Reactive Collision Avoidance using Real-Time Local Gaussian Mixture Model Maps

Aditya Dhawale, Xuning Yang and Nathan Michael

***Abstract*— In unknown, cluttered environments, robots require online real-time mapping and collision checking in order to navigate robustly. Discrete map representations are inefficient for collision checking as they are expensive in terms of memory and computation. This paper takes a probabilistic approach to local mapping by representing the environment as a Gaussian Mixture Model (GMM) and leverages its geometric properties to enable efficient collision checking given a time-parameterized trajectory. In contrast to current discretization-based methods, a GMM preserves geometric coverage of the environment without losing representation accuracy with varying map resolutions. We introduce a novel GMM local mapping algorithm that can be used with a single depth camera processed on a single CPU, and provide algorithms for collision avoidance given arbitrary trajectory representations. Finally, we provide experimentation results demonstrating safety, efficiency, and data coverage for real-time collision avoidance with a quadrotor navigating in a cluttered environment.**

I. INTRODUCTION

We propose a reactive, online collision checking approach to enable fast obstacle avoidance in unknown, cluttered environments. Mobile robots have moved from a paradigm of operating in well-defined environments to complex and dynamic environments, and require online, real-time mapping and collision checking that can operate at a minimum of 10Hz. State-of-the-art methods utilize laser range finders, monocular cameras, and depth cameras to populate a local map for collision avoidance [1–7]. However, discrete map representations have significant memory requirements and are computationally expensive. This work proposes a novel local map representation that reduces the memory footprint of the map storage and computational complexity, while preserving completeness in its representation.

Conventional means of representing maps use discrete data structures to store raw sensor measurements of the environment, relying on search efficient data representations such as KD-Trees or OctoMap [1, 5]. While the simplicity of fixed size grids allow for uniformity in resolution [4], many algorithms couple these methods with adaptive grid sizing [5] to allow data reduction. Such methods discretize space and encode occupancy leading to resolution dependent model fidelity and loss of memory efficiency. Instead of processing local scans, [6–8] limits collision avoidance to the field-of-view of the sensor over either a single frame [6, 8] or a sequence of frames [7], thus limiting the range of available motion to the field-of-view of the sensor. Our approach does not limit the number of sensor frames, but uses scans based on locality so as to not limit the range of available motion to only the recently obtained scans.

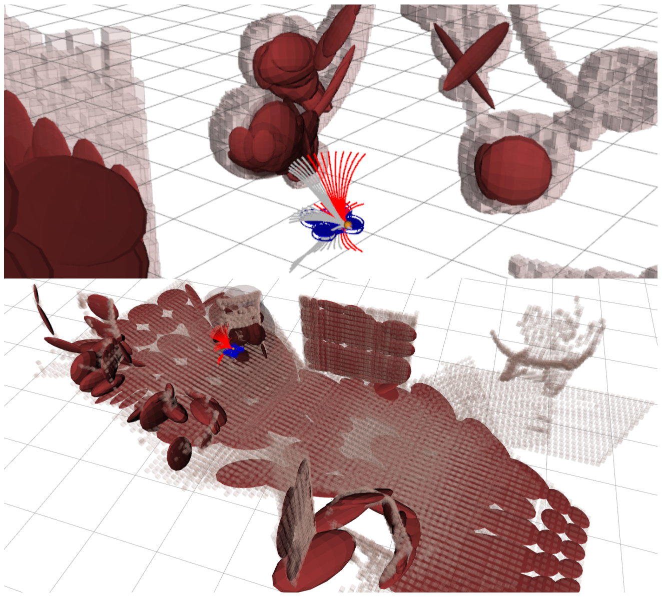


Fig. 1: A quadrotor navigates a cluttered environment via motion primitive based teleoperation. The dark red ellipsoids represent the Gaussian mixture model components that represent a local map (overlayed over a dense voxel grid representation for visualization). As the quadrotor navigates through the environment, trajectories that intersect with the Gaussian components are pruned (red lines) leaving only the safe, feasible trajectories (grey lines).

In this paper, we use a local probabilistic map representation that scales efficiently with the number of relevant sensor frames and represent the environment as a continuous distribution rather than a discrete set of points, and use this representation to perform reactive collision avoidance. The local map is represented as a Gaussian Mixture Model (GMM), which approximates the underlying distribution from which sensor measurements are sampled and provides an environment model that scales in fidelity with minimal information loss [9]. This map representation has been shown to outperform other representations in state estimation, large scale high fidelity mapping, and localization [9–13]. Instead of using a GPU to perform real-time GMM fitting [13], we use a single depth camera processed on an Intel Core i7-6700K CPU to generate the local map. We translate a Gaussian distribution's probabilistic spread to a geometric representation for collision checking. Specifically, given a set of local trajectories, we check trajectories for collision with the Gaussian components in the local map via intersection search.

