The experi-
mental results show that our method is relatively slow than the previous work but produces more accurate and robust robotic maps.

The rest of the paper is structured as follows. Section 2 describes the overview of our method. The procedures of our method, data pre-processing, hierarchical Gaussian processes, and post-processing, are explained in Section 3, 4, and 5, respectively. Experimental results and comparison with previous work will be described in Section 6. We conclude the paper with future work in Section 7.

2 Overview of Our Method

Fig. 1 shows the flow chart of our unified framework for occupancy mapping and surface reconstruction in three steps. First, we prepare training and test data. For local Gaussian processes, we partition the world with an octree into manageable subsets of training data (observations) and test data (query positions). On the other hand, we sample some global training data from the overall observations. Second, we predict each block map individually using hierarchical Gaussian processes which combine both global and local Gaussian processes. Finally, we collect the results from all blocks and build 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 Pre-Processing

We first begin with data pre-processing. This step assigns output values to hit points and surface normals to make training data of input and output pairs. Then, we prepare two types of training data, global and local, for each type of Gaussian process regression. The global training data conveys general trends over the whole space, while the local training data provides precise information about the local regions.

3.1 Target Function

In order to build training data, we need to define the target function or the output values of observations. Gaussian process occupancy maps [O’Callaghan and Ramos, 2012; Kim and Kim, 2012] applied the indicator function which maps occupied/empty points to $+1/−1$ based on the concept of occupancy. However, it is difficult to interpret the latent function values because they can be bigger than 1 or less than $−1$, and thus they are not occupancy values any longer. Instead, we adopt the signed distance function as a target function as shown in Fig. 2.

\[
f(x) = \begin{cases} 
-d, & \text{if } x \text{ is outside of the surface} \\
0, & \text{if } x \text{ is on the surface} \\
+d, & \text{if } x \text{ is inside of the surface}
\end{cases}
\]

where $d$ denotes the distance to the closest surface.

By doing this, we can build two kinds of map representations, occupancy maps and surface meshes with single Gaussian process prediction. This is because the result is a signed distance scalar field and zero-valued iso-surfaces can be easily extracted from it. The post processing procedures for occupancy mapping and surface reconstruction will be explained in Section 5.

Another benefit of the signed distance function is that it is continuous and its derivatives with respect to each axis are corresponding components of the negative surface normal vector,

\[
f(x) = −\hat{n} \cdot (x − x_0) ⇒ \nabla f = −\hat{n},
\]

where $\hat{n}$ denotes the unit surface normal vector at the tangen-
Suppose that in some extended block we have insufficient or no observations. Then, the prediction will be poor, or we cannot even predict the block map at all because we do not have any training data. In occupancy mapping or surface reconstruction, this causes erroneous or empty regions. Note that this is due to the local assumption; with single and global Gaussian process regression we get smooth and continuous prediction. However, we cannot use global Gaussian process regression directly due to the high computational complexity, and that is why we applied local Gaussian processes.

Therefore, we sample a manageable subset to represent the whole training data and combine global and local Gaussian processes, which will be explained in the next section. For sampling, we apply a voxel grid filter which partitions the space with an octree and approximates with their centroids. This is more efficient than random sampling because random sampling reflects the density of point clouds and thus we get more samples from dense areas, while a voxel grid filter summarizes the observations uniformly over the space.

4 Hierarchical Gaussian Process Regression

4.1 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. A Gaussian process prior on a function $f(x)$ is defined with a mean function $m(x)$ and a covariance function $k(x, x')$.

$$f(x) \sim GP(m(x), k(x, x')) .$$

Then, given $N$ observations $D = \{(x_i, f_i)\}_{i=1}^N = \{\textbf{X}, \textbf{f}\}$ and a test position $x_s$, the Gaussian process prior assumes a joint Gaussian distribution over the training outputs $\textbf{f}$ and the test output $f_s$,

$$
\begin{bmatrix}
\textbf{f} \\
f_s
\end{bmatrix} 
\sim \mathcal{N}
\left(
\begin{bmatrix}
m \\
m_s
\end{bmatrix},
\begin{bmatrix}
\textbf{K} & \textbf{k}_s \\
\textbf{k}_s^T & k_{ss}
\end{bmatrix}
\right),
$$

where $\textbf{K} \in \mathbb{R}^{N \times N}$, $[\textbf{K}]_{ij} = k(x_i, x_j)$, $\textbf{k}_s \in \mathbb{R}^N$, $[\textbf{k}_s]_j = k(x, x_j)$, $k_{ss} = k(x_s, x_s)$, $\textbf{m} \in \mathbb{R}^N$, $[\textbf{m}]_j = m(x_j)$, and $m_s = m(x_s)$. (see [Rasmussen and Williams, 2006] for details).
Since observations in reality are not perfect, we assume that each observation is corrupted with additive Gaussian noise,
\[ y = f(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2) \]
where \( \sigma^2 \) denotes the noise variance.

