2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Continuous Occupancy Maps using Overlapping Local Gaussian Processes

Soohwan Kim[1] and Jonghyuk Kim[2]

***Abstract*— This paper presents an efficient method of building continuous occupancy maps using Gaussian processes for large-scale environments. Although Gaussian processes have been successfully applied to map building, the applications are limited to small-scale environments due to the high computational complexity. To improve the scalability, we adopt a divide and conquer strategy where data are partitioned into manageable size of clusters and local Gaussian processes are applied to each cluster. Particularly, we propose overlapping clusters to mitigate the discontinuity problem that predictions of local estimators do not match along the boundaries. The results are consistent and continuous occupancy voxel maps in a fully Bayesian framework. We evaluate our method with simulated data and compare map accuracy and computational time with previous work. We also demonstrate our method with real data acquired from a laser range finder.**

I. Introduction

3D robotic mapping has gained significant attention recently due to the wide availability of 3D data obtained from various sensors such as Velodyne LIDAR scanners and Microsoft Kinect sensors. Among various map representations, this paper focuses on occupancy maps which distinguish between empty and occupied areas and have great flexibility in representing unstructured and complex 3D environments as shown in Fig. 1.

Conventional occupancy grid maps [1] discretize the world into independent grid cells and update occupancy of each cell individually. This independence assumption makes the algorithm run fast and easy to implement, but also leads to sparse results because only those cells which laser beams, for example, pass through or reflected at are updated. Octomaps [2] elaborate this approach to 3D occupancy voxel maps by applying an efficient data structure, an octree.

Recently, Gaussian processes [3], a Bayesian nonparametric approach to regression and classification in machine learning, have been applied to map building in the robotics literature. Elevation maps for outdoor terrains [4], [5] have been estimated using Gaussian process regression, but they are basically 2.5D and thus not suitable for modeling arbitrary 3D environments. For full 3D representation of the world, continuous occupancy maps [6], [7] have been predicted using Gaussian process classification. Since Gaussian processes capture spatial correlation between data, dense and accurate occupancy maps are obtained as well as map uncertainties. In the meanwhile, Gaussian process implicit surfaces have been reconstructed from point clouds [8], LIDAR data [9], [10], and hybrid data [11], while the uncertainties of mesh points have been utilized for path planning [12]. However, the main drawback of Gaussian processes is the high computational complexity of $O(N^3)$ where N is the number of training data and thus, not directly applicable for large-scale environments.

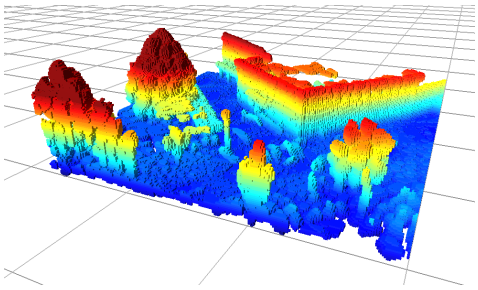


Fig. 1: 3D continuous occupancy map built from the dataset of University of Freiburg [18] with our proposed method.

To enhance the scalability, various approximation methods have been proposed. KD-trees [5], [7], [13] have been used to predict outputs of test positions only with their nearest training data, while a mixture of Gaussian processes [9], [14]–[16] partitions training data into manageable subsets and merges predictions of experts by a gating network. However, the former method needs to invert a new covariance matrix for each test point, while the latter requires each Gaussian process expert to predict over the whole input space. Thus, both are not appropriate for large-scale mapping where the sizes of training and test data are enormous. Pseudo-inputs [17] and active data selection [6], [10] approximate the training data with representatives. However, it is also time-consuming to search for best pseudo-inputs or basis vectors.

