Technical Report

Title

Fast Exploration Using Multirotors: Analysis, Planning, and Experimentation

Authors

Kshitij Goel
Micah Corah
Nathan Michael

CMU-RI-TR-19-03

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

February 2019

Abstract

High speed flight with multirotor aerial vehicles is limited by constraints on size, sensing range, on-board computation, accelerations, and velocities. For robotic exploration and operation in unknown environments, guaranteeing collision-free operation and selecting informative sensing actions further increase complexity. To this end, this work presents three contributions. First, we analyze the rate of reduction of the entropy of the map for idealized scenarios considering constraints on dynamics and sensing for a multirotor vehicle. Second, we propose an action representation that accounts for platform dynamics and provides actions that are useful for rapid exploration based on the prior analysis. Third, we present a receding-horizon sampling-based planner that uses this action representation, maximizes information gain, and ensures safe operation at high speeds. Finally, we present extensive simulation experiments in a complex 3D environment that demonstrate the significance of action design, horizon length, and replanning rate on exploration performance.

Contents

List of Figures

List of Tables

1 Introduction

Fast and safe motion planning policies for unknown environments can be useful when robots operate in challenging scenarios such as urban search-and-rescue and disaster response. In such scenarios, it is essential for a robot or a team of robots to find survivors or specific objects in an unstructured and unknown three-dimensional environment rapidly while maintaining collision-free operation. Aerial robots equipped with time-of-flight sensors have been used previously in a three-dimensional exploration context and have been shown to be useful in such hazardous scenarios [4, 5]. However, such platforms are often size, weight, and power (SWaP) constrained which places bounds on available actions during flight as well as limiting the choice of sensing modalities. At the same time, SWaP constraints also place bounds on the dynamics such as the maximum acceleration that the robot can achieve in flight. Rapid exploration using such constrained multirotors thus requires motion planning frameworks that enable safe navigation in unknown environments while accounting for model constraints and decreasing the uncertainty of the map [6].

Previous works have begun to consider the effects of platform dynamics on high speed exploration and navigation in unknown environments. Cieslewski et al. [6] take into account aerodynamic effects and sensing limits to place bounds on permissible velocities towards frontiers. While these upper bounds ensure that the multirotor can safely come to a halt within the maximum sensor range, the velocity bounds also depend on how fast the frontier locations are updated, i.e., the planning rate. Liu et al. [7] address this by accounting for time delay in processing sensor data in addition to model constraints to generate safe stopping trajectories for each planning iteration. However, an analysis on how planning rates and model constraints affect velocity bounds and exploration performance is lacking in the literature. In this work, we account for planning rates using a similar approach as Liu et al. [7] and provide analysis using a simplified quadrotor model to identify limits on exploration performance given model constraints. This analysis is later used to design an action representation as part of a proposed motion planning framework for rapid exploration.

Intuitively, information reward is high when the robot moves towards a frontier [6, 8]. Cieslewski et al. [6] pose exploration as a frontier selection problem and suggest selection of frontiers that lie in the direction of the flight path and adapt the velocity based on the sensor range and aerodynamic considerations. Although this approach provides insight into maintaining safety during rapid exploration, the motivation for the approach is incomplete as a more broad variety of actions such as yawing motions and motions parallel to a frontier can provide improved performance for rapid exploration. In the first part of this work, we

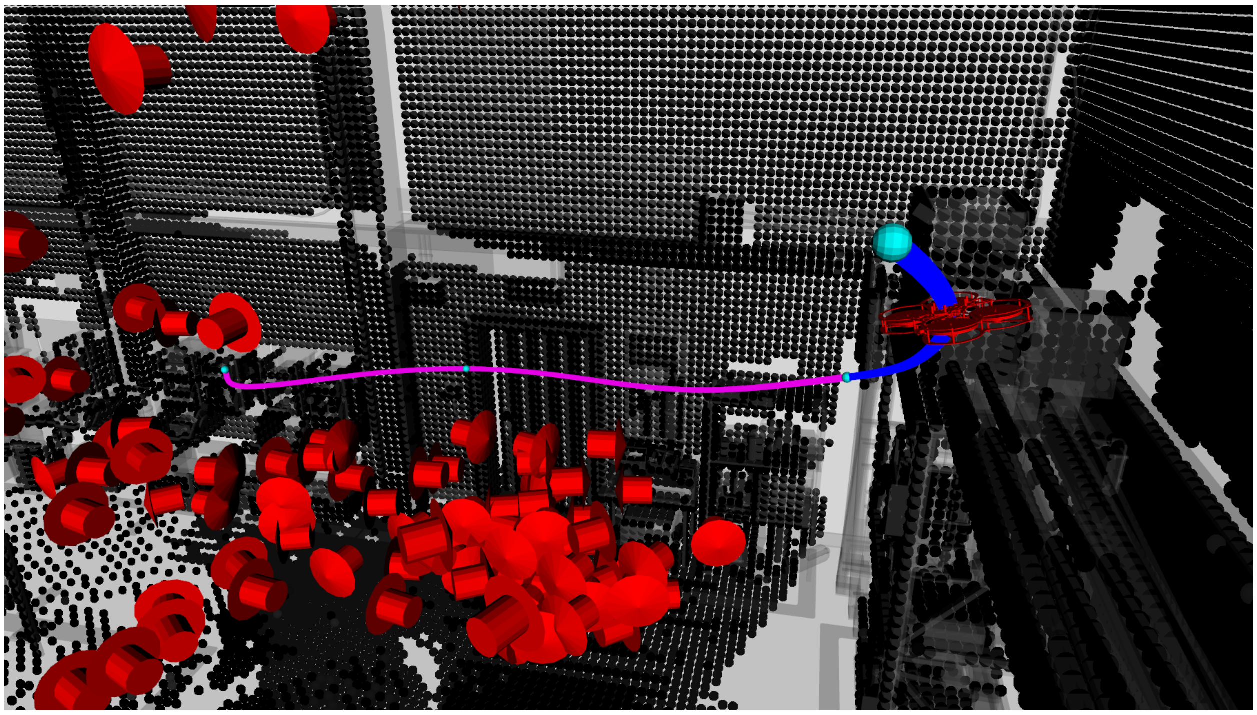


Figure 1: A multirotor explores a complex unstructured three-dimensional environment (occupied voxels in black) using the proposed planning approach. In this work, we first present analysis for bounds on exploration performance in ideal scenarios taking model constraints into account. Secondly, we propose a finite-horizon planning approach that leverages this analysis and ensures safety at high speeds. Last, we conduct extensive simulation experiments and draw conclusions on various factors affecting exploration performance.

address bounds on both linear and yaw rates and analyze rates of exploration given dynamics and sensing constraints. Specifically, given a constrained aerial platform, upper bounds on these velocities and rate of entropy reduction are derived for an idealized exploration scenario. Using these upper bounds, we propose a information-theoretic motion planning framework for high speed exploration that accounts for velocity and acceleration bounds in both action representation and selection. Lastly, we conduct simulation experiments in a large unstructured three-dimensional warehouse environment and analyze affects on exploration performance to show the significance of action design, finite-horizon length, and replanning rates.

This report proceeds as follows. Sect. 2 presents an idealized analysis on exploration performance bounds given platform constraints. Action representation for the proposed motion planning framework is developed in Sect. 3 followed by action selection policy description in Sect. 4. Sect. 5 details experimental results, with concluding remarks presented in Sect. 6.

