Search-Based Motion Planning for Aggressive Flight in SE(3)

Sikang Liu, Kartik Mohta, Nikolay Atanasov, and Vijay Kumar

***Abstract*—Quadrotors with large thrust-to-weight ratios are able to track aggressive trajectories with sharp turns and high accelerations. In this letter, we develop a search-based trajectory planning algorithm that exploits the quadrotor maneuverability to generate sequences of motion primitives in cluttered environments. We model the quadrotor body as an ellipsoid and compute its flight attitude along trajectories in order to check for collisions against obstacles. The ellipsoid model allows the quadrotor to pass through gaps that are smaller than its diameter with nonzero pitch or roll angles. Without any prior information about the location of gaps and associated attitude constraints, our algorithm is able to find a safe and optimal trajectory that guides the robot to its goal as fast as possible. To accelerate planning, we first perform a lower dimensional search and use it as a heuristic to guide the generation of a final dynamically feasible trajectory. We analyze critical discretization parameters of motion primitive planning and demonstrate the feasibility of the generated trajectories in various simulations and real-world experiments.**

***Index Terms*—Motion and path planning, autonomous vehicle navigation, aerial systems: applications.**

I. Introduction

MOTION planning, the problem of generating dynamically feasible trajectories that avoid obstacles in unstructured environments, for Micro Aerial Vehicles (MAVs), especially quadrotors, has attracted significant attention recently [1]–[4]. When the MAV attitude and dynamics are taken into account, the problem is challenging because there are no simple geometric conditions for identifying collision-free configurations [5]. Existing planning approaches usually model the MAV as a sphere or prism, which allows obtaining a simple configuration space (C-space) by inflating the obtacles with the robot size. As a result, the robot can be treated as a single point in C-space and the collision-checking even for trajectories that take dynamics into account is simplified. Even though this spherical model assumption is widely used in motion planing, it is very conservative since it invalidates many trajectories whose feasbility depends on the robot attitude (see Fig. 1). Several prior works have demonstrated aggressive maneuvers for quadrotors that pass through narrow gaps [6]–[8] but, instead of solving the planning problem, those works focus on trajectory generation with given attitude constraints. Those constraints are often hand-picked beforehand or obtained using gap detection algorithms which only works for specific cases.

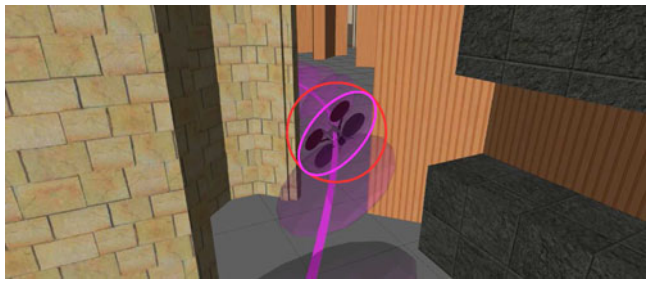

Fig. 1. By taking the shape and dynamics of a quadrotor into account, our planner is able to generate a trajectory that allows the quadrotor to pass through a door, narrower than robot's diameter. In contrast, existing methods that model the quadrotor as a sphere (red circle) would not be able to find a feasible path in this environment.

We are interested in designing a planner that considers the robot's actual shape and dynamics in order to obtain aggressive trajectories in cluttered environments. Since quadrotors are under-actuated systems, they cannot translate and rotate independently. this letter builds on our previous search-based trajectory planning approach [9] that utilizes motion primitives to discretize the control space and obtain a dynamically feasible resolution-complete (i.e., optimal in the discretized space) trajectory in cluttered environments. We extend our previous work by explicitly computing the robot attitude along the motion primitives and using it to enforce collision constraints. Furthermore, to reduce computation time for searching in high-dimensional (velocity, acceleration, jerk, etc.) space, we propose a novel hierarchical planning process that refines a dynamically feasible trajectory from a prior trajectory in lower dimensional space. The paper makes the following contributions:

1) A graph search algorithm that uses motion primitives to take attitude constraints into account and compute a dynamically feasible resolution-complete trajectory for a quadrotor is developed.
2) A hierarchical refinement process that uses prior lower-dimensional trajectories as heuristics to accelerate planning in higher dimensions is proposed.

