Dense Incremental Metric-Semantic Mapping via Sparse Gaussian Process Regression

Ehsan Zobeidi[1] Alec Koppel[2] Nikolay Atanasov[1]

***Abstract*— We develop an online probabilistic metric-semantic mapping approach for autonomous robots relying on streaming RGB-D observations. We cast this problem as a Bayesian inference task, requiring encoding both the geometric surfaces and semantic labels (e.g., chair, table, wall) of the unknown environment. We propose an online Gaussian Process (GP) training and inference approach, which avoids the complexity of GP classification by regressing a truncated signed distance function representation of the regions occupied by different semantic classes. Online regression is enabled through sparse GP approximation, compressing the training data to a finite set of inducing points, and through spatial domain partitioning into an Octree data structure with overlapping leaves. Our experiments demonstrate the effectiveness of this technique for large-scale probabilistic metric-semantic mapping of 3D environments. A distinguishing feature of our approach is that the generated maps contain full continuous distributional information about the geometric surfaces and semantic labels, making them appropriate for uncertainty-aware planning.**

I. Introduction

In the near future, robots may assist in many transportation, construction, security, and environmental monitoring services. Safe application of autonomous systems to tasks, specified in human-understandable terms, requires an understanding of both the 3-D geometry and the object identities, affordances, and operational context of the environment. This paper develops a metric-semantic mapping algorithm, using streaming RGB-D measurements onboard a robot, to reconstruct geometric surfaces and their semantic identity (e.g., chairs, tables, doors). We specifically focus on a probabilistic approach so that confidence and quantile information may be incorporated into decision-making in pursuit of certifiable safety and autonomous uncertainty reduction.

To build geometric and semantic understanding of the environment, robotic systems must discern patterns in information obtained with every measurement. While the amount of measured data keeps growing over time, the underlying geometric and semantic structure of the scene may not. Hence, an important objective for metric-semantic mapping algorithms is to build expressive maps whose memory and complexity requirements are affordable.

Occupancy mapping [1]–[8] is a problem in which one seeks to distinguish between occupied and free space. For each point in space, we explicitly store occupancy information, which may be made very accurate with sufficiently dense point clouds or voxel grids. However, the memory and computation requirements of such dense representations quickly become infeasible for large domains. Implicit function representations of geometric surfaces, for example, based on a Truncated Signed Distance Function (TSDF) [3], [9]–[12] are attractive because they enable accurate continuous surface representations with a finite number of parameters. TSDF representations are also useful for collision checking and human-understandable mesh generation. Existing work, however, either forgoes probabilistic representations in the interest of scalability or makes independence assumptions amongst the map elements (voxels, points).

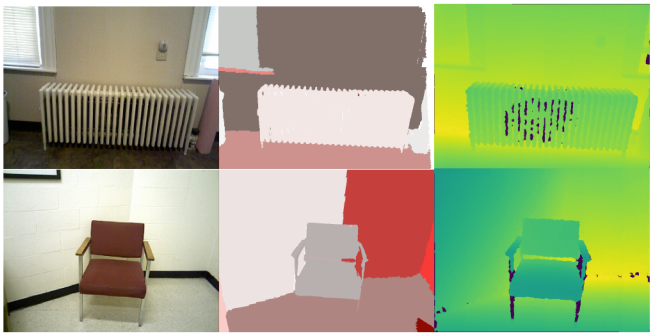

Fig. 1: RGB images (first column), segmented images (second column), and depth images (third column) used by the proposed approach for online construction of dense metric-semantic maps.

Classification of environment surfaces into semantic categories is important for context understanding and specification of complex robot tasks [13]–[15]. However, classification paradigms necessitate point estimates because efficient probabilistic classification remains an open challenge in machine learning, due to prior and data likelihoods not being conjugate (one may employ Laplace approximations to partially mitigate this challenge – see [16]).

For this reason, our focus is on TSDF *regression* for different semantic categories but in a Bayesian setting. Specifically, to incorporate spatial correlation into a probabilistic resolution-free TSDF map of the 3-D environment, we employ Gaussian Process (GP) regression. GPs are a probabilistic framework amenable to continuous map representations [17]–[19]. Unfortunately, onboard sensor data is not a direct observation of TSDF and, moreover, GP regression requires cubic complexity in the number of training examples. To ameliorate these challenges, first, we propose a transformation of the raw observations into a training dataset of TSDF values specific to different semantic classes.

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181, NSF NRI CNS-1830399, and ONR SAI N00014-18-1-2828.

[1]Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093, USA {ezobeidi,natanasov}@ucsd.edu

[2]Computational and Information Sciences Directorate, U.S. Army Research Laboratory, Adelphi, MD 20783, USA alec.e.koppel.civ@mail.mil

This data, however, grows over time, as the same scene is observed multiple times, and is not necessarily representative of the true underlying environment complexity. There are various ways to address bottlenecks of non-parametric statistics [20], [21], but in our setting, we observe that sensory data collected from identical spatial locations can be compressed significantly before GP training without affecting the TSDF posterior. Moreover, we notice that points that are far away in a map are unlikely to be correlated. In this regard, to reduce the complexity, one might consider local kriging, decomposing the spatial domain into subdomains and making predictions at a test location using only the training points contained within the subdomain. Unfortunately, discontinuities at the boundaries of the subdomains make stitching the domain together again difficult. Another ensemble methods construct multiple local estimators and use a weighted combination of their predictions, as in Bayesian committee machines [22], [23], sparse probabilistic regression, or infinite mixtures Gaussian process experts [24]. These approaches avoid the discontinuities present in local kriging at significant computational cost. Inspired by the Octomap framework [2], we propose an efficient alternative approach to remove discontinuities that decomposes the environment into an Octree of overlapping subdomains. Combining these ideas leads to an online probabilistic mapping approach that generates dense metric-semantic surfaces and, yet, remains computationally and memory efficient even in large-scale complex environments. Our main **contributions** are to:

- develop an online Gaussian Process training and inference algorithm that enables 3-D semantic segmentation of the environment through TSDF regression,
- ensure controllable computation and memory complexity while providing continuous probabilistic 3-D representations of large-scale environments,
- evaluate the proposed metric-semantic mapping approach in simulated and real-world public datasets.

Our algorithm retains full distributional information and may be used either offline, with all sensory data provided in advance, or online, processing RGB-D observations incrementally as they arrive.

II. Problem Formulation

Consider a robot navigating in an unknown environment represented as a subset of Euclidean space, $\mathcal{X} \subset \mathbb{R}^3$. The environment consists of two disjoint subsets, comprising obstacles and free space, i.e., $\mathcal{X} = \mathcal{O} \cup \mathcal{F}$. The obstacle region $\mathcal{O}$ is a closed set that is a pairwise disjoint union of N closed sets, i.e., $\mathcal{O} = \cup_{i=1}^{N} \mathcal{O}_i$. Each subset $\mathcal{O}_i$ denotes the region occupied by object instances from one semantic class. For example, $\mathcal{O}_1$ may be the space occupied by all chairs, while $\mathcal{O}_2$ may be the space occupied by all tables.

The robot is equipped with an RGB-D sensor that provides observations of the objects in its vicinity at each time step t. A semantic segmentation algorithm [25] is applied to the RGB images to obtain the classes of observed objects, while the depths of the object surfaces are provided by the depth images. Example images are provided in Fig. 1. The sensed information is defined formally below.