The paper's contributions are as follows. We present a novel algorithm for generating local maps using GMMs and show that the proposed method scales efficiently with incremental sensor measurements. We take a geometric ap-

The Authors are with the Robotics Institute at Carnegie Mellon University, Pittsburgh PA, 15213, USA {adityand, xuning, nmichael}@cmu.edu

proach for computing collisions given the $4\boldsymbol{\Sigma}$-probabilistic bound of each Gaussian component and a trajectory. We illustrate the proposed algorithm with motion primitive based teleoperation [14] to show real-time collision avoidance. Experimental results of a quadrotor teleoperated through a cluttered environment with online GMM based local map generation at > 40Hz (See Section IV-B) results in an average collision checking time of 0.150 to 0.204 milliseconds per trajectory, which outperforms collision checks against discrete world representations.

II. METHODOLOGY

We describe local map representation using a Gaussian mixture model and provide algorithms for online, real-time map generation and collision avoidance. We assume that the state estimates of the vehicle do not drift significantly, such that errors due to state estimation can be considered negligible in local map generation and collision avoidance and leave considerations of inconsistent state estimates as future work.

A. Map Representation via GMMs

A GMM represents data points using a finite set of Gaussian distributions with unknown parameters. GMM-based mapping assumes that sensor measurements are sampled from an underlying continuous distribution, and provides a probabilistic measure of spatial occupancy at any location with varying fidelity [9, 13]. For example, scene reconstruction may require a high fidelity in the map resolution; for the purpose of collision checking, a lower fidelity map can be used as the resolution does not change the geometric coverage of the map (See Section IV-C).

Given a GMM in euclidean space $\boldsymbol{\Theta} = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \lambda_i\}_{i=1,\ldots,M}$, the probability of a point $\mathbf{x} \in \mathbb{R}^3$ being sampled from this distribution is given by

$$p\left(\mathbf{x};\boldsymbol{\Theta}\right) = \sum_{i}^{M} \lambda_i \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right), \tag{1}$$

where λ_i is the importance weight, $\boldsymbol{\mu}_i \in \mathbb{R}^3$, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times 3}$ is the mean and covariance of the i^{th} Gaussian component respectively, and M is the total number of Gaussian components in the mixture.

B. Local Map

The local mapping framework generates a spatially consistent local map and a reduced local map for collision checking. Since dynamically feasible trajectories often extend past the current field-of-view of the sensor, it is necessary to create a local map that encloses the vehicle using all recent sensor observations. To achieve this, we dynamically select anchor frames and integrate subsequent sensor measurements that provide novel information about the environment to these anchor frames. This allows us to create spatially consistent maps with minimal information redundancy. Given a history and current sensor measurements as well as state estimates, we classify the current sensor frame as a keyframe *KF*, subframe *SF* or a bufferframe *BF*. Novel information is extracted from the current sensor data, to which a GMM is fitted. According to the type of the frame, the corresponding GMM is appended to the local map. For collision checking, a reduced local map that contains the vehicle's immediate surroundings is extracted from the current local map.

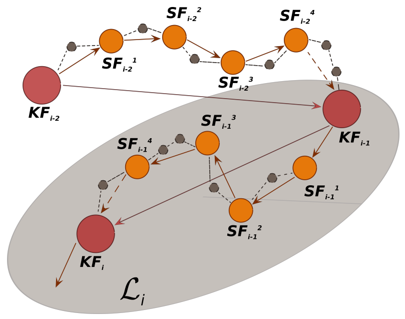


Fig. 2: A graphical representation of a local map $\mathcal{L}_i$. The vehicle poses are classified as *KF* (red), *SF* (orange) or a *BF* (brown) based on the euclidean distance between them. Each *SF* registers to a *KF* and a *BF* is used to be able to represent dynamic obstacles.

1) Frame Classification: Each incoming sensor frame is classified as either a

- *KF*: An anchor frame to which all the subsequent *SFs* and *BFs* in the local map $\mathcal{L}_i$ are registered to,
- *SF*: Sensor frames that provide sufficient novel information unobserved in the current map local map $\mathcal{L}_i$,
- *BF*: Sensor frames that are stored only for a single time step as to accommodate for dynamic obstacles; but do not get stored in $\mathcal{L}_i$

Given the current sensor location in the world frame $\mathbf{T}_{wc}$, the latest *KF* location $\mathbf{T}_{wk}$, and the latest *SF* location $\mathbf{T}_{ws}$, the current frame $\mathcal{F}_{wc}$ is classified according to a set of Euclidean distance thresholds, α_k, α_s and β_s, as

$$\mathcal{F}_{wc} = \begin{Bmatrix} \textit{KF}, & \text{if } \left\|\mathbf{t}\left(\nabla\{\mathbf{T}_{wk},\mathbf{T}_{wc}\}\right)\right\| \geq \alpha_k \\ \textit{SF}, & \text{if } \left\|\mathbf{t}\left(\nabla\{\mathbf{T}_{ws},\mathbf{T}_{wc}\}\right)\right\| \geq \alpha_s \\ & \text{or } \left\|\mathbf{R}\left(\nabla\{\mathbf{T}_{ws},\mathbf{T}_{wc}\}\right)\right\| \geq \beta_s \\ \textit{BF}, & \text{otherwise} \end{Bmatrix} \tag{2}$$

where $\mathbf{t}(\nabla\{\mathbf{T}_1,\mathbf{T}_2\})$ is the translation between transforms $\mathbf{T}_1$ and $\mathbf{T}_2$, and $\mathbf{R}(\nabla\{\mathbf{T}_1,\mathbf{T}_2\})$ is the change in the heading of the vehicle between transforms $\mathbf{T}_1$ and $\mathbf{T}_2$.

