**Robotics: Science and Systems 2023**
**Daegu, Republic of Korea, July 10-July 14, 2023**

# Autonomous Navigation, Mapping and Exploration with Gaussian Processes

Mahmoud Ali
Indiana University
alimaa@iu.edu

Hassan Jardali
Indiana University
hjardali@iu.edu

Nicholas Roy
MIT
nickroy@csail.mit.edu

Lantao Liu
Indiana University
lantao@iu.edu

***Abstract*—Navigating and exploring an unknown environment is a challenging task for autonomous robots, especially in complex and unstructured environments. We propose a new framework that can simultaneously accomplish multiple objectives that are essential to robot autonomy including identifying free space for navigation, building a metric-topological representation for mapping, and ensuring good spatial coverage for unknown space exploration. Different from existing work that model these critical objectives separately, we show that navigation, mapping, and exploration can be derived with the same foundation modeled with a sparse variant of a Gaussian process. Specifically, in our framework the robot navigates by following frontiers computed from a local Gaussian process perception model, and along the way builds a map in a metric-topological form where nodes are adaptively selected from important perception frontiers. The topology expands towards unexplored areas by assessing a low-cost global uncertainty map also computed from a sparse Gaussian process. Through evaluations in various cluttered and unstructured environments, we validate that the proposed framework can explore unknown environments faster and with a shorter distance travelled than the state-of-the-art frontier exploration approaches. Through field demonstration, we have begun to lay the groundwork for field robots to explore challenging environments such as forests that humans have yet to set foot in[1].**

## I. Introduction

3D LiDARs and other high-definition ranging sensors have long been used for autonomous systems to navigate and map unknown yet man-made environments such as indoor buildings and on freeways. LiDARs are used less often for robot navigation and exploration in highly unstructured environments where there are numerous indistinguishable small objects such as in crop fields and wild forests. Indeed, navigation in highly unstructured environment is challenging even for humans. Think of a scenario where humans need to cross a wild unknown forest with no road in it. We typically identify navigable spaces by identifying traversable passages across random vegetation in our field of view. In this case, obstacles such as clusters of shrubs are inherently grouped and *abstracted*, and a navigation decision is made by moving into the identified open space.

In previous studies, solving navigation, exploration, and mapping simultaneously is mostly structured on the paradigm of occupancy mapping and frontier based exploration [36, 7]. However, this framework has limitations when deployed in highly unstructured environments. The first issue is that the global occupancy map relies on an accurate representation of the environmental geometry and thus the number of map cells grows quadratically with the length-scale of the environment, incurring a prohibitive computational cost for a large environment. The second issue is the inappropriate assumption that the discretized occupancy/map cells are independent of each other. In the real world, space has continuity and features are correlated — these useful properties have not been sufficiently leveraged. The third issue is that a global occupancy map that attempts to memorize details of the whole environment typically assumes the environment is static and therefore becomes sensitive to any small changes. In unstructured environments this assumption can be easily violated because natural environments are constantly changing, for example foliage movements due to winds or raindrops, walking animals, flying birds/insects, etc.

In this work, we propose a principled framework that leverages the *abstracted* representation of multiple key autonomy components including perception, navigation and mapping, which is particularly useful for complex environments such as those off-road scenarios (e.g., forest trails). Our framework is based on a Gaussian process, which we use to model local perception, assess global exploration quality, and guide to map the navigable spaces. The proposed representation reduces the sensitivity to small objects and enables the robot to identify navigable passages in a noisy environment and adaptively construct a light-weight map that best characterizes the environmental navigability while ensuring full spatial coverage.

This paper includes the following contributions:

- We use a sparse Gaussian process (SGP) to model the local perception of 3D occupied points. The SGP correlates points of a neighborhood, eliminating the independence assumption of the prevalent occupancy representation framework. More importantly, the SGP also provides uncertainty assessment of the perception modeling. We show that the uncertainty can be used to uniquely *sketch* spatial navigability so that we can identify navigation guidance *frontiers* in an extremely convenient way. This model also reveals that local perception can naturally lead to local navigation.
- We demonstrate the SGP can provide guidance for constructing a (metric) topological map incrementally which

[1]Video material: https://youtu.be/WcbngSewXBw

can be viewed as a high-level characterization of the navigable spaces of the environment. The topology is very lightweight because its nodes are adaptively added only when the robot enters a new location that has a high exploration uncertainty (in contrast to pre-defined equi-distance node addition process). One might view the topological nodes as *condensed* and *salient* local perceptions filtered by the SGP.
- We show that by using the SGP, exploring very large environments becomes possible. The nodes of the topology map are passed to another separate layer of a *global* SGP whose uncertainty quantification represents the global exploration status in a computationally efficient way (computation bounded through SGP). By integrating the topology map with the global SGP model, we can define and identify topology frontiers that guide future exploration, which is different from existing exploration frontier mechanisms.
- We provide evaluation in multiple domains including a variety of simulated environments, structured indoor corridors and a challenging outdoor forest. The results demonstrate that our framework is able to solve simultaneously the navigation, exploration, and mapping tasks. Our algorithm explored the simulated environment faster, with a traveled distance less than the start-of-art frontier exploration approaches.

## II. Related Work

Most autonomous navigation, mapping and exploration approaches have been developed on top of occupancy grid maps where each cell is represented as free, occupied, or unknown [36, 7, 13, 10]. Occupancy grid maps ignore the structural dependency between cells and has a fixed resolution. To mitigate the issues brought by the discrete cells, kernel-based models such as Gaussian pocess have been proven robust in representing the dependencies of spatially correlated data [26, 38] overcoming the cell independence assumption. Continuous-map-based exploration approaches exploit the information-theoretic gain over a probabilistic map [6]. For example the continuous frontier map [15] represents the occupancy probability distribution as a continuous map. In general, the continuous approaches demonstrate efficient map building by reducing the entropy of a probabilistic map representation [6, 5], however, they depend on explicit path and trajectory planning which makes them scale poorly with environment size and require accurate metric-map estimates.

Another form of frontiers is proposed by [30] where frontiers are represented as particles placed in the free space, bounded by the occupied space, and growing into the unknown space. There also exist works that formulate the exploration problem as a next-best-view problem [12, 3] where a sequence of depth scans are optimized to reconstruct 3D objects [21]. Thus the exploration problem is also treated as an open space attraction and contours detection problem [16, 19, 20].

Another line of related exploration approaches represent the environment as a topological graph [9]. For example, a basic Voronoi topological map can be built on an existing occupancy map of the environment by identifying the corner points, where the Voronoi diagram vertices are used as the topological map nodes [32]. However, this representation can easily lead to cluttered graphs. Similar approaches have also been exploited to leverage the generalized Voronoi diagram [29], Voronoi random fields [11], and Voronoi decomposition and distance maps [37]. The graph-based exploration has been proven practical and efficient in tunnels [33, 8] and maze-like indoor environments [24]. Graphs are extremely efficient for planning in contrast to accurate trajectory computation over a metric map [4]. However, most existing graph-based explorations explicitly or implicitly rely on occupancy maps to establish the map's topology.

Different from existing works, our proposed framework combines the advantages of the continuous global uncertainty model and the graph-based approaches. First, our method includes a kernel-based global SGP uncertainty model that overcomes the discrete models' independence assumption of [36, 14] and reduces the computation cost of existing continuous models [26, 38, 15]. This is achieved by representing only the global exploration uncertainty information instead of the occupancy information. Second, our method consists of a local perception model which efficiently identifies local frontiers and exploits their abstract representation of the local observation to incrementally build a metric-topological map. The metric-topological map is built on top of a local occupancy model rather than an occupancy map [32, 33, 8]. Additionally, frontiers obtained from our local perception model serve as local navigation subgoals. The proposed local perception model is more robust and less susceptible to noise than traditional frontier extraction approaches.

In our framework, the most computationally costly components are the models represented with Gaussian processes. To overcome the computation complexity limitations of Gaussian processes [28], we select an efficient variational variant of SGPs [34] which jointly estimates the kernel hyperparameters and the inducing points. The variational approximation distinguishes between the inducing points (as a variational parameter) and the kernel hyperparameters, resulting in a flexible training process with high computational efficiency.

