Real-Time Information-Theoretic Exploration with Gaussian Mixture Model Maps

Wennie Tabib Kshitij Goel John Yao Mosam Dabhi Curtis Boirum Nathan Michael

***Abstract*—This paper develops an exploration framework that leverages Gaussian mixture models (GMMs) for high-fidelity perceptual modeling and exploits the compactness of the distributions for information sharing in communications-constrained applications. State-of-the-art, high-resolution perceptual modeling techniques do not always consider the implications of transferring the model across limited bandwidth communications channels, which is critical for real-time information sharing. To bridge this gap in the state of the art, this paper presents a system that compactly represents sensor observations as GMMs and maintains a local occupancy grid map for a sampling-based motion planner that maximizes an information-theoretic objective function. The method is extensively evaluated in long duration simulations on an embedded PC and deployed to an aerial robot equipped with a 3D LiDAR. The result is significant memory efficiency as compared to state-of-the-art techniques.**

I. Introduction

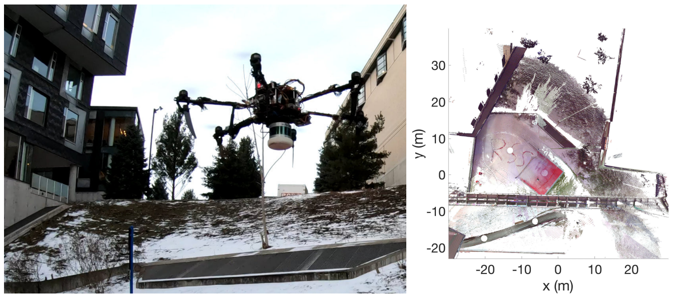
Robotic systems are critical for infrastructure inspection, search and rescue, and planetary exploration where limited bandwidth may preclude the transfer of high-fidelity perceptual models to human operators. For example, the data rate of the Mars-to-Earth direct communications channel varies between 500-32,000 bits per second [1]. On Earth, disaster-stricken environments often experience degraded or crippled telecommunications infrastructure across wide areas [2], which leads to preventable loss of life [3]. Robots that can explore and transmit high-resolution, memory-efficient maps are able to facilitate human response teams or receive guidance from operators to meet mission objectives.

This paper seeks to address the problem of constrained communications bandwidth robotic exploration by proposing an autonomous system that leverages compact perceptual models that generate high-resolution information about the robot's surroundings. Depth sensor observations are encoded as Gaussian Mixture Models (GMMs) and used to maintain a consistent local occupancy grid map. A motion planner is designed to select smooth and continuous trajectories that maximize an information-theoretic objective function.

The contributions of this work are: (1) a method for real-time occupancy reconstruction from Gaussian Mixture Models, (2) an information-theoretic exploration system that leverages the occupancy modeling technique, and (3) extensive evaluation of the exploration system in simulation and real-world scenarios using an aerial robot equipped with a 3D laser scanner. The paper is organized as follows: Section II surveys related work, Sections III, IV and V describe the proposed methodology, Section VI-A presents simulation results, Section VI-B presents field test results, and Section VII concludes with implications and future work.

This work was supported in part by NASA STTR Grant NNX16CK16C and industry.

The authors are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, USA. {wtabib, kgoel1, johnyao, mdabhi, cboirum, nmichael}@andrew.cmu.edu



(a) (b)

Fig. 1: (a) An autonomous aerial robot equipped with a 3D LiDAR explores an outdoor environment. (b) Top-down view of environment being explored.

II. Related Work

Planetary exploration systems to date have largely left questions about the cost of transmitting data to operators unaddressed. Arora et al. [4] leverage Bayesian networks to create a mapping between multi-modal sensor data and variables of interest coupled with a Monte Carlo Tree Search planner to maximize information gathering with the goal of limiting operator interaction given the significant delay induced by interplanetary travel. However, the cost of transmitting data to enable information sharing with human operators and planetary scientists is not discussed.

Hahn et al. [5] assume a tele-operated vehicle and develop a method to identify victims in a search and rescue scenario that fuses thermal and depth observations. Bandwidth constraints of the environment representation are not reported in the analysis of their approach. Papachristos et al. [6] develop an aerial robot to explore visually degraded, GPS-denied environments by fusing NIR camera and inertial data and build a high-resolution map by storing pointclouds, which is prohibitive to transmit across low-bandwidth communications infrastructure.

Many approaches leverage voxel-based occupancy modeling strategies for information-theoretic planning. Charrow et al. [7] develop a real-time information theoretic planning approach that maximizes the information gain in the environment by calculating the mutual information between a sensor observation and map represented as an occupancy grid. The occupancy grid map discretizes the environment into 3D voxels and stores a probability of occupancy for each voxel [8]. The downside of this map representation is that the memory demands become explosive as the environment to explore becomes larger or the desired resolution of the map becomes finer. To overcome this limitation, this work builds upon prior work by O'Meadhra et al. [9] that compresses sensor observations as GMMs for the purpose of occupancy reconstruction. This work extends these methods by developing real-time local occupancy mapping for information-theoretic planning.

The Normal Distribution Transform Occupancy Map (NDT-OM) was developed to overcome some of the limitations of the occupancy grid map. Saarinen et al. [10] discretize the environment into voxels in the same way that the occupancy grid map does, but also encode a Gaussian density into occupied voxels to better represent the surface. The motivation for this mapping approach is that because the surface is better represented, larger voxels may be used. However, for exploration in large environments, this technique suffers from the same drawbacks as the occupancy grid map.

Oleynikova et al. [11] develop Voxblox to produce Truncated Signed Distance Fields (TSDFs) from Euclidean Signed Distance Fields (ESDFs) and demonstrate the results on an aerial robot. The authors are able to outperform Octomap [12], which represents occupancy at multiple resolutions via an octree data structure. Voxblox may be dynamically resized, but is limited to a fixed-resolution voxel size. On the other hand, generative models such as GMMs learn a distribution over points in a sensor observation so that occupancy may be represented at arbitrary resolution [9].

GMMs have been used for compact perceptual modeling [13], occupancy modeling [9], and multi-robot exploration [14]. These models provide smoother surfaces with higher reconstruction accuracy at a significantly lower memory footprint [9] than discrete representations. Although communication-efficient exploration is proposed in [14], real-time operation is not demonstrated. This paper addresses this gap in the state of the art by proposing an exploration system that leverages GMMs for real-time 3D information-theoretic planning and perceptual modeling on computationally constrained platforms.

III. Overview

The proposed exploration system consists of mapping, information-theoretic planning, and a monocular visual-inertial navigation system (Fig. 2). The mathematical preliminaries of Gaussian mixture models (GMMs) are detailed in Section IV-A. Section IV-B develops the local grid mapping strategy via GMMs used by the planning approach to generate continuous trajectories that maximize an information-theoretic objective (Section V). Finally, the visual-inertial navigation and control subsystems are described in Section VI-B1.