2) Determining Novel Sensor Information: Sequential sensor observations often contain redundant information. To reduce the information processing, we estimate the subset of information in each sensor observation that has not been previously observed, and only learn GMMs $\boldsymbol{\Theta}$ over this subset.

Given a depth image measurement $\mathbf{Z}_c$ at the current sensor pose $\mathbf{T}_{wc}$, and a set of depth image measurements registered to the latest *KF* $\{\mathbf{Z}_0,\mathbf{Z}_1,\ldots,\mathbf{Z}_n\}$, we generate a sensor measurement hypothesis for the previous sensor measurement $\mathbf{Z}_{n-1}$ in the current sensor frame using a pinhole projection operator $\boldsymbol{\pi}$:

$$\begin{aligned} \mathbf{P}_{n-1} &= \boldsymbol{\pi}\left(Z_{n-1}\right) \\ \nabla\mathbf{T} &= \mathbf{T}_{wc}^{-1}\mathbf{T}_{w(n-1)} \\ \mathbf{Z}_{n-1,c} &= \boldsymbol{\pi}\left(\nabla\mathbf{T}\mathbf{P}_{n-1}\right), \end{aligned} \tag{3}$$

where, $\mathbf{Z}_{n-1,c}$ denotes the depth image of the previous

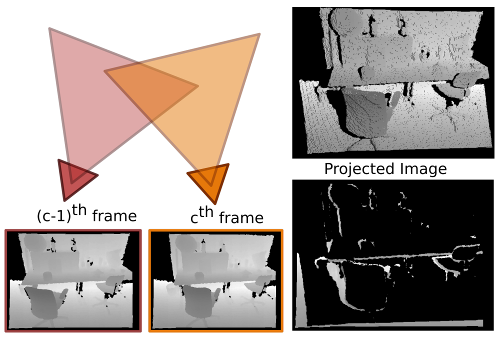


Fig. 3: The c^{th} frame (orange) represents the current sensor position and the image below it represents the corresponding depth image. The $(c\text{–}1)^{\text{st}}$ frame (red) represents the previous sensor position and the image below it represents its corresponding depth image. The projected image is the $(c\text{–}1)^{\text{st}}$ image projected in the c^{th} image frame, and the novel image $\hat{\mathbf{Z}}_c$ is the discrepancy between the projected image and the current image.

sensor measurement $\mathbf{Z}_{n-1}$ as viewed from current sensor pose $\mathbf{T}_{wc}$.

The novel sensor observations in the current depth image $\mathbf{Z}_c$ is then computed as

$$\mathbf{Z}_m(u,v) = \begin{cases} 0, & |\mathbf{Z}_c(u,v) - \mathbf{Z}_{\mathbf{n-1},\mathbf{c}}(u,v)| < \epsilon_d \\ 1, & |\mathbf{Z}_c(u,v) - \mathbf{Z}_{\mathbf{n-1},\mathbf{c}}(u,v)| \geq \epsilon_d \end{cases} \tag{4}$$

$$\hat{\mathbf{Z}}_c = \mathbf{Z}_c(u,v) \odot \mathbf{Z}_m \qquad \forall (u,v) \in (\mathbb{U},\mathbb{V})$$

Where $(\mathbb{U},\mathbb{V})$ denotes the image resolution.

Expectation-Maximization (EM) [15] is used to determine the GMM parameters $\mathbf{\Theta}$. EM assigns correspondence variables $\mathbf{C}_{ij}$ to each data point in the sensor observation, which encodes the responsibility of the j^{th} Gaussian component in representing the i^{th} data point. The EM algorithm iteratively estimates $\mathbf{C}$ in the E-step given the current set of GMM parameters $\mathbf{\Theta}$ by maximizing the log-likelihood of the sensor data given the current correspondence $\mathbf{C}$ and parameters $\mathbf{\Theta}$:

$$\log\left(\mathbf{Z}_c, \mathbf{C}; \mathbf{\Theta}\right) = \sum_{i}^{N}\sum_{j}^{J} \mathbf{C}_{ij}\left(\log\lambda_j + \log\mathcal{N}\left(\mathbf{Z}_c^i; \boldsymbol{\mu}_j, \mathbf{\Sigma}_j\right)\right). \tag{5}$$

Using the best estimate $\mathbf{C}$, the parameters $\mathbf{\Theta}$ are refined in the M-step.