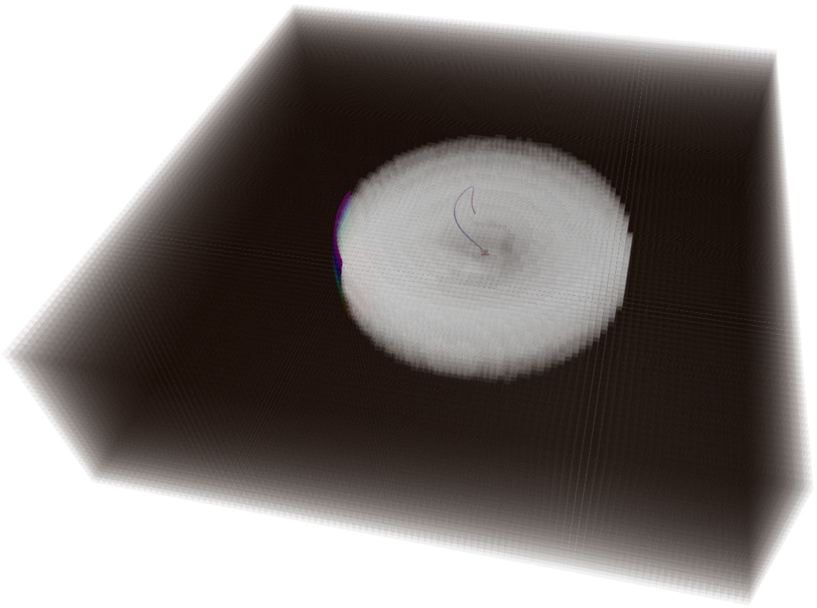


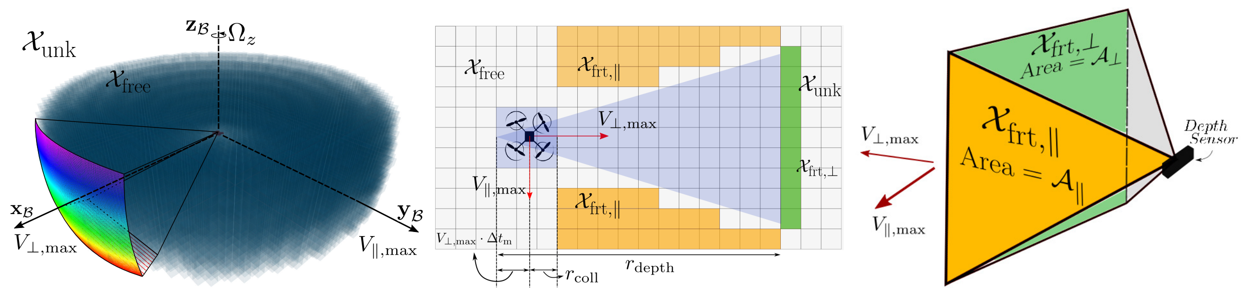
Figure 2: An ideal multirotor exploration scenario with no obstacles in the environment. The objective is to design a planner that takes actions such that over time, all of the unknown space (black voxels) is explored (white voxels) using a time-of-flight sensor while ensuring that the robot motion and planned trajectories are always within the free space (white voxels).

2 Idealized Exploration Analysis

The goal of this section is to investigate bounds on exploration performance for a constrained multirotor platform. We impose constraints based on dynamics, depth sensing, and compute bounds on the rate of entropy reduction for nominal conditions. We start with defining the exploration problem in Sect. 2.1, model simplifications and safety constraints in Sect. 2.2, followed by description of ideal exploration scenarios in Sect. 2.3. Bounds of velocity and exploration rate are calculated in Sect. 2.4 and Sect. 2.5 respectively. Later, in Sect. 3, we leverage these insights on velocities and exploration rates to design actions for the proposed motion planning strategy for exploration.

2.1 Problem Description

Figure 2 depicts a typical multirotor exploration scenario. Environment is modeled by an occupancy grid map with independent Bernoulli occupancy probabilities for voxels with a side-length c. Depending on user-specified thresholds for occupancy, the map is partitioned into three subsets: free space ($\mathcal{X}_{\text{free}}$, white voxels in Fig. 2), unknown space ($\mathcal{X}_{\text{unk}}$, black voxels


$\mathcal{X}_{\text{unk}}$
$\mathbf{z}_{\mathcal{B}}$
Ω_z
$\mathcal{X}_{\text{free}}$
$\mathbf{x}_{\mathcal{B}}$
$V_{\perp,\max}$
$\mathbf{y}_{\mathcal{B}}$
$V_{\|,\max}$
$\mathcal{X}_{\text{free}}$
$\mathcal{X}_{\text{frt},\|}$
$\mathcal{X}_{\text{unk}}$
$V_{\perp,\max}$
$V_{\|,\max}$
$\mathcal{X}_{\text{frt},\perp}$
$\mathcal{X}_{\text{frt},\|}$
$V_{\perp,\max}\cdot\Delta t_{\text{m}}$
r_{coll}
r_{depth}
$\mathcal{X}_{\text{frt},\perp}$
Area $= \mathcal{A}_{\perp}$
$V_{\perp,\max}$
$\mathcal{X}_{\text{frt},\|}$
Area $= \mathcal{A}_{\|}$
Depth Sensor
$V_{\|,\max}$

(a) Ideal exploration scenarios **(b)** Sensor top view **(c)** Sensor perspective view

Figure 3: Ideal exploration scenarios considered for exploration performance analysis in Sect. 2. Upper bounds on rate of novel voxels explored are computed for a double integrator system moving towards sensor scan ($V_{\perp,\max}$) and parallel to it ($V_{\|,\max}$) while performing rapid yaw motion. Environment is assumed completely free ($\mathcal{X}_{\text{unk}} = \mathcal{X}_{\text{free}}$) while sensor scan is delayed by Δt_{m}.

in Fig. 2), and occupied space ($\mathcal{X}_{\text{occ}}$). The voxels at the boundary of $\mathcal{X}_{\text{free}}$ and adjacent to voxels in $\mathcal{X}_{\text{unk}}$ are called frontier voxels, denoted by the set $\mathcal{X}_{\text{frt}}$ [8]. The objective of a motion planner in such exploration scenarios is to maximize the rate at which the amount of free space, $\mathcal{X}_{\text{free}}$, is explored. This rate is quantified as the rate of reduction of entropy of the occupancy map [9]. The entropy of the map decreases as the occupancy values of previously unknown voxels are updated based on sensor observations and become more certain, and there is at most one bit of entropy per voxel. Hence, we reason about the rate of reduction in entropy using bounds on the number of novel voxels covered per unit time.

Furthermore, the planned trajectories and the robot should always lie within the safe space $\mathcal{X}_{\text{free}}$ to ensure safety. This constraint, along with constraints specific to the multirotor platform, limit the minimum time in which an area can be explored. In this ideal analysis, we investigate what are these limits in ideal exploration scenarios and how can these limits inform the design of our proposed motion planner.

2.2 System Model and Safety Constraints

This work applies a simplified double-integrator quadrotor model with acceleration and velocity constraints for analysis of limits on exploration performance. Such a model can be thought of as a relaxation of dynamics models that are commonly used for position and attitude control of multirotor vehicles [10, 11]. Denote the position of the vehicle as $\mathbf{r} = [x, y, z]^{\mathrm{T}}$ in an inertial world frame $\mathcal{W} = \{\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}}\}$ and the yaw as ψ, assuming small roll and pitch angles. Given the yaw angle, we denote the state as $\boldsymbol{\xi} = [\mathbf{r}^{\mathrm{T}}, \psi, \dot{\mathbf{r}}^{\mathrm{T}}, \dot{\psi}]^{\mathrm{T}}$ and the body frame as $\mathcal{B} = \{\mathbf{x}_\mathcal{B}, \mathbf{y}_\mathcal{B}, \mathbf{z}_\mathcal{B}\}$. The derivatives of position and yaw satisfy

$$||\dot{\mathbf{r}}||_2 \leq V_\text{max} \qquad ||\ddot{\mathbf{r}}||_2 \leq A_\text{max} \qquad |\dot{\psi}| \leq \Omega_\text{max} \tag{1}$$

where $||\cdot||_2$ is the 2-norm.

For the purpose of this analysis, we are interested in computing ideal upper bounds on rate of novel voxels observed by the multirotor. In an ideal exploration scenario (Fig. 3a), unknown voxels are all unoccupied, and the robot is able to explore rapidly subject only to constraints on dynamics and safety.

However, the requirement for collision-free operation constrains the set of actions that a multirotor can safely execute while navigating in an unknown environments. In this case, the environment is incrementally revealed to the multirotor via measurements obtained from a depth sensor facing in the $\mathbf{x}_\mathcal{B}$ direction, which has a maximum range of r_depth (Fig. 3a, 3b). A planning policy can ensure collision-free operation by guaranteeing that the robot is able to stop within $\mathcal{X}_\text{free}$—given an appropriate collision radius r_coll (Fig. 3b)—such as in the work of Janson et al. [12]. In the worst case, any voxel in $\mathcal{X}_\text{unk}$ may be revealed to be occupied and so possibly force the robot to stop within $\mathcal{X}_\text{free}$ from any state $\boldsymbol{\xi}_t$ at time t. We define exploration scenarios and such velocity bounds later in Sect. 2.3.

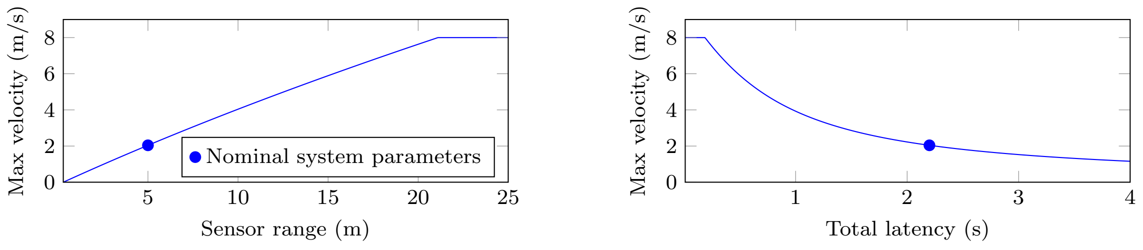
Plans are generated every Δt_p seconds, and, additionally, we assume bounded latency for acquiring depth information and integrating it into the occupancy map, denoted by Δt_m (Fig. 3b). As such, the action being executed at any given time is based on sensor data that is no older than $\Delta t_\text{l} = 2\Delta t_\text{p} + \Delta t_\text{m}$. The first planning duration accounts for planning computation, based on data that is at most Δt_m old. The second accounts for execution of the planned action while the robot re-plans again.

