2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Online Continuous Mapping using Gaussian Process Implicit Surfaces

Bhoram Lee, Clark Zhang, Zonghao Huang and Daniel D. Lee

***Abstract*— The representation of the environment strongly affects how robots can move and interact with it. This paper presents an online approach for continuous mapping using Gaussian Process Implicit Surfaces (GPISs). Compared with grid-based methods, GPIS better utilizes sparse measurements to represent the world seamlessly. It provides direct access to the signed-distance function (SDF) and its derivatives which are invaluable for other robotic tasks and it incorporates uncertainty in the sensor measurements. Our approach incrementally and efficiently updates GPIS by employing a regressor on observations and a spatial tree structure. The effectiveness of the suggested approach is demonstrated using simulations and real world 2D/3D data.**

I. Introduction

An accurate and efficient representation of spatial structures is crucial for successful planning and control in navigation or manipulation tasks. While a grid-based representation such as the occupancy grid (OG) is currently being employed [1], Gaussian Processes (GPs) which provide a compact representation of continuous functions, have been suggested as an alternative [2], [3]. Unlike grid-based methods, GP does not depend upon a particular choice of coordinates and can continuously interpolate surface structures. They can be naturally extended to incorporate environmental priors and derivatives as well.

This paper suggests an online framework to build a map as the zero level set of a GPIS. Our representation maintains a probabilistic estimate of the SDF to objects in the environment (Fig. 1). The representation itself includes distance and its gradient, which are critical information for online obstacle avoidance and trajectory planning. The update process we present modifies the existing map by comparing it with new observations in a Bayesian update scheme. We introduce a GP regressor on measurements and infer corresponding surface points and normals continuously. The measurement noise model considered is heteroscedastic and accounts for noise resulting from the geometric relationship between the surface and the observer's viewpoint. Additionally, our GPIS implementation consists of multiple small GPs that are updated and tested locally to mitigate the notorious complexity of the standard GP model. A spatial partitioning data structure is exploited to manage this data efficiently.

While prior research has focused on modeling the GP (Sec. II), this paper contributes a principled way to update GPIS models incrementally with noisy sensor measurements. This is necessary for a robot to continually plan while navigating in an uncertain environment.

B. Lee, C. Zhang and Z. Huang are with the GRASP Laboratory, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA, USA (e-mail: {bhorlee, clarkz, zoh}@seas.upenn.edu)

D. D. Lee is with CornellTech, 2 West Loop Rd, New York, NY, USA (e-mail: ddl46@cornell.edu)

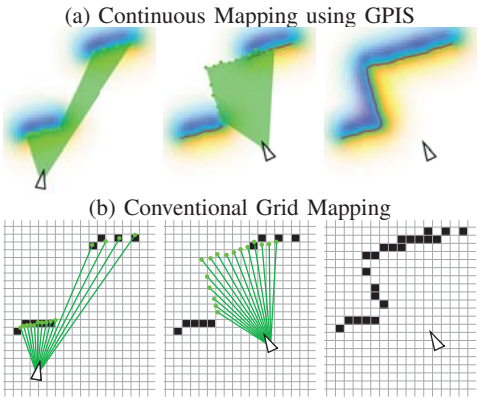

Fig. 1. The suggested method illustrated in (a) incrementally builds a SDF using GP from noisy sensor measurements. The yellow regions indicate areas with positive distance from obstacles, while the blue regions indicate areas inside obstacles (negative distance). Our approach is contrasted with the conventional grid representation shown in (b) which often suffers from discretization errors and disconnected cells. The probabilistic GPIS model provides continuous distance and gradient field estimates near the surface that are useful for motion planning and collision avoidance.

II. Related Work

There have been various suggested ways to incorporate local geometric properties and uncertainties into spatial representations. One of the most successful methods in robotics is OG map ([4], [1], [5]). OG map represents the world as discretized cells, each of which holds a binary random variable to indicate its occupied or free state. Since each cell is treated independently, an update or access of the occupancy value of the cells can be very fast. While having experienced considerable success with grid-based occupancy representations, recently there have been studies that addressed the problem of discretization. O'Callaghan *et al.* [3] pointed out that in OG mapping structural correlations between nearby cells are ignored and we often observe artifacts. They explored the idea of building continuous occupancy maps using GP that encode the correlation of observed points. Later, as an effort to avoid the cubic complexity of GP, Kim *et al.* [6], [7] suggested employing multiple local GPs with systematic spatial partitioning. Ramos *et al.* [8] also addressed the issue and presented kernel approximation methods as parametric alternatives. An efficient data handling method for GP occupancy map is presented in [9].