## III. Preliminaries on the Sparse Gaussian Process

The standard Gaussian process (GP) is defined as a set of random variables which is described by a mean function $m(\mathbf{x})$, and a co-variance function (kernel) $k(\mathbf{x},\mathbf{x}')$, where $\mathbf{x}$ is the GP input [27]:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x},\mathbf{x}'\right)\right). \tag{1}$$

Consider a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with $N$ training inputs $\mathbf{x}$ and their corresponding scalar outputs (observations) $y$. In GP regression, we assume that $y_i = f(\mathbf{x}_i) + \varepsilon_i$ where $f(\mathbf{x}_i)$ is the unknown underlying function and $\varepsilon_i$ is a Gaussian noise $\mathcal{N}\left(0, \sigma^2\right)$. The GP posterior can then be defined by a mean $m_{\mathbf{y}}(\mathbf{x})$ and a posterior co-variance function $k_{\mathbf{y}}(\mathbf{x},\mathbf{x}')$ [34],

$$
\begin{aligned}
m_{\mathbf{y}}(\mathbf{x}) &= K_{\mathbf{x}n}\left(\sigma^2 I + K_{nn}\right)^{-1}\mathbf{y}, \\
k_{\mathbf{y}}\left(\mathbf{x},\mathbf{x}'\right) &= k\left(\mathbf{x},\mathbf{x}'\right) - K_{\mathbf{x}n}\left(\sigma^2 I + K_{nn}\right)^{-1} K_{n\mathbf{x}'},
\end{aligned} \tag{2}
$$

where $K_{nn}$ is the $n \times n$ co-variance matrix of the training inputs, $K_{xn}$ is n-dimensional row vector of kernel function values between $\mathbf{x}$ and the training inputs; $K_{nx} = K_{xn}^T$. After training the GP, the prediction $y_*$ for any new input $\mathbf{x}_*$ is estimated using the GP prediction equation:

$$
p(y_*|\mathbf{y}) = N(y_*|m_{\mathbf{y}}(\mathbf{x}_*), k_{\mathbf{y}}(\mathbf{x}_*,\mathbf{x}_*) + \sigma^2). \tag{3}
$$

GP prediction depends on the kernel parameters $\Theta$ and the noise variance $\sigma^2$. The hyperparameters $(\Theta, \sigma^2)$ are estimated by maximizing the log marginal likelihood:

$$
\log p(\mathbf{y}) = \log\left[N\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + K_{nn}\right)\right]. \tag{4}
$$

A standard GP has a computation complexity of $\mathcal{O}(N^3)$ where $N$ is the number of the training samples. Many approximation approaches, known collectively as the *Sparse Gaussian Process (SGP)* [18, 31, 34], propose replacing the entire data set with only $M$ samples to represent the entire training data to overcome the computation cost of GP. These $M$ samples are called the *inducing points* $X_m$ and their corresponding values of the underlying function $f(\mathbf{x})$ are called the *inducing variables* $f_m$. We opt to jointly estimate the kernel hyperparameters $(\Theta, \sigma)$ and the inducing points $X_m$ by approximating the true exact posterior of a GP $p(f|y,\Theta)$ through a variational posterior distribution [34],

$$
q(f, f_m) = p(f|f_m)\phi(f_m), \tag{5}
$$

where $\phi(f_m)$ is the free variational Gaussian distribution. This variational approximation uses the Kullback-Leibler ($\mathbb{KL}$) divergence to describe the discrepancy between the approximated and the true posteriors, $\mathbb{KL}[q(f)||p(f|y,\Theta)]$. Minimizing the $\mathbb{KL}[q(f)||p(f|y,\Theta)]$ is identical to maximizing the variational lower bound of the true log marginal likelihood

$$
\begin{aligned}
F_V\left(X_m\right) &= \log\left[N\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + Q_{nn}\right)\right] - \frac{1}{2\sigma^2}\operatorname{Tr}(\widetilde{K}), \\
Q_{nn} &= K_{nm}K_{mm}^{-1}K_{mn}, \\
\widetilde{K} &= \operatorname{Cov}\left(\mathbf{f} \mid \mathbf{f}_m\right) = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn},
\end{aligned} \tag{6}
$$

where $F_V\left(X_m, \phi\right)$ is the variational objective function, $\operatorname{Tr}(\widetilde{K})$ is a regularization trace term, $K_{mm}$ is $m \times m$ co-variance matrix on the inducing inputs, $K_{nm}$ is $n \times m$ cross-covariance matrix between training and inducing points, and $K_{nm} = K_{mn}^T$.

## IV. Methodology

We propose a new framework that can simultaneously accomplish multiple autonomy objectives including identifying free space for navigation, building a metric-topological representation for mapping, and ensuring good spatial coverage for unknown space exploration. A systematic overview of the proposed framework is shown in Fig. 1. Briefly, the point cloud in Fig. 1(b) measured by the robot is converted and represented as a SGP occupancy model in Fig. 1(e) with its associated SGP variance surface in Fig. 1(f). The variance surface allows us to identify local navigable spaces (in white) and define local navigation frontiers (circles of color-scale representing different levels of navigability). A global uncertainty map in Fig. 1(h) is used to assess the exploration state of the whole environment. A metric-topological map in Fig. 1(i–m) is incrementally and adaptively built along the exploration path, where red vertices with low exploration uncertainty denote explored free spaces and green vertices with large exploration uncertainty are those to be explored.

Different from existing work that model these critical objectives separately, we show that navigation, mapping, and exploration can be derived with the same foundation modeled with a sparse variant of Gaussian processes. We present all modules in the following subsections.

### A. From Pointcloud to SGP Occupancy Model

Each local observation (pointcloud) is transformed into an occupancy surface by projecting the observed points onto a 2D circular surface with a predefined radius $r_{oc}$, see Fig. 1(c) and (d). Specifically, the sensor observation is converted to spherical coordinates, where each sensing point is described by a tuple $(\theta_i, \alpha_i, r_i)$ which represents the azimuth, elevation, and radius values, respectively. Each point defined in Cartesian coordinates $(x_i, y_i, z_i)$ can be transformed into spherical coordinates $(\theta_i, \alpha_i, r_i)$ using the following equations:

$$
r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}, \quad \theta_i = \tan^{-1}(y_i/x_i), \quad \alpha_i = \cos^{-1}(z_i/r_i). \tag{7}
$$

All observed points on or outside the circular occupancy surface (with a radius $r_i \geq r_{oc}$) are discarded and considered as the free space. The rest of the points that are inside the circular surface (with a radius $r_i < r_{oc}$) are projected on the occupancy surface and called the *occupied points*. Each occupied point $\mathbf{x}_i$ on the surface is defined by two attributes: the azimuth and elevation angles $\mathbf{x}_i = (\theta_i, \alpha_i)$, and assigned an *occupancy value* $f(\mathbf{x}_i)$. The probability of occupancy $f(\mathbf{x}_i)$ at each point on the occupancy surface is modeled by an SGP:

$$
f(\mathbf{x}) \sim \mathcal{SGP}\left(m(\mathbf{x}), k\left(\mathbf{x},\mathbf{x}'\right)\right). \tag{8}
$$

Given that the sensor measurements will have noise, we add white noise $\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ to the occupancy function $f(\mathbf{x})$, so the observed occupancy at any point $\mathbf{x}_i$ on the occupancy surface is described as $y_i = f(\mathbf{x}_i) + \varepsilon$.

The occupancy of a point $f(\mathbf{x}_i)$ is related to its radius $r_i$ by the following equation $f(\mathbf{x}_i) = r_{oc} - r_i$. This mapping is encoded in our SGP model as a zero-mean function $m(\mathbf{x}) = 0$ which sets the occupancy value of the non-occupied points to zero. This mapping process mimics the mechanism of the sensor itself. The final model of the occupancy surface is a 2D SGP, see Fig. 1(e), where the input is the azimuth and elevation angles, $\mathbf{x} \in \{(\theta, \alpha)\}_{i=1}^n$, and the corresponding output $y_i$ is the expected occupancy.