The contributions of this paper are two folds. First, we further reduce the computational complexity of occupancy mapping using Gaussian processes by partitioning test points together with training data and applying local Gaussian processes. Second, to mitigate the discontinuity problem along the boundaries we propose overlapping training data for local Gaussian processes. Experimental results with simulated data show that our method is more accurate than occupancy grid maps and more stable than the previous methods of occupancy mapping using Gaussian processes. We also demonstrate our method with real data acquired from a laser range finder.

The structure of the paper is as follows. In Section II we review occupancy mapping using Gaussian processes. We propose our local approximation method in Section III. In

[1]S. Kim is a PhD student of College of Engineering and Computer Science, The Australian National University, Canberra, Australia soohwan.kim at anu.edu.au

[2]J. Kim is a Senior Lecturer of College of Engineering and Computer Science, The Australian National University, Canberra, Australia jonghyuk.kim at anu.edu.au

Section IV we compare our method with previous work on simulated data and demonstrate our method with real data. We conclude the paper with future work in Section V.

II. Building Occupancy Maps using Gaussian Processes

In this section we review occupancy mapping using Gaussian processes [6], [14], [15]. We also calculate the computational complexity of this approach theoretically, which will be addressed by our method in Section III.

A. Occupancy Mapping

Occupancy mapping is a binary classification problem to find the posterior distribution of the occupancy m at every test position (test data), given robot pose states $\mathbf{s}$ and observations $\mathbf{z}$ of range measurements (training data) ,

$$p(m|\mathbf{s}, \mathbf{z}) \,, \tag{1}$$

where m=0 (free) or 1 (occupied). The robot poses $\mathbf{s}$ will be omitted hereinafter since they are embedded in the observations $\mathbf{z}$ while converting range measurements in sensor polar coordinates to line segments and hit points in the global Cartesian coordinates.

B. Gaussian Process Regression

Instead of applying Gaussian process classification directly, we first apply Gaussian process regression and then conduct probabilistic least square classification. This avoids time-consuming approximations such as Laplace's method or Expectation Propagation with tolerable loss of accuracy.

For now, we assume that range beams are discretized into several free points and predict the posterior of the continuous occupancy value $\tilde{m}$,

$$p(\tilde{m}|\mathbf{z}) \,, \tag{2}$$

where $\tilde{m} \in \mathbb{R}$, $\mathbf{z} = (z_1, ..., z_N)^{\mathrm{T}}$, $z_i \in \mathbb{R}$, $z_i = -1$ (free point) or $+1$ (hit point), and N is the number of training data. The discretization will be replaced by the integral kernels, and the regression results will be converted to class probabilities by the probabilistic least squares classification.

We consider the continuous occupancy value as a function of a location $\mathbf{x}$ and apply a Gaussian process prior,

$$\tilde{m}(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')\right), \tag{3}$$

where the mean function is chosen to be zero and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function between two locations $\mathbf{x}$ and $\mathbf{x}'$.

Since the estimation of robot poses and range measurements are not perfect, we assume that observations are corrupted with additive Gaussian noise,

$$z(\mathbf{x}) = \tilde{m}(\mathbf{x}) + \epsilon, \tag{4}$$

where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ and σ_n^2 denotes the noise variance. (For exact modeling of input noise, refer to [19].)

Then, the joint distribution of the observations $\mathbf{z}$ and the occupancy value $\tilde{m}$ is a Gaussian distribution,

$$\begin{bmatrix} \mathbf{z} \\ \tilde{m} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{k}_* \\ \mathbf{k}_*^{\mathrm{T}} & k_{**} \end{bmatrix}\right), \tag{5}$$

where $\mathbf{K} \in \mathbb{R}^{N \times N}$, $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_* \in \mathbb{R}^N$, $[\mathbf{k}_*]_i = k(\mathbf{x}_i, \mathbf{x}_*)$, and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$ while $\mathbf{x}_i$ is the location of the i-th observation z_i and $\mathbf{x}_*$ is the test position. (For details, refer to [3].)