2.3 Exploration Scenarios

We now define the two basic steady-state exploration scenarios that we consider in this analysis.

2.3.1 Perpendicular to frontier

In this scenario, the robot moves toward the frontier, in the direction of the unknown space, possibly while performing a rapid yaw motion. As discussed in Sect. 2.2, the robot must be able to stop before entering the unknown space which places an upper bound, denoted by $V_{\perp,\text{max}}$ in Fig. 3a and Fig. 3b. For this analysis, rather than allowing for the robot to



(a) Velocity ($V_{\perp,\max}$) vs. sensor range (r_{depth}) **(b)** Velocity ($V_{\perp,\max}$) vs. total latency (Δt_{l})

Figure 4: Maximum achievable velocity moving toward a frontier ($V_{\perp,\max}$) according to (2) based on cited parameters and varying (a) sensor range and (b) total latency (which consists of latency for the mapping system and time for planning). The circle marks the maximum velocity at which the robot can move toward unknown space for the parameters used in this paper (see Table 1) which is less than half the overall velocity bound. Approaching this velocity limit requires either sensor range exceeding 10 m, both higher planning rates and low mapping latency, or some combination of the two.

repeatedly accelerate and decelerate in anticipation of having to stop, we instead assume a single constant velocity toward the frontier.

2.3.2 Parallel to frontier

For the second scenario, the robot moves parallel to the frontier, again while possibly performing rapid yaw motion. In this case, the robot is is not moving toward unknown space and may potentially continue to do so indefinitely at the maximum allowable velocity. Practically, this scenario can also be thought of as the limit for a spiral motion after the curvature becomes very small. Velocity bound for this case is kept sufficiently higher than $V_{\perp,\max}$, denoted by $V_{\parallel,\max}$ in Fig. 3. Later in Sect. 4, we describe an action selection policy that ensures only actions in free space are selected for such high-speed parallel motions.

2.4 Bounds on Velocity

Given the system model and constraints for exploration scenarios, we now proceed with calculation of the velocity bounds for motion perpendicular and parallel to the frontier, $V_{\perp,\max}$ and $V_{\parallel,\max}$. We compute the maximum velocity toward the frontier based on motion at a constant velocity followed by stopping at maximum deceleration to satisfy the safety constraint. The expression for $V_{\perp,\max}$ is a function of acceleration ($A_{\max}$), maximum sensing range (r_{depth}), the collision radius (r_{coll}), and the latency in planning Δt_{l} (see Fig. 3b) and

Table 1: Ideal upper bounds on velocity and rate of entropy reduction for two ideal scenarios described in Sect. 2. All values are computed for a planning rate of 1 Hz, mapping latency 0.2 s, sensor point cloud of size 4.0 m × 3.15 m, occupancy grid resolution of 20 cm, overall bound on top speed $V_{\max}$ at 8 m/s, and collision radius r_{coll} set at 0.3 m.

Value/Cases	Area (m^2)	Velocity (m/s)	Entropy rate (bits/s)
Perpendicular ($\perp$)	12.60	2.04	3.21×10^3
Parallel ($\parallel$)	19.00	8.00	1.90×10^4
$\perp$, **rapid yaw**	31.50	2.04	8.03×10^3
$\parallel$, **rapid yaw**	50.90	8.00	5.09×10^4

is given by

$$\begin{aligned} V_{\perp,\max} &= \min(V_{\max}, V'_{\perp,\max}) \\ V'_{\perp,\max} &= A_{\max} \cdot \left(\sqrt{\Delta t_l^2 + 2\frac{r_{\text{depth}} - r_{\text{coll}}}{A_{\max}}} - \Delta t_l \right). \end{aligned} \tag{2}$$

Figure 4 shows the variation of this bound with r_{depth} and Δt_l for the parameters used in this work. In the ideal scenario for motion parallel to a frontier (see Sect. 2.3), there are no obstacles in the direction of motion. Therefore, the ideal upper bound on the velocity moving parallel to the frontier is identical to the maximum achievable by the system

$$V_{\parallel,\max} = V_{\max}. \tag{3}$$

2.5 Bounds on Rate of Exploration

2.5.1 Perpendicular to frontier

We calculate the number of voxels explored per unit time for the ideal scenario when robot moves towards the sensor observation while performing rapid yaw as follows. For an occupancy grid of cell resolution c and sensor point cloud dimensions W(width) × H(height), maximum rate of voxel discovery perpendicular to frontier (using (2)) is

$$\dot{\mathcal{N}}_\perp = \frac{\mathcal{A}_\perp}{c^3} A_{\max} \left(\sqrt{\Delta t_l^2 + 2\frac{r_{\text{depth}} - r_{\text{coll}}}{A_{\max}}} - \Delta t_l \right) \tag{4}$$

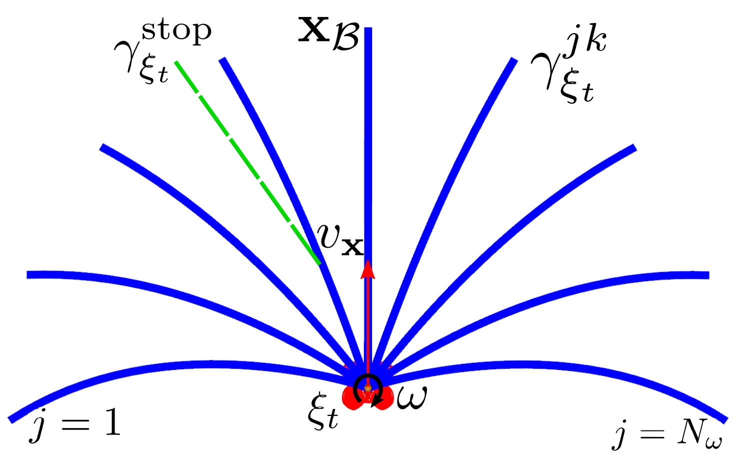
Here, $\mathcal{A}_\perp$ is the projection of the sensor cone in the direction of motion, $\mathbf{x}_{\mathcal{B}}$ if not yawing $\mathcal{A}_\perp = WH$ or, if yawing rapidly, $\mathcal{A}_\perp = 2r_{\text{depth}}H$.

2.5.2 Parallel to frontier

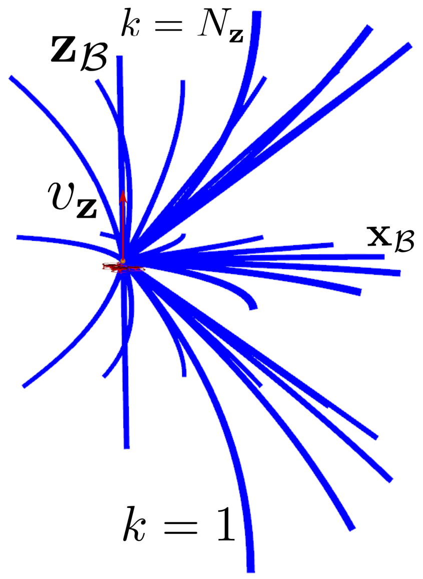
For the parallel case, area of the sensor wavefront changes to $\mathcal{A}_{\parallel} = (\sqrt{W^2 + H^2}) r_{\text{depth}}$ (see Fig. 3c) or $\mathcal{A}_{\parallel} = r_{\text{depth}} H$ for rapid yaw. Hence, maximum rate of voxel discovery in this case is (since $V_{\parallel,\max} = V_{\max}$):

$$\dot{\mathcal{N}}_{\parallel} = \frac{\mathcal{A}_{\parallel}}{c^3} V_{\max} \tag{5}$$

Substituting the values for sensor and the platform in (4) and (5), we get the required bounds on entropy reduction rate $\dot{\mathcal{H}}_{\max}$, shown in Table 1.



(a) Variation in ω



(b) Variation in $v_{\mathbf{z}}$

Figure 5: Available actions at the multirotor state $\boldsymbol{\xi}_t$ [1]. A set of forward arc motion primitives $\gamma_{\boldsymbol{\xi}_t}^{jk}$ (gray) are sampled using user specified action design parameters and discretized velocity bounds described in Sect. 3, and analysis presented in Sect. 2. The set of such primitives at each state is termed a motion primitive library (MPL) $\Gamma_{\boldsymbol{\xi}_t}$, which are sampled in $\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}$ and $-\mathbf{y}_{\mathcal{B}}$ directions (only $\mathbf{x}_{\mathcal{B}}$ shown for brevity). Dynamically feasible stopping trajectories $\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$ are also available for each primitive (green, dashed) for safety.