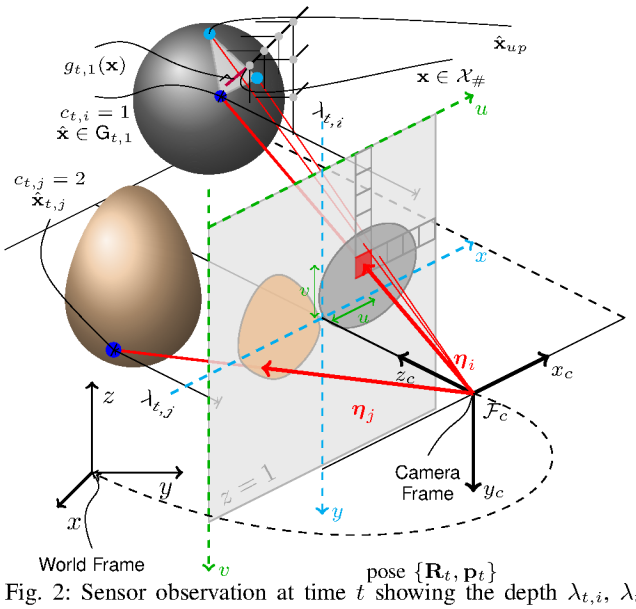

Fig. 2: Sensor observation at time t showing the depth $\lambda_{t,i}$, $\lambda_{t,j}$ and class $c_{t,i}$, $c_{t,j}$ measurements obtained along sensors rays $\boldsymbol{\eta}_i$, $\boldsymbol{\eta}_j \in \mathbf{E}$ when the sensor is at position $\mathbf{p}_t$ with orientation $\mathbf{R}_t$. The inducing points $\mathcal{X}_{\#}$ (c.f. Sec. IV-A) close to the observed surface are shown in gray.

Definition 1. A *sensor frame* $\mathbf{E} = \{\boldsymbol{\eta}_j\}_{j=1}^{M}$ is a set of vectors $\boldsymbol{\eta}_j \in \mathbb{R}^3$ such that $\mathbf{e}_3^\top \boldsymbol{\eta}_j = 1$, where $\mathbf{e}_j$ is the j-th standard basis vector.

Definition 2. A *sensor observation* at time t is a collection of depth $\lambda_{t,j} \in \mathbb{R}_+$ and object class $c_{t,j} \in \{1, ..., N\}$ measurements acquired by the RGB-D sensor along the directions of the sensor frame vectors $\boldsymbol{\eta}_j \in \mathbf{E}$.

At time t, the j-th sensor ray starts at the sensor position $\mathbf{p}_t \in \mathbb{R}^3$ and has direction $\mathbf{R}_t \boldsymbol{\eta}_j$, determined by the sensor orientation $\mathbf{R}_t \in SO(3)$. The ray may have finite length if it hits the obstacle set $\mathcal{O}$. We define the relationship among the object sets $\mathcal{O}_i$ and the sensor observations $\lambda_{t,j}$, $c_{t,j}$ next. The space $\mathcal{O}_i$, occupied by the i-th object class, is defined implicitly as the level set of a signed distance function.

Definition 3. The *directional truncated signed depth function* (DTSDF) $h_i(\mathbf{x}, \boldsymbol{\eta})$ of object class $\mathcal{O}_i$, is the signed depth from $\mathbf{x} \in \mathcal{X}$ to the boundary $\partial\mathcal{O}_i$ in direction $\boldsymbol{\eta} \in \mathbb{R}^3$, i.e.,

$$h_i(\mathbf{x}, \boldsymbol{\eta}) := \begin{cases} -\min\left(d_{\boldsymbol{\eta}}(\mathbf{x}, \partial\mathcal{O}_i), \bar{d}\right) & \text{if } \mathbf{x} \in \mathcal{O}_i \\ \min\left(d_{\boldsymbol{\eta}}(\mathbf{x}, \partial\mathcal{O}_i), \bar{d}\right) & \text{if } \mathbf{x} \in \mathcal{X} \setminus \mathcal{O}_i, \end{cases}$$

$$d_{\boldsymbol{\eta}}(\mathbf{x}, \partial\mathcal{O}_i) := \min\left\{d \geq 0 \;\middle|\; \mathbf{x} + d\boldsymbol{\eta} \in \partial\mathcal{O}_i\right\}, \tag{1}$$

where the depth is truncated to a maximum of $\bar{d} \in \mathbb{R}_+$.

Def. 3 states that $h_i(\mathbf{p}_t, \mathbf{R}_t \boldsymbol{\eta}_j)$ is the depth from sensor position $\mathbf{p}_t$ to obstacle set $\mathcal{O}_i$ along the direction $\mathbf{R}_t \boldsymbol{\eta}_j$ of the j-th ray at time t. The class observation $c_{t,j}$ is determined by the object set $\mathcal{O}_i$ with minimum absolute DTSDF to $\mathbf{p}_t$ along $\mathbf{R}_t \boldsymbol{\eta}_j$:

$$c_{t,j} = \underset{i \in \{1,...,N\}}{\arg\min} \; |h_i(\mathbf{p}_t, \mathbf{R}_t \boldsymbol{\eta}_j)|. \tag{2}$$

The depth observation $\lambda_{t,j}$ is a noisy measurement of the depth $h_{c_{t,j}}(\mathbf{p}_t, \mathbf{R}_t\boldsymbol{\eta}_j)$ to the nearest object class:

$$\lambda_{t,j} = h_{c_{t,j}}(\mathbf{p}_t, \mathbf{R}_t\boldsymbol{\eta}_j) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{3}$$

where σ^2 is the variance of the depth measurement noise. These definitions are illustrated in Fig. 2.

Given sensor poses $\mathbf{p}_t$, $\mathbf{R}_t$ and streaming onboard observations $\lambda_{t,j}$, $c_{t,j}$ for $j \in \{1, \dots, M\}$ and $t \in \{1, \dots, T\}$, the main objective of this work is to construct a metric-semantic map of the observed environment online by estimating the functions $h_i(\mathbf{x}, \boldsymbol{\eta})$ for $i = 1, \dots, N$. Note that $\{h_i\}$ implicitly define the object sets $\mathcal{O}_i = \{\mathbf{x} \in \mathcal{X} \mid \min_{\boldsymbol{\eta}} h_i(\mathbf{x}, \boldsymbol{\eta}) \leq 0\}$. The DTSDF $h_i(\mathbf{x}, \boldsymbol{\eta})$ is defined for an arbitrary direction $\boldsymbol{\eta}$. To reduce the dimension of its domain, we define its minimum over the directions $\boldsymbol{\eta}$.

Definition 4. The *truncated signed depth function* (TSDF) $f_i(\mathbf{x})$ of object class $\mathcal{O}_i$ is the signed depth from $\mathbf{x} \in \mathcal{X}$ to the boundary $\partial\mathcal{O}_i$, i.e.,

$$f_i(\mathbf{x}) := h_i(\mathbf{x}, \boldsymbol{\eta}^*) \text{ where } \boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} |h_i(\mathbf{x}, \boldsymbol{\eta})| \tag{4}$$

We propose an online Gaussian Process (GP) regression approach to maintain a distribution $\mathcal{GP}(\mu_{t,i}(\mathbf{x}), k_{t,i}(\mathbf{x}, \mathbf{x}'))$ over the TSDF functions $f_i(\mathbf{x})$ conditioned on the sensor observations $\{\lambda_{k,j}, c_{k,j}\}_{k=1,j=1}^{t,M}$ up to time t. In Sec. IV, we discuss techniques for online training over a finite set of inducing points and present the equations for inferring the depth and class distributions at arbitrary (test) locations $\mathbf{x}$ in the environment $\mathcal{X}$. Since we target online training and inference, in Sec. V, we partition the domain $\mathcal{X}$ into overlapping regions, organized in an octree data structure, and perform independent regression in each region.