Besides OG, SDF have been employed in many studies ([10], [11], [12]). SDF was suggested for the sake of accuracy and robustness in surface modeling [13]; Even in its discrete form, SDF can recover a surface more accurately

via interpolation (for example, [14]) than an OG map of the same grid resolution. In addition, the distance itself is informative. For example, when given the distance field, the Iterative Closest Point (ICP) algorithm [15] for surface registration can be performed without an expensive nearest neighbor search [16], [10]. Furthermore, the distance field and its gradients can be utilized in planning for collision checks and trajectory optimization [17], [18].

Although the values of a SDF are often stored in a discrete grid for fast lookup, there have been studies that treated SDF as continuous functions. In particular, GPIS is a non-parametric regression model for an implicit surface that provides the variance for every prediction. Dragiev *et al.* [2] utilized GPIS as shape representations for robotic grasping, and Hollinger *et al.* [19] used GPIS for planning in underwater inspection applications. Gerardo-Castro *et al.* [20] presented data fusion of laser and radar for shape representation within a GPIS framework. Kim *et al.* [21] and Martens *et al.* [22] explored using non-zero mean functions for a more complete shape representation. Whereas other studies lack the notion of building GPIS from online measurements, this paper contributes a method to enable incrementally building GPIS.

Applications of GP in robotics is not limited to OG map or SDF. Other related works include [23], where the authors introduced a GP regression for range measurement and demonstrated the method on localization and tracking problems. Our method also adopts a measurement regression, which is used to infer surface point locations and normals.

III. Representation

This section reviews the GPIS representation as an approximate SDF. Assume there exists a volumetric object and the surface that defines the boundary of the object in the D-dimensional Euclidean space. Consider a point $\boldsymbol{x}$ in the space and a function $f : \mathbb{R}^D \to \mathbb{R}$ such that

$$f = \begin{cases} +d & \text{outside surface} \\ 0 & \text{on surface} \\ -d & \text{inside surface,} \end{cases} \tag{1}$$

where d is the distance of the point from the surface. We aim to build a GP regressor that approximates this function near the surface of a structure. Our approach is based on the noisy input GP model [24], where the input $\tilde{\boldsymbol{x}}$ of GP is corrupted by a Gaussian noise $\boldsymbol{\epsilon}_x \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$, *e.i.*, $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}_x$. Then the observed output can be written as, $y = f(\tilde{\boldsymbol{x}} - \boldsymbol{\epsilon}_x) + \boldsymbol{\epsilon}_y$, where $\boldsymbol{\epsilon}_y \sim \mathcal{N}(\mathbf{0}, \sigma_y^2)$ is a noise to the output variable. Then, when its first order Taylor expansion is considered, the probability of an observation, y, can be expressed as,

$$P(y|f) = \mathcal{N}(f, \sigma_y^2 + \boldsymbol{\partial}_f^\top \Sigma_x \boldsymbol{\partial}_f), \tag{2}$$

where $\boldsymbol{\partial}_f$ is denoted as the derivative of the function with respect to the input. In our setting, Σ_x is modeled as an isotropic Gaussian covariance, with parameter σ_x, and the derivatives satisfy $||\boldsymbol{\partial}_f||_2 = 1$ by definition of surface normal. Thus, Eq. (2) can be rewritten as,

$$P(y|f) = \mathcal{N}(f, \sigma_y^2 + \sigma_x^2). \tag{3}$$

While the standard GP considers the output noise only, this allows us to incorporate the priors of measurement noise to the surface update. The noise may be modeled as another process as in [23]. However, in order to render the problem tractable, we assume that the noise is independent. Thus, the noise covariance of n input points can be written as $K_x = \text{diag}\{\sigma_{x_1}^2, \sigma_{x_2}^2, ..., \sigma_{x_n}^2\}$. Then the inference equations for the mean $\bar{f}_*$ and the variance $\mathbb{V}[f_*]$ of the function value given a test point $\boldsymbol{x}_*$ can be summarized as,

$$\begin{aligned} \bar{f}_* &= \mathbf{k}_*^\top (K + K_x)^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + K_x)^{-1} \mathbf{k}_*. \end{aligned} \tag{4}$$

We follow the notations $\mathbf{k}_*$, K, and $k(\mathbf{x}_*, \mathbf{x}_*)$ from [25], which represent the vector of covariances between $\boldsymbol{x}_*$ and n training points, the $n \times n$ covariance matrix of the training points, and the covariance function value of $\boldsymbol{x}_*$ respectively. $\mathbf{y}$ is the target vector of the training points.

Now let us consider available training data for the function f. While surface points are observed, we do not explicitly observe exemplars of off-surface or internal points. Some methods employ control points of the two types to express the directional information of the surface ([20]). In this study, we consider jointly modeling the SDF and the derivatives to encode the surface direction as in [2], [22]. In our implementation, a Matérn class covariance function ($\nu = 3/2$) is used with its first-order derivatives. Kernel parameter learning is one capability of GP that is not considered at this stage of the study.