Therefore, the conditional distribution of the occupancy value is also a Gaussian distribution,

$$p(\tilde{m}|\mathbf{z}) = \mathcal{N}\left(\tilde{m}; \mu, \sigma^2\right), \tag{6}$$

where $\mu = \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1}\mathbf{z}$ and $\sigma^2 = k_{**} - \mathbf{k}_*^{\mathrm{T}}[\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1}\mathbf{k}_*$.

1) Covariance Function: We use the Mátern covariance function with $\nu = 3/2$,

$$k_{\nu=3/2}(r) = \sigma_f^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(\frac{-\sqrt{3}r}{l}\right), \tag{7}$$

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

The choice of the covariance function is based on the fact that the squared exponential covariance function in the previous methods [6], [14] is too smooth to model sharp changes in occupancy values of arbitrary environments.

2) Integral Kernels: Originally, a covariance function defines similarity between two data points. Therefore, we should discretize continuous range beams into several free points, which makes the size of training data even larger. To avoid this problem, we employ the integral kernels [6] and treat a line segment as a single data point. The integral kernels are

$$\begin{aligned} k_{lp}(\mathbf{l}, \mathbf{x}) &= \int_0^1 k(\mathbf{l}(u), \mathbf{x}) du, \\ k_{ll}(\mathbf{l}, \mathbf{l}') &= \int_0^1 \int_0^1 k(\mathbf{l}(u), \mathbf{l}'(v)) du \, dv, \end{aligned} \tag{8}$$

where k_{lp} and k_{ll} are called line-to-point and line-to-line covariance functions, respectively, while $\mathbf{l}(u)$ and $\mathbf{l}'(v)$ denote line segments parameterized with $u, v \in [0, 1]$, respectively. Accordingly, the output of a line segment should be integrated as minus its length, $z(\mathbf{l}) = -\text{length}(\mathbf{l})$, while the output of a hit point is still $+1$.

Note that the integral kernels of the Mátern covariance function have no closed form solutions. Therefore, we employ Clenshaw-Curtis quadrature [20] to integrate point-wise covariance functions numerically.

C. Probabilistic Least Squares Classification

Now, we apply the probabilistic least squares classification [21] and return from the posterior distribution of the continuous occupancy value $p(\tilde{m}|\mathbf{z})$ to the binary occupancy $p(m|\mathbf{z})$ by squashing the mean with the variance through a cumulative Gaussian density function Φ,

$$p(m|\mathbf{z}) = \Phi\left(\frac{m(\alpha\mu + \beta)}{\sqrt{1 + \alpha^2\sigma^2}}\right), \tag{9}$$

where the parameters α and β are optimized by performing leave-one-out cross-validation on the training set.

TABLE I: Comparison of computational complexity between various occupancy mapping methods using Gaussian processes. ([methods] OM_GP: occupancy mapping using a Gaussian process, OM_MGP: using a mixture of Gaussian processes, and OM_LGP: using local Gaussian processes, [parameters] N: number of training data (line segments and hit points), M: number of test positions, and K: number of equally-divided clusters.) Note that $N \approx M$ in large-scale environments.

OM_GP	OM_MGP	OM_LGP (our method)
$O(N^3) + O(N^2M)$	$O\left(\frac{N^3}{K^2}\right) + O\left(\frac{N^2M}{K}\right)$	$O\left(\frac{N^3}{K^2}\right) + O\left(\frac{N^2M}{K^2}\right)$

D. Computational Complexity

It is worth examining the computational complexity of occupancy mapping using a Gaussian process (OM_GP). The occupancy map $p(m|\mathbf{z})$ in Eq. (9) requires the means and variances for all test positions. In Eq. (6), inverting the $N \times N$ matrix costs $O(N^3)$, while multiplying the cross covariance vector and the inverted matrix costs $O(N^2)$ per test position. Therefore, the total computational complexity is $O(N^3) + O(N^2M)$, where N and M are the number of training data and test positions, respectively.