3) Local map fusion: A new local map $\mathcal{L}_i$ is constructed when a new *KF* is spawned. $\mathcal{L}_i$ consists of the GMM components fused to the previous *KF* and the latest *KF*. The local map only contains GMM components learned from a *KF* or a *SF*. GMM components learned from a *BF* are only stored for the current time step in order to account for dynamic obstacles.

$$\mathcal{L}_i = \{\mathbf{\Theta}^i_{1,i-1}, \mathbf{\Theta}^i_{2,i-1}, .., \mathbf{\Theta}^i_{j,i-1}, ..\} \bigcup \{\mathbf{\Theta}^i_1, \mathbf{\Theta}^i_2, .., \mathbf{\Theta}^i_j, ..\} \tag{6}$$

where $\mathbf{\Theta}^i_{j,i-1}$ represents the j^{th} *SF* GMM in the $(i\text{–}1)^{\text{st}}$ *KF* transformed in the i^{th} *KF* and $\mathbf{\Theta}^i_j$ represents the j^{th} *SF* GMM learned in the i^{th} *KF*. A graphical representation of this process is shown in Fig. 2.

4) Reduced Local Map: As the local map $\mathcal{L}_i$ may be spatially expansive, we further reduce the size of the map for collision checking by creating a *reduced local map* $\mathcal{RL}_i$ that encapsulates the vehicle and trajectories. We store the means of the Gaussian components in the local map in a KD-Tree, and query an ϵ-ball around the current pose of the vehicle. The query radius ϵ is determined using the maximum trajectory distance. The Gaussian components $\mathbf{\Theta}_j$'s in the local map $\mathcal{L}_i$ that lie inside the $\boldsymbol{\epsilon}$-ball forms the reduced local map; i.e., $\mathcal{RL}_i = \{\mathbf{\Theta}_j : |\mathbf{\Theta}_j| < \epsilon, \mathbf{\Theta}_j \in \mathcal{L}_i\}$.

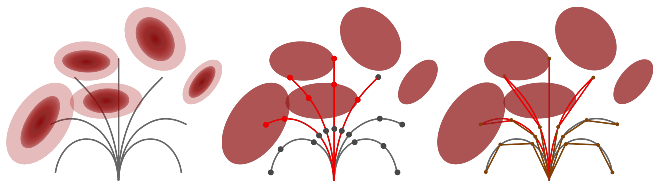


Fig. 4: A simplified 2D view of the proposed collision avoidance algorithm, illustrated using forward-arc motion primitives. A set of motion primitives interacting with a local GMM map (in burgundy) with configuration space inflation (light burgundy) is shown in (a). Each Gaussian component is reduced to its $4\mathbf{\Sigma}$ geometric representation for collision checking, via (b) sampling points along trajectories or (c) creating linear approximations to the trajectory based on curvature and solving for ellipsoid-line intersections. Rejected trajectories are shown in red.

Algorithm 1 Online GMM Local Map Generation

```
1: for $\mathbf{T}_{wb}$ the current sensor location in a consistent frame, $\mathbf{Z}_c$ the
   current depth image, and a query radius $\epsilon$ do
2:     Classify current frame according to Eq. (2)
3:     Compute the novel information $\hat{\mathbf{Z}}_c$ according to Eq. (4)
4:     Estimate the Gaussian mixture parameters parameters $\mathbf{\Theta}$
       that best represent $\hat{\mathbf{Z}}_c$ using Eq. (5)
5:     Integrate new Gaussian components into the latest local
       map $\mathcal{L}_i$ via Eq. (6)
6:     Store the means of the Gaussian components $\mathbf{\Theta}$ in a KD-Tree, and
       query a reduced local map given $\epsilon$.
7: end for
```

C. Trajectory Pruning

We present two ways of computing collisions given a time-parameterized trajectory and a GMM local map. Each component in the local map can be spatially represented by an ellipsoidal representation given a $\mathbf{\Sigma}$-bound of the distribution. We leverage this geometric property, and assume that each component can be represented as an ellipsoid as given by:

$$f(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^\top C(\mathbf{x} - \boldsymbol{\mu}) - 1, \tag{7}$$

where $\boldsymbol{\mu}$ is the center of the ellipsoid, $C = \text{diag}(c_1^{-2}, c_2^{-2}, c_3^{-2})$ and c_i are the major axes of the ellipsoid. We represent the configuration space of the vehicle by a sphere with radius r centered at the robot's geometric center. Then, we transform the local map to incorporate the configuration space by inflating the major axes:

$$f(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^\top D(\mathbf{x} - \boldsymbol{\mu}) - 1, \tag{8}$$

where $D = \text{diag}\left((c_1 + r)^{-2}, (c_2 + r)^{-2}, (c_3 + r)^{-2}\right)$. For sufficiency, we take the ellipsoid defined by the $4\mathbf{\Sigma}$ probability bound of each Gaussian component, which provides approximately 99.95% Chi-squared probabilistic coverage of the underlying point density.

Suppose a trajectory is given by $\mathbf{x}(t) = \gamma(t)$, where $\gamma(t)$ is a time-parameterized function with t defined over some interval $t \in [t_0, t_f]$, and $\mathbf{x}(t) = [x(t), y(t), z(t)]^\top$. Then, the ellipsoid-trajectory equation becomes:

$$f(t) = f(\mathbf{x}(t)) = (\mathbf{x}(t) - \boldsymbol{\mu})^\top R^\top D R(\mathbf{x}(t) - \boldsymbol{\mu}) - 1 \quad (9)$$

$$f(t) = (\mathbf{x}(t) - \boldsymbol{\mu})^\top A(\mathbf{x}(t) - \boldsymbol{\mu}) - 1, \quad (10)$$

where $R \in \mathbb{R}^{3\times 3}$ is the rotation matrix to transform the local trajectory into the frame of the mixture component. An intersection or collision occurs when $f(t) \leq 0$.