IV. Online Update

Now we will describe the main idea of the paper, an incremental update method for GPIS. Given a set of training surface points of GPIS and new observations, if the correspondences are known then the surface points can be updated in a Bayesian fashion. However, it is hard to find corresponding points directly from discrete measurements. We will show how to bypass the correspondence problem by using observation GP regressors, and describe the noise models used for the update. Note that all descriptions and illustrations are restricted to 2D for clarity, but can be easily extended to 3D as shown in Sec. VI. Fig. 2 illustrates what our method maintains and returns while being updated online.

A. Bayesian Update of Surface Points

Consider a surface point x_k^- from the GPIS model and assume a new observation of that surface point as $\tilde{x}_k$. Supposing that both the model point and observed point are Gaussian random variables with respective variances of $\sigma_{x_k}^{2-}$ and $\tilde{\sigma}_{x_k}^2$. The surface point in the GPIS can be updated to have the distribution $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{x}_k^+, \sigma_{x_k}^+)$ with,

$$\begin{aligned} \boldsymbol{x}_k^{2+} &= \frac{\tilde{\sigma}_{x_k}^2 \boldsymbol{x}_k^- + \sigma_{x_k}^{2-} \tilde{\boldsymbol{x}}_k}{\tilde{\sigma}_{x_k}^2 + \sigma_{x_k}^{2-}} \\ (\sigma_{x_k}^{2+})^{-1} &= (\tilde{\sigma}_{x_k}^2)^{-1} + (\sigma_{x_k}^{2-})^{-1} \end{aligned} \tag{5}$$

The GPIS model also contains surface normals which are updated separately from the points: a normal vector update

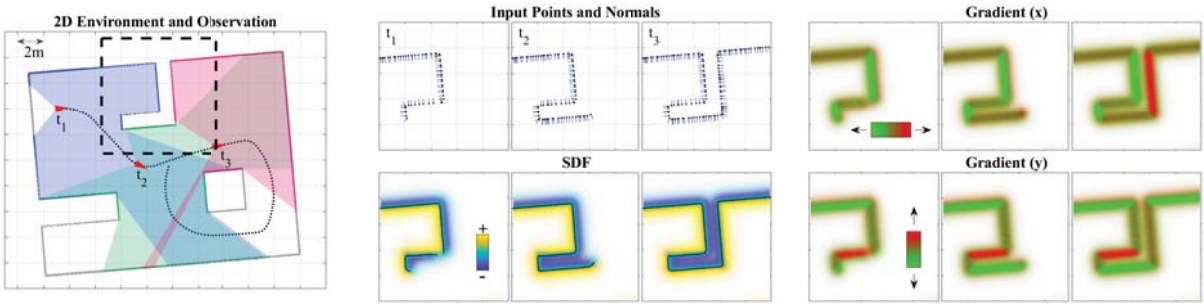

Fig. 2. The suggested online GPIS (Best viewed in color): (Left) The arrow indicates the pose of the robot, and the colored patches and points visualize the range measurement at time t_1, t_2, and t_3. The gray points are the accumulated range measurements over time, the dotted black line shows the trajectory of the robot. Other images on the right show the boxed area within the bold dashed line. (Middle Top) The method updates the training surface points and normals of GPIS with incremental observations. (Middle Bottom) The resulting continuous SDF is visualized in a yellow-blue colormap, and its sampled surfaces in black. (Right) The gradient fields of SDF are visualized in a red-green colormap.

is treated as weighted rotation instead of the arithmetic weighted average.

B. Observation GP

The Bayesian update is simple when we correspond a measurement $\tilde{x}_k$ with a point x_k in the GPIS model. However, this is not usually obvious. Instead of matching a measurement directly, we use a GP regressor on range measurements to infer $\tilde{x}_k$ and its surface normal.

Let z_i be the range measurement at angle θ_i in a local polar coordinate frame. Then the GP regressor is the function,

$$f_z : \theta \rightarrow z^{-1}. \tag{6}$$

Note that z^{-1} is used since it nicely scales from 0 to the inverse of the maximum limit range which is a finite positive value. This GP continuously predicts the range values at any given valid θ (Fig. 3 (a)). The Ornsten-Uhlenbeck covariance function is used because it expresses detailed curves without excessive smoothing.

Now, for a test point $\boldsymbol{x}_*$ (or equivalently $(\theta_*, {z_*}^{-1})$ in polar coordinates), if $\boldsymbol{x}_*$ is visible or free then ${z_*}^{-1} > \bar{f}_z(\theta^*)$. If it is unobservable then ${z_*}^{-1} < \bar{f}_z(\theta^*)$. We may soften this occupancy test by taking a logistic function along the ray as,

$$occ({z_*}^{-1}, \theta_*) = 1 - \frac{2}{1 + \exp\{-a({z_*}^{-1} - \bar{f}_z(\theta^*))\}}, \tag{7}$$

where a is a slope parameter. An example of this occupancy test using Eq. (7) is visualized in Fig. 3 (b). The angular ranges with very high variances are considered as being unobservable or invalid.