The previous work [14], [15] focused on the cubic complexity of the first term and partitioned the training data into manageable subsets and predicted occupancy maps using a mixture of Gaussian processes (OM_MGP). Suppose that the training data are equally divided into K clusters, then the number of training data would drop to N/K in each cluster. Thus, the computational complexity decreases significantly even though predictions are repeated K times by individual Gaussian processes. Table I summarizes the computational complexity of various occupancy mapping methods using Gaussian processes.

III. Building Occupancy Maps using Overlapping Local Gaussian Processes

The previous work (OM_MGP) aimed at partitioning large amount of training data but missed the fact that the number of test positions are also huge in large-scale environments. In fact, we found that the orders of magnitude of the training and test data are almost same ($N \approx M$) in reality in Section IV. Thus, the second term in the computational complexity of OM_MGP in Table I becomes more significant than the first one. Therefore, in order to further reduce the computational complexity, we propose to partition test positions as well as training data and apply local Gaussian processes to each cluster.

A. Partitioning Training Data

We use range measurements with returns as training data, and thus an observation is composed of a line segment and a hit point. Technically, the influence of a line segment on a test position depends on the distance between them. Therefore, we need to find all line segments close enough to each test position [7], but this is too time-consuming and does not have much impact when thresholding the final results.

Instead, we first divide hit points with the k-means clustering [22] and then assign line segments to the clusters including corresponding hit points. This successfully recovers sharp edges on the walls by balancing the influences of line segments (free) and hit points (occupied) in each cluster. The choice of the k-means clustering is based on scalability and feasibility; the Dirichlet process mixture models in the previous work [14], [16] takes too much time until Gibbs sampling converges, while the line tracking of the previous work [15] is not applicable for 3D data.

In addition, because the amount of training data is huge, all of them cannot be loaded on the memory at once. Thus, we take a coarse-to-fine clustering strategy; we roughly partition the world with boxes and finely cluster again with the k-means clustering in each box. Meanwhile, the cluster size depends on the distribution of the point clouds and thus varies cluster by cluster. Therefore, in order to guarantee each cluster to be less than a manageable size, we perform the k-means clustering iteratively on those clusters which are larger than a maximum limit.

B. Local Gaussian Processes

Suppose that the training data are partitioned into K subsets, $\mathbf{z} = \{\mathbf{z}_i\}_{i=1}^{K}$ associated with their cluster centers. Then, we assume that the occupancy value of a test position only depends on the closest cluster,

$$p(\tilde{m}|\mathbf{z}) \approx p(\tilde{m}|\mathbf{z}_k), \tag{10}$$

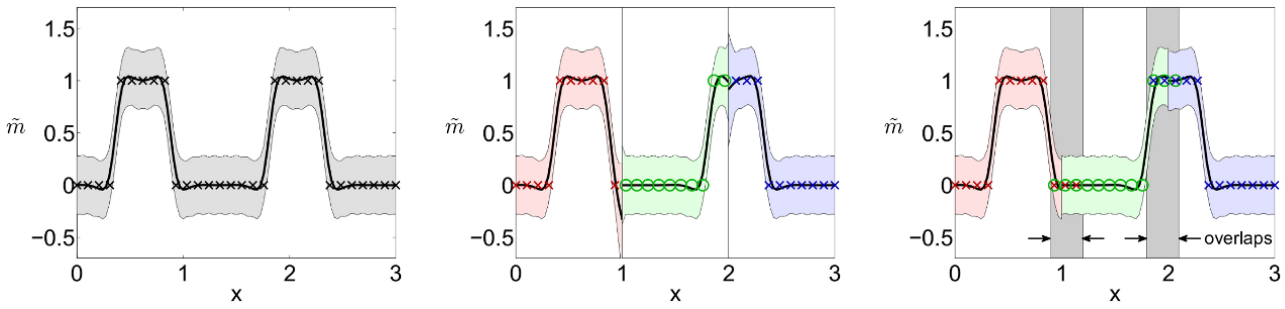
where the k-th cluster center is closest to the test point.