3 Action Representation

This section details the design of available actions for the proposed motion planning framework. We define a trajectory generation scheme, related parameters and conventions, and action design specifics leveraging insights gained in Sect. 2.

3.1 Motion Primitive Library Generation

High-speed flight requires large and smooth linear accelerations and angular velocity references. Smoothness for such references depends on higher derivatives of position, jerk and snap [13]. We use snap-continuous forward arc motion primitives to represent actions, which have been used in high-speed flight for multirotors by Spitzer et al. [1]. Given differentially-flat state of the multirotor at time t, $\boldsymbol{\xi}_t$, we can denote available 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 $\mathbf{x}_{\mathcal{B}}$ and $\mathbf{z}_{\mathcal{B}}$ directions, and ω is the body yaw rate. Actions are discretized using 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 (Fig. 5):

$$\Gamma_{\boldsymbol{\xi}_t} = \{\gamma_{\boldsymbol{\xi}_t}^{jk} \mid j \in [1, N_\omega], k \in [1, N_{\mathbf{z}}], |\mathbf{v}| \leq V_{\max}, |\omega| \leq \Omega_{\max}\} \tag{6}$$

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.

For a given action discretization, motion primitive $\gamma_{\boldsymbol{\xi}_t}^{jk}$ is generated as an 8th order polynomial in time using start and end point velocities, keeping position unconstrained. Velocity at end point in time, τ, is obtained by forward propagating a unicycle kinematics model using the current state and the available action parameterization while for other higher order derivatives up to snap, endpoints are kept zero:

$$\begin{aligned} \dot{\boldsymbol{\xi}}_\tau &= [v_{\mathbf{x}} \cos\theta, v_{\mathbf{x}} \sin\theta, v_{\mathbf{z}}, \omega] \\ \boldsymbol{\xi}_\tau^{(j)} &= \mathbf{0} \text{ for } j = 2, 3, 4 \end{aligned} \tag{7}$$

where $\{.\}^{(j)}$ denotes the j^{th} time derivative.

Stopping trajectories at any $\boldsymbol{\xi}_t$ ($\gamma_{\boldsymbol{\xi}_t}^{\text{stop}}$, Fig. 5) can be sampled by keeping $\dot{\boldsymbol{\xi}}_t = \mathbf{0}$. In contrast to the fixed duration (τ) primitives presented in [1], we search for the minimum duration for which the primitive is dynamically feasible. This is done to ensure that the vehicle is able to achieve the desired end point velocity $\dot{\boldsymbol{\xi}}_t$ in the minimum time possible from the current state. Furthermore, τ is lower bounded by the planning time and upper bounded by a user-specified maximum horizon $\tau_{\max}$. We describe this search in further in a later subsection.

3.2 Action Space for Fast Exploration

We now proceed to define the action space for the proposed fast exploration approach based on a collection of motion primitive libraries (MPLs), defined by (6), using the bounds on linear velocities obtained in Sect. 2 - $V_{\perp,\max}$ and $V_{\|,\max}$. The proposed planner uses 8 MPLs to represent the action space, $\mathcal{X}_{\text{act}} = \{\Gamma_{\boldsymbol{\xi}_t}^i \mid i \in [1, 8]\}$. These MPLs are obtained using set of different upper bounds on linear velocities as shown in Table. 2. We maintain MPLs for different velocity levels, some more than twice of $V_{\perp,\max}$, using $V_{\|,\max} = 8\,\text{m/s}$. In total, 166 primitives across 8 MPLs are available for the action selector choose from. Later, in Sect. 5, we show affects on exploration performance due to various sets of MPLs – especially the differences in parallel and perpendicular directions and low and high speed primitives.

Table 2: Discretization used to construct action space $\mathcal{X}_{\text{act}}$ using ideal analysis presented in Sect. 2. We compute velocity bounds for the planning rate of 1 Hz, i.e. $V_{\perp,\max} = 2.04\,\text{m/s}$ and $V_{\|,\max} = 8\,\text{m/s}$. For all action sets, maximum yaw rate is $1.0\,\text{rad/s}$ and maximum vertical $v_{\mathbf{z}}$ is kept at $1.5\,\text{m/s}$. Total number of primitives for a MPL are denoted by $N_{\text{prim}} = N_\omega \cdot N_{\mathbf{z}}$.

MPL ID	**Max. Linear Velocity**	Dir.	N_ω	$N_{\mathbf{z}}$	N_{prim}
1	0	yaw	1	1	1
2	$0.8 \cdot V_{\perp,\max}$	$\mathbf{x}_{\mathcal{B}}$	9	3	27
3	$V_{\perp,\max}$	$\mathbf{x}_{\mathcal{B}}$	9	3	27
4	$0.8 \cdot V_{\|,\max}$	$\mathbf{x}_{\mathcal{B}}$	9	3	27
5	$V_{\|,\max}$	$\mathbf{x}_{\mathcal{B}}$	9	3	27
6	$V_{\|,\max}$	$\mathbf{y}_{\mathcal{B}}$	9	3	27
7	$V_{\|,\max}$	$-\mathbf{y}_{\mathcal{B}}$	9	3	27
8	$v_{\mathbf{z}}$	$\mathbf{z}_{\mathcal{B}}$	1	3	3

Algorithm 1 Minimum feasible duration search for Action Generation.

```
    input: minimum duration ($\Delta t_{\text{p}}$), maximum duration ($\tau_{\max}$), search increment ($\Delta t$)
    output: minimum dynamically feasible duration
1:  for $t \in \{\Delta t_{\text{p}}, \dots, \Delta t_{\text{p}} + \Delta t, \dots, \tau_{\max}\}$ do
2:      $\gamma_{\boldsymbol{\xi}_t} \leftarrow$ SampleMoPrim($t$)   ▷ sample motion primitive with the candidate duration
3:  return $\arg\min_t$ DynamicsCheck($\gamma_{\boldsymbol{\xi}_t}$)   ▷ return the primitive
4:  function DynamicsCheck($\gamma_{\boldsymbol{\xi}_t}$)
        input: Candidate action.
        output: True if action is dynamically feasible.
5:      for $\boldsymbol{\xi}_i \in \gamma_{\boldsymbol{\xi}_t}$ do
6:          if $\ddot{\boldsymbol{\xi}}_i > A_{\max}$ and $\dddot{\boldsymbol{\xi}}_i > J_{\max}$ then   ▷ maximum acceleration and jerk feasibility
7:              return false
8:      return true
```

3.3 Minimum Feasible Duration Search

For all the actions designed in the preceding sections, the duration for the primitive is computed based on Alg. 1. Intuitively, minimum possible duration for a motion primitive can be the time it takes for the planner to output the next action – the planning time Δt_{p}. Using user-specified values for maximum duration ($\tau_{\max}$), we sample primitives at a given search increment (Δt) from the minimum duration (line 2), and return the dynamically feasible primitive at the minimum possible duration. Dynamics check is based on pre-specified empirically observed bounds on linear acceleration and linear jerk L^2-norms (line 4).

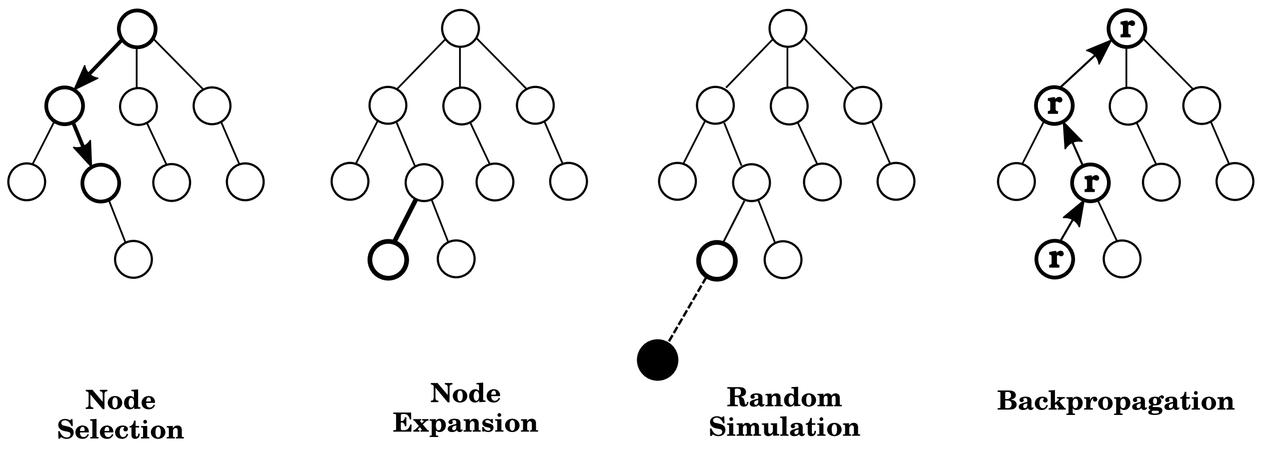


Figure 6: Brief overview of the general Monte Carlo tree search (MCTS) algorithm [2, 3]. Planner selects the deepest node that has unexpanded children based on the selection policy, which is usually based on Upper Confidence Bounds (UCB). From this selected node, children are expanded according to a random rollout policy until a terminal state or condition is reached. The cumulative reward and node visit statistics during this rollout are stored and back-propagated along the path to the root state. This process is iterated until a user-specified computational budget is reached, and the sequence of actions with the best rewards to visits ratio is returned.