III. Gaussian Process Regression Background

A *Gaussian Process* (GP) is a set of random variables such that the joint distribution of any finite subset of them is Gaussian. A GP-distributed function $f(\mathbf{x}) \sim \mathcal{GP}(\mu_0(\mathbf{x}), k_0(\mathbf{x}, \mathbf{x}'))$ is defined by a mean function $\mu_0(\mathbf{x})$ and a covariance (kernel) function $k_0(\mathbf{x}, \mathbf{x}')$. The mean and covariance are such that for any finite set $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the random vector $f(X) := [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^\top \in \mathbb{R}^n$ has mean with j-th element $\mu_0(\mathbf{x}_j)$ and covariance matrix with (j,k)-th element $k_0(\mathbf{x}_j, \mathbf{x}_k)$ for $j, k = 1, \dots, K$. Given a training dataset $\mathsf{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^n$ generated according to $y_j = f(\mathbf{x}_j) + \epsilon_j$ with noise $\epsilon_j \sim \mathcal{N}(0, \sigma_j^2)$, the posterior distribution of the random function $f(\mathbf{x})$ can be obtained from the joint distribution of the value $f(\mathbf{x})$ at an arbitrary location $\mathbf{x}$ and the random vector $\mathbf{y} := [y_1, \dots, y_n]^\top$ of the measurements. In detail, the joint distribution is:

$$\begin{bmatrix} f(\mathbf{x}) \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_0(\mathbf{x}) \\ \mu_0(X) \end{bmatrix}, \begin{bmatrix} k_0(\mathbf{x}, \mathbf{x}) & k_0(\mathbf{x}, X) \\ k_0(X, \mathbf{x}) & k_0(X, X) + D \end{bmatrix} \right), \tag{5}$$

where $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with $D_{i,i} = \sigma_i^2$, while the corresponding conditional distribution $f(\mathbf{x}) | \mathsf{D} \sim \mathcal{GP}(\mu_n(\mathbf{x}), k_n(\mathbf{x}, \mathbf{x}'))$ has mean and covariance functions:

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + k_0(\mathbf{x}, X)(k_0(X, X) + D)^{-1}(\mathbf{y} - \mu_0(X)),$$
$$k_n(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') - k_0(\mathbf{x}, X)(k_0(X, X) + D)^{-1}k_0(X, \mathbf{x}').$$

Computing the GP posterior has cubic complexity in the number of observations n due to the $n \times n$ matrix inversion.

IV. Probabilistic Metric-Semantic Mapping

While the sensor measurements $\{\lambda_{t,j}, c_{t,j}\}$ are generated according to the models in (2) and (3) that depend on $\{h_i(\mathbf{x}, \boldsymbol{\eta})\}$, we focus on approximating the TSDF functions $\{f_i(\mathbf{x})\}$, whose domains are lower-dimensional. Hence, the sensor data $\{\lambda_{t,j}, c_{t,j}\}_{j=1}^M$ is first transformed into training sets $\mathsf{D}_{t,i}$, suitable for updating the distribution of $\{f_i(\mathbf{x})\}$.

A. Training Set Construction

The class measurements allow us to associate the sensed data with a particular semantic class, while the depth measurements allow us to estimate the points where the sensor rays hit the object sets $\mathcal{O}_i$. In detail, we define the following point sets for each detected semantic class at time t:

$$\mathsf{G}_{t,i} = \{\hat{\mathbf{x}} \in \mathbb{R}^3 \,\big|\, \hat{\mathbf{x}} = \lambda_{t,j}\mathbf{R}_t\boldsymbol{\eta}_j + \mathbf{p}_t \text{ and } c_{t,j} = i\}. \tag{6}$$

The points $\hat{\mathbf{x}} \in \mathsf{G}_{t,i}$ lie in the continuous space $\mathcal{X}$ and the values $f_i(\hat{\mathbf{x}})$ of the TSDFs are close to zero since the sensor rays hit an object surface at these locations.

As shown in Prop. 1 below, the complexity of online GP training can be improved by forcing the training data to come from a finite set of points $\mathcal{X}_\# \subset \mathcal{X}$, such as a grid discretization (see Fig. 2). In detail, we choose points $\mathbf{x} \in \mathcal{X}_\#$ that are at most $\epsilon > 0$ away from the points in $\mathsf{G}_{t,i}$ and approximate their TSDF values $f_i(\mathbf{x}) \approx g_{t,i}(\mathbf{x})$. Precisely, the training data sets are defined at time t as:

$$\mathsf{D}_{t,i} = \{(\mathbf{x}, g_{t,i}(\mathbf{x})) | \mathbf{x} \in \mathcal{X}_\#, \exists \hat{\mathbf{x}} \in \mathsf{G}_{t,i} \text{ s.t. } ||\mathbf{x} - \hat{\mathbf{x}}||_2 \leq \epsilon\}. \tag{7}$$

The TSDF value $g_{t,i}(\mathbf{x})$ of a training point $\mathbf{x}$ is obtained by projecting $\mathbf{x}$ to the depth image plane and approximating its depth from the depth values of nearby pixels. In detail, suppose $\boldsymbol{\eta}_j$ is the pixel closest to the projection of $\mathbf{x}$ (red pixel in Fig. 2) and let $\hat{\mathbf{x}}_j \in \mathsf{G}_{t,i}$ be the coordinates of its ray endpoint (blue point in Fig. 2). Let $\hat{\mathbf{x}}_{right}$ and $\hat{\mathbf{x}}_{up}$ (two cyan points in Fig. 2) be the ray endpoints of two adjacent pixels. Then, $g_{t,i}(\mathbf{x})$ is the signed distance from $\mathbf{x}$ to the plane defined by $\hat{\mathbf{x}}_j$, $\hat{\mathbf{x}}_{right}$, and $\hat{\mathbf{x}}_{up}$:

$$g_{t,i}(\mathbf{x}) := \mathbf{n}^\top(\mathbf{x} - \hat{\mathbf{x}}_j), \quad \mathbf{n} := \text{sign}(\mathbf{m}^\top(\mathbf{p}_t - \hat{\mathbf{x}}_j))\mathbf{m},$$
$$\mathbf{m} = \frac{(\hat{\mathbf{x}}_{right} - \hat{\mathbf{x}}) \times (\hat{\mathbf{x}}_{up} - \hat{\mathbf{x}})}{\|(\hat{\mathbf{x}}_{right} - \hat{\mathbf{x}}) \times (\hat{\mathbf{x}}_{up} - \hat{\mathbf{x}})\|}, \tag{8}$$

where $\mathbf{m}$ is the normal of the plane and the signed distance from $\mathbf{p}_t$ to the plane is positive because the sensor is known to be outside of the object set $\mathcal{O}_i$.

B. Gaussian Process Regression

Let $\mathcal{GP}(\mu_{t-1,i}(\mathbf{x}), k_{t-1,i}(\mathbf{x}, \mathbf{x}'))$ be prior GP distributions over the signed depth functions $f_i(\mathbf{x})$ conditioned on past data $\mathsf{D}_{t-1,i}, \dots, \mathsf{D}_{1,i}$. We take $\mu_{0,i}(\mathbf{x}) = L$ where L is truncation value in TSDF, and we use sparse matern covariance function as the kernel $k_{0,i}(\mathbf{x}, \mathbf{x}')$. Given the training set $\mathsf{D}_{t,i}$ constructed at time t, we seek to compute the posteriors $f_i(\mathbf{x}) \,\big|\, \mathsf{D}_{t,i}, \dots, \mathsf{D}_{1,i} \sim \mathcal{GP}(\mu_{t,i}(\mathbf{x}), k_{t,i}(\mathbf{x}, \mathbf{x}'))$

for each semantic class i. The complexity of computing a GP posterior scales cubically with the number of observations in the training set $\mathsf{D}_{t,i}$, limiting the applicability to online settings as mentioned in Sec. III. We make a key observation that limiting the training data points $\mathbf{x}$ to come from a finite set $\mathcal{X}_{\#}$, as done in Sec. IV-A, may be used to reduce the GP traning complexity from cubic in the number of observations $\sum_t |D_{t,i}|$, which grows unbounded with time, to cubic in the number of distinct observed points from $\mathcal{X}_{\#}$. We formalize this in the following theorem, establishing that the GP posterior remains unchanged if we merge training data obtained from the same spatial locations.