This assumption is reasonable because the covariance of two data points drops quickly as the distance increases, especially in the Mátern covariance function. This independence factorizes the global Gaussian process into K local Gaussian processes (OM_LGP) with their own training and test data, which predict means and variances over their own expert domains, not over the whole input space. Suppose that all test positions are evenly partitioned to K clusters. Then, the number of test positions per cluster would decrease to M/K, which further reduces the computational time as shown in the last column of Table I.

C. Overlapping Training Data

The independence assumption, however, is violated near the boundaries of the partitioning boxes and clusters, because they are disjoint and thus the correlations of close training data are ignored. This causes the discontinuity problem that predictions do not match on the boundaries of local estimators. Therefore, we coarsely partition the training data with overlapping boxes and extend the ranges of clusters to share some training data near the boundaries.

Fig. 2 illustrates the effect of overlapping training data. Local Gaussian processes with disjoint training data suffer



(a) Global Gaussian process (b) Local GPs with disjoint training data (c) Local GPs with overlapping training data

Fig. 2: 1D example of overlapping training data. (a) Global Gaussian process with hit and free points ($+1$ and -1 cross points), (b) Local Gaussian processes with the same observations partitioned into three disjoint clusters (red, green, and blue), (c) Local Gaussian processes with the same clusters extended by 30%. Recognize that overlapping training data (red and blue crosses in green circles in gray overlapping regions) mitigate the discontinuity problem on the the boundaries. In each plot the black curve and the shaded region denote the mean and twice the standard deviation at each test point. The Mátern covariance function with $l = 1/3$ and $\sigma_f = 1$ is used.

from the discontinuity problem on the boundaries as shown in Fig. 2(b). The overlapping training data in Fig. 2(c) connect the predictions of local Gaussian processes seamlessly, and the results are comparable to the global Gaussian process in Fig. 2(a). The extent of extension should be determined carefully based on the hyperparameters and the density of training data, but we found that $20 \sim 30\%$ of extension is sufficient in most of the times.

IV. Experimental Results

In this section we evaluate our method with simulated data and compare run time and map accuracy with the previous methods. We also demonstrate our method with real data.

A. Experiments on Simulated Data

We simulated laser scanning from 28 robot poses in a virtual environment of $22 \times 18m^2$ as shown in Fig. 3(a). To imitate imperfection of robot pose estimation, we added independent Gaussian noise with zero means and standard deviations of $10cm$ and 2° to the true robot positions and rotations, respectively. Totally, $1,320$ observations (hits points and line segments), were obtained.

The training data were coarsely partitioned into 4 boxes of $11 \times 9m^2$ and finely clustered into 14 clusters with a maximum size of 100 hit points per cluster. The boxes and clusters are extended by 20%. With a map resolution of $20cm$, we generated $9,900$ test points in total and assigned them to the closest clusters. Fig. 3(b) depicts the partitioned training data (only disjoint clusters of hit points are shown for clarity) and test positions. Since the test point are allocated to the closest clusters, equidistance lines in each box are found like Voronoi diagrams.