For arbitrary trajectories $\mathbf{x}(t)$, no analytic solutions exist to Eq. (10) unless $\mathbf{x}(t)$ is affine. In the following subsections, we present two algorithms for collision checking for arbitrarily complex trajectories and provide a brief discussion on computational complexities.

1) Sampling based collision checking: Instead of computing an analytic solution to Eq. (10), a simple check would be to sample points along each trajectory. For M Gaussian mixture components, N local trajectories, and S samples per trajectory, the computational complexity would be $\mathcal{O}(MNS)$. This approach is delineated in Algorithm 2.

2) Piecewise affine trajectory approximation: If the trajectory is sufficiently smooth, one can generate piecewise affine (PWA) approximations to the trajectory using heuristics. For each trajectory, suppose s segments of affine approximations sufficiently approximate the trajectory. Then, over each segment, the affine approximation $\mathbf{x}_s(t) = \mathbf{a}_s t + \mathbf{b}_s$ with $\mathbf{a}_s, \mathbf{b}_s \in \mathbb{R}^3$ and $t \in [t_{s-1}, t_s]$ can be analytically solved in the frame of each Gaussian component.

The ellipsoid-line equation using Eq. (8), in the frame of the gaussian component, can be written as:

$$f(\mathbf{x}_s) = (\mathbf{x}_s)^\top D(\mathbf{x}_s) - 1 \quad (11)$$

$$f(t) = (\mathbf{a}_s t + \mathbf{b}_s)^\top D(\mathbf{a}_s t + \mathbf{b}_s) - 1, \quad (12)$$

Without loss of generality, a trajectory can always be transformed into the frame of the mixture component such that D is diagonal. Collisions are found via solutions to

$$0 = \left(\frac{a_1^2}{c_1^2} + \frac{a_2^2}{c_2^2} + \frac{a_3^2}{c_3^2}\right)t^2 + 2\left(\frac{a_1 b_1}{c_1^2} + \frac{a_2 b_2}{c_2^2} + \frac{a_3 b_3}{c_3^2}\right)t + \left(\frac{b_1^2}{c_1^2} + \frac{b_2^2}{c_2^2} + \frac{b_3^2}{c_3^2} - 1\right) \quad (13)$$

With the assumption that the trajectory does not begin inside a Gaussian component.

For M components, N trajectories, the number of segments is dependent on the curvature of the trajectory. The computation complexity would be $\mathcal{O}(MNS_n)$, where $S_n \leq S$, $n = 1, \ldots, N$ such that the worst case complexity collapses to that of the sample based approach (with S samples per trajectory). This approach is delineated in Algorithm 3. We provide a heuristic for determining number of segments for motion primitives in Sect. II-D.

Algorithm 2 Collision Checking with GMM Local Map via Sampling

```
 1: Given $M$ Gaussian mixture components, $N$ local trajectories, $S$
    samples per trajectory
 2: Discretize time interval $[t_0, t_f]$ into $\mathbf{t} = \{t_i\}$, $i = 1, \ldots, S$ s.t.
    $t_i \in [t_0, t_f]$
 3: for $n = 1 : N$ trajectories do
 4:     for $m = 1 : M$ Gaussian components do
 5:         Obtain the eigenvector matrix $R_m$, centers $\boldsymbol{\mu}_m$
 6:         compute $A_m = R_m{}^\top D_m R_m$ where
            $D_m = \mathrm{diag}\left((c_{m1}+r)^{-2}, (c_{m2}+r)^{-2}, (c_{m3}+r)^{-2}\right)$
 7:         for each $t \in \mathbf{t}$ do
 8:             Query point at time $t$: $\mathbf{x}_s = \mathbf{x}(t)$
 9:             if $f(\mathbf{x}_s) = (\mathbf{x}_s - \boldsymbol{\mu}_m)^\top A_m(\mathbf{x}_s - \boldsymbol{\mu}_m) \leq 1$ then
10:                 Reject trajectory and increment
11:             end if
12:         end for
13:     end for
14: end for
```

Algorithm 3 Collision Checking with GMM Local Map via PWA Trajectory Approximation

```
 1: Given $M$ Gaussian mixture components, $N$ local trajectories
 2: for $n = 1 : N$ trajectories do
 3:     Heuristically discretize trajectory into $S_n$ segments
 4:     Compute $S_n$ affine approximations
 5:     for $m = 1 : M$ Gaussian components do
 6:         Obtain the eigenvector matrix $R_m$, centers $\boldsymbol{\mu}_m$, and
            transform $\mathbf{x}$ into the frame of the Gaussian mixture
 7:         for each $s = 1 : S_n$ do
 8:             Solve Eq. (13) and denote solutions as $t^*_{1,2}$
 9:             if $t^*_{1,2} \in [t_{s-1}, t_s]$ then
10:                 Reject trajectory and increment
11:             end if
12:         end for
13:     end for
14: end for
```