IV. Mapping

A. Gaussian Mixture Models for Perception

The proposed approach leverages GMMs to compactly encode sensor observations for transmission over low-bandwidth communications channels. The GMM provides a generative

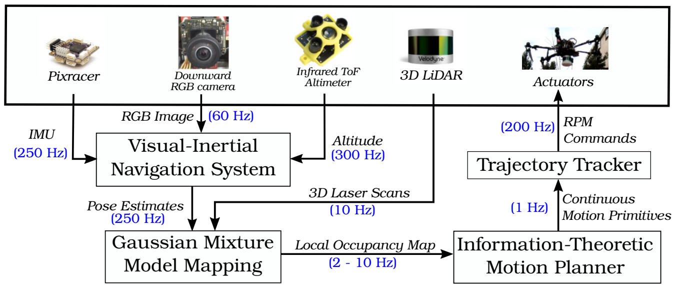


Fig. 2: Overview of the autonomous exploration system presented in this work. Using pose estimates from a visual-inertial navigation system (Section VI-B1) and 3D laser scans, the proposed mapping method (Section IV-A and Section IV-B) builds a memory-efficient continuous approximate belief representation of the environment while creating local occupancy grid maps in real-time. A motion primitives-based information-theoretic planner (Section V) uses this local occupancy map to generate snap-continuous forward-arc motion primitive trajectories that maximize the information gain over time.

model of the sensor observations from which occupancy may be reconstructed by resampling from the distribution and raytracing through a local occupancy grid map. Formally, the GMM is a weighted sum of M Gaussian probability density functions (PDFs). The probability density of the GMM is expressed as

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$$

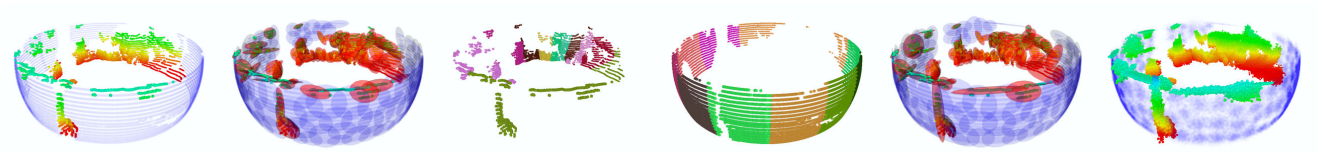
where $\boldsymbol{\Theta} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m\}_{m=1}^{M}$ compactly represent the parameters of the GMM. $\boldsymbol{x} \in \mathbb{R}^D$, $\pi_m \in \mathbb{R}^1$ is a weight such that $\sum_{m=1}^{M} \pi_m = 1$ and $0 \leq \pi_m \leq 1$, $\boldsymbol{\mu}_m$ is a mean, and $\boldsymbol{\Lambda}_m$ is a covariance matrix for the m^{th} D-dimensional Gaussian PDF of the distribution. The multivariate probability density for $\boldsymbol{x}$ is written as

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) = \frac{|\boldsymbol{\Lambda}_i|^{-1/2}}{(2\pi)^{D/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Lambda}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\Big).$$

Let $\mathcal{Z}_t = \{\boldsymbol{z}_t^1, \ldots, \boldsymbol{z}_t^n, \ldots, \boldsymbol{z}_t^N\}$ be a depth sensor observation consisting of N points, $\boldsymbol{z}_t^n \in \mathbb{R}^3$, taken at time t. Estimating optimal GMM parameters $\boldsymbol{\Theta}$ remains an open area of research [15]. This work employs the Expectation Maximization (EM) algorithm to solve the maximum-likelihood parameter estimation problem, which is guaranteed to find a local maximum of the log likelihood function [16]. To make the optimization tractable, EM introduces latent variables $\mathbf{C} = \{c_{nm}\}$ for each point $\boldsymbol{z}_t^n$ and cluster m and iteratively performs two steps [16, 17, 18].

The E step calculates the expected value of the complete-data log-likelihood $\ln p(\mathcal{Z}_t, \mathbf{C}|\boldsymbol{\Theta})$ with respect to the unknown variables $\mathbf{C}$ given the observed data $\mathcal{Z}_t$ and current parameter estimates $\boldsymbol{\Theta}^i$, which is written as $E[\ln p(\mathcal{Z}_t, \mathbf{C}|\boldsymbol{\Theta})|\mathcal{Z}_t, \boldsymbol{\Theta}^i]$ [17]. This amounts to evaluating the posterior probability, β_{nm}, using the current parameter values $\boldsymbol{\Theta}^i$ (shown in Eq. (1)) [16]

$$\beta_{nm} = \frac{\pi_m \mathcal{N}(\boldsymbol{z}_t^n|\boldsymbol{\mu}_m^i, \boldsymbol{\Lambda}_m^i)}{\sum_{j=1}^{M} \pi_j \mathcal{N}(\boldsymbol{z}_t^n|\boldsymbol{\mu}_j^i, \boldsymbol{\Lambda}_j^i)}, \tag{1}$$



(a) Pointcloud (b) Windowless GMM (c) Occupied Pointcloud (d) Free Pointcloud (e) Windowed GMM (f) Resampled model

Fig. 3: Overview of the approach to transform a sensor observation into a windowed GMM. (a) 3D LiDAR scan from Rapps Cave in Greenbrier, WV. Points at a distance larger than 5 m are normalized to a unit vector and projected to 5 m (dark blue) to represent free space. Points with a norm less than 5 m represent occupied space. (b) A 100-component GMM $\mathcal{F}(\boldsymbol{x})$ is learned over the free space points (dark blue) and 100-component GMM $\mathcal{G}(\boldsymbol{x})$ is learned over the occupied points (red). (c) and (d) illustrate the windowing technique on the occupied and free space points, respectively, by coloring each window in a different color. The GMM learned with windowed pointcloud data is shown in (e). The windowless GMM produces a tighter fit to the data than the windowed GMM which is exemplified in the occupied point component closest to the viewer. The resampled model produced by sampling 1×10^6 points from (e) is shown in (f). The number of points to resample is selected for illustration purposes and to highlight that the resampling process yields a map reconstruction with an arbitrary number of points.

where β_{nm} denotes the responsibility that component m takes for point $\boldsymbol{z}_t^n$. The M-step maximizes the expected log-likelihood using the current responsibilities, β_{nm}, to obtain updated parameters, $\boldsymbol{\Theta}^{i+1}$ via the following:

$$\boldsymbol{\mu}_m^{i+1} = \sum_{n=1}^{N} \frac{\beta_{nm}\boldsymbol{z}_t^n}{\sum_{n=1}^{N}\beta_{nm}} \tag{2}$$

$$\boldsymbol{\Lambda}_m^{i+1} = \sum_{n=1}^{N} \frac{\beta_{nm}(\boldsymbol{z}_t^n - \boldsymbol{\mu}_m^{i+1})(\boldsymbol{z}_t^n - \boldsymbol{\mu}_m^{i+1})^T}{\sum_{n=1}^{N}\beta_{nm}} \tag{3}$$

$$\pi_m^{i+1} = \frac{\sum_{n=1}^{N}\beta_{nm}\boldsymbol{x}_n}{\sum_{n=1}^{N}\beta_{nm}}. \tag{4}$$

Every iteration of EM is guaranteed to increase the log likelihood and EM is iterated until a local maximum of the log likelihood is achieved [16].