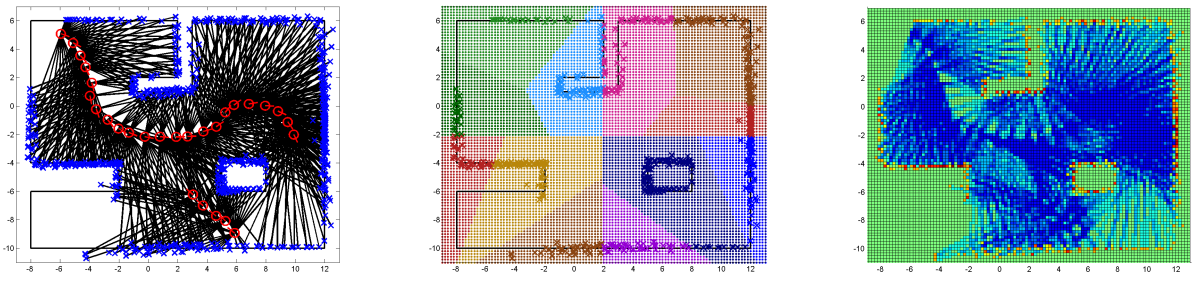
1) Comparison of Run Time: We implemented occupancy grid maps (OGM) and occupancy mapping using Gaussian processes (OM_GP, OM_MGP, and OM_LGP) in MATLAB as well as data partitioning in C++ on a computer with an Intel Core 2 Duo 3.0 GHz CPU and 3.25 GB RAM. The run time of each map building method is summarized in Table II. As expected, OGM ran the fastest, while OM_GP was the slowest. The map building time of OM_MGP was reduced with negligible overhead for clustering the training data. Our OM_LGP sped up more by partitioning the test data together with the training data. This result reflects the theoretical computational complexity in Table I.

TABLE II: Comparison of run time with the simulated data. (OGM: occupancy grid map, OM_GP: occupancy mapping using a Gaussian process, OM_MGP: a mixture of Gaussian processes, and OM_LGP: local Gaussian processes)

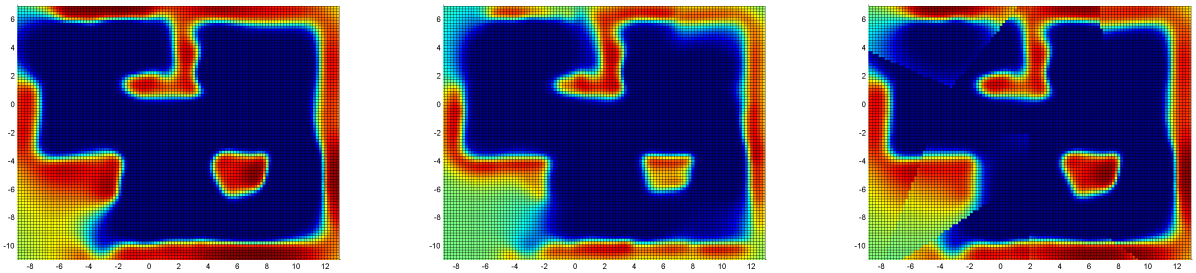
	Clustering	Mapping	Total
OGM	–	0.044 sec	0.044 sec
OM_GP	–	6.065 sec	6.065 sec
OM_MGP	0.007 sec	5.277 sec	5.284 sec
OM_LGP (ours)	0.011 sec	1.187 sec	1.198 sec

2) Comparison of Accuracy: The objective of this paper is to reduce the computational time, while retaining the map accuracy. So, now we turn our attention to the quality of the results. OGM in Fig. 3(c) is consistent with observations. However, there exist a lot of holes of unknown spots in empty spaces, which is getting worse in distant areas from robot positions. This may be not crucial because we usually threshold OGM with a value greater than 0.5 (unknown). The more serious problem is that not so many grid cells are predicted as occupied. Therefore, after thresholding, occupied cells are remained sparsely, which may mislead robots to plan a path through a wall. In order to avoid this problem, more observations should be acquired close to obstacles.