4 Action Selection

We formulate the action selection problem as a finite-horizon optimization seeking to maximize cumulative information gain [14], and build upon previous work [4, 15, 16] on robotic exploration using Monte Carlo tree search (MCTS).

Most MCTS-based planners follow four steps: selection, expansion, simulation playout, and backpropagation of statistics [2, 3]. Such planners usually construct a search tree iteratively by random rollout from a previously unexpanded node selected based on upper-confidence bounds for trees (UCT) [3]. Authors in [4, 15] show application of MCTS in planning for exploration using multirotors by using a UCT-based selection policy, information gain rewards, and random simulation playout over a finite horizon. We extend the approach proposed in [4, 15] by adding considerations for model constraints into node expansion phase of MCTS.

4.1 Information-Theoretic Exploration Objectives

In terms of defining the optimization objective for the proposed action selection policy, we follow a similar approach as our previous work [15]. There are two components to this

Algorithm 2 Overview of the MCTS-based Action Selection approach.

```
    input: action space $\mathcal{X}_{\text{act}}$, free space $\mathcal{X}_{\text{free}}$, root state $\boldsymbol{\xi}_0$, max. iterations $n_{\text{b}}$, max. horizon $\tau_{\max}$
    output: best sequence of actions from the root state $\Gamma^*_{\boldsymbol{\xi}_0} = \{\gamma^*_{\boldsymbol{\xi}_t}, t = 0 \dots \tau\}$
1: $\mathcal{T} \leftarrow$ initialize MCTS tree
2: while within computational budget or iteration $< n_{\text{b}}$ do
3:     $\boldsymbol{\xi}_{i-1} \leftarrow$ NodeSelectionUCT($\mathcal{T}$)    ▷ upper confidence bounds based node selection
4:     $\boldsymbol{\xi}_i \leftarrow$ NodeExpansion($\mathcal{T}$, $\boldsymbol{\xi}_{i-1}$)    ▷ ensures no inevitable collision states (ICS)
5:     $\boldsymbol{\xi}_{\tau_{\max}} \leftarrow$ RolloutPolicy($\mathcal{T}$, $\boldsymbol{\xi}_i$)    ▷ random simulation until terminal state
6:     $\mathcal{T} \leftarrow$ Backpropagation($\boldsymbol{\xi}_{\tau_{\max}}$, $\boldsymbol{\xi}_0$)    ▷ update statistics of visited nodes
7: return $\Gamma^*_{\boldsymbol{\xi}_0} \leftarrow \arg\max \frac{\boldsymbol{\xi}_{\text{rewards}}}{\boldsymbol{\xi}_{\text{visits}}}$    ▷ return actions connecting nodes with highest rewards/visits ratio
```

objective: a local information reward based on Cauchy-Schwarz quadratic mutual information (CSQMI) [14], and a global reward for the planner based on the decrease in the least distance to an informative view in the view library [15]. For any candidate action, $\gamma_{\boldsymbol{\xi}_t}$, we compute the local information gain $\mathcal{I}_\gamma$ over user-specified time intervals and treat the joint information reward as a heuristic for our MCTS planner. An additional reward based on informative views in view library, $\mathcal{V}_\gamma$, is computed based on the decrease in distance to the closest informative view. Specifically, let $d(\boldsymbol{\xi})$ denote the shorted-path distance to the closest informative view from any state $\boldsymbol{\xi}$. Then, the decrease in distance is based on start and end points of the candidate action $\gamma_{\boldsymbol{\xi}_t}$, i.e. $\mathcal{V}_\gamma = d(\boldsymbol{\xi}_\tau) - d(\boldsymbol{\xi}_0)$. This distance reward serves as a heuristic for exploration at the spatially global level. As such, competing routing and scheduling approaches [17] could be substituted for the distance cost with tradeoffs in computational cost and system design. For example, whereas selection of a destination view is implicit in computation of our distance cost, an approach based on a traveling salesman problem may commit to a fixed solution beforehand to avoid excessive computation. For further detail on the computation of these rewards, please refer to [15].

4.2 Monte Carlo Tree Search (MCTS)

Each *node* of the tree, $\mathcal{T}$, shown in Fig. 6 is the flat state of a multirotor, denoted by $\boldsymbol{\xi}$, with the root state being written as $\boldsymbol{\xi}_0$. *Edges* of the tree are the motion primitives available to execute from the originating node. Note that these motion primitives are of minimum dynamically feasible duration as shown in Sect. 3.3, so the durations of these edges in the tree varies with the node they originate from.

Action selection proceeds as shown in Alg. 2. After initializing the tree with the root state $\boldsymbol{\xi}_0$ (line 1), MCTS iterates until either a user specified number of maximum iterations or a time budget (usually set by the planning rate) condition is violated. Within each iteration, the tree is incrementally created in four steps. First, at line 3, a node with unexpanded children is selected using a criteria based on Upper Confidence Bounds for Trees (UCT) [3]. Second, from the resulting node, we choose a feasible action and expand the tree from $\boldsymbol{\xi}_{i-1}$ to $\boldsymbol{\xi}_i$ (line 4). Third, we insert nodes at random further down in the tree until a state is reached where the limit on maximum horizon is exceeded ($\boldsymbol{\xi}_{\tau_{\max}}$, line 5). And fourth, we backpropagate the node visits and reward statistics from $\boldsymbol{\xi}_{\tau_{\max}}$ to the root node $\boldsymbol{\xi}_0$ and continue further iterations (line 6). At the end, the sequence of nodes with the highest rewards to visits ratio is returned as the most informative trajectory from the current planning round ($\Gamma^*_{\boldsymbol{\xi}_0}$).

As mentioned before, this report mainly contributes to the node expansion step (lines 4). Specifically, we propose a node expansion approach that ensures no nodes in the tree are visited from which the multirotor can not stop within the sensor range, r_{depth}. In this way, we ensure that the robot never visits an inevitable collision state (ICS) [12]. Next, we describe this safe node expansion approach.

4.3 Safety at High Speeds

Using the finite-horizon action representation developed in Sect. 3, we present a safe node expansion algorithm that ensures the multirotor never reaches an inevitable collision state (ICS), essential to maintain safety at high speeds in unknown environments [12]. The node expansion policy for the proposed MCTS-based planning policy is shown schematically in Alg. 3. Candidate actions from the current node $\boldsymbol{\xi}_{i-1}$ are evaluated for expansion in three steps. First, the candidate action is checked to lie in the explored free space $\mathcal{X}_{\text{free}}$ (line 2). Specifically, we create a truncated signed distance field (TSDF) from locations of occupied and unknown spaces in the robot's local map and inflate our lookups in this grid by a user-specified collision radius, r_{coll}. Second, if the candidate action is inside $\mathcal{X}_{\text{free}}$, we sample a dynamically feasible stopping trajectory using the minimum feasible duration method shown in Sect. 3.3 at Δt_{p} (planning time) ahead of the current time (line 3). We choose this starting time to ensure that we have a safe trajectory to track and bring the robot to halt within the observed free space $\mathcal{X}_{\text{free}}$, in case the current planning round fails (further detail in Sect. 4.4). Third, this stopping primitive $\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$ is checked for collisions (line 4) using the same TSDF

Algorithm 3 Safety in Node Expansion.

```
1: function SafetyCheck($\gamma_{\boldsymbol{\xi}_t}$, $\mathcal{X}_{\text{free}}$)
       input: Candidate action.
       output: True if candidate action is feasible.
2:     if FreeSpace($\gamma_{\boldsymbol{\xi}_t}$) then                ▷ check if the action lies in free space
3:         $\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$ ← SampleStopPrim($\gamma_{\boldsymbol{\xi}_t}$)                ▷ stopping primitive
4:         return FreeSpace($\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$)
5:     else
6:         return false
7: function FreeSpace($\gamma_{\boldsymbol{\xi}_t}$)
       input: Candidate action.
       output: True if action is in explored free space.
8:     for $\boldsymbol{\xi}_i \in \gamma_{\boldsymbol{\xi}_t}$ do                ▷ sampled states along primitive
9:         if $\boldsymbol{\xi}_i \notin \mathcal{X}_{\text{free}}$ then                ▷ position should be in $\mathcal{X}_{\text{free}}$
10:            return false
11:    return true
```

as in line 2. Note that the stopping action is only sampled if the candidate action ($\gamma_{\boldsymbol{\xi}_t}$) passes the collision check at line 2. If $\gamma_{\boldsymbol{\xi}_t}$ is itself not in $\mathcal{X}_{\text{free}}$, it is marked infeasible without sampling a stopping primitive (line 6). Moreover, a dynamic feasibility check is not required here since the primitives are ensured to be within the user-specified acceleration and jerk limits given the duration of the primitive is calculated using the search method presented in Sect. 3.3.