Manuscript received September 10, 2017; accepted January 15, 2018. Date of publication January 23, 2018; date of current version March 29, 2018. This letter was recommended for publication by Associate Editor S. Chakravorty and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported by in part by ARL under Grant W911NF-08-2-0004, in part by DARPA under Grants HR001151626 and HR0011516850, in part by ARO under Grant W911NF-13-1-0350, and in part by ONR under Grant N00014-07-1-0829. *(Corresponding author: Sikang Liu.)*

S. Liu, K. Mohta, and V. Kumar are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: sikang@seas.upenn.edu; kmohta@seas.upenn.edu; kumar@cis.upenn.edu).

N. Atanasov is with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: natanasov@ucsd.edu).

Digital Object Identifier 10.1109/LRA.2018.2795654

3) The effect of motion primitive discretization parameters on the computation time, smoothness, and optimality of the generated trajectories is analyzed.

The code used in this work is open-sourced at https://github.com/sikang/mpl_ros. Users can easily test our planner and benchmark the performance against other planning algorithms. In addition, a video clip of the simulation and experimental results is published on https://youtu.be/V4Mha-KPtwc.

II. Related Works

Trajectories for MAVs or, more generally, differentially flat systems are represented as piecewise polynomials whose derivatives can be used to obtain explicit expressions for the system states and control inputs [10]. When collision avoidance is taken into account, more constraints need to be added to guarantee safety either through anchoring waypoints as in [2], [11] or building up a safe flight corridor as in [4], [12], [13]. These approaches require planning in a C-space in which the robot's attitude does not affect collision checking. Conservative symmetrical approximations of the robot body may ignore trajectories whose feasibility depends on the robot attitude. Hence, planning in SE(3) is necessary in order to obtain agile trajectories in cluttered environments. Planning with 6 DOF has been addressed in several works [14], [15] via sampling techniques but these do not translate immediately to our problem, where the rotation and translation are coupled and a smooth, deterministic trajectory is desired. Methods based on motion primitives are a promising approach for planning dynamically feasible and collision-free trajectories. For example, lattice search with predefined primitives [16], [17] may be used to plan trajectories for non-circular robots in obstacle cluttered environments. In our previous work [9], we developed an approach for quadrotors based on lattice search by using motion primitives generated via optimal control [18]. In this work, we extend [9] to account for attitude constraints by explicitly computing the robot attitude along the motion primitives based on the desired acceleration and gravity.

While randomized sampling approaches have been effective at solving very high dimensional planning problems, they take a long time to converge to an optimal solution [19] and intermediate solution quality might be unpredictable. Hence, randomized approaches are not suitable for fast navigation in unknown environments where frequent, predictable re-planning is necessary. Traditional graph search techniques are considered inefficient in high dimensional spaces but appropriate heuristic design [20]–[22] may accelerate their speed. Using weighted heuristics, however, produces sub-optimal solutions and does not always reduce planning time [23]. An interesting, alternative idea for accelerating motion planning is based on adaptive dimensionality [24], which exploits preliminary search results in lower dimensions to accelerate the planning process in high dimensions. In this work, we use a hierarchical planning procedure—plan a trajectory in low dimensional space and use it as a heuristic to guide the search in high dimensional space—to improve the refinement step in [9] which can potentially lead to unsafe and infeasible trajectories, while guaranteeing dynamical feasibility, safety, and resolution completeness.

III. Motion Planning With Attitude Constraints

In this section, we introduce our trajectory planning framework based on motion primitives. While our previous work [9] guarantees safety, dynamical feasibility and optimality, it assumes a spherical robot body. Here, we introduce a way to account for the robot attitude during planning based on the desired acceleration and gravity. Since the quadrotor yaw is decoupled and does not affect system dynamics, we assume it remains constant during planning.

A. System Dynamics in Planning

Before introducing the planning approach, we inspect the relation between polynomial trajectories and system dynamics. The position $\mathbf{x} = [x, y, z]^\mathsf{T}$ in $\mathbb{R}^3$ of the quadrotor can be defined as a differentially flat output as described in [11]. The associated velocity $\mathbf{v}$, acceleration $\mathbf{a}$ and jerk $\mathbf{j}$ can be obtained by taking derivatives with respect to time as $\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}}$ respectively. The desired trajectory for the geometric SE(3) controller as described in [25] can be written as $\Phi(t) = [\mathbf{x}_d^\mathsf{T}, \mathbf{v}_d^\mathsf{T}, \mathbf{a}_d^\mathsf{T}, \mathbf{j}_d^\mathsf{T}]^\mathsf{T}$. According to [26], we assume the force and angular velocity are our control inputs to the quadrotor. Ignoring feedback control errors, the desired mass-normalized force in the inertial frame can be obtained as