On the other hand, Gaussian processes produced dense and more accurate occupancy maps given the same observations because the correlations between training data are considered. OM_GP in Fig. 3(d) successfully classifies occupied spaces from empty ones even with the noise in robot poses. There are some mis-predictions due to insufficient



(a) Simulated data (b) Partitioned hit points and test points (c) Occupancy grid map (OGM)



(d) OM_GP (e) OM_MGP (f) OM_LGP (our method)

Fig. 3: 2D Simulation Results. (a) Laser hit points (blue crosses) and corresponding laser beams with returns (black lines) obtained from noisy robot poses (red circles), (b) Partitioned training and test data (Neither laser beams nor shared hit points are drawn for clarity.), (c) Occupancy grid map, (d) Occupancy mapping using a Gaussian process, (e) A mixture of Gaussian processes, (f) Local Gaussian processes (our method). Note that occupancy is color-coded; red/green/blue denotes occupied/unknown/empty. Hyperparamters are trained as $l = 2.66$, $\sigma_f = 2.81$ and $\sigma_n = 0.99$, and the squashing parameters are learned as $\alpha = 7.94$ and $\beta = -1.42$.

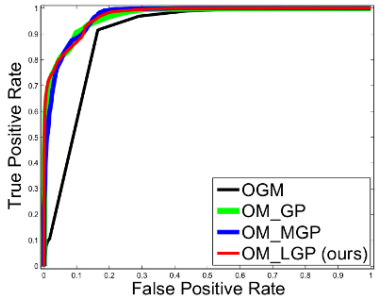


Fig. 4: ROC curves of occupancy mapping methods with the simulated data. (Refer to Fig. 3 for the abbreviations of method names.)

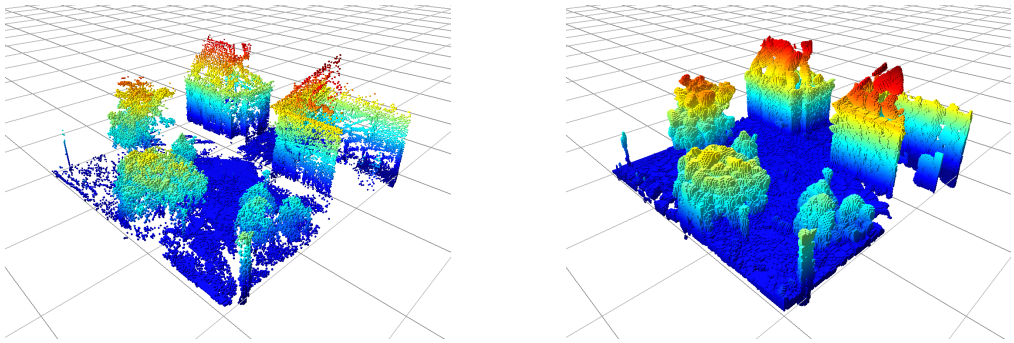
observations in some areas such as the right face of the L-shape wall in the middle and the right corner of the rectangular pole. OM_MGP in Fig. 3(e) suffers from loss of details when merging predictions of individual Gaussian processes. The sharp changes of occupancy on the walls are also little blurred. Finally, our OM_LGP in Fig. 3(f) recovers the sharpness but shows some breaks on predictions, which is not significant especially when thresholding the results.

For more precise comparison of map accuracy, we plot the Receiver Operating Characteristic curves based on the ground truth of the simulated data. The ROC curve in Fig. 4 confirms that our OM_LGP is comparable to OM_GP and OM_MGP, while outperforms OGM. From the comparison results of map accuracy and run time, we can say that our method is more accurate than OGM and more scalable than OM_GP and OM_MGP.

B. Experiments on Real Data

We demonstrated our method with the laser dataset of University of Freiburg [18]. A region of $45 \times 45 \times 30m^3$ containing buildings, trees and roads was selected as a representative for outdoor environments. Totally, $3,676,474$ training data (laser hit points and laser beams with returns) were collected, while $7,593,750$ test positions were generated with a map resolution of $20cm$. Recognize that the number of test positions are even bigger than the number of training data.