Proposition 1. *Consider $f(\mathbf{x}) \sim \mathcal{GP}(\mu_0(\mathbf{x}), k_0(\mathbf{x}, \mathbf{x}'))$. Let:*

$$X = \{\mathbf{x}_1 \ , \ldots, \mathbf{x}_1 \quad , \mathbf{x}_2 \ , \ldots, \mathbf{x}_2 \quad , \ldots, \mathbf{x}_n \ , \ldots, \mathbf{x}_n \quad \}$$
$$Y = \{y_{1,1}, \ldots, y_{1,m_1}, y_{2,1}, \ldots, y_{2,m_2}, \ldots, y_{n,1}, \ldots, y_{n,m_n}\}$$

be data generated from the model $y_{i,j} = f(\mathbf{x}_i) + \eta_{i,j}$ with $\eta_{i,j} \sim \mathcal{N}(0, \sigma_i^2)$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$. Let:

$$\hat{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \ \hat{Y} = \left\{ \frac{1}{m_1} \sum_{j=1}^{m_1} y_{1,j}, \ldots, \frac{1}{m_n} \sum_{j=1}^{m_n} y_{n,j} \right\}$$

be a compressed version of the data generated from the model $f(\mathbf{x}_i)$ with noise $\hat{\eta}_i \sim \mathcal{N}(0, \frac{\sigma_i^2}{m_i})$. Then, $f(\mathbf{x})|X, Y$ and $f(\mathbf{x})|\hat{X}, \hat{Y}$ have the same Gaussian Process distribution $\mathcal{GP}(\mu_n(\mathbf{x}), k_n(\mathbf{x}, \mathbf{x}'))$ with:

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + k_0(\mathbf{x}, \hat{X})(\hat{D} + k_0(\hat{X}, \hat{X}))^{-1}(\hat{\boldsymbol{\zeta}} - \mu_0(\hat{X})),$$
$$k_n(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') - k_0(\mathbf{x}, \hat{X})(\hat{D} + k_0(\hat{X}, \hat{X}))^{-1} k_0(\hat{X}, \mathbf{x}'),$$

where $\hat{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with $\hat{D}_{i,i} = \frac{\sigma_i^2}{m_i}$ and $\hat{\boldsymbol{\zeta}} \in \mathbb{R}^n$ is a vector with elements $\hat{\zeta}_i := \frac{1}{m_i} \sum_{j=1}^{m_i} y_{i,j}$.

Proof. See Appendix A. □

Prop. 1 shows that the complexity of online GP training can be reduced without changing the posterior if the training datasets $\cup_t \mathsf{D}_{t,i}$ are compressed. In detail, the theorem allows us to summarize the data by simply keeping set of distinct training points $\mathsf{P}_{t,i} \subset \mathcal{X}_{\#}$, the number of times $m_{t,i}(\mathbf{x})$ that each point $\mathbf{x} \in \mathsf{P}_{t,i}$ has been observed up to time t and the average of the measured TSDF values $\bar{g}_{t,i}(\mathbf{x}) := \frac{1}{m_{t,i}(\mathbf{x})} \sum_{\tau=1}^{m_{t,i}(\mathbf{x})} g_{\tau,i}(\mathbf{x})$ at the observed points. Assuming same noise variance σ^2 at all primary observations, the precision matrix in Prop. 1 is $\boldsymbol{\Omega}_{t,i} = (\sigma^2 \text{diag}(m_{t,i}(\mathsf{P}_{t,i}))^{-1} + k_{0,i}(\mathsf{P}_{t,i}, \mathsf{P}_{t,i}))^{-1}$. Given these statistics, the mean functions $\mu_{t,i}(\mathbf{x})$ and covariance functions $k_{t,i}(\mathbf{x}, \mathbf{x}')$ of the GP distributions of $f_i(\mathbf{x})$ can be obtained at any time t from Prop. 1.

$$\mu_{t,i}(\mathbf{x}) = \mu_0(\mathbf{x}) + k_0(\mathbf{x}, \mathsf{P}_{t,i}) \boldsymbol{\Omega}_{t,i} (\bar{g}_{t,i}(\mathsf{P}_{t,i}) - \mu_0(\mathsf{P}_{t,i})),$$
$$k_{t,i}(\mathbf{x}, \mathbf{x}') = k_0(\mathbf{x}, \mathbf{x}') - k_0(\mathbf{x}, \mathsf{P}_{t,i}) \boldsymbol{\Omega}_{t,i} k_0(\mathsf{P}_{t,i}, \mathbf{x}'), \tag{9}$$

We discuss the implementation details of online training and prediction next.

1) Online GP Training: For each class $i \in \{1, \ldots, N\}$, we keep two hashmap data structures over $\mathcal{X}_{\#}$ to represent the number of observations $m_{t,i}(\mathbf{x})$ and the average TSDF values $\bar{g}_{t,i}(\mathbf{x})$. Given the sensor data $\mathsf{D}_{t+1,i}$ at time $t+1$, we update the hashmaps for each $(\mathbf{x}, g) \in \mathsf{D}_{t+1,i}$ as:

$$\begin{aligned} m_{t+1,i}(\mathbf{x}) &= m_{t,i}(\mathbf{x}) + 1 \\ \bar{g}_{t+1,i}(\mathbf{x}) &= \bar{g}_{t,i}(\mathbf{x}) + \frac{1}{m_{t+1,i}(\mathbf{x})} (g - \bar{g}_{t,i}(\mathbf{x})) \end{aligned} \tag{10}$$

2) Online GP Prediction: If we need the online prediction we should keep track of $\boldsymbol{\Omega}_{t,i}$, otherwise $\boldsymbol{\Omega}_{T,i}$ is sufficient. At time step $t+1$, in order to achieve $\boldsymbol{\Omega}_{t+1,i}$ from $\boldsymbol{\Omega}_{t,i}$, we have two kind of new observation in $\mathsf{D}_{t+1,i}$. We either have seen some of training points in $\mathsf{P}_{t,i}$ again, let call them $\tilde{\mathsf{P}}$, or we have seen new training points $\mathsf{P}' = \mathsf{P}_{t+1,i} \setminus \mathsf{P}_{t,i}$. First we update precision matrix regarding the first kind, let $\boldsymbol{\Omega}'_j$ be the updated precision matrix for first j elements of $\tilde{\mathsf{P}}$. Let $\mathbf{x}$ be the $j+1$-th element of $\tilde{\mathsf{P}}$ in order to update precision matrix regarding $\mathbf{x}$ and achieving $\boldsymbol{\Omega}'_{j+1}$ from $\boldsymbol{\Omega}'_j$, let $\delta = m_{t+1,i}(\mathbf{x}) - m_{t,i}(\mathbf{x}), \epsilon = \frac{\sigma^2}{m_{t+1,i}(\mathbf{x})} - \frac{\sigma^2}{m_{t,i}(\mathbf{x})}$, and l to be the index of $\boldsymbol{\Omega}_{t,i}$ corresponding with $\mathbf{x}$:

$$\boldsymbol{\Omega}'_{j+1} = ({\boldsymbol{\Omega}'_j}^{-1} + \delta \mathbf{e}_l \mathbf{e}_l^\top)^{-1} = \boldsymbol{\Omega}'_j - \frac{(\boldsymbol{\Omega}'_j \mathbf{e}_l)^\top \boldsymbol{\Omega}'_j \mathbf{e}_l}{\frac{1}{\epsilon} + \mathbf{e}_l^\top \boldsymbol{\Omega}'_j \mathbf{e}_l} \tag{11}$$

3) Batch GP Prediction: Let $\tilde{\boldsymbol{\Omega}} = \boldsymbol{\Omega}'_{|\tilde{\mathsf{P}}|}$, then if we update precision matrix $\tilde{\boldsymbol{\Omega}}$ with second kind of new observation we will have $\boldsymbol{\Omega}_{t+1,i}$, let $\boldsymbol{\Sigma}' = \sigma^2 \text{diag}(m_{t+1,i}(\mathsf{P}'))^{-1} + k_{0,i}(\mathsf{P}', \mathsf{P}'), \mathbf{K} = k_{0,i}(\mathsf{P}_{t,i}, \mathsf{P}'), \mathbf{C} = \tilde{\boldsymbol{\Omega}}\mathbf{K}, \mathbf{E} = (\boldsymbol{\Sigma}' - \mathbf{K}^\top \mathbf{C})^{-1}, \mathbf{F} = -\mathbf{C}\mathbf{E}$ then:

$$\boldsymbol{\Omega}_{t+1,i} = \begin{bmatrix} \tilde{\boldsymbol{\Omega}}^{-1} & \mathbf{K} \\ \mathbf{K}^T & \boldsymbol{\Sigma}' \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{\boldsymbol{\Omega}} - \mathbf{F}\mathbf{C}^T & \mathbf{F} \\ \mathbf{F}^T & \mathbf{E} \end{bmatrix} \tag{12}$$