The E step is computationally expensive because a responsibility β_{nm} is calculated for each cluster m and point $\boldsymbol{z}_t^n$, which amounts to NM responsibility calculations. In the M step, every parameter must be updated by iterating over all N samples in the dataset. In practice, a responsibility matrix $\mathbf{B} \in \mathbb{R}^{N\times M}$ is maintained whose entries consist of the β_{nm} to estimate the parameters $\boldsymbol{\Theta}$. To reduce the computational complexity and enable online calculation, an approximation is made to partition the data into windows, learn a distribution for each window, and merge the results.

Let $\mathcal{G}_i(\boldsymbol{x})$ be a GMM trained from N_i points in window i and let $\mathcal{G}_j(\boldsymbol{x})$ be a GMM trained from N_j points in window j, where $\sum_{w=1}^{W} N_w = N$ for sensor observation $\mathcal{Z}_t$ and W windows. $\mathcal{G}_j(\boldsymbol{x}) = \sum_{k=1}^{K} \tau_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\nu}_k, \boldsymbol{\Omega}_k)$ may be merged into $\mathcal{G}_i(\boldsymbol{x}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Lambda}_m)$ by concatenating the means, covariances, and weights. However, care must be taken when merging the weights as they must be renormalized to sum to 1 [19]. The weights are renormalized via Eqs. (5) and (6):

$$N^* = N_i + N_j \tag{5}$$

$$\boldsymbol{\pi}^* = \begin{bmatrix} \frac{N_i\pi_1}{N^*} & \ldots & \frac{N_i\pi_m}{N^*} & \frac{N_j\tau_1}{N^*} & \ldots & \frac{N_j\tau_k}{N^*} \end{bmatrix}^T \tag{6}$$

where $m \in [1, \ldots, M]$ and $k \in [1, \ldots, K]$ denote the mixture component in GMMs $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$, respectively. $N^* \in \mathbb{R}^1$ is the sum of the support sizes of $\mathcal{G}_i(\boldsymbol{x})$ and $\mathcal{G}_j(\boldsymbol{x})$. $\boldsymbol{\pi}^* \in \mathbb{R}^{M+K}$ are the renormalized weights. The means and covariances are merged by concatenation. Following the work of O'Meadhra et al. [9], distinct free $\mathcal{F}(\boldsymbol{x})$ and occupied $\mathcal{G}(\boldsymbol{x})$ GMMs are maintained to compactly represent the density of points observed in the environment. The process by which $\mathcal{F}(\boldsymbol{x})$ and $\mathcal{G}(\boldsymbol{x})$ are created is illustrated in Figs. 3a and 3e. Because the GMM is a generative model, one may sample from the distribution to generate points associated with the surface model and reconstruct occupancy (detailed in Section IV-B). Figure 3 illustrates the model used to compactly represent the sensor observation.

B. *Local Occupancy Grid Map*

The occupancy grid map [20] is a probabilistic representation that discretizes 3D space into finitely many grid cells $\mathbf{m} = \{m_1, ..., m_{|\mathbf{m}|}\}$. Each cell is assumed to be independent and the probability of occupancy for an individual cell is denoted as $p(m_i|\mathbf{X}_{1:t}, \mathcal{Z}_{1:t})$ where $\mathbf{X}_{1:t}$ represents all vehicle states up to and including time t and $\mathcal{Z}_{1:t}$ represents the corresponding observations. Unobserved grid cells are assigned a uniform prior of 0.5 and the occupancy value of the grid cell m_i at time t is expressed using log odds notation for numerical stability.

$$l_{t,i} \triangleq \log\left(\frac{p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}{1 - p(m_i|\mathcal{Z}_{1:t}, \mathbf{X}_{1:t})}\right) - l_0$$

When a new measurement $\mathcal{Z}_t$ is obtained, the occupancy value of cell m_i is updated as

$$l_{t,i} \triangleq l_{t-1,i} + L(m_i|\mathcal{Z}_t)$$

where $L(m_i|\mathcal{Z}_t)$ denotes the inverse sensor model of the robot and l_0 is the prior of occupancy [20].

Instead of storing the occupancy grid map $\mathbf{m}$ that represents occupancy for the entire environment viewed since the start of exploration onboard the vehicle, a local occupancy grid map $\bar{\mathbf{m}}_t$ is maintained centered around the robot's pose $\mathbf{X}_t$. The local occupancy grid map moves with the robot, so when regions of the environment are revisited, occupancy must be reconstructed from the surface models $\mathcal{G}(\boldsymbol{x})$ and $\mathcal{F}(\boldsymbol{x})$. To reconstruct occupancy at time $t+1$ given $\bar{\mathbf{m}}_t$, the set difference of the bounding boxes b_t and b_{t+1} for $\bar{\mathbf{m}}_t$

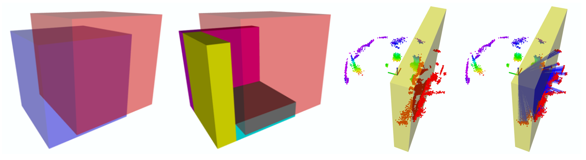

(a) (b) (c) (d)

Fig. 4: Overview of the method by which occupancy is reconstructed (a) The blue bounding box b_{t+1} is centered around $\mathbf{X}_{t+1}$ and red bounding box b_t is centered at $\mathbf{X}_t$. (b) illustrates the novel bounding boxes in solid magenta, teal, and yellow colors that represent the set difference $b_{t+1} \setminus b_t$. (c) Given a sensor origin shown as a triad, resampled pointcloud shown in Fig. 3f, and novel bounding box shown in yellow, each ray from an endpoint to the sensor origin is tested to determine if an intersection with the bounding box occurs. The endpoints of rays that intersect the bounding box are shown in red. (d) illustrates how the bounding box occupancy values are updated. Endpoints inside the yellow volume update cells with an occupied value. All other cells along the ray (shown in blue) are updated to be free.

and $\mathbf{m}_{t+1}$, respectively, are used to compute at most three non-overlapping bounding boxes (see Figs. 4a and 4b for example). The intersection of the bounding boxes remains up-to-date, but the occupancy of the novel bounding boxes must be reconstructed using the surface models $\mathcal{G}(\boldsymbol{x})$ and $\mathcal{F}(\boldsymbol{x})$. Raytracing is an expensive operation [21], so time is saved by removing voxels at the intersection of b_t and b_{t+1} from consideration.

The local occupancy grid map at time $t+1$, $\bar{\mathbf{m}}_{t+1}$, is initialized by copying the voxels in local grid $\bar{\mathbf{m}}_t$ at the intersection of b_{t+1} and b_t. In practice, the time to copy the local occupancy grid map is very low (on the order of a few tens of milliseconds) as compared to the cost of raytracing through the grid. In order to identify the GMM components that intersect the bounding boxes, a KDTree [22] stores the means of the densities. A radius equal to twice the sensor's max range is used to identify the components that could affect the occupancy value of the cells in the bounding box. A ray-bounding box intersection algorithm [23] checks for intersections between the bounding box and the ray from the sensor origin to density mean. Densities that intersect the bounding box are extracted into local submaps $\bar{\mathcal{G}}(\boldsymbol{x})$ and $\bar{\mathcal{F}}(\boldsymbol{x})$. Points are sampled from each distribution and raytraced to their corresponding sensor origin to update the local grid map (example shown in Figs. 4c and 4d).