These three steps ensure that from any node visited during MCTS iterations, the vehicle can always come to complete stop within free space. Thus, no ICS is visited during the search, ensuring output actions are safe to execute [12].

4.4 Scheduling Planner Output

The output from MCTS planner is a sequence of actions, $\Gamma^*_{\boldsymbol{\xi}_0}$, that maximize the information-theoretic objectives presented in Sect. 4.1, and the corresponding safety actions. Figure 7 provides an overview of how this output is scheduled for the trajectory tracker to follow. From the root state $\boldsymbol{\xi}_0$, the segment (a) + (c) denotes the sequence $\Gamma^*_{\boldsymbol{\xi}_0}$ while (b) denotes the feasible stopping primitive $\gamma^{\text{stop}}_{\boldsymbol{\xi}_t}$. Notice that (b) is scheduled at $\boldsymbol{\xi}_{\text{p}}$, one planning round ahead of the root state $\boldsymbol{\xi}_0$. This is done to ensure that, in case the next planning round

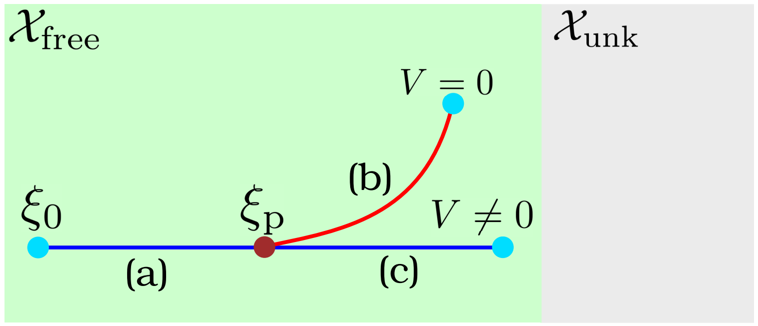
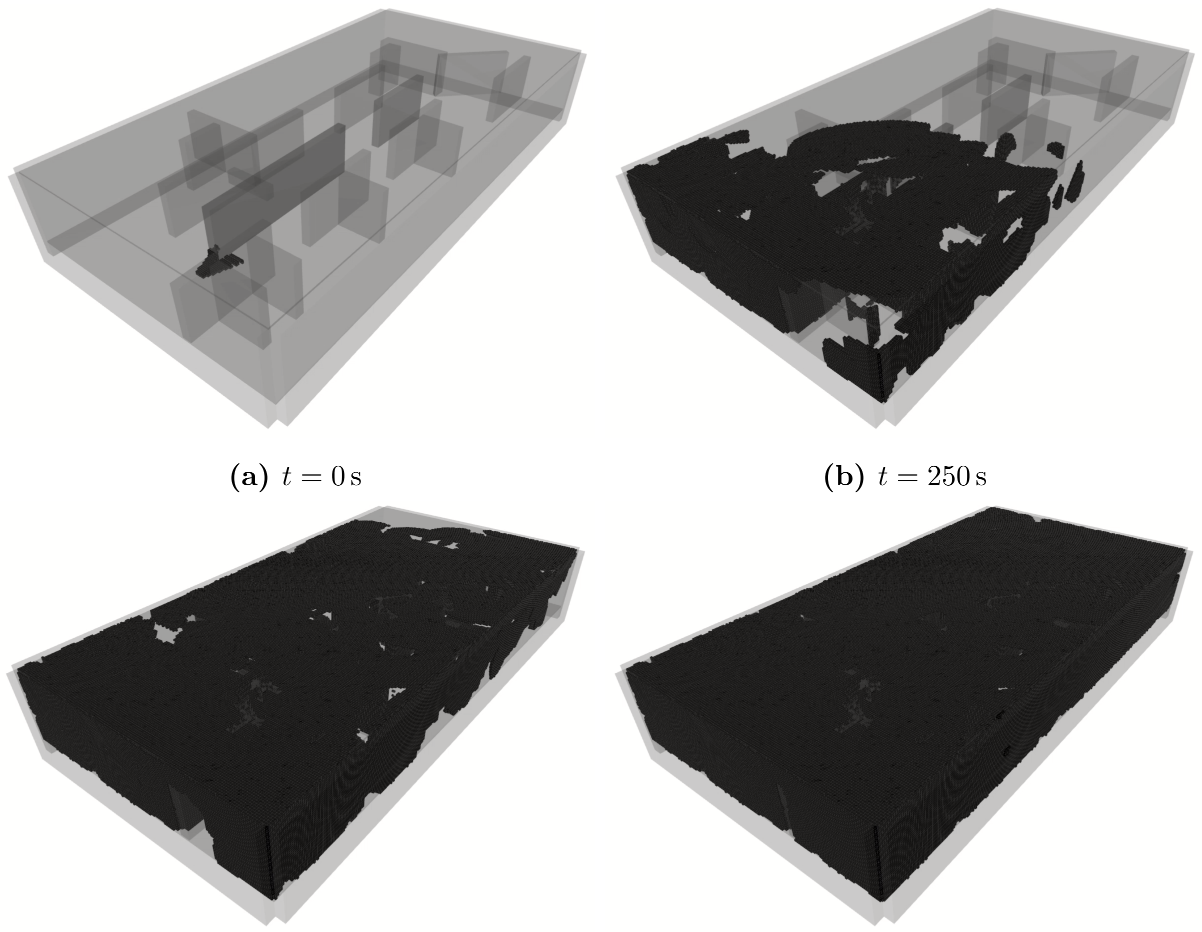


Figure 7: Overview of safe node expansion in the proposed MCTS-planner and the scheduling of stopping trajectories. Safety checks detailed in Alg. 3 and Sect. 4.3 ensure that all trajectory segments (a), (b), and (c) lie in free space, $\mathcal{X}_{\text{free}}$. Segment (a) is scheduled at root node $\boldsymbol{\xi}_0$, while segments (b) and (c) are scheduled at the end of the current planning round $\boldsymbol{\xi}_{\text{p}}$. One of the segments (b) or (c) are selected for tracking at $\boldsymbol{\xi}_{\text{p}}$ based on the output of the next planning round. This approach to scheduling and safety checking ensures that the vehicle is always able to stop within $\mathcal{X}_{\text{free}}$, even there is a failure in finding a feasible action during a one of the planning rounds.

fails, the trajectory tracker switches to tracking the segment (b) instead of (c) (which has a non-zero end-point velocity). The segment (b) ensures that the vehicle stops within $\mathcal{X}_{\text{free}}$ with velocity and all higher-order derivatives zero at the end-point.

As described earlier in Sect. 3, these actions are 8$^{\text{th}}$ order polynomials in velocity, smooth and continuous up to snap. Such high-order continuity ensures that there is no abrupt jump in references for the trajectory tracker when switching between trajectory segments (for example, from (a) to (b) in Fig. 7).

5 Results



(a) $t = 0\,\mathrm{s}$ **(b)** $t = 250\,\mathrm{s}$

(c) $t = 900\,\mathrm{s}$ **(d)** $t = 1800\,\mathrm{s}$

Figure 8: Different stages of exploration with respect to time for three-dimensional unstructured environment considered for evaluation of the proposed motion planning framework. Black voxels denote space marked as occupied. We consider a large room in 3D warehouse environment with dimensions $60\,\mathrm{m} \times 30\,\mathrm{m} \times 11\,\mathrm{m}$.

This section details the quantitative evaluation of the proposed exploration strategy. Specifically, we demonstrate variation of exploration performance due to changes in action design (Sect. 3) using the rate of reduction in Shannon entropy of the map [9]. For all experiments, map is constructed using depth sensor data from the multirotor exploring a large unstructured three-dimensional warehouse environment (see Fig. 8). In all our experiments, we use the action representation presented in Sect. 3 and action selection strategy presented in Sect. 4.

5.1 Simulation Experimentation Setup

In this section, various parametric details are presented that are common to all the experiments conducted in the later sections. Specifically, we state the configurations for multirotor inertial properties and constraints, sensing modality specifications, and assumptions in simulation experiments.

Table 3: Inertial properties and dynamics constraints for the multirotor used in the simulation experiments.

Configuration	**Weight**	**Thrust/Weight**	$A_{\max}$	$J_{\max}$
Value	30.45 N	3.5	$10\,\mathrm{m/s^2}$	$35\,\mathrm{m/s^3}$

5.1.1 Inertial and Sensing Parameters

Table. 3 lists a few of the critical dynamics parameters for the multirotor used in our experiments. Intuitively, to be able to execute fast actions on a multirotor, we need a high thrust to weight ratio to be able to change attitude of the multirotor rapidly. Therefore, we use the parameters from the high speed teleoperation platform presented in Spitzer et al. [1]. Furthermore, empirically determined acceleration ($A_{\max}$) and jerk ($J_{\max}$) constraints for this platform are taken into account in both the proposed action representation (Sect. 3) and action selection (Sect. 4).