C. Semantic Class Prediction

In this section, we discuss how to predict the semantic class labels on the surfaces of the implicitly estimated object sets $\mathcal{O}_i$. While we did not explicitly model noise in the class observations in (2), in practice, semantic segmentation algorithms may produce incorrect pixel-level classification. This may lead to some sensor observations $\lambda_{t,j}$, $c_{t,j}$ being incorrectly included into the training set $\mathsf{D}_{t,i}$ of a different object class. This happens, for example, if objects from two different classes, say i_1 and i_2, are spatially close to each other and, in an RGB image observation, parts of the boundary of one are classified as belonging to the other class. Over time, with multiple sensor observations, the TSDF approximations for both classes i_1 and i_2 may contain training points $\mathbf{x} \in \mathcal{X}_{\#}$ with small TSDF values, indicating an object class surface. In order to predict the correct object class when such a situation happens, we compare the likelihoods of the different classes at surface points given the posterior TSDF distributions of the functions $f_i(\mathbf{x})$. In detail, consider a surface point, i.e., $\mathbf{x} \in \mathcal{X}$ such that $f_i(\mathbf{x}) = 0$ for some class i. The likelihood that the class label of $\mathbf{x}$ is $c \in \{1, \ldots, N\}$ is provided below.

Proposition 2. *Let $\mathcal{GP}(\mu_{t,i}(\mathbf{x}), k_{t,i}(\mathbf{x}, \mathbf{x}'))$ be the distributions of the signed depth functions $f_i(\mathbf{x})$ at time t, determined according to* (9)*. Consider an arbitrary point $\mathbf{x} \in \mathcal{X}$ on the*

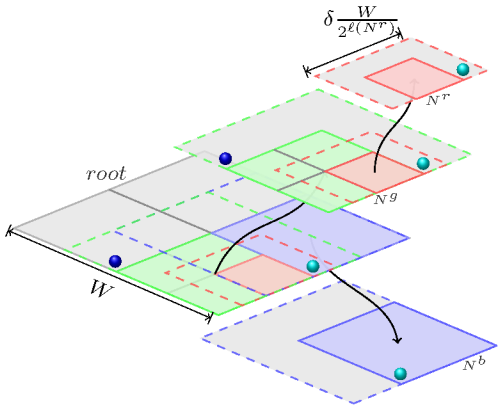

Fig. 3: Illustration of the octree data structure in two dimensions. For three nodes N^r, N^g, N^b, the regions for their $\mathcal{S}(\cdot), \mathcal{T}(\cdot)$ are respectively filled and dashed with red, green, blue colors. For all nodes $Max(N) = 1$, so node N^g splits. Note that there is no training point in the filled green region ($\mathcal{T}(N^g)$), but two training points observed in $\mathcal{S}(N^g)$. Additionally $N^r \in children(N^g), \ell(N^r) = 2$. We see the cyan training point is in both $\mathsf{P}^{N^b}_{t,c}, \mathsf{P}^{N^r}_{t,c}$.

surface of the obstacle set $\mathcal{O}$, i.e., $\mathbf{x}$ is such that $f_i(\mathbf{x}) = 0$ for some class $i \in \{1, \dots, N\}$. Then, the probability that the true class label of $\mathbf{x}$ is $c \in \{1, \dots, N\}$ is:

$$\mathbb{P}\left(\arg\min_i |f_i(\mathbf{x})| = c \;\middle|\; \min_i |f_i(\mathbf{x})| = 0\right) = \frac{\frac{1}{\sigma_{t,c}(\mathbf{x})}\phi\left(\frac{\mu_{t,c}(\mathbf{x})}{\sigma_{t,c}(\mathbf{x})}\right)}{\sum_i \frac{1}{\sigma_{t,i}(\mathbf{x})}\phi\left(\frac{\mu_{t,i}(\mathbf{x})}{\sigma_{t,i}(\mathbf{x})}\right)},$$

where $\phi(\cdot)$ is the density of the standard normal distribution and $\sigma_{t,i}(\mathbf{x}) := \sqrt{k_{t,i}(\mathbf{x}, \mathbf{x})}$.

Proof. See Appendix B. □

The class distribution for an arbitrary point, not lying on an object surface, may also be obtained, as shown in the proof of Prop. 2, but is both less efficient to compute and rarely needed in practice.

V. Octree of Gaussian Processes

Prop. 1 allows us to compress the TSDF training data over time to a finite set of distinct training points. However, the complexity of GP training still scales cubically in the number of distinct training points. To ensure that online training is possible, we propose an octree data structure with overlapping blocks to store the training points. We train independent GPs in each of these blocks, which is efficient since the number of training points per block is small. The block overlap serves to eliminate discontinuities in the resulting TSDF estimate. At test time, the TSDF value of a query point is inferred using only the parameters of the corresponding block according to Eq. 9.

An octree of training points is a tree data structure such that each internal node has eight children, and recursively subdivided into octants based on their number of training points in order to partition a W-length side cube out of environment. Each node N of the octree with point $ctr(N)$ as center of its cube has following specifications:

1) $\ell(N)$: level of node N in the tree is $\ell(N) \geq 0$. In level zero there is one node which is the root of the tree.
2) $\mathcal{S}(N) := \{\mathbf{x} \in \mathcal{X} | \ \|\mathbf{x} - ctr(N)\|_\infty \leq \delta \frac{W}{2^{\ell(N)+1}}\}$ is support of node N. Set of points in the $\mathcal{X}_\#$ that are potent to be assigned to this node as observed training points is $\mathcal{S}(N) \cap \mathcal{X}_\#$. Node N splits into eight children if number of observed training points in its support exceeds $Max(N).(\delta > 1)$
3) $\mathcal{T}(N) := \{\mathbf{x} \in \mathcal{X} | \ \|\mathbf{x} - ctr(N)\|_\infty \leq \frac{W}{2^{\ell(N)+1}}\}$, prediction for test points in this region is evaluated by GP trained by observed training points in $\mathcal{S}(N)$.
4) $children(N)$: It is empty if N is a leaf in the octree, otherwise is a set of eight nodes with their centers in $\{ctr(N) + s_x\mathbf{e}_1 + s_y\mathbf{e}_2 + s_z\mathbf{e}_3 | s_x, s_y, s_z \in \{+\frac{W}{2^{\ell(N)+1}}, -\frac{W}{2^{N_{lvl}+1}}\}\}$. Their level is $\ell(N) + 1$.

GP in each node will be calculated with observed training points in $\mathcal{S}(N)$. At time step t, for class i, let show observed training points in node N, with $\mathsf{P}^N_{t,i} = \mathsf{P}_{t,i} \cap \mathcal{S}(N)$. The other GP variables like $\mathbf{\Omega}^N_{t,i}$ are defined accordingly.

VI. Evaluation

In this section, we apply our method to a 2-D simulation, compare its performance to the incremental Euclidean signed distance mapping method Fiesta [12] on the Cow and Lady dataset [11], and demonstrate its 3-D semantic reconstruction performance on the SceneNN dataset [26]. In all experiments, sparse Matérn kernel ($v = 3/2$) [23] is used.

We set $\mathcal{X}_\#$ to be a grid with resolution $voxel\ size$. Given a query point $\hat{\mathbf{x}}$, we choose a cubic frame around it such that $(frame\ size - 1) \times voxel\ size \geq 2 \times \epsilon$. All points from $\mathcal{X}_\#$ that lie in the cubic frame are chosen as training points associated with $\hat{\mathbf{x}}$.

A. 2-D Simulation

In this section, we generate a random 2-D environment (see Fig. 4), obtain random observations using a simulated depth-class sensor, and then apply our method to obtain a map. Then we compare it with the ground truth.

1) TSDF Accuracy: A sample of 2-D simulation with its TSDF and boundaries (ground truth and constructed) is shown in Fig. 4. Truncation value of TSDF is very dependent on the $frame\ size$. Larger $frame\ size$ allows estimating larger truncation value, but incurs additional computational cost. Our method provides continuous TSDF. In order to evaluate our method's precision in estimating TSDF and its resistance against noise, all test points are picked within truncation value, out of a grid with its resolution to be half of the $voxel\ size$. The result is shown upright in Fig. 5.