As the number of mixture components in the distribution increases over time in one region, updating the occupancy becomes increasingly expensive as the number of points needed to resample and raytrace increases. To limit this potentially unbounded number of points, a small, fixed-size bounding box around the current pose with half-lengths h_x, h_y, and h_z is used to determine if a prior observation was made within the confines of the bounding box. This bounding box approach works for sensors that have a 360° field of view such as the 3D LiDAR used in this work, but does not readily extend to depth sensors with smaller fields of view. If a prior observation was made within the bounding box, the current observation, $\mathcal{Z}_t$ is not stored as a GMM.

V. Planning for Exploration

The planning framework consists of action generation and action selection. Action generation (detailed in Section V-A) refers to the design of candidate actions for the planner, while action selection in the exploration context refers to the planning policy that selects safe, feasible trajectories to minimize the uncertainty of the map over time (see Sections V-B and V-C).

A. Action Generation

Accurate position control of multirotors presumes continuity in supplied references up to high-order derivatives of position [24]. Actions that satisfy continuity requirements must be computable in real-time. This paper utilizes forward arc primitives for trajectory representation, first introduced by Yang et al. [25], for smooth and continuous teleoperation of multirotors. These primitives are generated via forward propagation of unicycle kinematics with higher-order endpoint constraints and have been successfully demonstrated in high-speed multirotor exploration and teleoperation scenarios [26, 27]. What follows is a brief overview of the concepts required to generate these motion primitives. For further detail please refer to [25, 26]. For the differentially-flat multirotor state at time t, $\boldsymbol{\xi}_t$ denote the action parameterization as $\mathbf{a} = [v_{\mathbf{x}}, v_{\mathbf{z}}, \omega]$ where $v_{\mathbf{x}}$ and $v_{\mathbf{z}}$ are velocities in the body frame in the $\mathbf{x}_{\mathcal{B}}$ and $\mathbf{z}_{\mathcal{B}}$ directions, respectively, and ω is the body frame angular rate about $\mathbf{z}_{\mathcal{B}}$ axis. Actions are discretized using the user-specified maximum velocity bounds in $\mathbf{x}_{\mathcal{B}} - \mathbf{y}_{\mathcal{B}}$ plane (ω variation, N_ω primitives) and $\mathbf{z}_{\mathcal{B}}$ plane ($v_{\mathbf{z}}$ variation, $N_{\mathbf{z}}$ primitives) to obtain a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$ given by (Figs. 5a and 5b):

$$\Gamma_{\boldsymbol{\xi}_t} = \{\gamma^{jk}_{\boldsymbol{\xi}_t} \mid j \in [1, N_\omega], k \in [1, N_{\mathbf{z}}], |\mathbf{v}| \le V_{\max}, |\omega| \le \Omega_{\max}\} \quad (7)$$

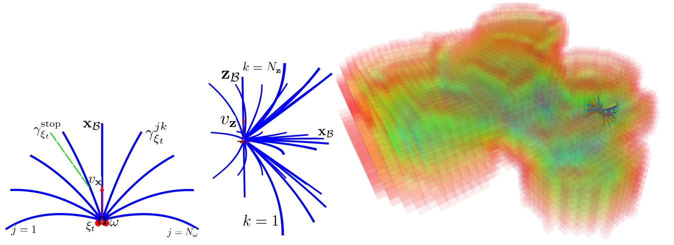
where $|\mathbf{v}|$ is the norm of $v_{\mathbf{x}}$ and $v_{\mathbf{z}}$, and $V_{\max}$ and $\Omega_{\max}$ are user-specified bounds on linear and angular velocities, respectively.

For a given action discretization, motion primitive $\gamma^{jk}_{\boldsymbol{\xi}_t}$ is generated as an 8th order polynomial in time using start- and end-point velocities, keeping position unconstrained. End-point velocity is obtained by forward propagating a unicycle kinematics model using the current state, maximum duration of the motion primitive (τ), and the available action parameterization. Higher-order derivatives from acceleration to snap are constrained to zero at endpoints:

$$\begin{aligned} \dot{\boldsymbol{\xi}}_\tau &= [v_{\mathbf{x}} \cos\theta, v_{\mathbf{x}} \sin\theta, v_{\mathbf{z}}, \omega] \\ \boldsymbol{\xi}^{(n)}_\tau &= \mathbf{0} \text{ for } n = 2, 3, 4 \end{aligned} \quad (8)$$

where $\{.\}^{(n)}$ denotes the nth time derivative.

For each MPL $\Gamma_{\boldsymbol{\xi}_t}$, an additional MPL containing stopping trajectories at any $\boldsymbol{\xi}_t$ can be sampled by fixing $\dot{\boldsymbol{\xi}}_\tau = \mathbf{0}$ ($\Gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$, Fig. 5a). These stopping trajectories are appended to the motion primitives in $\Gamma_{\boldsymbol{\xi}_t}$ at a timestep one planning round away from the starting time. These actions ensure safety if the planner fails to compute an optimal action, as described in Section V-C. The final action space $\mathcal{X}_{\text{act}}$ available for the



(a) ω variation (b) $v_{\mathbf{z}}$ variation (c) Frontier Distance Field

Fig. 5: Forward-arc motion primitives from the multirotor state $\boldsymbol{\xi}_t$ [26] are obtained after varying yaw rate (a) and vertical velocity (b). Stopping trajectories $\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$ are always computed for each primitive (green, dashed) to ensure safety. An explored local occupancy map is used to construct a distance field for frontiers (c) [14], which enables computation of a global reward as detailed in Section V-B.

MPL ID	Velocity and Duration	Direction	N_ω	$N_{\mathbf{z}}$	N_{prim}
1	$v_{\mathbf{x}}, \tau$	$\mathbf{x}_{\mathcal{B}}$	3	5	15
2	$v_{\mathbf{x}}, 2\tau$	$\mathbf{x}_{\mathcal{B}}$	3	5	15
3	$v_{\mathbf{x}}, \tau$	$-\mathbf{x}_{\mathcal{B}}$	3	5	15
4	$v_{\mathbf{x}}, 2\tau$	$-\mathbf{x}_{\mathcal{B}}$	3	5	15
5	$v_{\mathbf{z}}$	$\mathbf{z}_{\mathcal{B}}$	1	5	5

TABLE I: Discretization used to construct action space $\mathcal{X}_{\text{act}}$ for the simulation and hardware experiments. Total number of primitives for a MPL are denoted by $N_{\text{prim}} = N_\omega \cdot N_{\mathbf{z}}$. Base duration τ was kept 3 s for all experiments.

motion planner is composed of the MPLs shown in Table I [1]. The total number of actions available in this configuration of MPLs is chosen such that computation of rewards, described later in Section V-B, is real-time feasible.