The rational quadratic (RQ) covariance function is selected as the SGP kernel, $k_{\mathrm{RQ}}\left(\mathbf{x},\mathbf{x}'\right)$, because a GP prior with an RQ kernel is expected to have functions that vary across different

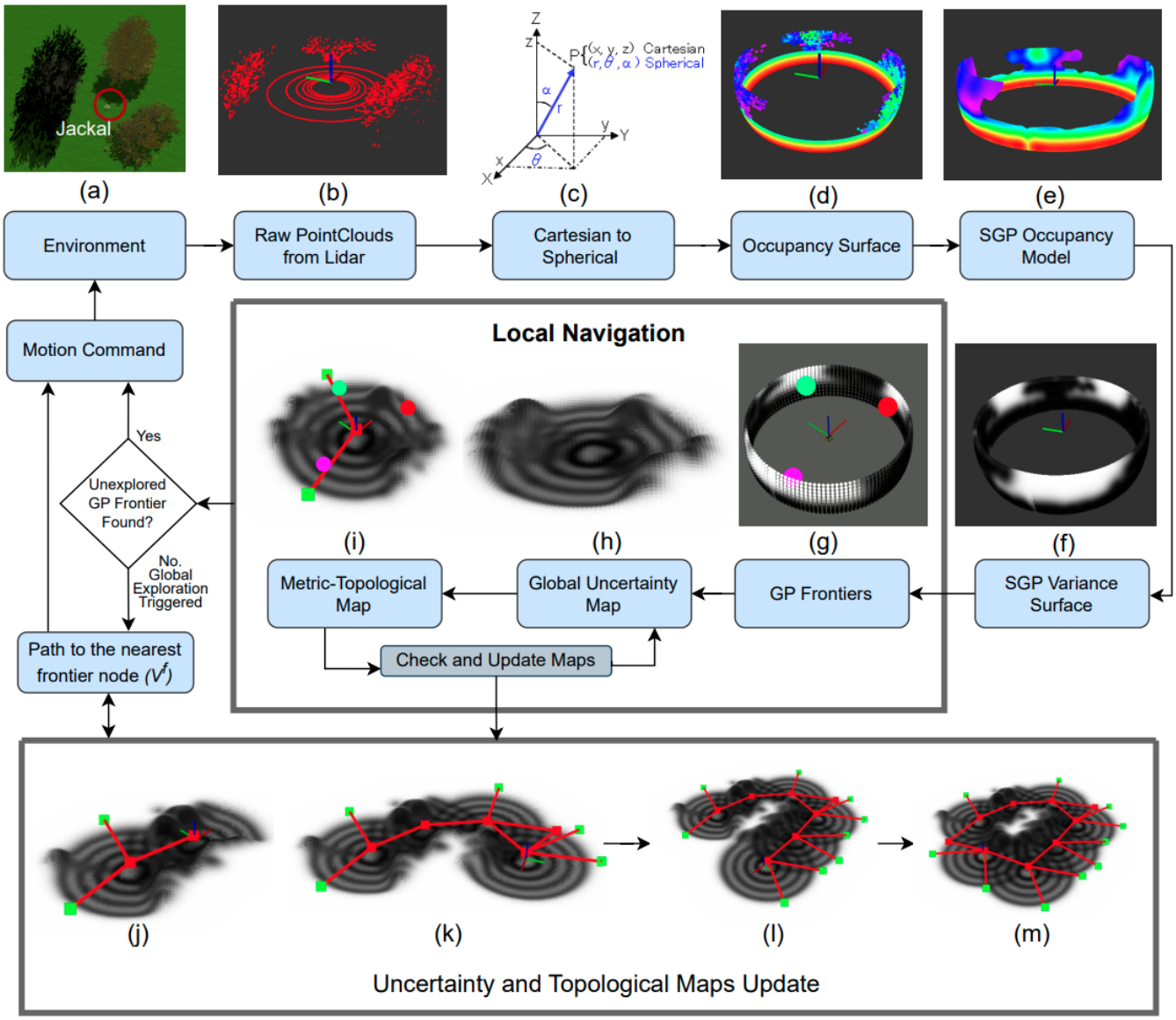



Fig. 1: Block flow diagram of our system modules.

length-scales [27]. Formally,

$$k_{\mathrm{RQ}}\left(\mathbf{x},\mathbf{x}'\right)=\sigma_k^2\left(1+\frac{(\mathbf{x}-\mathbf{x}')^2}{2\alpha_k\ell^2}\right)^{-\alpha_k}, \tag{9}$$

where $\sigma_k^2$ is the signal variance, $\ell$ is the length-scale, and $\alpha_k$ sets the relative weighting of large vs. small scale variations. Given the RQ kernel, the kernel parameters $\Theta$ are defined as $\sigma_k^2$, $\ell$ and $\alpha_k$. We believe that the RQ kernel is more expressive for modeling the occupancy surface than the commonly used Squared Exponential (SE) kernel because the RQ is equivalent to a scaled mixture of SE kernels with mixed characteristic length-scales [27]. We also exploit our knowledge about the resolution of the LiDAR to initiate different length-scales along the azimuth and the elevation axes. (The capability of the SGP occupancy model to re-construct the original pointcloud has been investigated in [1], where an SGP occupancy model with only 400 inducing points has an average error of around 12 cm compared with the original ground-truth pointcloud.)

One advantage of the GP and its variants over other modeling approaches is that GPs provide a variance value that assesses the uncertainty level for each prediction. Therefore, the construction of the SGP occupancy surface is accompanied by a corresponding SGP variance (uncertainty) surface, as shown in Fig. 1(f). Intuitively, while the occupancy surface can be considered as a *local* 3D map projected on a circular surface of the robot locality, the variance surface associated with the reconstructed occupancy surface can be viewed as a local circular uncertainty-map around the robot, see Fig. 2. It is also worth mentioning that, since the GP utilizes a kernel function that correlates neighboring data points, spatial continuity is thus leveraged. This also allows the SGP occupancy model

to possess a *smoothing* property where cluttered objects (e.g., clusters of shrubs) are grouped as one entity and small, noisy features such as individual grass blades or leaves can be ignored.

### *B. Gaussian Process Frontier for Navigation*

It is interesting to observe that the SGP variance surface can model local free space effectively. The high-variance values of the SGP variance surface correspond to either large free space area (i.e., no obstacles nearby, and thus no data points) or gaps between obstacles (with discontinuities on occupancy values), see bright regions on the variance surface in Fig. 1(f). A Gaussian Process frontier (GP frontier) is defined as the point with the greatest variance in a local high variance region, as illustrated in Fig. 1(g). Since high variance regions indicate local free space, GP frontiers thus act as local navigation points directing towards such open/free spaces, as illustrated in Fig. 3. (The terms GP frontiers and local navigation points are used interchangably throughout the remainder of the paper.) Note that the GP frontier is different from the well-known frontier concept introduced in [35] and the C-frontier [15], as both of these techniques rely on a global occupancy map. In contrast, our proposed GP frontier considers only the current observation and is defined by variance (uncertainty) quantification.

Formally, a GP frontier $f_i$ is defined by its centroid point on the variance surface $(\theta_{fi}, \alpha_{fi})$, and the radius (distance) $r_{fi}$ between the frontier centroid and the occupancy surface origin (sensor origin). This radius is estimated using the SGP occupancy model, where the occupancy at the GP frontier is $oc_{fi} = SGP((\theta_{fi}, \alpha_{fi}))$ and $r_{fi} = r_{oc} - oc_{fi}$. As a consequence, when the GP frontier lies in the free space, the frontier radius is equal to the occupancy radius ($oc_{fi} = 0$, and $r_{fi} = r_{oc}$), while when the GP frontier lies in the gaps between obstacles, the frontier radius is less than the occupancy radius ($r_{fi} < r_{oc}$, and $oc_{fi} > 0$). When there are two or more GP frontiers, we can choose the one that is closest to the robot's current heading, enabling continuous and smooth motion at high speeds. When the task is exploring an unknown environment, the selection among multiple GP frontiers will also take into account exploration status, which is discussed in later subsections.

### *C. Exploration with Global Uncertainty Map*

We model the global exploration state for the whole environment with a separate SGP. We again leverage the uncertainty quantification of this *global SGP* to guide exploration towards unexplored regions with no observations. Therefore, we refer to this SGP model as the *global uncertainty map*. Formally, the global uncertainty map is a SGP whose training inputs $\mathbf{Z}$ are the $(x,y)$ coordinates of the robot poses where different observations were acquired, $\mathbf{Z} = \{(x_i, y_i)\}_{i=1}^{K}$, where $K$ is the number of observations. In our framework, observations are acquired *if and only if* the robot enters an uncertain (unexplored) area. The global uncertainty map predicts the exploration uncertainty $g(\mathbf{z})$ of any point $\mathbf{z} = (x,y)$ in the continuous domain, where regions with high certainty mean explored and otherwise unexplored. Specifically,

$$g(\mathbf{z}) \sim \mathcal{SGP}\left(m(\mathbf{z}), k\left(\mathbf{z}, \mathbf{z}'\right)\right), \tag{10}$$

where $m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}')$ are the mean and co-variance functions, respectively.