C. Surface Point and Normal Inference

With this continuous local representation of the occupancy, a corresponding measurement point can be numerically inferred (Fig. 3(c)) as the following. Given an old point $\boldsymbol{x}_k^-$ with normal $\boldsymbol{n}_k^-$,

1) Test its occupancy with Eq. (7).
2) If the absolute value of occupancy is larger than a small number ϵ, and
   a) if the value is negative (beyond 'surface') then move a step along the direction of $\boldsymbol{n}_k^-$, or
   b) if the value is positive ('free') then move a step along the opposite direction of $\boldsymbol{n}_k^-$.
3) Otherwise, stop sampling points. Take the current location with a small occupancy value as $\tilde{x}_k$.

We may iterate this step to obtain a new point very close to the surface.

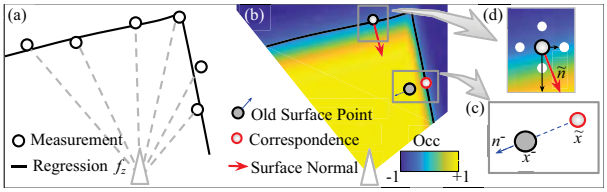

Fig. 3. (a) The range regression and (b) occupancy test based on Eq. (7) are computed in local polar coordinates and converted to Cartesian coordinates in the figure. Illustration for the computation of (c) point location and (d) surface normal. (Best viewed in color)

Next, the gradients of a measurement point $\tilde{x}_k$ can be inferred by computing (Fig. 3(d)),

$$\frac{\partial \tilde{x}_k}{\partial v} \approx \frac{occ(\tilde{x}_k + \delta \boldsymbol{e}_v) - occ(\tilde{x}_k - \delta \boldsymbol{e}_v)}{2\delta} \tag{8}$$

where v indicates an axis of the D-dimensional Cartesian space and $\boldsymbol{e}_v$ is the unit vector along the axis. Also δ is a small positive value that represents a reasonable neighborhood range. Then the normalized vector of the gradients is used as the surface normal $\boldsymbol{\partial}_f$. The absolute gradient values are determined by a of Eq. (7) which is somewhat arbitrary. However, this is not problematic since we only need the directional information of the gradient near the surface. This way, the normal vector can be obtained at any surface point within the visible range.

D. Noise Models

As mentioned in Sec. III, we treat a training surface point $\boldsymbol{x}$ as a Gaussian variable. The noise variance is modeled with two additive terms,

$$\sigma_x^2 = \sigma_z^2 + \sigma_\phi^2. \tag{9}$$

The first term, σ_z^2, is a function of the range value at the time of measurement, and the second term, σ_ϕ^2, is a function of the relative view angle (ϕ) of the surface. Intuitively, a

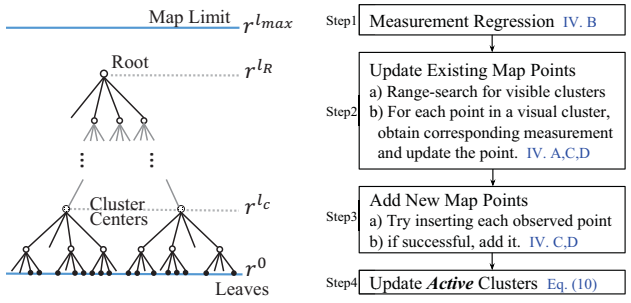

Fig. 4. (Left) Data Structure Implementation in QuadTree: The clusters for local GPs are centered at the level l_c, being a distance r^{l_c} apart from each other. The radius of each cluster is larger than r^{l_c} to allow proper overlapping areas between clusters. (Right) The Update Process

measurement will be less noisy when observed at a closer distance and orthogonal to the observer. One way to implement this model is to use $\sigma_z^2 = \alpha_z z^{-2}$ and $\sigma_\phi^2 = \alpha_\phi \tan(\phi)$, where α_z and α_ϕ are parameters.

The surface normal requires a separate noise model from the point itself. Although the noise for each axis is not independent because of the constraint $|\boldsymbol{\partial}_f| = 1$, we assume a spherical Gaussian noise for simplicity. We model the uncertainty σ_∂ considering that the normals are numerically obtained. In an ideal case, the mean occupancy values of the neighbor points, $\sum_v occ(\boldsymbol{x}_* + \delta \boldsymbol{e}_v) + \sum_v occ(\boldsymbol{x}_* - \delta \boldsymbol{e}_v)$ would be exactly zero. However, if for some reason the neighbor is not representing the local surface properly, it will have some non-zero values. We use this mean value to indicate how much the surface normal computation can be trusted. This gives a reasonable range of absolute value of σ_∂^2 which is the variance of surface normal vectors.