D. Local Trajectories: Motion Primitive Library

An example of a family of local trajectories is a motion primitive library. Motion primitives are dynamically feasible local trajectories parameterized by the input space of the dynamics, which have been shown to be amenable to online autonomous exploration [16] and teleoperation [14, 17]. This paper follows [14] and uses *forward-arc motion primitives*. These local trajectories are formed by propagating the dynamics of a unicycle model with a constant linear velocity v_x, angular velocity ω, and vertical velocity v_z for a specified amount of time, T [18]. A motion primitive γ is generated by parameterizing a kinematic or dynamic model of the robot using a single action $\mathbf{a}$, i.e. $\gamma(\mathbf{a})$, where $\mathbf{a} = \{v_x, \omega, v_z\}$ representing a combination of the discretized input. The forward-arc motion primitive is given by the solutions to the unicycle model:

$$\gamma(t) = \mathbf{x}(0) + \begin{bmatrix} \frac{v_x}{\omega}(\sin(\omega t + \theta) - \sin(\theta)) \\ \frac{v_x}{\omega}(\cos(\theta) - \cos(\omega t + \theta)) \\ v_z t \\ \omega t \end{bmatrix}, \quad (14)$$

where $t \in [0, T]$. The pose of the vehicle at time t is given by $\mathbf{x}(t) = [x(t), y(t), z(t), \theta(t)]^\top$, and v_{xt}, v_{zt}, and ω_t are the linear and angular velocities of the vehicle at time t in the body frame, respectively. A motion primitive library is then given by discretizing the input space, resulting in a set of local trajectories $\mathit{\Gamma} = \{\gamma_i(\mathbf{a}_i)\}_{i=1,\ldots,N}$ where the input space is defined by $\{\mathbf{a}_i\}_{i=1,\ldots,N}$.

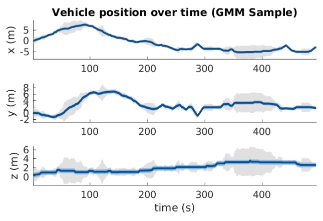
For forward-arc motion primitives, the curvature of the trajectory is correlated with the angular velocity that is used to generate the motion primitive. As such, our heuristic for generating the number of segments is defined as follows:

$$S_n = \lceil 1 + k\,|\omega_n| \rceil \quad (15)$$

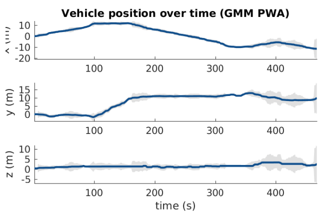
where $k = 3$ is empirically chosen and $\lceil\cdot\rceil$ is the ceiling operator.

III. IMPLEMENTATION

Fitting GMMs over depth sensor scans in real-time is costly and requires GPU-accelerated desktops [13]. We achieve online GMM fitting performance using an Intel i7-6700K CPU in real-time over each depth scan via down-sampling and parallelization. First, we downsample each original $(640, 480)$ resolution depth image using Gaussian pyramids [19] to a $(160, 120)$ resolution image without loss of map fidelity or computational efficiency (See Section IV-C). Second, we exploit the knowledge about the diminishing support of Gaussian distributions by further partitioning the depth image into smaller 4×5 patches with $(32, 30)$ pixel resolution, and specify 3 GMM components per patch. This allows parallel GMM fitting over each independent patch. Partitioning the continuous sensor data into smaller patches may break the geometric continuity of the scene along the patch edges. However, ensuring that $> 99.95\%$ coverage of the scan data points are covered by the GMM representation is sufficient for collision checking.



(a)



(b)

Fig. 5: A visualization of free space around each vehicle vs. configuration space for example trials with (a) GMM local map with sampling, and (b) GMM local map with PWA approximations based collision avoidance. The blue line denotes the pose of the vehicle and the light blue shading denotes the configuration space of 0.5m.

IV. EXPERIMENTS AND RESULTS

We evaluate our proposed algorithm in simulation in a cluttered environment as shown in Fig. 6 and compare it to KD-Tree maps in our local mapping framework approach, in terms of computational complexity of map generation and collision checking, and also show that our method provides safety guarantee of at least the configuration bound."

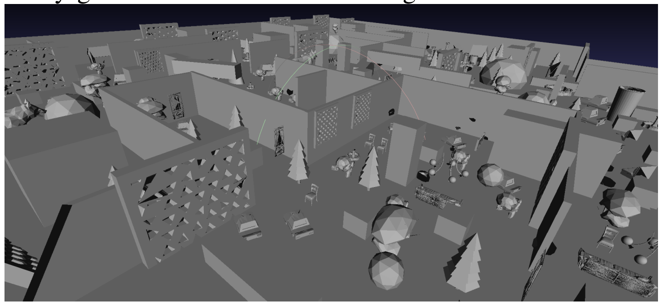


Fig. 6: The cluttered environment.[1]

In the simulation scenario, an operator teleoperates the vehicle using forward-arc motion primitives [14]. We generate a library of 155 motion primitives, using 31 linearly spaced angular velocities $\omega \in \{-3, 3\}$rad/s, 5 linearly spaced vertical velocities $v_z \in \{-1, 1\}$m/s, and limit the vehicle to a maximum linear velocity of 2m/s. We assume a configuration radius of 0.5m. The heuristics for frame categorization are: α_k = 1.0m, α_s = 0.2m, and β_s = 0.2 radians.