Then, given \( N \) noisy observations \( D = \{(x_i, y_i)\}_{i=1}^N = \{\mathbf{X}, \mathbf{y}\} \) and a test position \( \mathbf{x}_* \), the joint distribution over the training outputs \( \mathbf{y} \) and the test output \( f_* \) still remains as Gaussian,
\[
\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mathbf{m} \\ \mathbf{k}^T \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathbf{I} & \mathbf{k}^T \\ \mathbf{k} & k_{**} \end{bmatrix} \right).
\]

Therefore, we can infer the predictive distribution of the test output which is also Gaussian,
\[
f_* \mid \mathbf{y} \sim \mathcal{N} \left( \mu_*, \sigma_*^2 \right),
\]
where the mean and variance are
\[
\mu_* = \mathbf{m}_* + \mathbf{k}^T \left( \mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} (\mathbf{y} - \mathbf{m}), \quad \sigma_*^2 = k_{**} - \mathbf{k}^T \left( \mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{k}.
\]

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 \( K \) subsets. For the hierarchical Gaussian processes (glocal = global + local), \( G \) denotes the number of global training data.

<table>
<thead>
<tr>
<th>Gaussian Processes</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>( O(N^3 + NM^2) )</td>
</tr>
<tr>
<td>Local</td>
<td>( O((N^3 + N^2M)/K^2) )</td>
</tr>
<tr>
<td>Glocal</td>
<td>( O((N^3 + N^2M)/K^2 + G^3 + G^2(N + M)) )</td>
</tr>
</tbody>
</table>

On the other hand, Fig. 4(b) describes the graphical model of local Gaussian processes with \( k = 1, \ldots, K \), independent training and test subsets. Note that input and output pairs are omitted in the graphical model, and overlapping training subsets are ignored for brevity.

### 4.2 Mean Function: Gaussian Process

Usually, for the mean function a zero or constant mean function is used. However, since the mean function in the Gaussian process regression is also a function, we can impose another Gaussian process prior to the mean function again,
\[
m(x) \sim GP \left( 0, c(x, x') \right),
\]
where \( c(x, x') \) is another covariance function and the zero mean function is selected.

Therefore, given \( G \) noisy global observations \( D = \{(x_i, y_i)\}_{i=1}^G \),
\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^G = \{\tilde{X}, \tilde{y}\}$, the means in Eq. [1] are
\begin{align*}
m_x &= \mathbf{c}^T [\tilde{\mathbf{C}} + \alpha_s^2 \mathbf{I}]^{-1} \tilde{y}, \\
m &= \mathbf{c}^T [\tilde{\mathbf{C}} + \alpha_s^2 \mathbf{I}]^{-1} \tilde{y},
\end{align*}
where $\tilde{\mathbf{C}} \in \mathbb{R}^{G \times G}$, $|\tilde{\mathbf{C}}|_{ij} = \sigma(x_i, x_j)$, $\mathbf{c}_s \in \mathbb{R}^G$, $|\mathbf{c}_s|_i = \sigma(x_i, x)$, and $\mathbf{C} \in \mathbb{R}^{G \times N}$, $|\mathbf{C}|_{ij} = \sigma(x_i, x_j)$.

Recognize that even if there exist no observations in an extended block, we can calculate the mean value from Eq. [3] and [4]. Now, for the global Gaussian process, we have to invert a $G \times G$ matrix which costs $O(G^3)$ and to predict the means of $M$ test outputs, which costs $O(G^2 M)$. In addition, we have to predict the means of $N/K$ training outputs for $K$ blocks, which costs $O(G^2 N)$. Therefore, compared to the local Gaussian processes, our hierarchical Gaussian processes additionally cost $O(G^3 + G^2 (N + M))$.

Fig. 4(c) depicts the graphical model of hierarchical Gaussian processes or glocal (global + local) Gaussian processes. By incorporating global training data, the model has both global and local properties.

### 4.3 Covariance Functions

#### Rational Quadratic Covariance Function

In the hierarchical Gaussian process, the global Gaussian process requires to handle the overall data trend. Thus, we apply a rational quadratic covariance function for smooth prediction,
\[ k(r) = \sigma_r^2 \left( 1 + \frac{r^2}{2 \alpha l_r^2} \right)^{-\alpha}, \]
where $r = |x - x'|$, and the hyperparameters $\sigma_r$, $\alpha$ and $l_r$ are positive.