B. Information-Theoretic Objective

The action selection policy uses an information-theoretic objective to maximize the information gain over time. The choice of this objective is motivated by computational feasibility onboard a compute-constrained platform. Charrow et al. [7] show that Cauchy-Schwarz Quadratic Mutual Information (CSQMI) is a computationally efficient alternative to the Shannon mutual information.

To minimize redundant computation, CSQMI is computed only at the end point of the primitive $\gamma_{\boldsymbol{\xi}_t}$. In the planner, this metric is used to measure the information gain a candidate action will return locally, however, this design may result in myopic decision-making and only reason locally. Therefore, an additional global distribution of information is incorporated via frontiers [28]. This global reward, denoted by $\mathcal{V}_\gamma$, is calculated based on the change in distance towards a frontier along a candidate action. Using the node state $\boldsymbol{\xi}_0$, endpoint state $\boldsymbol{\xi}_\tau$, and a distance field constructed based on the position of the frontiers (see Fig. 5c), this reward can be calculated as $\mathcal{V}_\gamma = d(\boldsymbol{\xi}_0) - d(\boldsymbol{\xi}_\tau)$, where $d(\boldsymbol{\xi}_t)$ denotes the distance to the nearest voxel in the distance field from state $\boldsymbol{\xi}_t$ [14].

[1]This design is dependent on the sensor model and SWaP constraints of the aerial robot. We follow the analysis presented in [27] for the selection of these design parameters.

Algorithm 1 Overview of Action Selection for Exploration

```
1:  input: $\mathcal{X}_{\text{act}}$, $\mathcal{X}_{\text{free}}$
2:  output: $\gamma^*_{\boldsymbol{\xi}_t}$                                  ▷ best action
3:  for $\Gamma_{\boldsymbol{\xi}_t} \in \mathcal{X}_{\text{act}}$ do
4:      for $\gamma_{\boldsymbol{\xi}_t} \in \Gamma_{\boldsymbol{\xi}_t}$ do
5:          feasible ← SafetyCheck($\gamma_{\boldsymbol{\xi}_t}$, $\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$, $\mathcal{X}_{\text{free}}$)
6:          if feasible then
7:              $\mathcal{I}_\gamma \leftarrow$ InformationReward($\gamma_{\boldsymbol{\xi}_t}$)
8:              $\mathcal{V}_\gamma \leftarrow$ FrontierDistanceReward($\gamma_{\boldsymbol{\xi}_t}$)
9:          else
10:             $\mathcal{I}_\gamma \leftarrow 0.0$, $\mathcal{V}_\gamma \leftarrow 0.0$
11: return $\gamma^*_{\boldsymbol{\xi}_t} \leftarrow \arg\max_{\gamma_{\boldsymbol{\xi}_t} \in \mathcal{X}_{\text{act}}} [\mathcal{I}_\gamma + \mathcal{V}_\gamma]$
```

C. Action Selection

Using the rewards described in the preceding section, the objective for the motion planner is defined as follows [27, 14]:

$$\begin{aligned} \arg\max_{\gamma_{\boldsymbol{\xi}_t}} \; & \mathcal{I}_\gamma + \alpha\, \mathcal{V}_\gamma \\ \text{s.t.} \; & \gamma_{\boldsymbol{\xi}_t} \in \mathcal{X}_{\text{act}} \end{aligned} \tag{9}$$

where α is a weight that adjusts the contribution of the frontier distance reward. As stated earlier, the goal is to maximize this reward function in real-time on a compute-constrained aerial platform. Previous information-theoretic approaches that construct a tree and use a finite-horizon planner either do not use a global heuristic [29] or are not known to be amenable for operation on compute-constrained platforms [14]. Keeping real-time operation as a key goal in this work, a single-step planner is used with the action space $\mathcal{X}_{\text{act}}$ consisting of motion primitives of varying duration (see Table I). Due to this design of candidate actions, the planner is able to compute rewards over candidate actions further into the explored map from the current position. This way, even in a single-step planning formulation, longer duration candidate actions provide a longer lookahead than the case when all candidate actions are of the same duration (see Table I).

The action selection procedure is detailed in Algorithm 1. For every candidate action $\gamma_{\boldsymbol{\xi}_t}$ in the action space $\mathcal{X}_{\text{act}}$, a safety check procedure is performed to ensure that this candidate and the associated stopping action ($\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$) are dynamically feasible and lie within free space $\mathcal{X}_{\text{free}}$ (Line 5). The free space check is performed using a Euclidean distance field created from locations of occupied and unknown spaces in the robot's local map given a fixed collision radius [30]. Checking that the stopping action is also feasible ensures that the planner never visits an inevitable collision state, which is essential for safe operation, as shown in [31]. If the action is feasible, the local information reward ($\mathcal{I}_\gamma$, Line 7) and frontier distance reward ($\mathcal{V}_\gamma$, Line 8) are determined as described in Section V-B. The planner then returns the action with the best overall reward (Line 11).

VI. Experimental Design and Results

This section details the experimental design to validate the proposed approach[2]. Results are shown for 25 hours of real-time simulation trials in unstructured, 3D rich environments, as well as for field test experiments where the approach is deployed on hardware. The following shorthand is introduced for this section only: MCG will refer to the Monte Carlo GMM mapping approach and OG mapping will refer to the Occupancy Grid mapping approach. The mapping and planning software is run on an embedded Gigabyte Brix 6500U with four cores. The simulation computer has 8GB RAM and the hardware computer has 16GB RAM. All other technical specifications are identical for the simulation and hardware setups.

All simulation and hardware experiments use 10 windows each with 10 mixture components for the occupied- and free-space GMMs (200 mixture components total) with $h_x = h_y = 2\,\mathrm{m}$ and $h_z = 1\,\mathrm{m}$. The LiDAR has a max range of $5.0\,\mathrm{m}$ and operates at $10\,\mathrm{Hz}$ for both simulation and hardware. The local occupancy grid maps for both MCG and OG mapping span $20\,\mathrm{m} \times 20\,\mathrm{m} \times 12\,\mathrm{m}$ at a $0.2\,\mathrm{m}$ resolution. To calculate memory requirements for the OG mapping approach, the incremental OG map is transmitted as a changeset pointcloud where each point consists of 4 floating point numbers: $\{x, y, z, \text{logodds}\}$. The changeset is computed after insertion of every pointcloud and a floating point number is assumed to be 4 bytes, or 32 bits. For the MCG approach, the cumulative data transferred is computed by summing the cost of transmitted GMMs. Each mixture component is transmitted as 10 floating point numbers: 6 numbers for the covariance matrix (because the covariance matrix is symmetric), three numbers for the mean, and one number for the mixture component weight. One additional number is also stored per GMM that represents the number of points from which the GMM was learned.

Before discussing the results of the exploration approaches, the choice of the windowing strategy presented in Section IV-A is analyzed for reconstruction accuracy. Fig. 6 employs the Area under ROC Curve (AUC) to evaluate the accuracy of the occupancy reconstruction using the windowed and windowless GMM approaches. Ground truth probabilities for the occupancy grid map are computed by raytracing the sensor observation through a traditional occupancy grid map and updating the probabilities of occupancy via the inverse sensor model.