Table 4: Sensing constraints for the forward facing depth camera used in for the exploration experiments.

Configuration	r_{depth}	**Rate**	**Width**	**Height**	**Noise**
Value	5.0 m	10 Hz	160 pixels	120 pixels	N/A

Following inertial constraints, we now list key sensing modality parameters and constraints in Tab. 4. We use a single forward facing depth sensor for all experiments with maximum range capped at r_{depth}. To enable a faster mapping rate we keep the rate of sensor observations at 10 Hz. For all sensor observations, we do not consider noise in depth measurements to keep the focus on planning in a deterministic map.

Table 5: Average rates at which various types of high and low speed primitives are selected by the action selector. These rates are calculated after averaging number of primitives selected of the said type with respect to the total number of planning rounds across trials.

Experiments	$\perp$	$\|$	Stop	Yaw	High	Low
E1: $\|, \perp$, low speed	0.30	0.57	8.85×10^{-4}	0.13	-	1.0
E2: $\perp$, low speed	0.57	-	3.40×10^{-3}	0.43	-	1.0
E3: $\perp$, high speed	0.54	-	6.80×10^{-3}	0.45	0.31	0.69
E4: $\|, \perp$, high speed	0.45	0.25	0.01	0.29	0.54	0.46

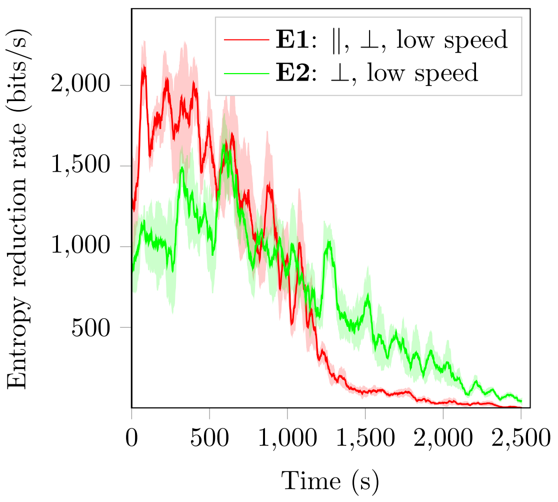
5.1.2 Triggered Simulation

To analyze variation in exploration performance due to increasing planning rates and finite-horizon lengths, we trigger the simulation based on the time taken by the planner. Specifically, number of iterations taken by the MCTS planner is kept consistent across trials and the simulation clock is slowed down to accommodate it accordingly. All trials are 2500 seconds in duration with robots starting at different initial positions in the environment.

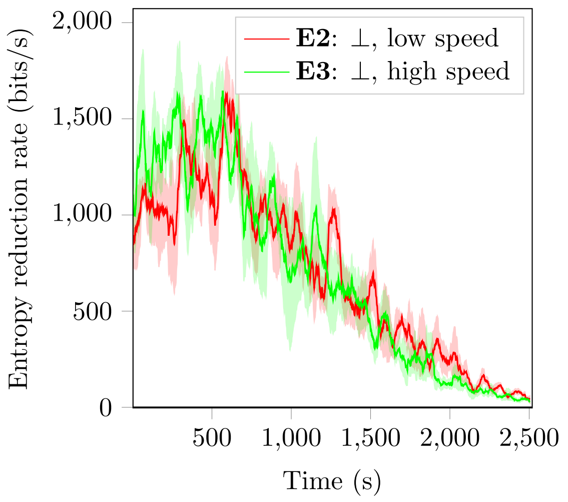
5.2 Effects of Action Design

In this section, we identify the effects of the variation in available motion primitive libraries that are available to the MCTS planner have on the following experiments (all experiments were performed using a MCTS-based planner with a 2 second planning horizon):

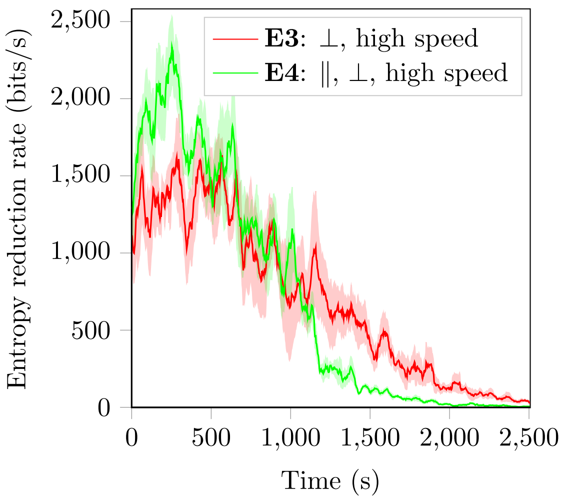
- **E1**: Action design with low speed actions in both perpendicular and parallel directions to the sensor observation, no high speed actions.
- **E2**: Action design with low speed actions in only perpendicular direction to the sensor observation, no actions in parallel direction.
- **E3**: Action design with low and high speed actions in perpendicular direction to the sensor observation, no actions in parallel direction.
- **E4**: Action design with low and high speed actions in perpendicular direction, while only high speed actions in parallel direction to the sensor observation.



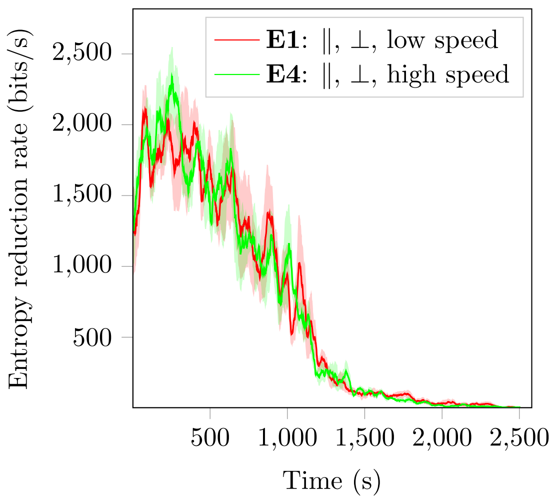
(a) Effect of ∥ direction at low speed



(b) Speed comparison in ⊥ direction



(c) Effect of ∥ direction at high speed



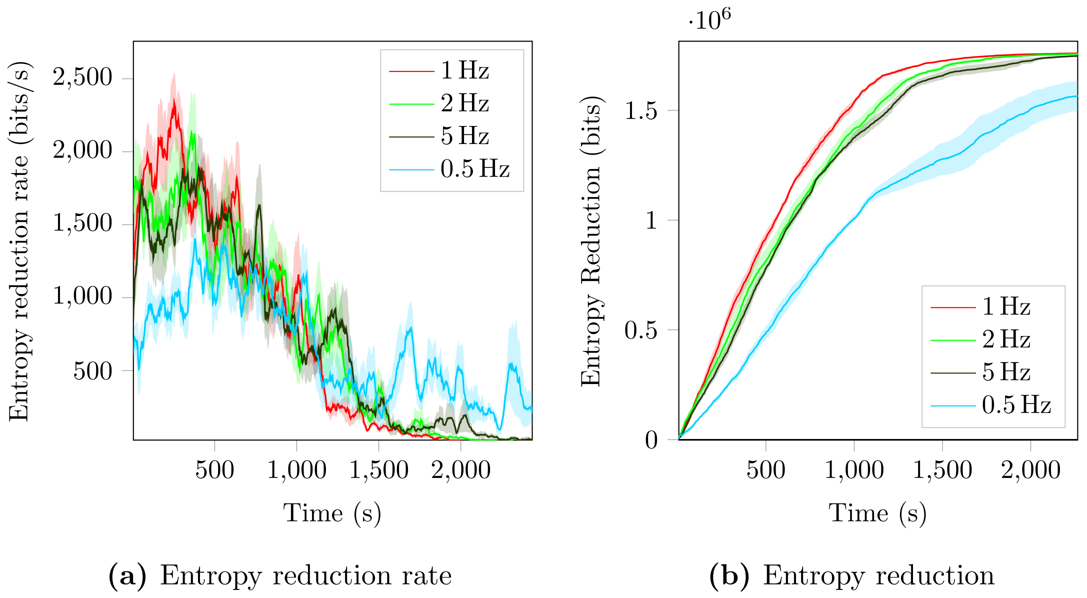
(d) Speed comparison in both directions

Figure 9: Rate of exploration variations with time to investigate effects of changes in action design on exploration performance. We observe substantial improvement in performance due to availability of parallel motions **(a, c)**, while performance increment due to high speed perpendicular actions is observed to be minimal **(b, d)**.