V. Implementation

The foremost concern in using GP would be its computational complexity. Given N points, computing the inverse covariance matrix in a standard GP regression model takes time $\mathcal{O}(N^3)$ and testing M points takes time $\mathcal{O}(MN^2)$. There have been various approximation approaches for this issue [25], and we chose a method similar to the spatial partitioning described in [7], [21]. The basic idea is to divide points into clusters and train each cluster locally. This reduces the time for computing an inverse to $\mathcal{O}(KN_k^3)$ and testing to $\mathcal{O}(MKN_k^2)$ where K is the number of clusters and $N_k \ll N$ is the largest number of points per cluster. Note that in this approach, each cluster must overlap with its neighbors to prevent discontinuity.

The practical issue of this idea is then how to divide space and to what extent the clusters should overlap. We used the QuadTree [26] and OcTree [27] to partition and access the surface points, as well as perform a range search for clustering efficiently. Specifically, we set the minimum resolution of the leaf node r^0, the maximum limit of map size $r^{l_{max}}$, the intervals of local GP clusters $r^{l_c} > r^0$ (Fig. 4, left). Note that the radius of each cluster R_c must satisfy $R_c > r^{l_c}$ to allow overlapped areas with neighboring clusters. When there are changed points, any local GPs within the range R_c from the points are re-trained. In order to make updating efficient, the algorithm keeps track of cluster nodes that are affected by inserting or removing points, which we call *active* clusters.

The overall online update process is summarized on the right in Fig. 4. When new range observations are received, we first compute a GP regression on them (Step 1). Given the current pose, we may query potentially visible clusters within the maximum range measurement value, and reject those that are beyond visible angular ranges. If an existing surface point is considered visible, the measurement regression is used to find the correspondence of the point, as well as to infer surface normals (Step 2). After moving existing surface points, test if newly observed points can be inserted into the map (Step 3). Only the GPs in the *active* set are re-computed (Step 4). The practical computation of Eq. (4) involves,

$$L := Chol(K + K_x) \text{ and } \alpha := L^\top \backslash (L \backslash y). \quad (10)$$

When testing a point, a range search is performed to find nearby local GPs first. Then, the predictive mean and variance (Eq. (4)) of the point can be computed as,

$$\bar{f}_* = \mathbf{k}_*^\top \alpha \text{ and } \mathbb{V}[f_*] = L \backslash \mathbf{k}_*. \quad (11)$$

on the n closest GPs. Depending on the confidence level, our method chooses the best inference if the variance is below a certain value, or takes a weighted sum otherwise. This preserves the continuity of the map.

VI. Results

In this section, we demonstrate the characteristics and performance of the suggested method. We evaluated the accuracy in structure prediction on a simulation. Some of the visual results and computational times on real data are also reported. All programs are written in Matlab and C++ [1] and all computations are performed on a CPU of a laptop (Intel i7-6700HQ @ 2.60GHz).

Structure Prediction: We compared our method with OG mapping [28], Hilbert maps [8], and TSDF [13]. The three methods are chosen to have all combinations of discrete/continuous and occupancy/SDF representations. The simulation data of range measurements and ground truth poses were obtained in Gazebo using a Hokuyo range sensor model on a Turtlebot with a Gaussian noise $\sigma = 0.01m$. The simulated environment (Fig. 5) was made to duplicate the one from [8], [29], [3] with slightly rotation to avoid any artifacts caused by axis-aligned data. The angular range of the lidar sensor was $-135°$ to $+135°$ and the sensing resolution was $1°$. The accumulated raw measurements were dense enough to cover the structure as seen in Fig. 2 (a). The ground truth SDF of the environment is shown in Fig. 5 (a). The distance values of the 1D dotted cross-section is visualized under the map. The four other graphs Fig. 5 (b)-(e) display the results of the compared methods in the form of either occupancy or SDF.

The first two gray-scale maps are occupancy maps. The OG map in Fig. 5 (b) has a grid resolution of $0.1m$ and

[1]The source code and the parameter settings can be found at `https://github.com/leebhoram/GPisMap`