#### Matérn Covariance Function

In order to make the local Gaussian processes deal with sharp changes in object shapes, we use the Matérn covariance function with $\nu = 3/2$,
\[ k(r) = \sigma_m^2 \left( 1 + \frac{\sqrt{3}r}{l_m} \right) \exp \left( -\frac{\sqrt{3}r}{l_m} \right), \]
where the hyperparameters $\sigma_m$ and $l_m$ are also positive.

#### Sparse Covariance Function

Note that even for a large distance, the Matérn covariance function still has some positive value. Thus, to reduce the correlation between extended blocks, we also use the sparse covariance function,
\[ k(r) = \begin{cases} \sigma_r^2 \left( \frac{2 + \cos(2 \pi r / l_r)}{3} (1 - \frac{r}{l_r}) + \frac{2}{3} \sin \left( \frac{2 \pi r}{l_r} \right) \right), & \text{if } r < l_r, \\
0, & \text{if } r \geq l_r, \end{cases} \]
where the hyperparameters $\sigma_r$ and $l_r$ are also positive. Recognize that the covariance of two outputs is now zero when their distance is greater than the threshold $l_r$. In order to take advantages of both covariance functions, we use the product of the Matérn and sparse covariance functions (with a single signal variance $\sigma_f$) because positive semidefinite kernels are closed under additions and multiplications. [Taylor and Cristianini, 2004] Fig. 3(b) compares their behaviors with hyperparameters set to $\sigma_m = \sigma_s = 1$, $l_m = 0.2$, and $l_s = 0.5$.

### 4.4 Derivative Observations

For the global training data, we estimate surface normals and sample a small subset from them. This is because derivative observations are more efficient to represent global trends. Since a Gaussian process is a linear estimator, the derivative of a Gaussian process is still a Gaussian process. In fact, one can easily show that the partial derivative only affects the covariance function in Eq. [3] and [4]. Thus, the covariances between function values and partial derivatives, and between partial derivatives are well-defined as
\[ \text{cov} \left( \frac{\partial f}{\partial x_j}, \frac{\partial f}{\partial x_k} \right) = \frac{\partial^2 k(x, x')}{\partial x_j \partial x_k}, \quad \text{cov} \left( \frac{\partial f}{\partial x_j}, \frac{\partial f}{\partial x_k} \right) = \frac{\partial^2 k(x, x')}{{\partial x_j}^2} \frac{\partial^2 k(x, x')}{{\partial x_k}^2}. \]

(For details, refer to [Solak et al., 2003].)

However, the estimated surface normals are generally very noisy. Therefore, we assume a different level of measurement noise for the derivative observations,
\[ \frac{\partial y}{\partial x_j} = \frac{\partial f(x)}{\partial x_j} + \varepsilon_{dn}, \quad \varepsilon_{dn} \sim \mathcal{N}(0, \sigma^2_{dn}). \]

### 4.5 Training Hyperparameters

So far, several hyperparameters are introduced from the covariance functions and observation noise. We train them by maximizing the log marginal likelihood (or evidence),
\[ \log p(y \mid X, \Theta) = \frac{1}{2} y^T K^{-1} y - \frac{1}{2} \log |K_n| - \frac{N}{2} \log 2 \pi, \]
where $\mathbf{K}_n = \mathbf{K} + \sigma_s^2 \mathbf{I}$ and $\Theta$ is the set of hyperparameters. In order to make the problem simple, we train the hyperparameters of the global Gaussian process first and then train the hyperparameters of the local Gaussian processes based on them.

### 5 Post Processing

Now, we are given the means and variances at grid test positions. 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.

#### 5.1 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],

\[ p(m_s = 1|D) = \Phi \left( \frac{\alpha m_s + \beta}{\sqrt{1 + \alpha^2 \sigma_n^2}} \right), \tag{5} \]

where \( m_s \) denotes the binary random variable (1: occupied, -1: empty) for each test position.

5.2 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 iso-surface. This is because by definition of the signed distance function, the function values 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 using Gaussian processes.

6 Experimental Results

In this section, we demonstrate our method with a real dataset and compare the accuracy and run time with OctoMaps [Rusu et al., 2010] and our previous work, GPmaps [Kim and Kim, 2013b]. Our method was implemented in C++ and the source code is open to the public [Kim, 2015]. The experiments were performed on a laptop computer with an Intel Core i7 2.0 GHz CPU and 8 GB RAM.