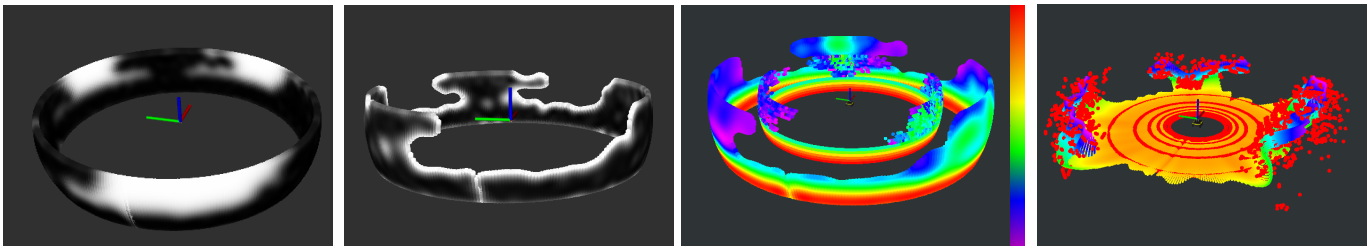
An exploration utility function $U_{exp}$ is used to select the GP frontier $f_i$ with the maximum exploration gain as the next navigation subgoal. The exploration utility function $U_{exp}$ maximizes the exploration gain based on three attributes of the GP frontier: i) Exploration uncertainty $u_{fi}$ of the GP frontier which is calculated using the global uncertainty map. This attribute is prioritized as it differentiates between explored vs. unexplored GP frontier. ii) The size (area) $a_{fi}$ of the high-variance region, in the SGP variance surface where the GP frontier lies. Larger gaps indicate higher exploration gain. iii) GP frontier direction $\theta_{fi}$ with respect to the robot heading, where the GP frontier that is more aligned with the robot's heading (smaller azimuth angle) leads to lower total energy cost to be reached. Formally,

$$\begin{aligned} U_{exp}(f_i) &= k_u u_{fi} + k_a a_{fi} - k_d \theta_{fi}^2, \\ f^* &= \arg\max_{f_i \in \mathcal{F}} \left(U_{exp}(f_i)\right), \end{aligned} \tag{11}$$

where $k_u$, $k_a$, and $k_d$ are weighting factors associated with the exploration uncertainty, area, and direction of the GP frontier, respectively. The azimuth angle $\theta_{fi}$ is squared to indicate the absolute direction and to imply a low penalty for small directions ($\leq$ 1 radian).

Our SGP uncertainty map offers significant computational efficiency, with an update time significantly faster than other mapping techniques (e.g., an octomap [14], GPOM [26], and fGPOM [38]). The uncertainty map is essentially a sparse GP, allowing us to predict the exploration state (explored vs. unexplored) with *bounded computation* (as explained in Section III), while other mapping techniques encode unexplored, occupied, and free information. Also, our global uncertainty map is updated *sparsely*, meaning that the update is triggered only when the robot enters an area that is sufficiently uncertain, while other techniques continuously update the map with new sensor observations. Finally, a complete LiDAR observation is represented in our uncertainty map as one "condensed" point, regardless of its occupancy information (free/occupied), to indicate that the location where the observation was acquired has been explored. While other occupancy mapping techniques necessitate separating occupancy information into occupied and free points and subsequently updating cells/voxels occupancy (in the case of a discrete map e.g. OGM, octomap) or the free and occupied training points (in the case of a continuous map e.g. GPOM, fGPOM). Therefore, a point used to update our global uncertainty map is equivalent to the combined set of the occupied $N_{oc}$ and the free points $N_{fr}$ needed to update the occupancy map,

$$\eta_\theta . \eta_\alpha \leq N_{oc} + N_{fr} \leq (1 + (\frac{L}{\delta - 1})) \eta_\theta . \eta_\alpha, \tag{12}$$



(a) (b) (c) (d)

Fig. 2: The SGP occupancy model as a local 3D map. (a) Variance (uncertainty) surface associated with the SGP occupancy model (same as Fig. 1(f)); (b) Variance surface after removing high variance regions; (c) SGP occupancy model (outer surface) compared with the original occupancy (inner surface). Warmer colors indicate higher occupancy values; (d) Reconstructed pointcloud from the SGP occupancy model (rainbow-colored, where color represents $z$-axis) versus the raw pointcloud of the scene shown in Fig. 1(a) (in red).

where $\eta_\theta$ and $\eta_\alpha$ represent the number of the horizontal and vertical beams of LiDAR, respectively. $L$ is LiDAR's maximum range, and $\delta$ represents the distance between free points sampled along LiDAR's beam (in our case $\eta_\theta.\eta_\alpha$=16000).

In addition, in the context of an expansive/large environment where a high volume of observations are received, the accuracy of the global uncertainty map tends to remain relatively stable, even though we bound the number of the inducing points. This is explained as the inducing points are used to characterize mainly the "borderline" between explored and unexplored spaces, which necessitates fewer points than the scenario for characterizing explored "areas". Consequently, even the exploration space enlarges, the impact on the performance of the global uncertainty map does not exhibit severe deterioration.

### D. Uncertainty-Driven Metric-Topological Mapping

To represent the explored spaces, we also propose a mapping framework in the paradigm of metric-topological mapping [25, 23], that is, a graph representation of the free space with metric position information attached to the nodes of the graph. In contrast to dense representations of environment geometry, a metric-topological map allows us to represent both the topological connectivity of the free space and the metric paths that move through the free space. One may view the metric-topological map as a high-level characterization of the navigable spaces of the environment.

Formally, a metric-topological map is an undirected graph $\mathcal{M} = (V, E)$, where $V$ denotes the vertices (nodes), and $E$ represents the edges between nodes. Each node $v \in V$ encodes the $(x, y)$ location of the node, and each edge $e \in E$ encodes the connectivity between two connected nodes (and is also labeled with the Euclidean distance between the two nodes). The vertex set $V$ includes two types of nodes, *explored nodes* set $V^e \in V$ (red squares in Fig. 1(i-m)), and *map frontier nodes* set $V^f \in V$ (green squares in Fig. 1(i-m)). The map frontier nodes keep track of the unexplored spaces.

While a metric-topological map is computationally efficient, we are faced with the challenge of building it with a reasonable number of nodes that can sufficiently represent the environment. To incrementally build $\mathcal{M}$, instead of adding equi-distance nodes, we adaptively add a new node only when the robot enters a new location that has a high exploration uncertainty (greater than a predefined threshold $U_{th}$). The acquired observation at that high uncertain area is used to update the global uncertainty map and subsequently reduces the total uncertainty of the global uncertainty map and to expand the topological map based on the following rules:

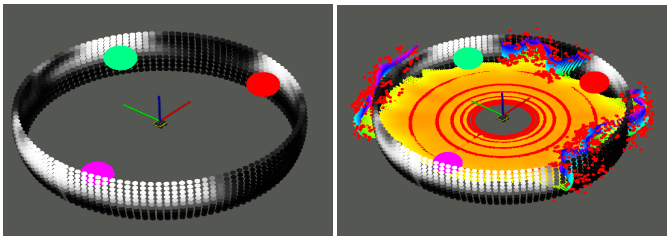
- A node $v_t$ added at time $t$ will be connected to any node $v_i$ that lies inside the current field of view (FoV) if the length of the edge $e_{t,i}$, that connects the two nodes, is less than the radius predicted by the SGP occupancy model along the edge direction, which means that there is no obstacle between the two nodes.
- When two or more GP frontiers have high exploration uncertainty values (unexplored), the GP frontier that has the highest exploration gain $U_{exp}$, is selected as the next navigation subgoal. Other high uncertain GP frontiers are added to $\mathcal{M}$ as map frontier nodes $v^f$ to be revisited later.
- At any time, if a map frontier node $v^f$ lies inside the robot's current FoV, this map frontier node is converted to an explored node $v^e$.

An illustration of the update of the global uncertainty map and the metric-topological map is shown in Fig. 1(i–m). For the uncertainty map, darker regions mean certain (explored) areas and brighter regions mean uncertain (unexplored) areas. The number of the inducing points of the global uncertainty map is bounded to a maximum number $J_{max}$ for real-time computation. However, the size of the topological map (number of nodes) is not restricted and it depends on the area and the structure of the navigable space.