[1]Available at: https://github.com/vibhavg/simulation_environments

A. Safety

Safety is evaluated using the minimum distance of the vehicle pose to its surroundings. We use a dense pointcloud representation as a baseline and query the radius of free space around the vehicle at each iteration. In Fig. 5, two 6-8 minute example trials are shown. Throughout each trial, the vehicle's configuration space, denoted in blue, is contained within the free space around the vehicle, denoted in grey, indicating that the vehicle is safe at all times.

B. Efficiency

We analyze timing and memory efficiency of GMM local maps as compared to KD-Tree local maps. Each KD-Tree and GMM local map is learned over a downsampled data set of 19200 data points with an average of 14 frames per local map, resulting in a maximum of 249600 points per local map. The KD-Tree local map creates a new KD-Tree at each iteration using the raw data using a discretization of 0.1m. The GMM local map generates 60 components per frame, which results in approximately 300 components per local map. Each reduced local map contains approximately 30–40 components.

We observe that each GMM local map can be generated in approximately 22.89ms on a single CPU (Fig. 7, Top). To store the same amount of data, KD-Tree requires approximately 41.92ms. GMM based local map is agnostic to the number of frames in the local map; as new components only need to be appended to the current local map. In contrast, a new KD-Tree needs to be generated with each new sensor measurement and scales poorly over increasing amounts of data.

Collision checking timing analysis averaged over 10000 trajectories is shown in Fig. 7 (Bottom). We observe 0.737ms for KD-Tree based collision check per trajectory, 0.204ms for GMM local map with sampling-based collision check per trajectory, and 0.150ms for GMM local map with PWA trajectory approximation. Sufficiently representing the local map using a low number of components (See Section IV-C) contributes to significant speed-ups over KD-Tree queries.

C. Map Fidelity and Coverage

Given a set of 3D points $\mathbf{P}$ and a GMM $\mathbf{\Theta}$, the log-likelihood of the points $\mathbf{P}$ having been sampled from the distribution $\mathbf{\Theta}$ is given by,

$$\mathbf{S} = \sum_{i}^{N} \ln \sum_{j}^{M} \lambda_j \mathcal{N}(\mathbf{P}_i; \boldsymbol{\mu}_j, \mathbf{\Sigma}_j) \quad (16)$$

We define $\mathbf{S}$ as the *scan score* of the points $\mathbf{P}$. Higher scan score suggests that the probability of points $\mathbf{P}$ having been sampled from GMM $\mathbf{\Theta}$ is high.

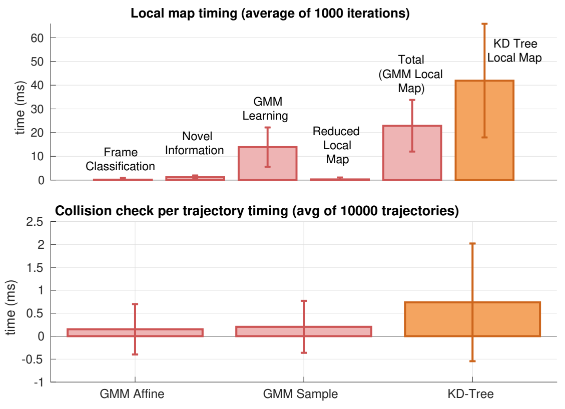


Fig. 7: Top: Timing analysis for GMM local map generation, averaged over 1000 local maps containing 249600 points. GMM local map takes 22.89ms to learn, whereas KD-Tree local map would take 41.92ms to generate. Bottom: Timing analysis for per trajectory collision checking with samples and PWA trajectory approximations, as compared to using KD-Tree representations. GMM methods take 0.25ms for collision checking per trajectory, whereas KD-Tree takes 0.75ms per trajectory. Error bars report standard deviation of the mean.

We evaluate the loss of information via computing the score $\mathbf{S}$ of a GMM learned over the downsampled data versus a GMM learned over the full resolution data. For each point in the full resolution depth image, we compute the scores from both the GMM learned from a downsampled data and the full resolution data. As shown in Fig. 9, GMM learned over the downsampled data achieves similar scores as the GMM learned over the full data, implying minimal loss of information. The comparable performance may be the result of smoothing, as reducing the number of points reduces high frequency noise in the data, possibly leading the EM algorithm to more robust GMM representations. These experiments suggest that uniformly downsampling the depth image does not cause a significant loss in the map fidelity thus enabling a substantial increase in the computational efficiency.