Given the pointcloud in Fig. 3a consisting of N points, the N_o occupied space points shown in colors varying from red to green are separated into 10 windows of points (shown in Fig. 3c). For each window $i \in [1, \ldots, 10]$, a GMM $\mathcal{G}_i(\boldsymbol{x})$ consisting of 10 mixture components is learned and merged into a single GMM $\mathcal{G}(\boldsymbol{x})$ consisting of a total of 100 mixture components (shown in red in Fig. 3e). The same process is repeated to generate a free space GMM $\mathcal{F}(\boldsymbol{x})$ with free space points (illustrated in blue in Fig. 3e). The occupancy

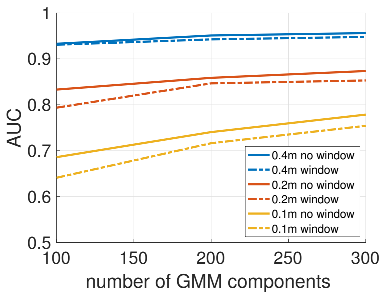


Fig. 6: The quality of reconstruction when opting for the windowed strategy (dashed lines) shown in Fig. 3e as opposed to learning a distribution on all of the data (solid lines) as in Fig. 3b is presented. The effect of partitioning data into 10 windows, learning a distribution on each of the windows, and merging the result does not significantly reduce the quality of the reconstruction. The gains in speed are significant enough to motivate deploying the windowing strategy for real-time operations.

is reconstructed by sampling and raytracing N_o points from $\mathcal{G}(\boldsymbol{x})$ and N_f points from $\mathcal{F}(\boldsymbol{x})$ such that $N = N_o + N_f$ through an occupancy grid map. Figure 6 illustrates the occupancy reconstruction accuracy for varying cell sizes ($0.1\,\mathrm{m}$, $0.2\,\mathrm{m}$, and $0.4\,\mathrm{m}$). All simulation and hardware experiments in this chapter use $0.2\,\mathrm{m}$ voxel sizes. The windowing approach results in a 1.4% performance decrease and a $10\times$ speed up in calculation ($1.22\,\mathrm{s}$ for the windowless approach vs. $0.12\,\mathrm{s}$ for the windowed approach) for 100-component $\mathcal{G}(\boldsymbol{x})$ and 100-component $\mathcal{F}(\boldsymbol{x})$, reducing computation time from seconds to milliseconds. Exploiting the spatial locality through windowing reduces the computational complexity associated with learning the distribution. For example, the responsibility matrix learned on all of the occupied data is of size $N_o \times M$ but each windowed GMM has a responsibility matrix of size $\frac{N_o}{10} \times \frac{M}{10}$.

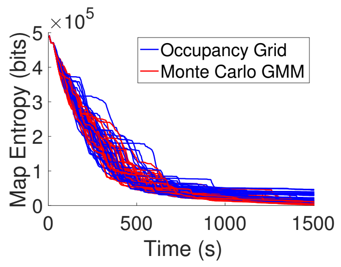
A. Simulation Design and Experiments

1) Simulation Setup: The proposed exploration strategy is evaluated with 60 real-time simulation trials over approximately 25 hours in a $30\,\mathrm{m} \times 40\,\mathrm{m} \times 6\,\mathrm{m}$ environment constructed from colorized FARO[3] pointclouds of Rapps Cave, located in Greenbrier, West Virginia (see Fig. 8a). In each simulation, the multirotor robot begins exploration from one of three pre-determined starting positions and explores for $1500\,\mathrm{s}$. This end time for the exploration experiments is set empirically, based on the total time required to fully explore the cave environment. Note that ground truth state estimates are used for these simulation experiments, while the hardware experiments detailed in Section VI-B use a visual-inertial odometry estimator (see Section VI-B1).

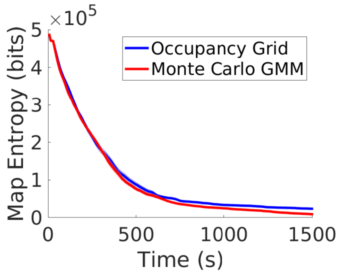
2) Results: Map entropy reduction over time is used to compare exploration performance of the MCG and OG approaches over 30 simulation trials (see Fig. 7a). Map entropy is measured from a global occupancy grid map that is maintained separately from the simulated robot's onboard map [32]. The simulation trials demonstrate that MCG achieves similar performance as OG, which indicates that the approximations

[2]Video of the simulation and hardware experiments may be found at https://youtu.be/-XADDZa4J0Y

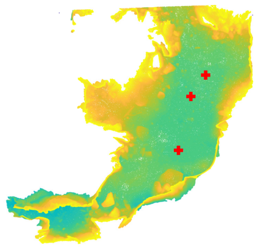
[3]A FARO is a survey grade 3D laser that emits pointclouds with color. https://www.faro.com/

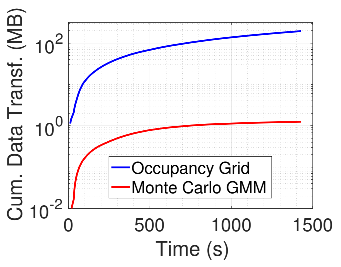


(a) Map Entropy



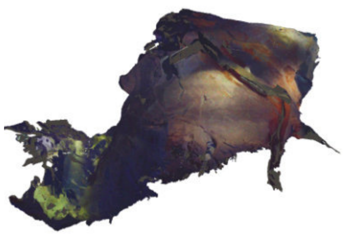
(b) Map Entropy

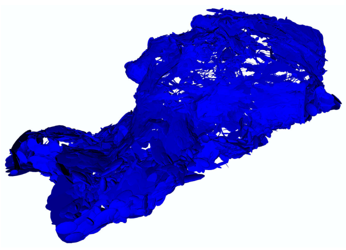

(c) Starting Positions

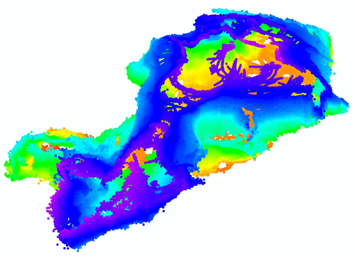


(d)

Fig. 7: Exploration statistics for simulation experiments. (a) Map entropy over time for 30 trials, (b) mean map entropy over time for each method, and (c) three different starting positions (10 trials per starting position per method). Although both methods achieve similar entropy reduction, MCG uses significantly less memory according to the average cumulative data transferred shown in (d). The total amount of transmitted map data after $1500\,\mathrm{s}$ is $1.3\,\mathrm{MB}$ for MCG, $204\,\mathrm{MB}$ for OG, and $4.5\,\mathrm{GB}$ (not shown) for raw pointclouds. The maximum variance for the MCG approach is $4.2 \times 10^{-3}\,\mathrm{MB}^2$ and $1.2 \times 10^{2}\,\mathrm{MB}^2$ for the OG approach. The proposed MCG method represents a decrease of two orders of magnitude as compared to the OG method.