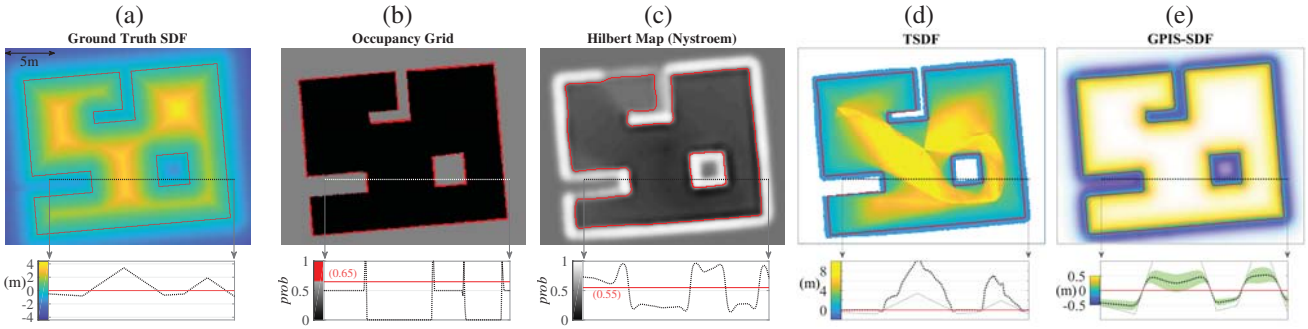

Fig. 5. Surface Modeling in 2D (best viewed in color): This comparison illustrates the characteristics of different mapping/modeling methods. The true SDF is shown in (a). While (b) and (c) model the structure using the occupancy representation, (d) and (e,Ours) model the surface using the SDF representation. On the other hand, (b) and (d) are grid-based implementations whereas (c) and (e-Ours) are continuous functions. The red solid lines on the 2D maps are the surface (or occupied) parts of the structure. The occupancy/distance values of the dotted section are visualized under the maps. The gray lines in (d) and (e) are the ground truth scaled properly for comparisons. Note that the colormap scales vary according to the range of each result.

cells that have probability values above a threshold (0.65, default by [28]) are considered occupied and are marked in red. Although the measurements completely covered the structure, a close look into the map reveals missing surface cells on the bottom left side. This is due to the fact that those areas were only observed from afar for a short period of time so that the cells could not accumulate enough measurements to trust its occupancy. This occurs frequently in OG maps. Fig. 5 (c) shows the Hilbert map using the Nyström method, for which we set *Kernel width* $\gamma = 2.5$, *number of features* $= 3000$, and default values by the authors for other parameters. As shown in the 1D visualization, this method expresses probabilities which ranges between 0 to 1 as a smooth function. As a result, its predictions for surfaces come as wide regions and it is hard to obtain a precise surface (curve) in the same way as done in Fig. 5 (b). Instead, we visualized the boundary pixels of empty space with the threshold 0.55 to match with other methods.

The next two maps are SDFs. Fig. 5 (c) shows the TSDF map of resolution $0.1m$. In the figure, cells with zero weights are plotted in white. Our test adopted the moving average with the unit weight as done in [10] for simplicity. The surface estimate as the zero-level isocontour is shown in red. Often the distance update is restricted to the vicinity of the hit points, but we did not restrict it so that we could better observe the SDF values. The obtained distance values tend to be overestimated as shown in the 1D plot. This is because distances are taken along the the line of sight to the sensor rather than the closest distance to the hit point. Lastly, the SDF and surface estimates by GPIS is shown in Fig. 5 (e). The minimum resolution r was 0.1 and the scale parameter of the covariance function was 1.2. The map is masked with an alpha (transparency) layer of the variance values, and thus regions of high variances appear white. The variance is visualized as the green envelop in the 1D profile plot. It gives high uncertainty in free space since our implementation does not explicitly model those free regions. However, the predicted signed-distances at the vicinity of the structure are close to actual values with high confidence.

The accuracy of surface predictions are evaluated with reduced data rates. The Hilbert map was omitted since its

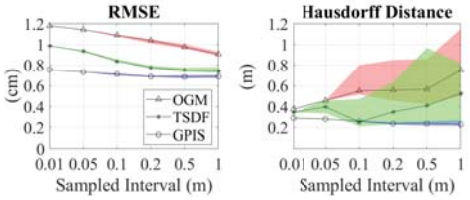

Fig. 6. Surface Prediction Accuracy vs. Reduced Data Rate. The colored patch represents minimum/maximum intervals for each case.

result was not comparable without devising a method to precisely extract the surface. The original data used in Fig. 5 was collected roughly every $1cm$ while the robot traveled $37m$. We skipped regular numbers of frames to simulate longer sampling intervals (Fig. 6). For instance, in the '$1m$' case, the robot is assumed to travel $1m$ before it receives new measurement. Often, in practical situations, only glimpses of parts of the surrounding are available for a robot due to high-speed motion, occlusions, or degraded bandwidth. The experiment simulated such a condition, in which the quantity of observation is degraded. Two metrics, the root-mean-square error (RMSE) and the Hausdorff distance (d_H), are considered. The RMSE was computed by reading the ground truth distance values at the location of predicted surface samples for each method. It indicates how accurately each sample was predicted in an average sense. On the other hand, d_H measures an adversarial case of errors and becomes large when there are consecutive missing points or large deviations. Overall, the accuracy of TSDF and GPIS was higher than OG map because TSDF achieves the sub-grid precision by interpolation and GPIS deals with continuous values. With lesser data, RMSE appears to decrease and d_H becomes larger with larger variances in the case of OGM and TSDF. Interestingly, when more data are used, OGM and TSDF had thicker or more jittered surface contours resulting in slightly higher RMSE. When less data is employed, there were higher chances to miss each cell and d_H was largely affected by the luck on which specific data frames are received. While the grid-based methods suffered a lack of repeated observations, the suggested GPIS showed robust and consistent results for sparse observations.