We validate the geometric local map coverage of the sensor data points by computing the percentage of points that are within $4\mathbf{\Sigma}$ distance from the mean of all components. We empirically evaluate geometric coverage for using 1 to 6 components per patch for several randomized runs. As shown in Fig. 8, even 2 Gaussian components per patch ensures $> 99.95\%$ coverage.

V. CONCLUSION

We propose a novel real-time framework for collision avoidance using GMM based local maps generated. We presented two analytic methods for collision checking given arbitrary trajectories, leveraging the geometric properties of Gaussian distribution spread. Such local maps provides an elegant solution to represent the environment in a continuous representation, while providing high fidelity information and reducing memory and computational complexity.

As GMMs provide a continuous representation of the environment, probabilistic interactions with the world can be readily accommodated. Future work includes extending the proposed collision avoidance methodology to accommodate uncertain state estimates with respect to GMM local map representations.

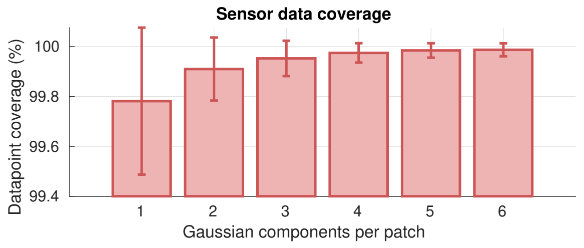


Fig. 8: Geometric coverage of sensor data represented by N number of Gaussian components per image patch. For all $N = 1, \ldots, 6$, 99.95% of data lie within $4\mathbf{\Sigma}$-probabilistic bound.

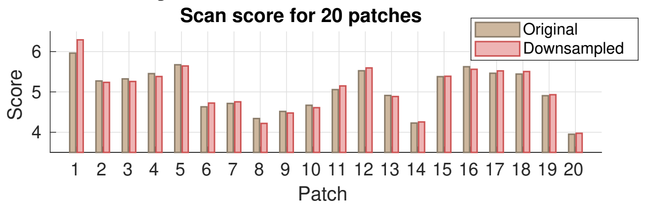


Fig. 9: Comparison of scan score for a GMM learned over the original full resolution data, versus a GMM learned over a downsampled data. Scores for the downsampled GMM perform comparable to the full resolution GMM. Downsampling reduces the number of points and leads the EM algorithm to a more robust GMM representation.

REFERENCES

[1] D. Maier, A. Hornung, and M. Bennewitz, “Real-time navigation in 3D environments based on depth camera data,” in *2012 12th IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids)*, 2012, pp. 692–697.
[2] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, “Stereo vision-based obstacle avoidance for micro air vehicles using disparity space,” in *2014 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 3242–3249.
[3] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, “Robust monocular flight in cluttered outdoor environments,” *arXiv preprint arXiv:1604.04779*, 2016.
[4] S. Liu, M. Watterson, S. Tang, and V. Kumar, “High speed navigation for quadrotors with limited onboard sensing,” in *2016 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 1484–1491.
[5] J. Chen, T. Liu, and S. Shen, “Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments,” in *2016 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 1476–1483.
[6] P. Florence, J. Carter, and R. Tedrake, “Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps,” in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.
[7] P. R. Florence, J. Carter, J. Ware, and R. Tedrake, “Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data,” in *2018 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018.
[8] B. T. Lopez and J. P. How, “Aggressive 3-D collision avoidance for high-speed navigation,” in *2017 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.
[9] S. Srivastava and N. Michael, “Approximate Continuous Belief Distributions for Precise Auton. Inspection,” in *Proc. of the IEEE Intl. Sym. on Safety, Security and Rescue Robotics*, 2016.
[10] A. Dhawale, K. Shaurya Shankar, and N. Michael, “Fast Monte-Carlo Localization on Aerial Vehicles Using Approximate Continuous Belief Representations,” in *The IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, June 2018.
[11] W. Tabib, C. O’Meadhra, and N. Michael, “On-Manifold GMM Registration,” *IEEE Robotics and Automation Letters*, pp. 1–1, 2018.
[12] 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 Intl. Conf. on 3D Vision (3DV)*, 2015, pp. 241–249.
[13] ——, “Accelerated generative models for 3D point cloud data,” in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, 2016.
[14] X. Yang, K. Sreenath, and N. Michael, “A Framework for Efficient Teleoperation via Online Adaptation,” in *Proc. of the 2017 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.
[15] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[16] W. Tabib, M. Corah, N. Michael, and R. Whittaker, “Computationally efficient information-theoretic exploration of pits and caves,” in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ Intl. Conf. on*, 2016, pp. 3722–3727.
[17] X. Yang, A. Agrawal, K. Sreenath, and N. Michael, “System-Agnostic Adaptive Teleoperation for High-Dimensional Systems,” *Special Issue on Learning for Human-Robot Collaboration, Auton. Robots*, 2018.
[18] M. Pivtoraiko, I. A. Nesnas, and A. Kelly, “Autonomous robot navigation using advanced motion primitives,” in *Proc. of the IEEE Aerospace Conf.*, Big Sky, USA, 2009, pp. 1–7.
[19] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, “Pyramid methods in image processing,” *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.