2) Classification Accuracy: In each map we pick random points out of boundaries, and calculate the error in the signed distance field and accuracy of class detection. For each class we calculate precision and recall. Since everything is symmetric for both classes red and blue, we present the average over two classes as precision and recall. In Fig. 5 we produce 50 random maps and take the average of mentioned variables over all maps. As we see in all curves, they are not very sensitive to class error probability. In Fig. 6 we investigate the effect of the parameters of our algorithm on measures Misclassification Rate i.e. the ratio of samples with

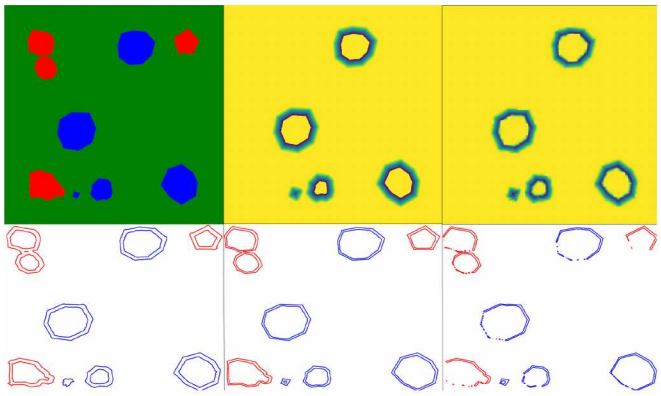

Fig. 4: On the first row, from left, first two are ground truth of environment and TSDF of the blue class, the third is constructed TSDF with $frame\ size = 10$. On the second row, we see constructed boundaries for different $frame\ size = 10, 3, 2$. The sharp edges are captured better in $frame\ size$ 3 vs. 10, but using $frame\ size$ less that 3 caused missing parts of the boundaries.

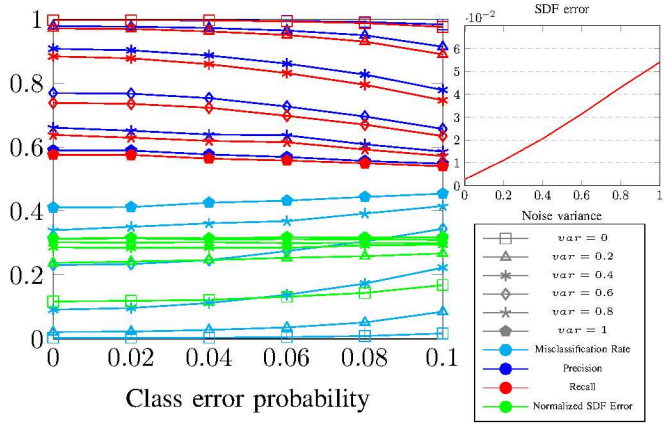

Fig. 5: On the left Misclassification Rate, Precision, Recall, Normalized SDF Error are plotted for different class error probability and noise variance on the rays. Upright shows SDF error, average of SDF errors over 10 random maps with 100 random observations for each, with $voxel\ size = 0.1$, $Max(N) = 100$, $\delta = 1.2$.

wrong detected class, Normalized SDF Error$= \frac{\text{SDF error}}{voxel\ size}$ in which SDF error is average of absolute value of difference between estimated signed distance value and the real signed distance value, FDR $= 1 -$ Precision, and FNR $= 1 -$ Recall. We see in all figures Misclassification Rate, FDR, and FNR behave closely. Increasing $Max(N)$ improves Normalized SDF Error. At first the improvement is considerable, but then even with exponential increase in $Max(N)$, Normalized SDF Error does not change significantly. Class measures at first slightly improve. Increasing δ also has a similar effect on all the measures. Increasing Gaussian Process noise variance at first improves all the measures, then it worsens them. Its wrong chosen value is very crucial to misclassification rate, but it changes very smoothly, i.e. if the value is in the right region, its exact optimum value does not matter.

B. Cow and Lady Dataset

In this section, we investigate the performance of our method on the cow and lady dataset, and compare it to Fiesta which builds ESDF map incrementally for voxels, by updating the effect of observed obstacles in two independent

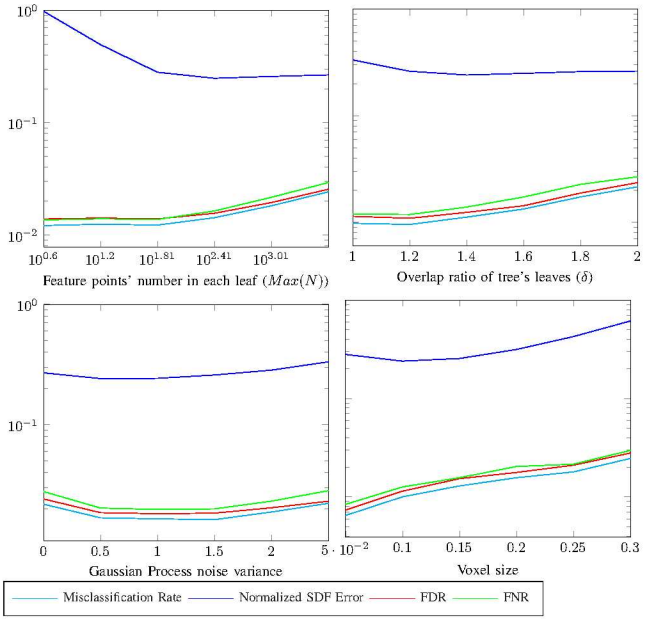

Fig. 6: The default values are $\delta = 1.5$, $Max(N) = 100$, Gaussian Process noise variance $= 1$, $voxel\ size = 0.1$. The class error probability is 0.05 and the variance noise of the rays is 0.5. The test points are out of boundaries consecutive average distance 0.05. The measures are from 100 random observations in each map, then getting average over 50 random maps, and classes.

queues, for inserting and deleting them. Increasing $Max(N)$ improves SDF error which is significant at first, but then improvement is not considerable. The time at first decreases, because the number of leaves decreases, then it increases, since the GP kernel matrix gets bigger. Increasing δ improves the SDF error, but not significantly. Although, it increases time significantly. Increasing the GP noise variance improves the SDF error at first, then it worsens it. It does not have a meaningful effect on the time. When it is close to zero, it affects the SDF error significantly. In Fig. 7 we can see our method improved the error of Fiesta significantly, when the $voxel\ size$ varies. Fiesta's parameters are set as their default.

C. SceneNN Dataset

For the 3-D reconstruction with classification evaluation of our method we apply it to the dataset SceneNN [26]. In this part we evaluate on a test grid with resolution $\frac{voxel\ size}{2}$, then we consider the points out of the new grid with confidence more than some threshold. Then we take the mesh of zero value to be the surface. For the classification we use Prop. 2. We can see the effect of different parameters on the performance in Fig. 8. Increasing $Max(N)$ improves both classification and signed distance field. The improvement after 100 is negligible, but time increases significantly. Increasing δ improves the TSDF specifically, since the shape representation improves, but after 1.4 the improvement is negligible. As we have seen in the 2-D simulation, choosing the wrong value of GP variance is crucial to both TSDF and the classification, but it is not sensitive to the optimal value.