Comparison between exploration performance across these experiments is shown in Fig. 9. As we have shown in the idealized analysis (Sect. 2), parallel actions also influence entropy reduction rate (measured in bits/s) significantly in addition to actions perpendicular to a frontier or a sensor observation in general. This analysis is reflected in the results shown in

Fig. 9a, where the use of parallel direction primitives in experiment sets **E1, E4** provides a nearly 1.5 times improvement in entropy reduction performance over **E2** and **E3** respectively (see Fig. 9a and Fig. 9c). This behavior is expected since parallel motions have a larger effective area during exploration and are allowed to achieve higher speeds than perpendicular actions (Sect. 2). However, introducing high speed actions in either directions seems to impart only a little advantage over the low speed actions (see Fig. 9b and Fig. 9d), which is in contrast to our analysis in Sect. 2.

We can investigate why including a high speed action representation not been effective in exploration performance, by looking at the rate at which a particular type of primitive is selected. Table 5 shows these rates for each of the experiments averaged across trials. For **E3**, we observe that only about one-third of the high speed primitives are actually selected by the planner, while for **E4** we observe about half of them being chosen. This follows from our earlier observation in Sect. 2 that *in perpendicular direction to a frontier, the motion planner favors slower actions due to the safety constraints.* In addition, comparing **E1** and **E4**, there is a 2 times decrease in the probability of a parallel direction primitive being selected. This means that *at higher speeds, the action selector is not able to select parallel actions as often as at low speeds.* These two factors counteract each other, leaving a negligible performance increment due to higher speeds. We currently suspect that this behavior is either due to enhanced feasibility constraints at higher speeds or excessive length of the motion primitive library available to the finite horizon planner, see the limitations section (Sect. 6) for further discussion.



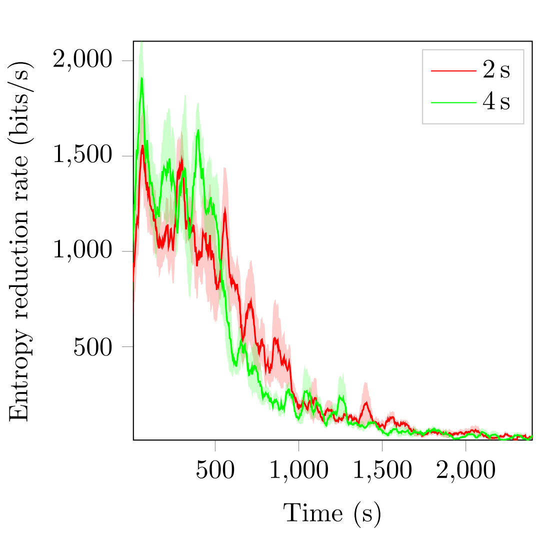
(a) Entropy reduction rate **(b)** Entropy reduction

Figure 10: Results for effects of planning rate on exploration performance in a three-dimensional scenario. Two metrics, entropy reduction rate **(a)** and entropy reduction **(b)**, are shown with respect to time. We observe a peculiar behavior – entropy reduction rate maximizes for 1 Hz planning rate, performing better than 2 Hz and 5 Hz. Furthermore, a substantial part of the environment is left unexplored at 0.5 Hz **(b)**.

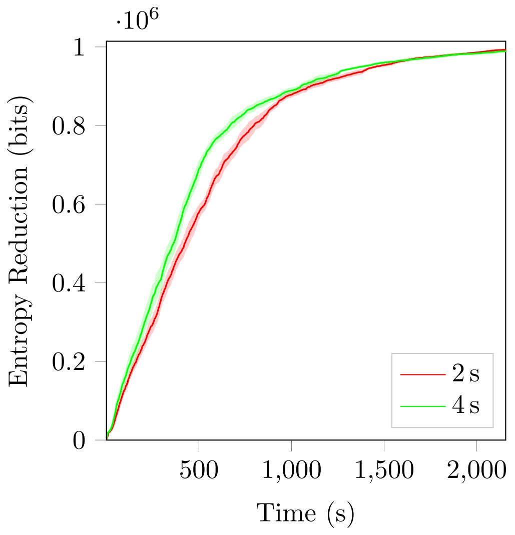
5.3 Effects of Planning Rate

This section deals with analyzing exploration performance with changing replanning rate. As analyzed in Sect. 2, faster rate of planning means the multirotor can execute higher velocity actions while remaining anytime safe. Thus, the rate of exploration should increase as we move towards faster planning rates. To this end, using our simulation setup, we conduct the 2500 seconds long trials with planning rates 0.5 Hz, 1 Hz, 2 Hz, and 5 Hz. Note that all trials in these experiments use the action design used in experiment **E4** previously.

We observe that the exploration performance peaks at 1 Hz, while the performance at 0.5 Hz degrades to an extent that a significant part of the environment is left unexplored (Fig. 10). In contrast to our analysis in Sect. 2, the entropy reduction rate does not get better with increasing planning rate after 1 Hz. This result does not follow the expected behavior shown in Sect. 2. The cause of this trend might be linked with the high speed primitives not being effective in increasing exploration performance, as suggested in the preceding section. However, further investigation is required to get a better insight into this variation, and we leave it as a future work.



(a) Entropy reduction rate



(b) Entropy reduction

Figure 11: Variation in length of the finite-horizon and its effects on exploration performance in a two-dimensional scenario. Two metrics, entropy reduction rate **(a)** and entropy reduction **(b)**, are shown with respect to time. As expected, longer finite horizon results in significantly better exploration rate (here we compare 2 second with a 4 second horizon).

5.4 Effects of Finite-Horizon Length

As detailed in Sect. 4, we use a Monte-Carlo Tree Search (MCTS) based finite horizon planning strategy to select actions that are feasible and maximize information gain. In this section, effects of the horizon length of this receding-horizon planner on the exploration performance are analyzed. We start with a simplified two-dimensional scenario, where action space is constrained to lie only in the $\mathbf{x}_{\mathcal{B}}-\mathbf{y}_{\mathcal{B}}$ plane (so, no motion primitives in $\mathbf{z}_{\mathcal{B}}$ direction). In Fig. 11, results for simulation experiments at two different horizon lengths – 2 and 4 seconds – are shown. For a 4 seconds horizon, exploration rate is nearly 75% faster than that of 2 seconds case during the first 800 seconds of exploration.

It is worth noting here that as the horizon length increases, the computational cost per planning iteration of our MCTS-based planner also increases significantly. Thus, there is a trade-off between computational complexity and the exploration performance that needs to be considered while trying to use this planner in a real time use case.

6 Conclusion

6.1 Contributions

We presented three main contributions in this report. First, we analyzed how exploration performance varies using a few idealized scenarios considering constraints on dynamics and sensing for a multirotor vehicle. Second, we proposed a strategy to design candidate actions for the motion planner that accounts for platform dynamics and provides actions that are useful for rapid exploration based on the prior analysis. Third, we presented a receding-horizon sampling-based planner that uses this action representation, maximizes information gain, and ensures safe operation at high speeds. Finally, through extensive simulation experiments in a complex three-dimensional environment, we demonstrated the significance of action design, finite-horizon length, and replanning rate on exploration performance.

6.2 Limitations & Future Work

Sect. 5 highlights a few limitations in our current approach to action representation (Sect. 3) and action selection (Sect. 4). First, the variation in the exploration performance due to increase in planning rate is in contrast to the ideal exploration analysis in Sect. 2. We note that the cause of this behavior might be due to excessive length of the primitives in perpendicular direction at higher speeds. Another reason might include large spaces between poses on which CSQMI is evaluated. One solution might be discounting the information rewards with respect to time. Second, the computational cost of our MCTS-based action selection strategy scales with the number of desired iterations as well as finite-horizon length. Solving this limitation might require using intelligent biasing techniques to search for informative regions in the search tree and hence generate a better planning policy in lesser number of iterations.

References

[1] 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.

[2] G. Chaslot, "Monte-Carlo tree search," Ph.D. dissertation, Universiteit Maastricht, 2010.

[3] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. on Comput. Intell. and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

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

[5] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *J. Field Robot.*, Jun. 2012.

[6] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, Sep. 2017.

[7] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016.

[8] 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.

[9] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York, NY: John Wiley & Sons, 2012.

[10] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 56–65, 2010.

[11] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles,” *IEEE Robot. Autom. Mag.*, vol. 20, no. 32, 2012.

[12] 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.

[13] 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.

[14] 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.

[15] M. Corah, C. O’Meadhra, K. Goel, and N. Michael, “Communication-efficient planning and mapping for multi-robot exploration in large environments,” *IEEE Robot. Autom. Letters*, 2019, submitted for publication.

[16] M. Lauri and R. Ritala, “Planning for robotic exploration based on forward simulation,” *Robot. Auton. Syst.*, vol. 83, 2016.

[17] M. Kulich, J. Faigl, and L. Přeučil, “On distance utility in the exploration task,” in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* Shanghai, China: IEEE, May 2011.