$$\mathbf{f}_d = \mathbf{a}_d + g\mathbf{z}_w. \tag{1}$$

where g is the gravitational acceleration and $\mathbf{z}_w = [0, 0, 1]^\mathsf{T}$ is the z-axis of the inertial world frame. Similar to [25], given a specific yaw ψ, the desired orientation in SO(3) can be written as $\mathbf{R}_d = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ where

$$\mathbf{r}_3 = \mathbf{f}_d / \|\mathbf{f}_d\|, \quad \mathbf{r}_1 = \frac{\mathbf{r}_{2c} \times \mathbf{r}_3}{\|\mathbf{r}_{2c} \times \mathbf{r}_3\|}, \quad \mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1 \tag{2}$$

and

$$\mathbf{r}_{2c} = [-\sin\psi, \cos\psi, 0]^\mathsf{T}. \tag{3}$$

which is assumed to be not parallel to $\mathbf{r}_3$. The associated angular velocity in the inertial frame, $\dot{\mathbf{R}}_d = [\dot{\mathbf{r}}_1, \dot{\mathbf{r}}_2, \dot{\mathbf{r}}_3]$, can be calculated as

$$\begin{aligned}
\dot{\mathbf{r}}_3 &= \mathbf{r}_3 \times \frac{\dot{\mathbf{f}}_d}{\|\mathbf{f}_d\|} \times \mathbf{r}_3, \\
\dot{\mathbf{r}}_1 &= \mathbf{r}_1 \times \frac{\dot{\mathbf{r}}_{2c} \times \mathbf{r}_3 + \mathbf{r}_{2c} \times \dot{\mathbf{r}}_3}{\|\mathbf{r}_{2c} \times \mathbf{r}_3\|} \times \mathbf{r}_1, \\
\dot{\mathbf{r}}_2 &= \dot{\mathbf{r}}_3 \times \mathbf{r}_1 + \mathbf{r}_3 \times \dot{\mathbf{r}}_1
\end{aligned} \tag{4}$$

where

$$\dot{\mathbf{r}}_{2c} = [-\cos\psi, -\sin\psi, 0]^\mathsf{T}\dot{\psi}, \quad \dot{\mathbf{f}}_d = \mathbf{j}_d^\mathsf{T}. \tag{5}$$

Therefore, the desired angular velocity $\mathbf{w}_d$ in body frame is obtained as:

$$[\mathbf{w}_d]_\times = \mathbf{R}_d^\mathsf{T}\dot{\mathbf{R}}_d. \tag{6}$$

Once the desired force $\mathbf{f}_d$, orientation $\mathbf{R}_d$ and angular velocity $\mathbf{w}_d$ are defined, it is straightforward to compute the desired control inputs for the quadrotor system. Notice that: 1) orientation is algebraically related to the desired acceleration and gravity and 2) angular velocity is algebraically related to the desired jerk.

B. Search-Based Planning Using Motion Primitives

As mentioned in the previous section, the desired trajectory can be defined as

$$\Phi(t) := [\mathbf{x}^\mathsf{T}, \dot{\mathbf{x}}^\mathsf{T}, \ddot{\mathbf{x}}^\mathsf{T}, \dddot{\mathbf{x}}^\mathsf{T}]^\mathsf{T} = [\mathbf{x}^\mathsf{T}, \mathbf{v}^\mathsf{T}, \mathbf{a}^\mathsf{T}, \mathbf{j}^\mathsf{T}]^\mathsf{T} \tag{7}$$

and each component of $\Phi(t)$ can be represented by a polynomial parameterized in time t. Position can be defined as

$$\mathbf{x}(t) := \sum_{k=0}^{K} \mathbf{d}_k \frac{t^k}{k!} = \mathbf{d}_K \frac{t^K}{K!} + \ldots + \mathbf{d}_1 t + \mathbf{d}_0 \tag{8}$$

where $\mathbf{d}_k \in \mathbb{R}^3$ are the coefficients. The corresponding velocity, acceleration and jerk can be obtained by taking the derivative of (8). A polynomial trajectory from one state to the other within a specified time duration is called a *motion primitive*. Our approach uses primitives generated as the solutions to an optimal control problem [9] to build a graph from an initial state to a goal state and search for the optimal sequence of primitives. Technical details and proof of optimality can be found in our previous work [9]. In this letter, we give the explicit solution for generating the optimal trajectory using jerk as the control input.