The training data were coarsely partitioned into 18 overlapping boxes of $15 \times 15 \times 15m^3$ and finely clustered into $11,497$ clusters with a maximum number of 500 hit points per cluster in 22.6 minutes. The size of boxes was determined by the memory capacity and the density of 3D point cloud. The maximum size of clusters, on the other hand, was chosen based on the trade off between run time and map accuracy. As mentioned in Section II-D, the computational time for



(a) Occupancy grid map (OGM) (b) Our method (OM_LGP)

Fig. 5: Results with the laser dataset of the University of Freiburg [18]. Note that results are color-coded by height.

inverting a square matrix grows cubically, and we tested and found that it increased sharply after $1,500$. Thus, we selected $1,000$ number of training data (hit points and line segments) as a limit and extended each cluster by 15%.

As with the simulated data, our OM_LGP in Fig. 5(b) is more accurate than sparse OGM in Fig. 5(a); the holes on the ground and buildings were tightly filled. OGM was generated in 66.3 seconds, while OM_LGP was built in 1.3 hours, but we expect significant speed-up by converting the MATLAB codes to C/C++ in future work. The hyperparameters were learned as $l = 1.33$, $\sigma_f = 2.81$ and $\sigma_n = 0.99$.

V. Conclusions

In this paper, we proposed to apply local Gaussian processes with overlapping training data for occupancy mapping. Previous work focused on the huge size of training data and partitioned them to reduce the cubic complexity of Gaussian processes. In contrast, we pointed out the fact that the orders of magnitude of training data and test positions are almost same in large-scale environments and partitioned test positions together with training data. Therefore, we achieved to enhance the scalability by further speeding up the computational time. In addition, we proposed overlapping training data to overcome the discontinuity problem on boundaries of local Gaussian processes.

Experimental results on simulated data show that our method is more accurate than occupancy grid maps and more scalable that the previous work of occupancy mapping using Gaussian processes. We also demonstrated our method with real data obtained from a laser range finder and showed the feasibility of our method in reality. As future work, we plan to employ an efficient data structure like octomaps to reduce the memory usage and to consider online updates with sequential observations in dynamic environments.

References

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

[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*, pp. 1–18, 2013.

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

[4] T. Lang, C. Plagemann, and W. Burgard, "Adaptive non-stationary kernel regression for terrain modeling," in *Proceedings of Robotics: Science and Systems*, 2007.

[5] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte, "Gaussian process modeling of large-scale terrain," *Journal of Field Robotics*, vol. 26, no. 10, pp. 812–840, 2009.

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

[7] ——, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

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

[9] M. Smith, I. Posner, and P. Newman, "Generating implicit surfaces from LIDAR data," in *Proceedings of Towards Autonomous Robotic Systems*, 2010.

[10] ——, "Adaptive compression for 3d laser data," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 914–935, 2011.

[11] 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*, 2011, pp. 2845–2850.

[12] G. Hollinger, B. Englot, F. Hover, U. Mitra, and G. Sukhatme, "Active planning for underwater inspection and the benefit of adaptivity," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 3–18, 2013.

[13] Y. Shen, A. Ng, and M. Seeger, "Fast Gaussian process regression using *kd*-trees," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, pp. 1225–1232.

[14] S. Kim and J. Kim, "Towards large-scale occupancy map building using Dirichlet and Gaussian processes," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2011.

[15] ——, "Building occupancy maps with a mixture of Gaussian processes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012, pp. 4756–4761.

[16] ——, "Building large-scale occupancy maps using an infinite mixture of Gaussian process experts," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2012.

[17] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006, pp. 1257–1264.

[18] B. Steder and R. Kümmerle, "The outdoor dataset of the university of freiburg," http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/.

[19] A. Girard and R. Murray-Smith, "Learning a Gaussian process model with uncertain inputs," University of Glasgow, Tech. Rep. TR-2003-144, 2003.

[20] W. Gentleman, "Implementing Clenshaw-Curtis quadrature, I methodology and experience," *Communications of the ACM*, vol. 15, no. 5, pp. 337–342, 1972.

[21] J. C. Platt, "Probabilities for SV Machines," in *Advances in Large Margin Classifiers*. MIT Press, 2000, pp. 61–74.

[22] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.