VII. Conclusion

This paper developed a Bayesian inference method for online probabilistic metric-semantic mapping via scalable

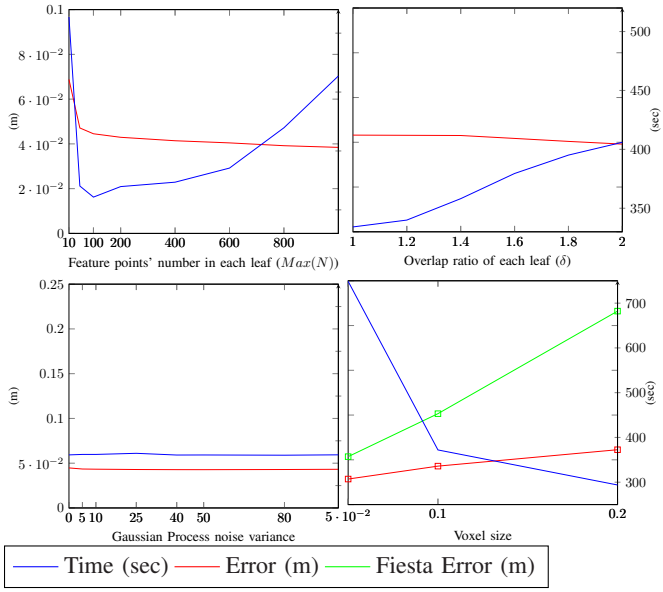

Fig. 7: Training is done with 829 depth images, and their synchronized pose out of Cow and Lady dataset. The left and right axises determine the SDF error (m), and time(seconds). The default parameters for our algorithm is $Max(N) = 200$, $\delta = 1.5$, Gaussian Process noise variance $= 25$, $voxel\ size = 0.1$, $frame\ size = 5$, and truncation value is $3 \times voxel\ size$. The error is verified with respect to ground truth point cloud that is provided by the dataset.

Gaussian Processes regression of semantic class signed distance functions. Our method offers a promising direction for semantic task specifications and uncertainty-aware task planning. Future work will focus on improving the inference speed and precision by considering alternative implicit functions and incorporating probabilistic representation of other environment characteristics such as texture and temperature.

Appendix A Proof of Prop. 1

Without loss of generality, assume $\mu_0(\mathbf{x}) = 0$. The general result can be concluded by change of variables $f(\mathbf{x}) - \mu_0(\mathbf{x})$. Let X_* be an arbitrary finite set of query points, and $\mathbf{f}_*$ be the vector of evaluations of f over X_*. To prove the posterior GPs are identical, regardless of which dataset is used for training, we will show that the Gaussian distribution of $\mathbf{f}_*$ conditioned on either dataset is the same. Let $\mathbf{f} := f(X)$, $\hat{\mathbf{f}} = [\mathbf{f}^\top, \mathbf{f}_*^\top]^\top$, $\boldsymbol{\zeta} = [y_{1,1}, ..., y_{n,m_n}]^\top$, and $D \in \mathbb{R}^{\sum_{j=1}^{n} m_j \times \sum_{j=1}^{n} m_j}$ be a diagonal matrix such that $D_{\sum_{k=0}^{i-1} m_k + j, \sum_{k=0}^{i-1} m_k + j} = \frac{1}{\sigma_i^2}$ for $j = 1, ..., m_i$. Our approach is to calculate $p(\mathbf{f}_*, \mathbf{f}|\boldsymbol{\zeta})$ from the joint distribution $p(\mathbf{f}_*, \mathbf{f}, \boldsymbol{\zeta})$ for the first dataset. Then, if we repeat the process for the second data set, calculating $p(\mathbf{f}_*, \mathbf{f}|\hat{\boldsymbol{\zeta}})$, we end up with the same normal distribution. If the marginal of $\mathbf{f}_*$ is obtained by integrating out $\mathbf{f}$, the distributions $p(\mathbf{f}_*|\boldsymbol{\zeta})$, and $p(\mathbf{f}_*|\hat{\boldsymbol{\zeta}})$ remain the same. From conditional probability and since $\boldsymbol{\zeta}$ depends only on $\mathbf{f}$, we have $p(\mathbf{f}_*, \mathbf{f}, \boldsymbol{\zeta}) = p(\boldsymbol{\zeta}|\mathbf{f})p(\mathbf{f}_*, \mathbf{f})$. Let $\mathbf{z} = [\boldsymbol{\zeta}^\top, \hat{\mathbf{f}}^\top]^\top, \hat{\mathbf{z}}^\top = [\hat{\boldsymbol{\zeta}}^\top, \hat{\mathbf{f}}^\top]^\top$. The log likelihood of $\mathbf{z}$ is:

$$\log p(\mathbf{z}) \propto \sum_{i=1}^{n} \sum_{j=1}^{m_i} -\frac{1}{2} \frac{(y_{i,j} - f(\mathbf{x}_i))^2}{\sigma_i^2} - \frac{1}{2} \hat{\mathbf{f}}^\top \boldsymbol{\Omega} \hat{\mathbf{f}}, \quad (13)$$

where $\begin{bmatrix} k_0(X,X) & k_0(X,X_*) \\ k_0(X_*,X) & k_0(X_*,X_*) \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} \\ \boldsymbol{\Omega}_{12}^\top & \boldsymbol{\Omega}_{22} \end{bmatrix} = \boldsymbol{\Omega}$. For the first dataset, let $\mathbf{H}$ be a matrix with elements $\mathbf{H}_{i,(\sum_{k=0}^{i-1} m_k + j)} = \frac{-1}{\sigma_i^2}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$ with zero at the rest of its entries. Similarly, for the second dataset, let $\hat{\mathbf{H}}$ to be a matrix with elements $\hat{\mathbf{H}}_{i,i} = \frac{-m_i}{\sigma_i^2}$ for $i = 1, ..., n$ and zero elsewhere. Rewrite (13):

$$\log p(\mathbf{z}) \propto -\frac{1}{2} \mathbf{z}^\top \boldsymbol{\Omega}_{jnt} \mathbf{z} \quad (14)$$

$$\boldsymbol{\Omega}_{jnt} := \begin{bmatrix} D^{-1} & \mathbf{H}^\top & \mathbf{0} \\ \mathbf{H} & \mathrm{diag}(\mathbf{H1}) + \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} \\ \mathbf{0} & \boldsymbol{\Omega}_{12}^\top & \boldsymbol{\Omega}_{22} \end{bmatrix} = \begin{bmatrix} D^{-1} & \mathbf{G}^\top \\ \mathbf{G} & \boldsymbol{\Omega}_{cnd} \end{bmatrix},$$

where $\mathbf{1}$ is a vector of ones. This means $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_{jnt}^{-1})$, and hence $\hat{\mathbf{f}}|\boldsymbol{\zeta} \sim \mathcal{N}(-\boldsymbol{\Omega}_{cnd}^{-1}\mathbf{G}\boldsymbol{\zeta}, \boldsymbol{\Omega}_{cnd}^{-1})$. Similarly for second dataset, define $\hat{\boldsymbol{\Omega}}_{jnt} := \begin{bmatrix} \hat{D}^{-1} & \hat{\mathbf{G}}^\top \\ \hat{\mathbf{G}} & \hat{\boldsymbol{\Omega}}_{cnd} \end{bmatrix}$, where $\hat{\mathbf{G}} := [\hat{\mathbf{H}}^\top, \mathbf{0}]^\top$ and $\hat{\boldsymbol{\Omega}}_{cnd}$ is defined by adding $\mathrm{diag}(\hat{\mathbf{H}}\mathbf{1})$ to the top left block of $\boldsymbol{\Omega}$. Again, $\hat{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Omega}}_{jnt}^{-1})$, so we can conclude $\hat{\mathbf{f}}|\hat{\boldsymbol{\zeta}} \sim \mathcal{N}(-\hat{\boldsymbol{\Omega}}_{cnd}^{-1}\hat{\mathbf{G}}\hat{\boldsymbol{\zeta}}, \hat{\boldsymbol{\Omega}}_{cnd}^{-1})$. The equivalence of the covariance matrices and means of these two normal distributions follows from $\hat{\mathbf{H}}\mathbf{1} = \mathbf{H}\mathbf{1}$ and $\hat{\mathbf{H}}\hat{\boldsymbol{\zeta}} = \mathbf{H}\boldsymbol{\zeta}$. □

Appendix B Proof of Prop. 2

Let $l_c(z) := \mathbb{P}\left(\arg\min_i |f_i(\mathbf{x})| = c \text{ and } \min_i |f_i(\mathbf{x})| \leq |z|\right)$. Since $\mathbb{P}(\min_i |f_i(\mathbf{x})| \leq |z|) = \sum_i l_i(z)$:

$$\mathbb{P}\left(\arg\min_i |f_i(\mathbf{x})| = c \,\middle|\, \min_i |f_i(\mathbf{x})| \leq |z|\right) = \frac{l_c(z)}{\sum_i l_i(z)}$$