We define the state

$$\mathbf{s}(t) := [\mathbf{x}(t)^\mathsf{T}, \dot{\mathbf{x}}(t)^\mathsf{T}, \ddot{\mathbf{x}}(t)^\mathsf{T}]^\mathsf{T} = [\mathbf{p}^\mathsf{T}, \mathbf{v}^\mathsf{T}, \mathbf{a}^\mathsf{T}]^\mathsf{T} \tag{9}$$

as a subset of the trajectory $\Phi(t)$ that excludes the jerk. From an initial state $\mathbf{s}_0 = [\mathbf{p}_0^\mathsf{T}, \mathbf{v}_0^\mathsf{T}, \mathbf{a}_0^\mathsf{T}]^\mathsf{T}$, we apply a constant jerk input $\mathbf{u}_m$ from a pre-defined control set $\mathcal{U}_M$ for a short duration $\tau > 0$. The resulting curve between $\mathbf{s}_0$ and the end state is a motion primitive such that for $t \in [0, \tau]$ the system state $\mathbf{s}(t)$ can be written as

$$\mathbf{s}(t) = F(\mathbf{u}_m, \mathbf{s}_0, t) := \begin{bmatrix} \mathbf{u}_m \frac{t^3}{6} + \mathbf{a}_0 \frac{t^2}{2} + \mathbf{v}_0 t + \mathbf{p}_0 \\ \mathbf{u}_m \frac{t^2}{2} + \mathbf{a}_0 t + \mathbf{v}_0 \\ \mathbf{u}_m t + \mathbf{a}_0 \end{bmatrix}. \tag{10}$$

It has been shown in [18] and [9] that $F(\cdot)$ provides the minimum jerk trajectory between $\mathbf{s}_0$ and $\mathbf{s}(\tau)$.

The finite control input set $\mathcal{U}_M$ and duration τ define a graph $\mathcal{G}(\mathcal{S}, \mathcal{E})$, where $\mathcal{S}$ is the set of reachable states in $\mathbb{R}^9$ and $\mathcal{E}$ is the set of edges connecting those states. The states in $\mathcal{S}$ are generated by applying each element of $\mathcal{U}_M$ at each state iteratively, and each element in $\mathcal{E}$ is a primitive as defined in (10). A breadth-first-search (BFS) of a finite horizon leads to the graphs shown in Fig. 2.

We are interested in finding a trajectory from $\mathbf{s}_0$ to $\mathbf{s}_g$ that is optimal in terms of total control effort J and time T taken to reach the goal. According to [9], a desired optimal trajectory is obtained as

$$\Phi^*(t) = \underset{\Phi(t)}{\arg\min}\, J + \rho T = \underset{\Phi(t)}{\arg\min} \int_0^T \|\mathbf{j}\|^2 dt + \rho T$$
$$\text{s.t. } \mathbf{s}_0 \leftarrow \Phi(0),\ \mathbf{s}_g \leftarrow \Phi(T) \tag{11}$$

where ρ is the weight that decides the trade-off between effort and time.

For the primitive defined in (10), $J = \|\mathbf{u}_m\|^2 \tau$ and $T = \tau$. Thus, the cost of a primitive of applying $\mathbf{u}_m$ from state $\mathbf{s}_n \in \mathcal{S}$ is defined as

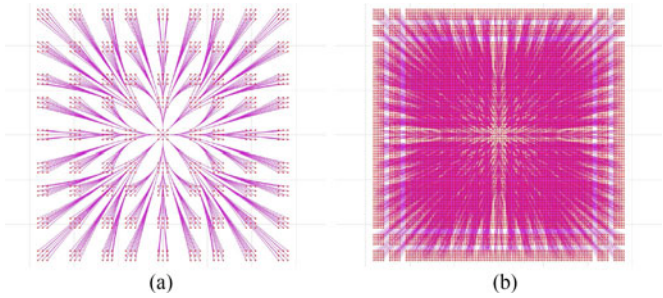

Fig. 2. Graph $\mathcal{G}(\mathcal{S}, \mathcal{E})$ generated by applying BFS for a finite planning horizon over a set of motion primitives $\mathcal{U}_M$ with 9 elements (a) and 25 elements (b). Red dots represent states in $\mathcal{S}$ and magenta splines represent edges in $\mathcal{E}$. (a) $\tau = 0.5$, $|\mathcal{U}_M| = 9$, (b) $\tau = 0.5$, $|\mathcal{U}_M| = 25$.