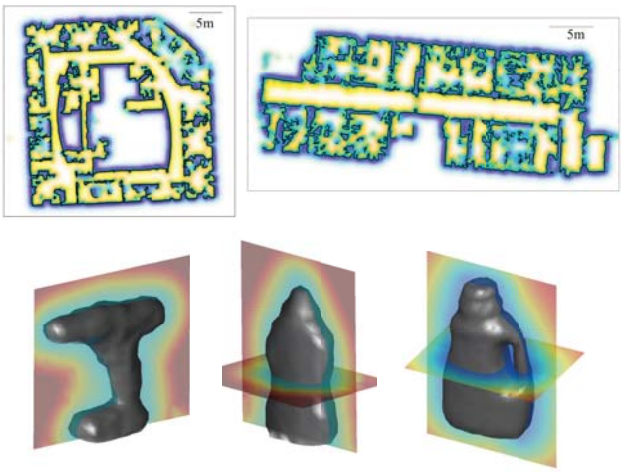

Fig. 7. Surface Modeling Results on Real 2D/3D Data: (Top-Left) Intel Lab and (Top-Right) Freiburg Building 079 from [30], (Bottom) Daily Objects from YCB [31] and BigBIRD [32] datasets. The SDF is visualized in the 'jet' colormap with an arbitrary scale. (Best viewed in color)

Experiments on Real Data: Unlike simulation with nicely arranged surfaces and simpler sensor models, real world data includes uncertain pose estimates, structures of various sizes, non-Gaussian sensor noise. We tested our method on public datasets and some results are shown in Fig.7. The top two images show mapping results using 2D lidar data. Although the details beyond the used map resolution ($0.1\ m$) are limited, we can clearly see the surfaces of the walls for the rooms and hallways. The limitation by a single resolution could be overcome by a hierarchical implementation of GPIS with varied scales [22], [21]. The bottom images are object surface modeling results using 3D depth images. The zero-level iso-surfaces are visualized as a mesh and the SDF values of selected planes are colored in an arbitrary scale. In these cases, about forty viewpoints were used and the resolution of the model was chosen as $r = 0.01m$ to reflect the scale of objects. The SDF with its gradients will be useful for precise interactions such as grasping.

Computation Time: The computation time per frame for 2D cases including real data usually took less than 20 ms with a multi-thread implementation of Step 4 (Fig. 8 (a)). In order to observe the computational complexity with respect to the map size, we generated a large 2D map by increasingly tiling the Intel Lab map of Fig. 7 which is roughly $1,000m^2$. The parameters used to obtain the result were as follows: the minimum resolution $r = 0.1$, the cluster interval $r^{l_c} = 0.4$, and the cluster radius $R_c = 2r^{l_c}$. Fig. 8 (b) shows the resulting median update time for completing each *tile*. The majority of the time was spent in Step 2 and 4. The testing time per point did not vary much as the map grows, ranging from 4 to 6 μsec. Of course, the absolute computing time will vary according to parameter settings and the complexity of scenes. However, this logarithmic tendency demonstrates the potential scalability of our method. The computation time for our current 3D implementation took longer, < 0.6 seconds per frame, since the dimension of the covariance

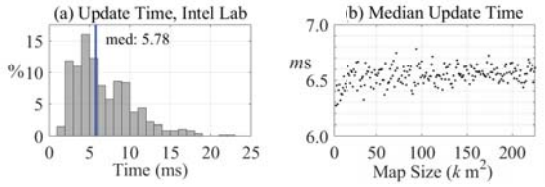

Fig. 8. Computing Time: (a) Update time distribution for mapping Intel Lab (Fig. 7 Top-left). (b) Median update time for a large map growing in 2D. Median values are reported since the distribution is not normal.

matrix was larger in addition to a larger number of points per cluster. Having said that, the reported numbers are based on serial computation of the standard Cholesky decomposition. A potential parallel Cholesky factorization [33], [34] and a more systematic multi-threaded implementation could further speed up the process.