If all GP frontiers are located in explored (certain) locations, the nearest map frontier node from $\mathcal{M}$ is selected to be explored next. The nearest map frontier node is selected based on the distance information encoded in $\mathcal{M}$. The shortest path to the nearest map frontier is calculated by running the A* algorithm on the previously-generated nodes of the metric-topological map. The A* path consists of explored nodes that lead to the nearest map frontier node, and those explored nodes act as intermediate navigation waypoints (no new nodes need to be created during the navigation to the nearest map frontier).

### E. Analysis

Our algorithm consists of two main computational steps: computing the local SGP perception model and updating the global uncertainty map.



(a) (b)

Fig. 3: (a) Local navigation points (colored spheres) on a variance surface represent the GP frontiers; (b) The GP frontiers are assigned in the middle of the free space which maximizes the distance to surrounding obstacles when the robot moves towards the free space.

*1) Computation complexity of local SGP perception model:* The local SGP perception model is computed for each observation independently, and it is trained using only the occupied points $N_{oc}$. $N_{oc}$ is bounded to $0 \leq N_{oc} \leq \eta_\theta . \eta_\alpha$, where $N_{oc\_max} = \eta_\theta . \eta_\alpha$ means all beams return hit points. If $M$ is the number of the inducing points for the local SGP model, then the local SGP computation complexity is $\mathcal{O}(N_{oc}M^2)$.

*2) Computation complexity of global uncertainty map:* Our global uncertainty map uses only one point to represent one observation. Let $K$ be the number of observations and $J$ be the number of the inducing points for the global map, the computation complexity of the uncertainty map is $\mathcal{O}(KJ^2)$.

Combining the computational complexities of these two main steps, we have

***Remark** 4.1:* The proposed navigation, mapping and exploration algorithm has an overall computational complexity of $O(N_{oc}M^2 + KJ^2)$.

We are also interested in understanding the exploration performance:

***Remark** 4.2:* The proposed SGP exploration algorithm with the metric-topological map $\mathcal{M} = (V, E)$ achieves an approximation of guarantee of $(1 - \frac{1}{e})OPT$ where $OPT$ is the optimality of exploration gain of all possible configurations/solutions of the $|V|$ map nodes.

*Proof:* Our SGP exploration problem can be reduced to the sensor placement and coverage problem [2, 17] with a sufficient number of sensors. For the proposed metric-topological map $\mathcal{M} = (V, E)$, one might view each node as a sensor that observes surrounding space defined by the proposed local SGP perception model. The observation radius $r$ of the sensor is determined by the global SGP's length-scale. In this way, we can transform each node $v \in V$ into a "disk" model with its sensing area $A_v$. For all nodes $V$, the covered area $C_A(V) = \bigcup_{v \in V} A_v$. Recall that any function $f$ from sets to real values is called *submodular* if and only if it satisfies the "diminishing returns" property, i.e., $f(S \bigcup \{v\}) - f(S) \geq f(T \bigcup \{v\}) - f(T)$ whenever $S \subset T$. We can see that $C_A(V)$ well satisfies this property and is thus submodular. This means that, optimizing placements (measured by information gain) for the disk model is a submodular maximization problem with a well-known approximation guarantee of $\left(1 - (\frac{|V|-1}{|V|})^{|V|}\right) OPT \geq (1 - \frac{1}{e})OPT$ [22], where $e$ is the base of the natural logarithm and $OPT$ is the optimality of all possible configurations of the $|V|$ nodes. ■

## V. Results

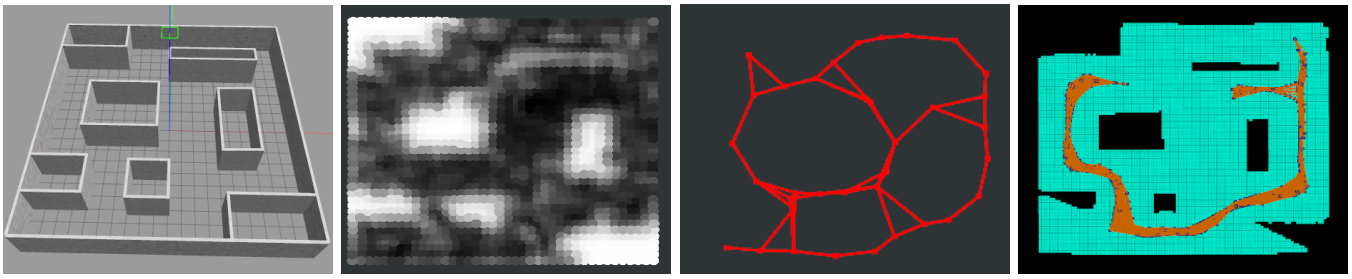
We perform both simulation and real-world experiments to demonstrate the performance of our framework in multiple environments with different configurations. To validate our framework, we compare our method with two baseline approaches including the standard and widely used frontier exploration method [36] as well as the recent state-of-the-art exploration algorithm GBPlanner [8]. The GBPlanner is a graph-based exploration method that, similar to our method, keeps a graph representation of the environment during the exploration. We consider the GBPlanner as our baseline because of its proven reliable performance in exploring unknown cave environments in DARPA Subterranean Challenge. We also demonstrate the performance of our method in a real forest environment. The values of critical parameters and the method for tuning them are outlined in the Appendix.

### A. Simulation

We first evaluate our framework in two simulated environments: a cluttered indoor environment and a forest trail environment. For GBPlanner, we use the ETHZ SuperMegaRobot (SMB) robot with the default configuration [8] which requires two 3D LiDARs. Each simulated experiment includes 15 trials for an average performance. The maximum linear and angular velocities are set to 1.0 meters per second and 1.0 radians per second, respectively. These maximum velocities are applied for both our approach and the comparative baselines.

*1) Simulated cluttered indoor environment:* We test the framework in an indoor space with a few obstacles distributed, see Fig. 4a. The simulated indoor environment is used for comparing our approach with the GBPlanner and the frontier exploration (two baselines). For our global uncertainty map, we calculate the explored (certain) area as the area that has uncertainty value under the predefined threshold $U_{th}$. For the 3D voxblox map generated by GBPlanner, instead of calculating the volume of the explored space in 3D, we are only interested in the explored area and compare that between our approach and the other baseline. The GBPlanner explored area is calculated as the explored area in XY plane (z=0) of the 3D voxblox map, see Fig. 4d. Two metrics are used to evaluate the three approaches: the explored area and the traveled distance. We also compare our metric-topological map with the global graph generated by the GBPlanner.

Fig. 5 shows that our algorithm outperforms the baselines in terms of exploration rate and total traveled distance. Specifically, our algorithm explores the whole environment (around 300 $m^2$) in an average time of 93 seconds with an average traveling distance of 74 meters, while the GBPlanner needs an average time of 169 seconds with an average traveling distance of 91 meters. The standard frontier exploration algorithm explores the environment in 257 seconds with an average traveling distance of 103 meters.



(a) (b) (c) (d)

Fig. 4: (a) Simulated cluttered indoor environment; (b) Global uncertainty map (darker colors indicate more certain regions due to past exploration); (c) Metric-topological map; (d) Global graph generated by the GBPlanner (vertices in blue and edges in light-brown ), and the background (cyan color) represents the explored area (the XY plane of the 3D voxblox map).

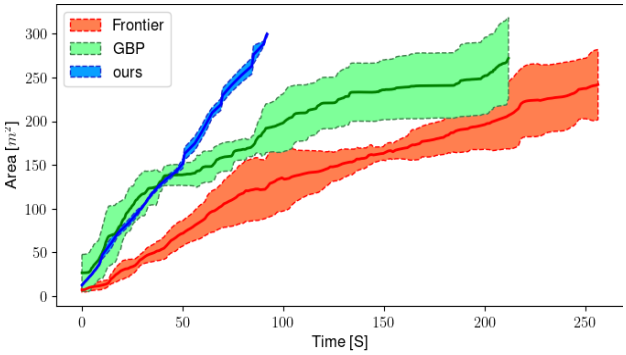



Fig. 5: Exploration rate (explored area versus time) in the indoor environment. Our method completed the task (end of the blue lines) earlier than the two baselines.

| Algorithm | Ours | GBPlanner | Frontier |
|---|---|---|---|
| Time[s] | 93 ± 1.30 | 169.20 ± 9.20 | 257 ± 29.50 |
| Distance[m] | 73 ± 3.90 | 91.50 ± 18.90 | 103 ± 39.10 |

TABLE I: Average time and average distance traveled by the robot to complete the exploration.