$$C(\mathbf{s}_n, \mathbf{u}_m) = C(\mathbf{u}_m) = (\|\mathbf{u}_m\|^2 + \rho)\tau. \tag{12}$$

The cost of the individual primitive is independent of the current state and only depends on the set $\mathcal{U}_m$ and τ. In addition, it can be shown by Pontryagin' minimum principle that (10) is the optimal solution of (11). Details of the proof can be found in [9]. Therefore, search for an optimal trajectory of (11) is equivalent to find the optimal solution to the following deterministic shortest path problem:

Problem 1: Given an initial state $\mathbf{s}_0$, a goal region $\mathcal{X}^{goal}$, a free space $\mathcal{X}^{free}$ and motion primitives based on a finite set of control inputs $\mathcal{U}_M$ with duration $\tau > 0$, choose a sequence of control inputs $\mathbf{u}_{0:N-1}$ of length N such that:

$$\min_{N, \mathbf{u}_{0:N-1}} \left(\sum_{n=0}^{N-1} \|\mathbf{u}_n\|^2 + \rho N \right) \tau$$
$$\text{s.t. } F_n(t) := F(\mathbf{u}_n, \mathbf{s}_n, t),\ \mathbf{u}_n \in \mathcal{U}_M$$
$$\mathbf{s}_{n+1} = F_n(\tau) = F_{n+1}(0),\ \mathbf{s}_N \in \mathcal{X}^{goal}$$
$$F_n(t) \subset \mathcal{X}^{free} \tag{13}$$

We are able to solve this problem through a graph search algorithm like A*. The optimal trajectory $\Phi^*(t)$ can be recovered by applying the optimal control solution $\mathbf{u}^*_{0:N-1}$ with (10) from the start $\mathbf{s}_0$ as

$$\Phi^*(t) \leftarrow [\mathbf{s}_0 \xrightarrow{\mathbf{u}_0^*} \mathbf{s}_1 \ldots \xrightarrow{\mathbf{u}_{N-1}^*} \mathbf{s}_N]. \tag{14}$$

When planning dynamic trajectories, traditional distance-based heuristics are not effective since short-distance trajectories may require sudden changes in velocity, acceleration or orientation. Instead, we use a heuristic, proposed in [9], which is based on the solution of a *Linear Quadratic Minimum Time* (LQMT) problem and takes trajectory smoothness into account. Given the current state $\mathbf{s}$ and the goal state $\mathbf{s}_g$, the LQMT solution provides an explicit formula for the $H(\mathbf{s}, \mathbf{s}_g)$ as described in the appendix.

C. Feasibility Checking

When checking if a motion primitive is contained in the free space $\mathcal{X}^{free}$ in Problem 1, we need to consider both dynamical

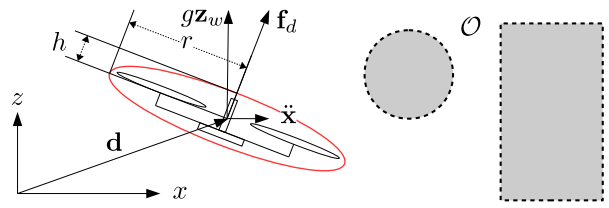

Fig. 3. A quadrotor can be modeled as an ellipsoid with radius r and height h. Its position and attitude can be estimated from the desired trajectory. A point cloud $\mathcal{O}$ is used to represent obstacles.

constraints that arise from system dynamics and geometric constraints due to physical obstacles.

1) Dynamically Feasible Primitives: The dynamical constraints on a quadrotor system are the min/max thrust and torques that can be provided by the motors [18]. However, it is hard to examine the true specification for each quadrotor and apply correct non-linear constraints. In fact, it is reasonable to utilize the property of differential flatness and apply velocity, acceleration, and jerk constraints on each axis independently. This leads to componentwise inequalities of the form:

$$|\dot{\mathbf{x}}(t)| \preceq \mathbf{v}_{\max},\ |\ddot{\mathbf{x}}(t)| \preceq \mathbf{a}_{\max},\ |\dddot{\mathbf{x}}(t)| \preceq \mathbf{j}_{\max}. \tag{15}$$

Polynomial expressions for $\dot{\mathbf{x}}, \ddot{\mathbf{x}}, \dddot{\mathbf{x}}$ allow us to check (15) in *closed-form* for each axis by finding the min/max value on time interval $[0, \tau]$. The latter is equivalent to finding the roots of the corresponding derivatives. Thus, we can guarantee that the planned trajectories always stay within the bounds $\mathbf{v}_{\max}, \mathbf{a}_{\max}, \mathbf{j}_{\max}$. More specifically, we define

$$\mathbf{v}_{\max} = v_{\max}\mathbf{1},\ \mathbf{a}_{\max} = a_{\max}\mathbf{1},\ \mathbf{j}_{\max} = j_{\max}\mathbf{1} \tag{16}$$