(a) Simulated Cave Environment


(b) Resulting GMM after 1500 seconds


(c) Resampled map from GMM

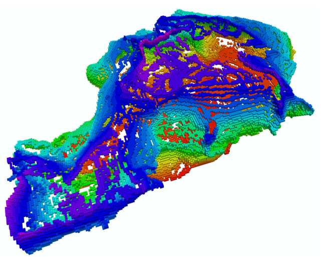

(d) Dense voxel map after 1500 seconds

Fig. 8: Environment used for simulation experiments, shown in (a), is based on a real cave environment in West Virginia. After 1500 seconds of autonomous exploration, the resulting MCG map (b) is densely resampled to 10 million points to obtain the reconstruction shown in (c). The output of the OG mapping strategy consists of $50,398$ voxels and is shown in (d). Both mapping strategies leverage voxels with $20\,\mathrm{cm}$ resolution. (c) and (d) are colored according to z-height.

made by the former enables real-time performance without compromising exploration or map reconstruction quality.

Figure 7d depicts the cumulative amount of data that must be transferred to reproduce the OG and MCG maps remotely. After $1500\,\mathrm{s}$, transferring the MCG map requires $1.3\,\mathrm{MB}$ as compared to $204\,\mathrm{MB}$ to incrementally transfer the OG map and $4.5\,\mathrm{GB}$ to transfer the raw pointcloud data. The MCG approach significantly outperforms the other approaches in terms of cumulative data transfer requirements.

A representative example of the reconstructed GMM map for one trial from Fig. 7b is shown in Fig. 8b. Resampling one million points from this distribution yields the map shown in Fig. 8c. Note that the MCG reconstruction is smoother and has a higher fidelity than the OG reconstruction because the generative model used by the former does not assume independence between voxels.

B. Hardware Design and Experiments

A $6.7\,\mathrm{kg}$ aerial robot (see Fig. 9b) equipped with a downward-facing global shutter camera, IMU, and 3D LiDAR is used to conduct field experiments.

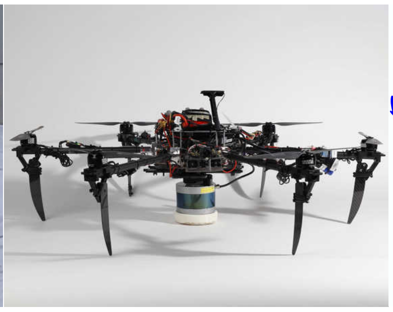
1) Visual-Inertial Navigation and Control: State estimates are computed from IMU and downward facing camera observations via VINS-Mono [33], a tightly-coupled visual-inertial odometry framework that jointly optimizes vehicle motion, feature locations, camera time delay, and IMU biases over a sliding window of monocular images and pre-integrated IMU measurements. An auxiliary state estimator runs during takeoff to provide smooth odometry when the vehicle has not yet experienced sufficient motion excitation for VINS-Mono to initialize. The auxiliary state estimator is an unscented Kalman filter that fuses downward rangefinder altitude, downward optical flow, and IMU observations to estimate vehicle odometry. The loop closure functionality of VINS-Mono is disabled to avoid having relocalization-induced discontinuities in the trajectory estimate, which would have significant implications for occupancy mapping and is left as future work.

For accurate trajectory tracking, a cascaded Proportional-Derivative (PD) controller is used with a nonlinear Luenberger observer to compensate for external acceleration and torque disturbances acting on the system [34]. To improve trajectory tracking, the controller uses angular feedforward velocity and acceleration terms computed from jerk and snap references sampled from the reference trajectory's 8th order polynomial (Fig. 2). For the one minute flight shown in Fig. 9, the vehicle traveled a total distance of $28\,\mathrm{m}$ and experienced about $2\,\mathrm{m}$ position drift and 17° heading drift.

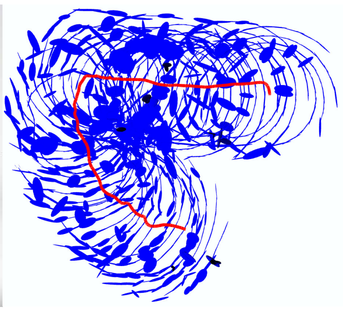
Additionally, a state machine enables the user to trigger transitions between the following modes of flight operation: (1) take-off, (2) hover, (3) tele-operation, (4) autonomous exploration, and (5) landing. The results presented in the next section all pertain to the autonomous exploration mode.

2) Results: Four experimental trials are conducted and evaluated outdoors in inclement weather with duration of

(a) Robot trajectory

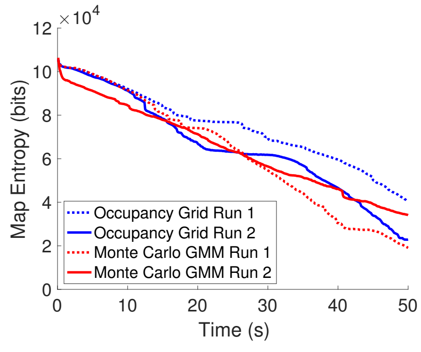


(b) Aerial robot with 3D LiDAR

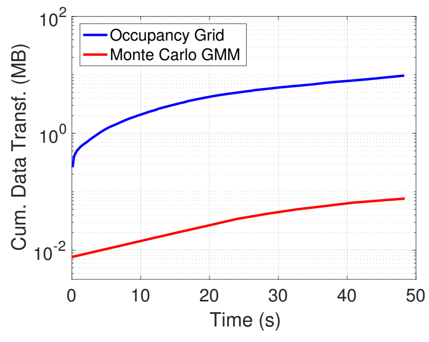


(c) GMM Map constructed during flight

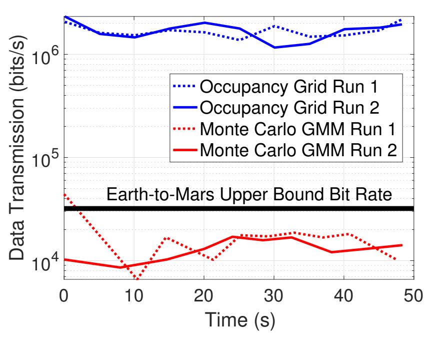
Fig. 9: Experimental setup for hardware trials. (a) Test site for outdoor exploration experiments with robot trajectory highlighted in red. (b) Hexrotor aerial robot used in field tests. (c) Top-down view of the GMM map (blue) constructed during the 60 s flight and the estimated trajectory (red).



(a)



(b)



(c)

Fig. 10: Exploration statistics for hardware experiments. (a) illustrates the map entropy reduction for two trials of each approach. (b) represents the average cumulative data transferred at a given time in a semi-logarithmic plot where the vertical axis is logarithmic labeled with successive powers of 10. The maximum variance is $2.9 \times 10^{-5}\,\mathrm{MB}^2$ and $7.2\,\mathrm{MB}^2$ for the MCG and OG approaches, respectively. The total amount of data transferred at the end of each hardware trial on average is approximately 78 kB for the MCG mapping approach, 9.7 MB for the OG approach, and 154 MB when transmitting raw pointclouds (not shown). (c) illustrates the bit rate for each approach in a semi-logarithmic plot where the vertical axis is logarithmic labeled with successive powers of 10. The communications bandwidth required for the MCG approach is just under the upper bound on the available Earth-to-Mars bit rate.