Fig. 4d shows the global RRT-based graph generated by the GBPlanner. The GBPlanner exploits its graph for both local and global path planning. The GBPlanner graph can be described as the combination of the local RRT trees used for local planning task which however cannot effectively represent the environment. While the GBPlanner global graph has an average of 205 vertices and 902 edges for the indoor environments, our metric-topological map represents the environment with an average of only 35 nodes and 44 edges which however covers all navigable spaces with their connectivity, see Fig. 4c.

We also compare the average update time of our global uncertainty map with different mapping algorithms; octomap [14], GPOM [26], and fGPOM [38]. Fig. 6 shows that the global uncertainty map update time (0.065 Sec) requires around one quarter of the fGPOM update time (0.27 Sec). The average update time for octomap and GPOM are 0.1 and 1.6 seconds, respectively. For a fair comparison, all the mapping algorithms were executed in the same CPU, including our uncertainty map.

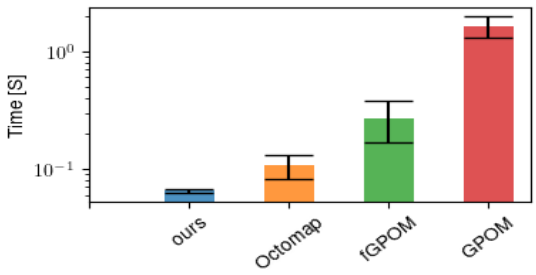
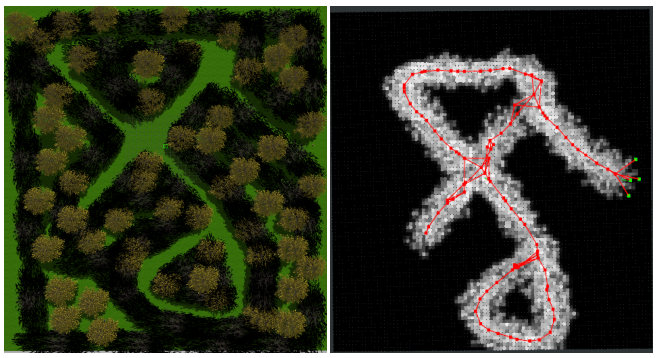


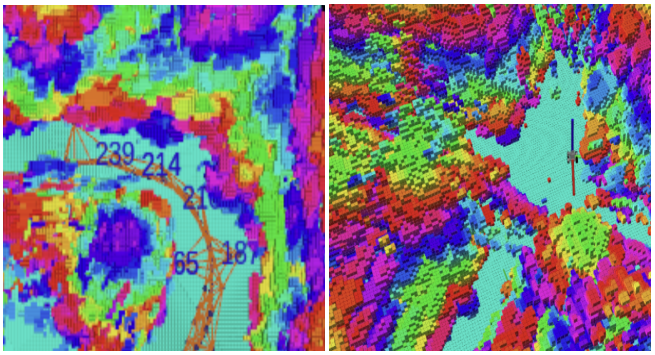

Fig. 6: Update time for different mapping algorithms.



(a) (b)

Fig. 7: Simulated forest environment with navigable trails. (a) An air-view of the scene; (b) Metric-topological map generated using our approach (in red) on top of the global uncertainty map (brighter color means more certain area). The frontier nodes (green squares) represent the exit of the trail.

*2) Simulated trail environment:* We then test in a forest-like environment where the navigable space is composed of narrow trails surrounded by trees and bushes, see Fig. 7. This environment is a more challenging scenario because trees and bushes produce more noisy LiDAR observations, and the environment has a much longer traversal distance than the indoor case. The navigable trails, deemed as the area to explore, are approximately 2500 $m^2$. Our algorithm explores these trails in an average time of 430 seconds with an average travel distance of 300 meters. We run the GBPLanner




(a) (b)

Fig. 8: GBPlanner exploring the simulated trail environment. (a) GBPlanner detects fake frontiers inside the tree branches (e.g. frontiers that are marked with information gains of 65 and 187); (b) A set of voxels generated by tree leaves block a true frontier. We visualize the 3D voxblox here to show the fake frontiers inside the tree branches.

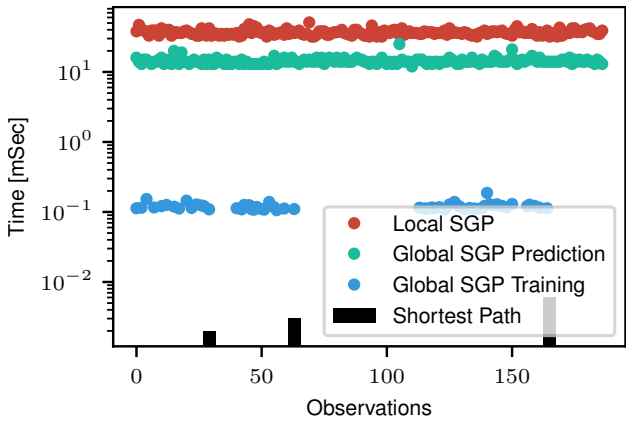
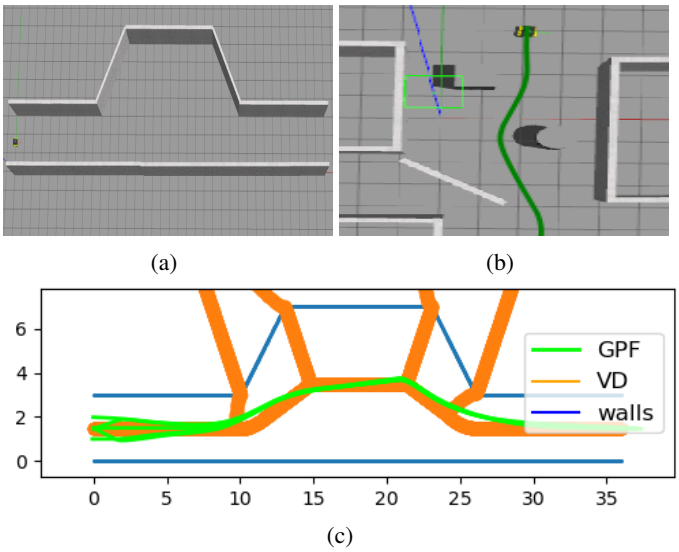



Fig. 9: Time cost for each step for different components during the simulated trail experiment.




(a) (b)

(c)

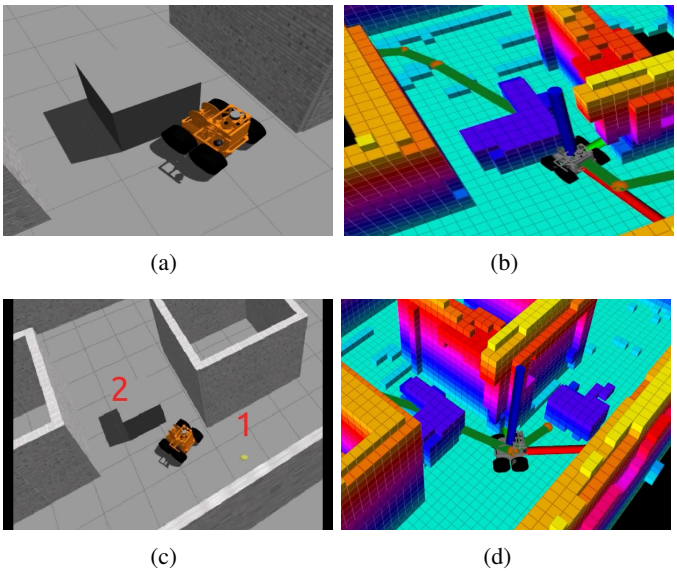
Fig. 10: GP frontier navigation behavior: (a) Simulated environment with variable passage width; (b) Static obstacles avoidance, the path generated by the local navigation algorithm is shown in green; (c) 5 paths (green lines) were generated by the GP frontier. For each path, the robot has a different starting pose; The Voronoi Diagram (VD) is plotted in orange to highlight the optimal path, considering optimality as maximizing distance to nearby obstacles.

in the simulated trail environment, but it does not complete the exploration. When using the GBPlanner with the default parameters, it gets stuck before completing the exploration because it cannot process the noisy pointcloud generated from trees and bushes. We tuned different parameters for the GBPLanner to adapt to the noisy pointcloud, but it was not able to complete the exploration because GBPlanner either: i) detects (fake) frontiers between tree branches and bushes but the robot cannot go inside as its physical dimension is bigger than the fake frontier, or ii) GBPlanner detects tree leaves from both sides of the trail as obstacles, and connects them to form a set of voxels, which blocks the trail and prevents the algorithm from detecting (true) frontiers, see Fig. 8.