2) Collision Free Primitives: As indicated in Section I, traditional collision checking though inflating obstacles is too conservative and not suitable for planning agile trajectories in cluttered environments since it fails to take the actual robot shape and attitude into account. In this letter, we model the quadrotor as an ellipsoid ξ in $\mathbb{R}^3$ with radius r and height h and the obstacle map as a point cloud $\mathcal{O} \subset \mathbb{R}^3$ (see Fig. 3). Given a quadrotor state $\mathbf{s}$, its body configuration ξ at $\mathbf{s}$ can be obtained as

$$\xi(\mathbf{s}) := \{\mathbf{p} = \mathbf{E}\tilde{\mathbf{p}} + \mathbf{d} \mid \|\tilde{\mathbf{p}}\| \leq 1\} \tag{17}$$

where

$$\mathbf{d} = \mathbf{x}(t),\ \mathbf{E} = \mathbf{R}\begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & h \end{bmatrix}\mathbf{R}^{\mathsf{T}} \tag{18}$$

and the orientation $\mathbf{R}$ can be calculated from $\ddot{\mathbf{x}}(t)$ and gravity as shown in (2).

Checking whether the quadrotor hits obstacles while following a trajectory is equivalent to checking if there is any obstacle inside the ellipsoid along the trajectory. In other words, we need to verify that the intersection between ξ and the point cloud $\mathcal{O}$ is empty:

$$\mathcal{O} \cap \xi = \{\mathbf{o} \mid \|\mathbf{E}^{-1}(\mathbf{o} - \mathbf{d})\| \leq 1,\ \forall \mathbf{o} \in \mathcal{O}\} = \varnothing \tag{19}$$

Instead of checking through every point in $\mathcal{O}$, it is more efficient to use *KD-tree* [27] to crop a subset $\mathcal{O}_{r,\mathbf{d}}$ of $\mathcal{O}$ at first and then check the intersection between ξ and obstacles inside $\mathcal{O}_{r,\mathbf{d}}$. The subset $\mathcal{O}_{r,\mathbf{d}}$ is created by looking for neighbor points around $\mathbf{d}$ within radius r, assuming $r \geq h$.

Since the contour of an ellipsoid following a primitive is not convex, we sample I states in time along a primitive F_n and consider the primitive F_n collision-free if

$$\mathcal{O} \cap \xi(\mathbf{s}_{i,n}) = \varnothing,\ \forall i = \{0, 1, \ldots, I-1\} \tag{20}$$

where $\mathbf{s}_{i,n}$ is the i-th sampled state on F_n.

In sum, the explicit formulation of the feasibility constraints $F_n(t) \subset \mathcal{X}^{free}$ in Problem 1 is written as:

$$\begin{aligned} F_n(t) &\preceq [\mathbf{v}_{\max}^{\mathsf{T}}, \mathbf{a}_{\max}^{\mathsf{T}}, \mathbf{j}_{\max}^{\mathsf{T}}]^{\mathsf{T}}, \\ \mathcal{O} \cap \xi(\mathbf{s}_{i,n}) &= \varnothing,\ \forall i = \{0, 1, \ldots, I\}. \end{aligned} \tag{21}$$

IV. Trajectory Refinement

In the proposed planning approach, the dimension of the state space increases with increasing requirements on the continuity of the final trajectory. More precisely, if C^2 continuity is required for the final trajectory, jerk should be used as a control input and the state space of the associated second order system would be $\mathbb{R}^9$ (position, velocity acceleration). Generally, planning in higher dimensional spaces (e.g., snap input) requires more time and memory to explore and store lattices/states. In this section, we introduce a hierarchical approach to planning a feasible trajectory in high dimensional space by utilizing guidance from a trajectory planned in lower dimensional space. We show that the overall computation time of this hierarchical planning is shorter than the total time it takes to plan a optimal trajectory directly. Due to the fact that the final trajectory is calculated from a trajectory in lower dimensional space, similar to the refinement process in [9], we call this hierarchical planning process as trajectory refinement.