50 s. This duration is chosen keeping in mind the safety constraints imposed on the operation of the multirotor due to degraded battery performance in inclement weather. Given that re-localization is not enabled in state estimation, the pose estimates drift over time. To compensate for this in the analysis of exploration quality, raw sensor observations are logged onboard and post-processed to estimate ground truth poses using a FARO ground truth map (shown in Fig. 1b). Ground truth estimates are computed by manually aligning the first LiDAR sensor observation with the FARO map to obtain an initial pose estimate $\mathbf{T}_{f\mathcal{Z}_1} \in \mathrm{SE}(3)$, where f denotes the FARO frame and $\mathcal{Z}_1$ denotes the sensor frame of the first LiDAR observation. Successive LiDAR observations are registered using GICP [35] to estimate $\mathbf{T}_{\mathcal{Z}_t\mathcal{Z}_{t+1}} \in \mathrm{SE}(3)$ and the transform $\mathbf{T}_{f\mathcal{Z}_t}\mathbf{T}_{\mathcal{Z}_t\mathcal{Z}_{t+1}}$ is used to seed the GICP registration between the sensor observation at time $t+1$ and the FARO map. The ground truth pose estimates are used to provide a fair comparison between the two exploration approaches and generate the map entropy reduction plots shown in Fig. 10a. These plots illustrate almost equivalent performance between the MCG and OG mapping approaches, however, Fig. 10b demonstrates the MCG approach requires $100\times$ less memory as compared to the OG approach. Fig. 10c illustrates the reduction in the communication bandwidth required by the MCG approach compared to the OG approach, which brings the former just under the upper bound on Earth-to-Mars bit rate [1].

VII. Conclusions

This paper presented a method of high-fidelity perceptual modeling that is amenable to transmission across low-bandwidth communications channels. In analyzing the results of the hardware trials in the context of the Mars-to-Earth lowest communications bit rate (500 bits per second), it would take almost 30 days to transmit the raw pointcloud data, 45 hours to transmit the occupancy grid map incrementally, and 21 minutes to transmit the GMM map [1]. At the highest bit rate of 32,000 bits per second, those numbers are changed to approximately 0.5 days, 0.7 hours, and 20 seconds, respectively. This mapping capability is an enabling technology for search and rescue, planetary exploration, and tactical operations where humans and robots must share information in real-time. Future work will consist of introducing re-localization strategies to curb drift over long duration flights.

VIII. Acknowledgments

The authors thank Xuning Yang for fruitful discussions about motion primitives-based planning and Aditya Dhawale for feedback on this manuscript.

References

[1] (2019) Mars science laboratory data rates/returns. [Online]. Available: https://marsmobile.jpl.nasa.gov/msl/mission/communicationwithearth/data/

[2] V. H. Cid, A. R. Mitz, and S. J. Arnesen, "Keeping communications flowing during large-scale disasters: leveraging amateur radio innovations for disaster medicine," *Disaster medicine and public health preparedness*, vol. 12, no. 2, pp. 257–264, 2018.

[3] A. M. Townsend and M. L. Moss, *Telecommunications Infrastructure in Disasters: Preparing Cities for Crisis Communication*. Robert F. Wagner Graduate School of Public Service, New York University, 2005.

[4] A. Arora, P. M. Furlong, R. Fitch, S. Sukkarieh, and T. Fong, "Multi-modal active perception for information gathering in science missions," *arXiv preprint arXiv:1712.09716*, 2017.

[5] R. Hahn, D. Lang, M. Häselich, and D. Paulus, "Heat mapping for improved victim detection," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 116–121.

[6] C. Papachristos, S. Khattak, and K. Alexis, "Autonomous exploration of visually-degraded environments using aerial robots," in *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*. IEEE, 2017, pp. 775–780.

[7] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, WA, May 2015.

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

[9] C. O'Meadhra, W. Tabib, and N. Michael, "Variable resolution occupancy mapping using Gaussian mixture models," *IEEE Robotics and Automation Letters*, p. 1, 2018, early access.

[10] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal, "Normal distributions transform occupancy maps: Application to large-scale online 3d mapping," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2233–2238.

[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 *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 1366–1373.

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

[13] S. Srivastava and N. Michael, "Efficient, multifidelity perceptual representations via hierarchical gaussian mixture models," *IEEE Transactions on Robotics*, 2018.

[14] M. Corah, C. OMeadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1715–1721, April 2019.

[15] R. Hosseini and S. Sra, "An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization," *Mathematical Programming*, pp. 1–37, 2017.

[16] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag New York, 2007.

[17] J. A. Bilmes *et al.*, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, p. 126, 1998.

[18] B. Eckart, K. Kim, A. Troccoli, A. Kelly, and J. Kautz, "Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration," in *2015 International Conference on 3D Vision (3DV)*. IEEE, 2015, pp. 241–249.

[19] S. Srivastava, "Efficient, multi-fidelity perceptual representations via hierarchical gaussian mixture models," Master's thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, August 2017.

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

[21] J. Amanatides, A. Woo *et al.*, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[22] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees," https://github.com/jlblancoc/nanoflann, 2014.

[23] A. Williams, S. Barrus, R. K. Morley, and P. Shirley, "An efficient and robust ray-box intersection algorithm," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005, p. 9.

[24] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011.

[25] X. Yang, K. Sreenath, and N. Michael, "A framework for efficient teleoperation via online adaptation," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5948–5953.

[26] A. Spitzer, X. Yang, J. Yao, A. Dhawale, K. Goel, M. Dabhi, M. Collins, C. Boirum, and N. Michael, "Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor," in *Proc. of the Intl. Sym. on Exp. Robot.* Buenos Aires, Argentina: Springer, 2018, to be published.

[27] K. Goel, M. Corah, and N. Michael, "Fast exploration using multirotors: Analysis, planning, and experimentation," The Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-19-03, 2019.

[28] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. of the Intl. Sym. on Comput. Intell. in Robot. and Autom.*, Monterey, CA, Jul. 1997.

[29] W. Tabib, M. Corah, N. Michael, and R. Whittaker, "Computationally efficient information-theoretic exploration of pits and caves," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, Korea, Oct. 2016.

[30] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice," *Auton. Robots*, 2018.

[31] L. Janson, T. Hu, and M. Pavone, "Safe motion planning in unknown environments: Optimality benchmarks and tractable policies," in *Proc. of Robot.: Sci. and Syst.*, Pittsburgh, PA, Jul. 2018.

[32] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[33] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robotics*, vol. 34, no. 4, pp. 1004–1020, August 2018.

[34] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," in *IEEE Robot. Autom. Mag.*, Sep. 2010.

[35] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4, 2009, p. 435.