In this experiment, where the algorithm runs for a longer duration than the cluttered environment experiment, we evaluated the runtime performance of the individual components of our framework. As illustrated in Fig. 9, the runtime of the main components did not exhibit exponential growth. The *Local SGP* curve represents the time taken to fit an occupancy surface into an SGP model and construct the variance surface, while the *Global SGP Training* curve shows the time required to re-tune the global SGP parameters based on new observations. The gaps in the *Global SGP Training* curve that coincide with the activation of the A* algorithm reflect the moments when the robot travels through a previously explored area towards the nearest map frontier without incorporating new observations into the global model. The computation time for the A* algorithm is measured in range of microseconds and it is only triggered a few times when the robot is in an already explored area. Note that, our framework based on SGP can be implemented in GPU, thus the update time for the uncertainty map can be further reduced, resulting 0.11 milliseconds in Fig. 9. This is less than the average update time (65 milliseconds) shown in Fig. 6 where we run it on CPU to compare with other mapping techniques which do not have GPU versions.

### B. Navigation Performance

We are also interested in the robot local navigation behavior guided by the GP frontiers. We demonstrate that the robot always attempts to maximize the distance to stationary obstacles nearby, leading to safe navigation behavior. The GP frontier local navigation behavior is investigated through three scenarios. In the first scenario, shown in Fig. 10a, the robot moves along a corridor with variable width. The optimal path, where the distance to obstacles is maximized, can be described by the Voronoi Diagram (VD) of the environment, as illustrated in Fig. 10c. In order to test whether our GP frontier algorithm can recover to the VD path, we demonstrate through four challenging starting poses that are not aligned with the optimal VD path. Specifically, two of the four poses are shifted




(a) (b) (c) (d)

Fig. 11: GBPlanner behavior towards dynamic obstacles. (a) and (b) show the behavior of the GBPlanner when a dynamic obstacle is added in the environment after a path is planned, the robot keeps following the old planned path and as a consequence, the robot hits the obstacle. (c) and (d) show that when a dynamic obstacle is moved from position '1' to position '2', the occupancy map behind the robot is not updated accordingly.

from the VD path by a half meter in positive and negative directions, and the other two poses have a wrong heading with respect to the VD path, where the robot heading is shifted by 45 degrees clockwise and counterclockwise, respectively. The results show that the paths followed by the robot are aligned with the VD optimal path, even when the robot started in a pose far from the VD path.

In the second scenario, a few stationary obstacles are placed in the environment. The robot shows a similar behavior where the GP frontier sets the navigation subgoals in the middle of the free space to maximize the distance to surrounding obstacles, see Fig. 10b.

In the third scenario, we observe the robot's reactive behavior by inserting new obstacles while the robot is moving. The robot can successfully avoid randomly inserted objects in front of it. The behavior can be observed in the supplementary video. We have also compared the behavior of our algorithm with that of the GBPlanner. The results reveal that the GBPlanner can run into dynamic obstacles that block the way, see Fig. 11.

### C. Demonstration in Real Environments

We then validate the proposed method through testing on a real mobile robot—Jackal from Clearpath Robotics. The robot is equipped with a Velodyne VLP-16 LiDAR and an Intel® Core i7 PC with 32 GB RAM and 6 GB Geforce RTX2060 GPU. For robot pose estimation including both position and orientation, we use the LOAM library [39]. (We believe other pose estimation schemes such as a fusion of GPS and IMU will also work.) An RGB camera is mounted in front of the robot for first-person-view recording only. The maximum linear and angular velocities are set to 0.4 meters per second and 1.0 radians per second, respectively.

The robot explores two different environments. The first environment is an indoor environment where the robot explores the main corridor of the lab building, and the second environment is a real forest.

*1) Corridor navigation, mapping and exploration:* In this experiment, we demonstrate that our algorithm is able to i) detect unexplored frontiers; ii) navigate the robot to the nearest frontier when the robot enters an explored area; and iii) connect unexplored frontier when they become inside the robot FOV, which results in a connected metric-topological map. Fig. 12a shows a camera view of the corridor which has a traversal distance of 95 meters. This experiment is repeated 5 times, and the robot shows a consistent behavior when it starts from the same initial pose. Fig. 12b shows the metric-topological map generated by our algorithm on top of the global uncertainty map.

It is worth mentioning that, our framework excels in cluttered, unstructured, and noisy environments, both indoors and outdoors. Yet, in vast obstacle-free spaces (e.g., a large empty parking lot), the performance declines due to high uncertainty in the local SGP perception model, causing navigation to resemble a random walk. To tackle this, we predefined a certain number of GP frontiers (e.g., front, back, left, right) within the circular SGP model, facilitating topological map creation and space exploration. In such cases, the map's resolution depends on the preset number of frontiers.

We have also evaluated the runtime performance of the hardware demonstration during the corridor experiment. Fig. 12c shows that the runtime performance of the local and global SGP models in the real hardware experiment is similar to that of the simulation evaluation conducted previously. Fitting the local SGP occupancy surface and reconstructing the variance surface take around 40 milliseconds, whereas updating the global uncertainty map with one observation takes approximately 0.11 milliseconds. The corridor experiment has more local observations than the simulated trail due to the slower movement of the robot in the real experiment, however, the number of observations used to update the global uncertainty map is not affected by the robot velocity, because the uncertainty map is only updated if and only if the robot enters an uncertain area.

*2) Real forest trails:* To validate the effectiveness of our algorithm in a challenging real environment, we have conducted a field trial in a local forest. Our goal is to compare the results of real-world experiments to those from simulated experiments. Since our neighborhood does not have a forest environment like the simulated scenario with circular trails, we have simplified the task to finding and marking unexplored trails as GP frontiers. In the first field trial, the robot safely navigated through an intersection of three trails and successfully recognized the unexplored GP frontiers. Additionally, it efficiently updated the global uncertainty map in real-time, see

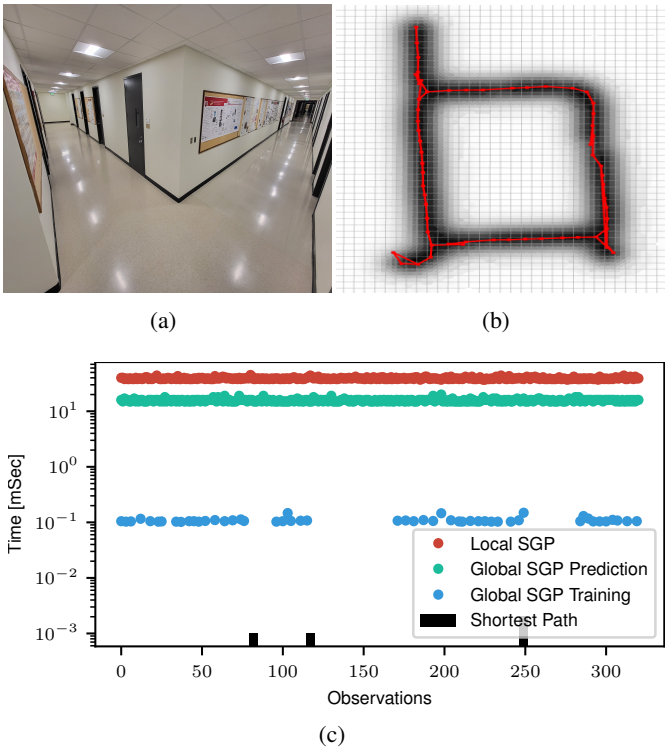



Fig. 12: Experiment in a corridor. (a) Camera view of the corridor; (b) Metric-topological map generated using our approach. (c) Time cost for each step for different components.

Figs. 13a and 13b.

In the second field trial, the task is to navigate a long, curved trail with a narrow passage, as shown in Figs. 13c and 13d. The objective of this demonstration is to showcase the ability of our framework to navigate safely in unstructured and noisy environments, as well as to construct a topological map in an adaptive manner based on exploration information encoded in the global uncertainty map. The GP frontier navigation is robust against noisy pointcloud data and the robot's path is aligned with the mid-line of the trail, resembling the Voronoi navigation demonstrated in the simulation experiments. These behaviors can be observed in the supplementary video.