A. Trajectories Planned in Different Control Spaces

Denote the trajectories planned using velocity, acceleration or jerk inputs as $\Phi^j, j = 1, 2, 3$ respectively. Given the same start and goal, dynamics constraints and discretization, examples of the optimal trajectories in each case are plotted in Fig. 4, where the control effort $J^j, j = 1, 2, 3$ of the whole trajectory is measured as

$$J^j = \int_0^T \|\mathbf{x}^{(j)}\|^2 dt. \tag{22}$$

Denote the execution and computation time of the trajectory as T^j and $t^j, j = 1, 2, 3$ accordingly. From the planning results in Fig. 4, two conclusions can be drawn with increasing j:

1) The execution time increases, i.e., $T^1 < T^2 < T^3$;
2) The computation time increases, i.e., $t^1 < t^2 < t^3$.

Note that the computation time increases dramatically as j increases.

B. Using Trajectories as Heuristics

The fact that searching a optimal trajectory in the lower dimensional space is much faster than in a higher dimensional space leads to the approach described in this subsection to speed up the planning speed for the actual MAV system.

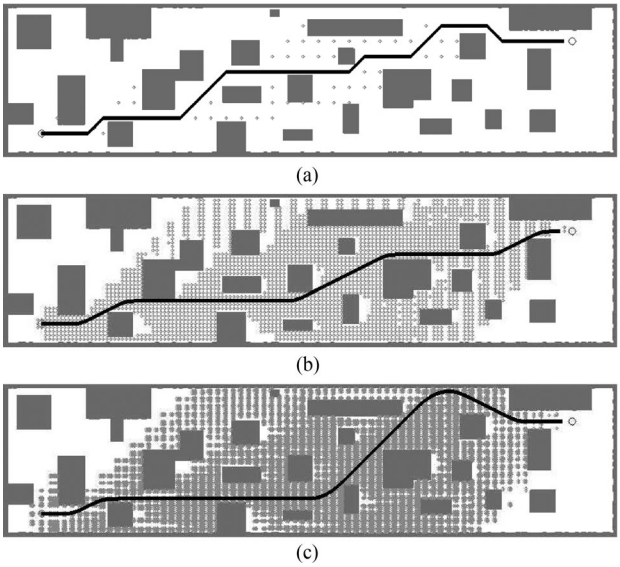

Fig. 4. Optimal trajectories planned using piecewise constant (a) velocity, (b) acceleration, (c) jerk from a start (blue dot) to a goal (red dot) state. Grey dots indicates explored states. (a) $\Phi^1 : T^1 = 32$ s, $J^1 = 42, t^1 = 2$ ms. (b) $\Phi^2 : T^2 = 33$ s, $J^2 = 2.25, t^2 = 60$ ms. (c) $\Phi^3 : T^3 = 34$ s, $J^3 = 3.75, t^3 = 1646$ ms.

Denote the prior trajectory in lower dimensional space as Φ^p, we are searching for a trajectory in higher dimensional space Φ^q $(q > p)$. Assume the duration of each primitive in Φ^q is τ, each lattice $\mathbf{s}_n^q$ in the graph is associated with a time T_n which is the minimum time it takes from the start to the current lattice. T_n is an integer multiplication of τ. Instead of calculating the heuristic $H(\mathbf{s}_n^q)$ from current state $\mathbf{s}_n^q$ to the goal $\mathbf{s}_g$ directly as described in [9], we propose to use the intermediate goal $\mathbf{s}_n^p = \Phi^p(T_n)$ evaluated from trajectory Φ^p at T_n such that the heuristic value is calculated as below:

$$H(\mathbf{s}_n^q, \Phi^p) = H_1(\mathbf{s}_n^q, \mathbf{s}_n^p) + H_2(\mathbf{s}_n^p, \mathbf{s}_g). \tag{23}$$

The first term $H_1(\cdot)$ on the RHS of (23) is proposed in the appendix where $\mathbf{s}_n^q$ is fully defined but $\mathbf{s}_n^p$ has undefined states. The second term $H_2(\cdot)$ is given directly as the cost from $\mathbf{s}_n^p$ to the goal by following Φ^p, to be more specific:

$$H_2(\mathbf{s}_n^p, \mathbf{s}_g) = J^q(\mathbf{s}_n^p, \mathbf{s}_g) + \rho(T^p - T_n) \tag{24}$$