6.1 Real Dataset with Missing Observations

For a real dataset, we use the Stanford Bunny dataset [Turk and Levoy, 1994]. 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 employed to obtain very dense point clouds about 74 cm away from the model (16×12×15 cm³).

To simulate conventional situations of robotic mapping with a laser scanner of a 0.1° angular resolution, we randomly sampled 10% of the hit points. Moreover, in order to simulate missing data we removed all the points inside of a sphere with a radius 0.035 m at (0.045, −0.015, 0.082). Totally 14,259 hit points were used for map building as shown in the middle of Fig. 6(a).

6.2 Data Pre-processing

For global training data, we estimated surface normals using the moving least square method [Rusu and Cousins, 2011] with a search radius 0.01 m and downsampled 178 data points in total using a voxel grid filter with an octree of a leaf node size 0.02 m. The left image of Fig. 6(a) shows the derivative observations for the global Gaussian process. Note that the surface normals are pointing outward of the surface and still no observations are acquired from the middle hole. On the other hand, we partitioned the world with an octree of a leaf node size 0.02 m and created 629 blocks in total. The middle image of Fig. 6(a) shows the function observations for the local Gaussian processes. The global hyperparameters were trained by maximizing the marginal distribution to \( l = 0.07820, \alpha = 0.11098, \sigma_f = 0.03511, \sigma_n = 0.00087, \) and \( \sigma_{nd} = 0.07653, \) while the local hyperparameters were trained by maximizing the marginal distributions of all grid blocks to \( l_s = 0.07082, l_M = 0.03267, \sigma_f = 0.08725, \) and \( \sigma_n = 0.00051. \)

6.3 Accuracy Comparison

Let us begin with the OctoMap in the right image of Fig. 6(a). It is accurate but sparse given the global and local training data. More importantly, it can not infer those areas with no observations because it only updates grid cells where each ray passes through or returns back. The resolution of occupancy maps is 2 cm.

As described in Section 3, we applied hierarchical Gaussian process regression. Using a global Gaussian process with the global training data, we predicted the mean values in Eq. 3. The left images of Fig. 6(b) and 6(c) show the occupancy map and surface mesh built with these global mean values only. Recognize that the maps are very smooth and the hole in the center are filled with high uncertainty. Local Gaussian processes with local training data also predict the mean and covariance values as in Eq. 1 and 2. The middle images of Fig. 6(b) and 6(c) describe the occupancy map and surface mesh built with local Gaussian processes only, which is exactly the same as our previous work, GPmaps [Kim and Kim, 2013b]. As expected, local Gaussian processes only deal with local observations and thus, the results are accurate but no prediction on the missing part. Finally, hierarchical Gaussian processes combine global and local Gaussian processes and produce the right images of Fig. 6(b) and 6(c). Note that we obtained more accurate results than the OctoMap. Even the hole in the center is filled with high uncertainty.
Figure 6: Experimental results with the real dataset.

Table 2: Run times for each map building method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>OctoMap</td>
<td>4.54 sec</td>
</tr>
<tr>
<td>GPmap</td>
<td>19.5 min</td>
</tr>
<tr>
<td>Our method</td>
<td>47.1 min</td>
</tr>
</tbody>
</table>

6.4 Time Comparison

Table 2 compares run times of various mapping methods. OctoMap is the fastest due to its simple update rule per observation. Our previous work, GPmap, only uses local Gaussian processes and runs relatively slow than OctoMap. However, its results are more dense and accurate than OctoMaps as shown in the middle images in Fig. 6(b) and 6(c). Finally, our new method using hierarchical Gaussian processes are slow but recovers the missing observations.

7 Conclusions

In this paper, we proposed a new method for occupancy mapping and surface reconstruction using hierarchical Gaussian processes. Previously, we focused on the computational complexity of Gaussian processes and proposed local Gaussian processes. However, this local approach produces erroneous or no predictions where insufficient or no observations are obtained in the local region. Therefore, we combined a global Gaussian process with local Gaussian processes to enhance the accuracy and robustness. To be more specific, we imposed a Gaussian process prior on the mean function for sampled global training data. Thus, the hierarchical Gaussian processes hold both global and local properties.

We demonstrated our method with a real dataset and compared accuracy and speed with the previous work, OctoMaps
and GPmaps. Our method run relatively slow compared to the previous work, but generated more accurate and robust maps given some missing observations. More comprehensive experiments with real datasets from large-scale environments are remained as future work. One limitation of this approach is that it is a batch procedure. In other words, we do not incrementally update the map, but collect sequential data and build the map at once. To deal with sequential observations, we plan to apply Bayesian recursive estimation in the near future.

References