VII. Conclusion and Future Work

We suggested an online approach to build surface models using GPIS. The GPIS representation jointly learns an approximate SDF and its derivatives near the surface. Based on the noisy input GP model, the method keeps the surface training points updated as more and more measurements are received. By introducing a regression on the discrete range measurements, our method infers surface points and normals continuously in space. Then the training points of GPIS are updated in a Bayesian fashion with noise models based on geometry at the time of observation.

In order to focus on online mapping with GPIS, we assumed a known pose from a simulator or another algorithm in this study. One of the interesting and promising issues for future study will be to add pose estimation in this framework. Performing pose estimation while structure building (a.k.a SLAM) is an expected function for autonomous robots. One fruitful approach is to use ICP-like registration methods for localization. In our representation, the distance field is readily available which can accelerate ICP computations. Thus, our immediate future study will be to extend the suggested method to include pose estimation via registration.

This study contributes a method enabling online updates of GPIS, benefits of which include direct access to the distance field and the derivatives, probabilistic inference, and continuous expression of structure. This is different from an occupancy representation which requires a distance transform or nearest-neighbor search to obtain distance information. TSDF discretizes the distance function and overestimates its values, while GPIS provides precise and continuous values near the obstacles. The ability to provide reliable distance information online is valuable to robotic tasks such as collision avoidance and precise planning. The high accuracy for structure modeling under degraded data rate was demonstrated in comparison with the other methods. The effectiveness of our method is also tested on real world data for robotics applications.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[2] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian Process implicit surfaces for shape estimation and grasping," in *IEEE International Conference on Robotics and Automation*, pp. 2845–2850, 2011.

[3] S. T. OCallaghan and F. T. Ramos, "Gaussian Process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[4] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, no. 2, pp. 61–74, 1988.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[6] S. Kim and J. Kim, "Building occupancy maps with a mixture of Gaussian Processes, year=2012, pages=4756-4761,," in *2012 IEEE International Conference on Robotics and Automation*.

[7] "Continuous occupancy maps using overlapping local Gaussian Processes, author=Kim, Soohwan and Kim, Jonghyuk, booktitle=Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, pages=4709–4714, year=2013,,"

[8] F. Ramos and L. Ott, "Hilbert Maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[9] J. Wang and B. Englot, "Fast, accurate Gaussian Process occupancy maps via test-data octrees and nested Bayesian fusion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1003–1010, 2016.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pp. 127–136, IEEE, 2011.

[11] M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially extended kinectfusion," *MIT CSAIL Tech. Rep.*, 2012.

[12] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3D reconstruction using signed distance functions.," in *Robotics: Science and Systems*, vol. 2, 2013.

[13] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, ACM, 1996.

[14] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM SIGGRAPH*, pp. 163–169, 1987.

[15] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[16] H. Pottmann, S. Leopoldseder, and M. Hofer, "Registration without icp," *Comput. Vis. Image Underst.*, vol. 95, no. 1, pp. 54–71, 2004.

[17] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, IEEE International Conference on*, pp. 489–494, 2009.

[18] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*, 2016.

[19] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. 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.

[20] M. P. Gerardo-Castro, T. Peynot, and F. Ramos, "Laser-radar data fusion with Gaussian Process implicit surfaces," in *Field and Service Robotics*, pp. 289–302, Springer, 2015.

[21] S. Kim and J. Kim, "Hierarchical Gaussian Processes for robust and accurate map building," in *Australasian Conference on Robotics and Automation*, pp. 117–124, 2015.

[22] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, "Geometric priors for Gaussian Process implicit surfaces," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 373–380, 2017.

[23] C. Plagemann, K. Kersting, P. Pfaff, and W. Burgard, "Gaussian Beam Processes: a nonparametric bayesian measurement model for range finders.," in *Robotics: Science and Systems*, 2007.

[24] A. McHutchon and C. E. Rasmussen, "Gaussian Process training with input noise," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1341–1349, 2011.

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

[26] R. A. Finkel and J. L. Bentley, "Quad Trees: a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[27] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[28] "Matlab Robotics System Toolbox." `https://www.mathworks.com/products/robotics.html`. Last Accessed: 2018-09-12.

[29] S. Kim and J. Kim, "GPmap: A unified framework for robotic mapping based on sparse Gaussian Processes," in *International Conference on Field and Service Robots*, 2013.

[30] "Cyrill Stachniss—Robotics Datasets." `http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html`. Last Accessed: 2018-09-12.

[31] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *Advanced Robotics (ICAR), 2015 International Conference on*, pp. 510–517, IEEE, 2015.

[32] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 509–516, IEEE, 2014.

[33] G. H. Golub and C. F. Van Loan, *Matrix Computations, Ch. 6*. JHU Press, 1996.

[34] A. George, M. T. Heath, and J. Liu, "Parallel Cholesky factorization on a shared-memory multiprocessor," *Linear Algebra and its applications*, vol. 77, pp. 165–187, 1986.