The term we are interested in computing is $\lim_{z \to 0} \frac{l_c(z)}{\sum_i l_i(z)}$. Let $\mathbf{x}$ be an arbitrary (test) point, define $\mu_i := \mu_{t,i}(\mathbf{x})$ and $\sigma_i := \sigma_{t,i}(\mathbf{x})$ for $i = 1, \ldots, N$. The GP prior of f_i stipulates that its value at $\mathbf{x}$ has a density function $p(t) = \frac{1}{\sigma_i}\phi\left(\frac{t-\mu_i}{\sigma_i}\right)$. Hence, $\mathbb{P}(|f_i(\mathbf{x})| \geq t) = 1 - \Phi(\frac{|t|-\mu_i}{\sigma_i}) + \Phi(\frac{-|t|-\mu_i}{\sigma_i})$. Note that $l_c(z)$ corresponds to the probability that $|f_c(\mathbf{x})| \leq |f_i(\mathbf{x})|$ for all i, since all f_i are independent of each other:

$$l_c(z) = \int_{-z}^{z} \frac{\phi\left(\frac{t-\mu_c}{\sigma_c}\right)}{\sigma_c} \prod_{i \neq c} \left(1 - \Phi\left(\frac{|t| - \mu_i}{\sigma_i}\right) + \Phi\left(\frac{-|t| - \mu_i}{\sigma_i}\right)\right) dt$$

The claim is concluded by $\lim_{z \to 0} \frac{l_c(z)}{2z} = \frac{1}{\sigma_c}\phi\left(\frac{-\mu_c}{\sigma_c}\right)$. □

References

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[2] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.

[3] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *IEEE Int. Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[4] L. Teixeira and M. Chli, "Real-time mesh-based scene estimation for aerial inspection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4863–4869.

[5] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale three-dimensional mesh reconstruction," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1584–1591, 2018.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[7] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

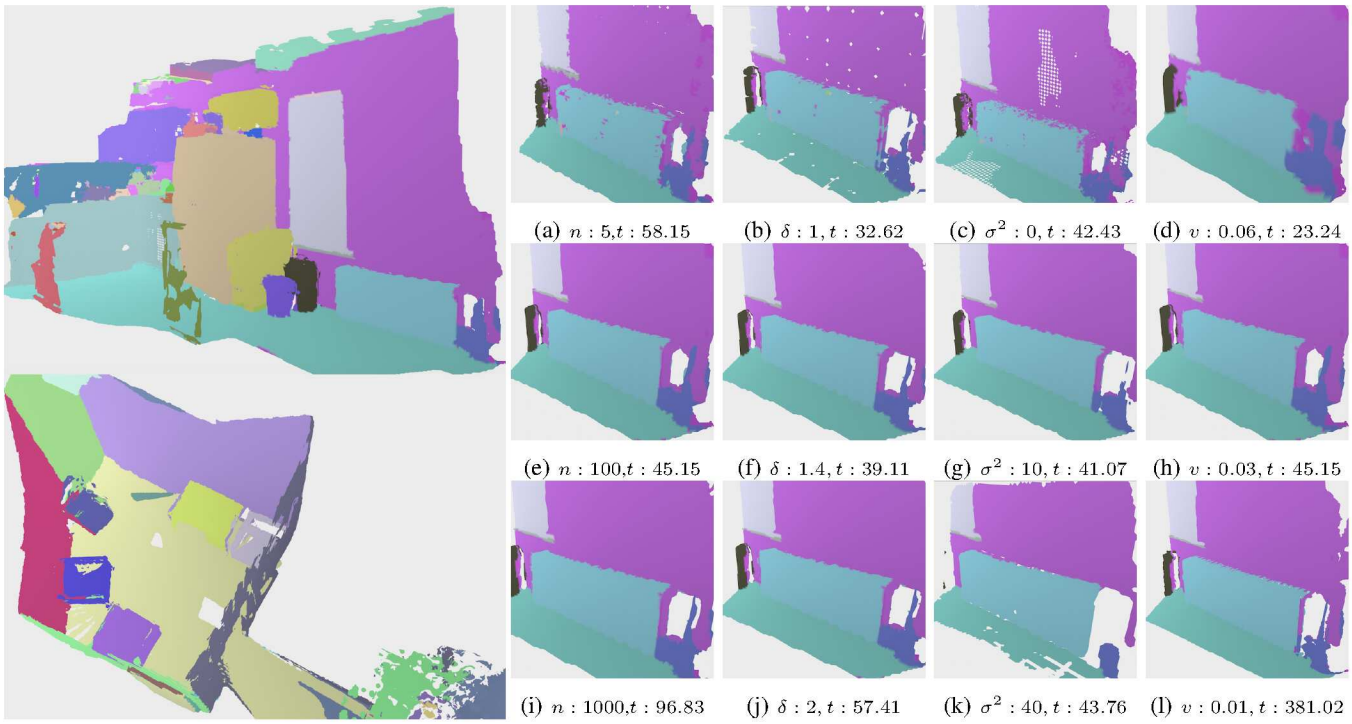

Fig. 8: The default parameters are $\delta = 1.5$, $Max(N)(n) = 100$, GP noise variance $(\sigma^2) = 3$, $voxel\ size(v) = 0.03$, $frame\ size = 1$. On the right we see the change of parameters over 140 RGB-D images (t is time in seconds). Left up is the final reconstruction on sequence 255, containing 2450 RGB-D images. The number of detected classes is 85, shown in random colors. This reconstruction took 1040.41 seconds. Left down is the final reconstruction on sequence 011, containing 3700 RGB-D images. The number of detected classes is 61. This reconstruction took 1885.72 seconds. On the right we see the change of parameters over 140 RGB-D images.

Fig. 9: Reconstruction of the Cow and Lady dataset. Red hues indicate lower TSDF uncertainty.

[8] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conf. on Computer Vision*, 2014.

[9] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Eurographics Symposium on Geometry Processing*, 2006.

[10] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," 1996.

[11] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017.

[12] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019.

[13] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

[14] A. Hermans, G. Floros, and B. Leibe, "Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2631–2638.

[15] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, "Joint semantic segmentation and 3d reconstruction from monocular video," in *European Conference on Computer Vision*. Springer, 2014, pp. 703–718.

[16] M. Jadidi, L. Gan, S. Parkison, J. Li, and R. Eustice, "Gaussian Processes Semantic Map Representation," *arXiv:1707.01532*, 2017.

[17] M. G. Jadidi, J. V. Miró, R. Valencia, and J. Andrade-Cetto, "Exploration on Continuous Gaussian Process Frontier Maps," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 6077–6082.

[18] S. Kim and J. Kim, "Occupancy mapping and surface reconstruction using local Gaussian processes with kinect sensors," *IEEE Trans. on Cybernetics*, vol. 43, no. 5, pp. 1335–1346, 2013.

[19] S. O'Callaghan and F. Ramos, "Gaussian process occupancy maps," *International Journal of Robotics Research*, vol. 31, no. 1, 2012.

[20] A. Koppel, "Consistent online Gaussian Process regression without the sample complexity bottleneck," in *American Control Conference (ACC)*, 2019, pp. 3512–3518.

[21] A. Koppel, A. S. Bedi, K. Rajawat, and B. M. Sadler, "Optimally compressed nonparametric online learning," *IEEE Signal Processing Magazine*, 2020.

[22] V. Tresp, "A bayesian committee machine," *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[23] S. Kim and J. Kim, "Recursive bayesian updates for occupancy mapping and surface reconstruction," in *Australasian Conference on Robotics and Automation*, 2014.

[24] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," in *Advances in neural information processing systems*, 2002, pp. 881–888.

[25] A. Milioto and C. Stachniss, "Bonnet: An Open-Source Training and Deployment Framework for Semantic Segmentation in Robotics using CNNs," in *IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[26] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "Scenenn: A scene meshes dataset with annotations," in *International Conference on 3D Vision (3DV)*, 2016.