## VI. Conclusion

This paper presents a new framework for a mobile robot to autonomously navigate, map, and explore unstructured environments. In contrast to existing work, our algorithm is derived from the sparse Gaussian process and does not rely on a costly detailed occupancy map, instead, our final output is a high-level structural representation of the environment delivered as a metric-topological map, which utilizes a global uncertainty map to identify the environmental locations that have the highest valuable observations (information) as the topological map nodes. Our experimental results demonstrate that we are able to explore environments more rapidly and cover more of the unknown space with a lower computational cost than existing techniques.

## Appendix

In this appendix we discuss how different parameters are selected:

- **M**: the number of the inducing points for the local SGP occupancy map is chosen to compromise on the computations complexity and the accuracy of the occupancy model. More inducing points will result in higher computations complexity $\mathcal{O}(N_{oc}.M^2)$, however, more inducing points will increase the accuracy of the reconstructed pointcloud. The number of the inducing points, $M = 250$, was chosen to keep the average deviation between the occupancy model and the original occupancy surface below 0.2 (20 cm in terms of radius).
- $\mathbf{V_{th}}$: the variance threshold that differentiates between high (GP frontier) and low variance regions on the variance surface. In fact, the variance related to the SGP occupancy model is different from one sensor observation to another, and it is affected by both the number of the occupied points $N_{oc}$, and their distribution over the occupancy surface. Therefore, we choose the variance threshold $V_{th}$ as a variable that changes with the distribution of the variance over the variance surface. $V_{th}$ is proportional to the variance mean $v_m$ over the variance surface, $V_{th} = K_m * v_m$ where $K_m$ is constant. High variance regions with a width less than a predefined threshold (based on robot size and occupancy radius $r_{oc}$) are neglected and are not considered GP frontiers. Also, a high variance region with a width greater than $\pi$ is divided into multiple GP frontiers based on its width.
- **J**: the number of the inducing points for the global SGP uncertainty map. As the exploration starts with zero acquired observations $K = 0$, we choose an adaptive value of $J$ where its value will be equal to the number of observations till it reaches a maximum number of $J_{max} = 500$.

$$J = \begin{cases} K & \text{if } K < J_{max} \\ J_{max} & \text{otherwise} \end{cases} \tag{13}$$

Table II shows the parameters we used during the experiments.


## Acknowledgement

This work is supported by the National Science Foundation with grant number 2047169 and in part by the U.S. Army Research Laboratory with grant number W911NF-22-2-0018.


## References


[1] Mahmoud Ali and Lantao Liu. Light-weight pointcloud representation with sparse gaussian process. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[2] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten H Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 131–142, 2006.

(a) (b) (c) (d)

Fig. 13: Demonstrations in real forests. (a) Intersection of 3 trail branches; (b) Pointcloud, grey-coded variance surface, and navigation subgoals (3 color-coded spheres) correspond to (a). (c) A trail scenario with curved and narrow passage. (d) Pointcloud, grey-coded variance surface, and navigation subgoal (2 color-coded spheres) correspond to (c).

| Component | Parameter | Meaning | Values |
|---|---|---|---|
| Local SGP occupancy model | $L_\theta, L_\alpha$ | *Lenghtscales* | 0.09, 0.12 |
| | $M$ | *Inducing points* | 250 |
| | $N_{oc\_max}$ | *Maximum number of occupied points* | 16000 |
| | $K_m$ | *Variance threshold for GP frontier* | 0.035 |
| Global SGP uncertainty map | $L_x, L_y$ | *Lenghtscales* | 2.5 , 2.5 |
| | $J$ | *Inducing points* | $\leq 500$ |
| | $U_{th}$ | *Exploration uncertainty threshold* | 7.8 |
| Others | $Ka$ | *Area weighting factor* | 0.1 |
| | $Kd$ | *Direction weighting factor* | 5 |

TABLE II: Parameters values used in experiments.


[3] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.

[4] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4):707–717, 2011.

[5] Henry Carrillo, Philip Dames, Vijay Kumar, and José A Castellanos. Autonomous robotic exploration using a utility function based on rényi's general theory of entropy. *Autonomous Robots*, 42(2):235–256, 2018.

[6] Benjamin Charrow, Sikang Liu, Vijay Kumar, and Nathan Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4791–4798. IEEE, 2015.

[7] Titus Cieslewski, Elia Kaufmann, and Davide Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135–2142, 2017. doi: 10.1109/IROS.2017.8206030.

[8] Tung Dang, Frank Mascarich, Shehryar Khattak, Christos Papachristos, and Kostas Alexis. Graph-based path planning for autonomous robotic exploration in subterranean environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3105–3112, 2019. doi: 10.1109/IROS40897.2019.8968151.

[9] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robotic exploration as graph construction. *J. Comput., vol*, 7(3), 1978.

[10] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564, 2012. doi: 10.1109/IROS.2012.6385934.

[11] Stephen Friedman, Hanna Pasula, and Dieter Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *IJCAI*, volume 7, pages 2109–2114, 2007.

[12] Héctor H González-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[13] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8. VDE, 2010.

[14] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[15] Maani Ghaffari Jadidi, Jaime Valls Miró, Rafael Valen-

cia, and Juan Andrade-Cetto. Exploration on continuous gaussian process frontier maps. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6077–6082, 2014. doi: 10.1109/ICRA.2014.6907754.

[16] Christoforos Kanellakis, Petros Karvelis, and George Nikolakopoulos. Open space attraction based navigation in dark tunnels for mavs. In *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings*, pages 110–119. Springer, 2019.

[17] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008.

[18] Neil Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Proceedings of the 16th annual conference on neural information processing systems*, pages 609–616, 2003.

[19] Sina Sharif Mansouri, Miguel Castaño, Christoforos Kanellakis, and George Nikolakopoulos. Autonomous mav navigation in underground mines using darkness contours detection. In *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings 12*, pages 164–174. Springer, 2019.

[20] Sina Sharif Mansouri, Petros Karvelis, Christoforos Kanellakis, Anton Koval, and George Nikolakopoulos. Visual subterranean junction recognition for mavs based on convolutional neural networks. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 192–197. IEEE, 2019.

[21] Jasna Maver and Ruzena Bajcsy. Occlusions as a guide for planning the next view. *IEEE transactions on pattern analysis and machine intelligence*, 15(5):417–433, 1993.

[22] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14:265–294, 1978.

[23] Michael T Ohradzansky, Andrew B Mills, Eugene R Rush, Danny G Riley, Eric W Frew, and J Sean Humbert. Reactive control and metric-topological planning for exploration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4073–4079. IEEE, 2020.

[24] Helen Oleynikova, Zachary Taylor, Roland Siegwart, and Juan Nieto. Sparse 3d topological graphs for micro-aerial vehicle planning. in 2018 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9.

[25] Teddy Ort, Krishna Murthy, Rohan Banerjee, Sai Krishna Gottipati, Dhaivat Bhatt, Igor Gilitschenski, Liam Paull, and Daniela Rus. Maplite: Autonomous intersection navigation without a detailed prior map. *IEEE Robotics and Automation Letters*, 5(2):556–563, 2019.

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

[27] Williams C. K. I. Rasmussen, C. E. *Gaussian processes for machine learning*. MIT press, 2005.

[28] Carl Rasmussen and Zoubin Ghahramani. Infinite Mixtures of Gaussian Process Experts. In *Advances in Neural Information Processing Systems*, 2002.

[29] Vachirasuk Setalaphruk, Atsushi Ueno, Izuru Kume, Yasuyuki Kono, and Masatsugu Kidode. Robot navigation in corridor environments using a sketch floor map. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No. 03EX694)*, volume 2, pages 552–557. IEEE, 2003.

[30] Shaojie Shen, Nathan Michael, and Vijay Kumar. Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle. *The International Journal of Robotics Research*, 31(12):1431–1444, 2012.

[31] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18:1257, 2006.

[32] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[33] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, D Montemerlo, Aaron Morris, Zachary Omohundro, et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91, 2004.

[34] Michalis K Titsias. Variational model selection for sparse gaussian process regression. *Report, University of Manchester, UK*, 2009.

[35] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

[36] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998.

[37] Yijun Yuan and Sören Schwertfeger. Incrementally building topology graphs via distance maps. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 468–474. IEEE, 2019.

[38] Yijun Yuan, Haofei Kuang, and Sören Schwertfeger. Fast gaussian process occupancy maps. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1502–1507. IEEE, 2018.

[39] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.