where T^p is execution time of Φ^p and $J^q(\mathbf{s}_n^p, \mathbf{s}_g)$ is the control effort from $\mathbf{s}_n^p$ to $\mathbf{s}_g$ along Φ^p as expressed in (22). This formulation is consistent with the cost function defined before in (11). As the prior trajectory is in the lower dimensional space, J^q for Φ^p is always zero (e.g., for planning a optimal trajectory Φ^2 that uses acceleration input, the corresponding control efforts of Φ^1 is zero as there is no acceleration along Φ^1). Thus $H_2(\cdot)$ turns out to be only the execution time between $\mathbf{s}_n^p$ and goal:

$$H_2(\mathbf{s}_n^p, \mathbf{s}_g) = \rho(T^p - T_n). \tag{25}$$

Fig. 5 shows an example of applying (23) to search a trajectory Φ^2 using acceleration with a prior trajectory Φ^1 planned using velocity. The new trajectory Φ^2 tends to stick with the prior trajectory Φ^1 due to the effect of $H_1(\cdot)$. $H_2(\cdot)$ will push the searching moving forward towards the goal. In fact, the heuristic function defined in (23) is not admissible since it may not necessarily be the under-estimation of the actual cost-to-goal.

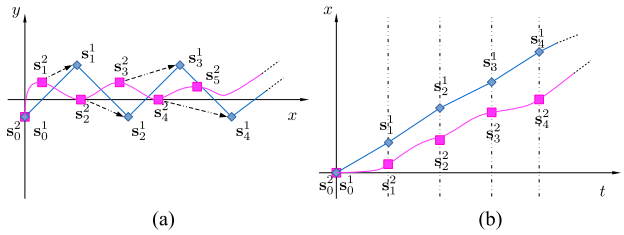

Fig. 5. Search Φ^2 (magenta) using Φ^1 (blue) as the heurisric. Left figure plots the trajectories in $x - y$ plane, the black arrows indicate the H_1. Right figure shows the corresponding x position with respect to time t along each trajectory, for states with the same subscript, they are at the same time T_n. (a) $x - y$ plot. (b)$t - x$ plot.

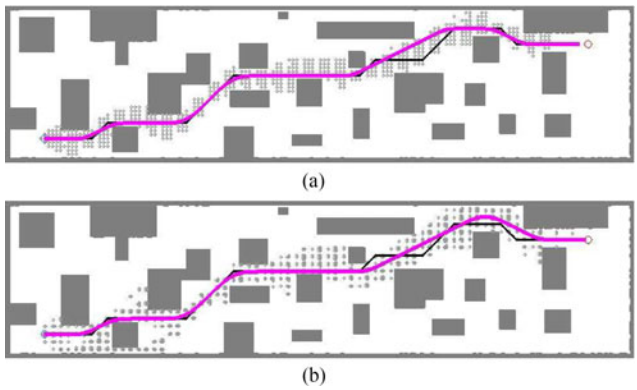

Fig. 6. Trajectories (magenta) planned using Φ^1 (black) as the heuristic. The computation time t^q and the number of expanded nodes are much less than the searching results in Fig. 4. (a) $\Phi^2 : T^2 = 35$ s, $J^2 = 3.0, t^2 = 11$ ms. (b) $\Phi^3 : T^3 = 36$ s, $J^3 = 4.25, t^3 = 98$ ms.

However, we are able to search for trajectories in higher dimensional space in a much faster speed by searching the neighboring regions of the given trajectory instead of exploring the whole state space with the same priority.

The results of applying (23) for the same planning tasks in Fig. 4 are shown in Fig. 6, in which Φ^1 is used to plan for both trajectory Φ^2 and Φ^3. Comparing Figs. 6 to 4, the total cost of control effort and execution time, namely $J^q + \rho T^q$, of the new trajectories Φ^q in Fig. 6 are greater than the optimal trajectories in Fig. 4, but the computation time t^q and the number of expanded nodes are much less.

V. Evaluation

A. 2-D Planning

2-D planning is efficient and useful in 2.5-D environments where the obstacles are vertical to the floor. We start by showing 2-D planning tasks of flying though gaps with different widths. In Fig. 7 shows how planned trajectories Φ^3 using jerk as a control input vary as the gap in a wall is shrinking (left wall moves closer to the right wall from (a) to (f)). Accordingly, the angle of the desired roll at the gap ϕ_{gap} increases. Assume the robot has radius $r = 0.35$ m, height $h = 0.1$ m, and the maximum acceleration in each axis is $a_{\max} = g$. Denoting the roll along trajectory as ϕ, according to (1) and (2), we have

$$-\arctan\frac{a_{\max}}{g} \leq \phi \leq \arctan\frac{a_{\max}}{g} \tag{26}$$

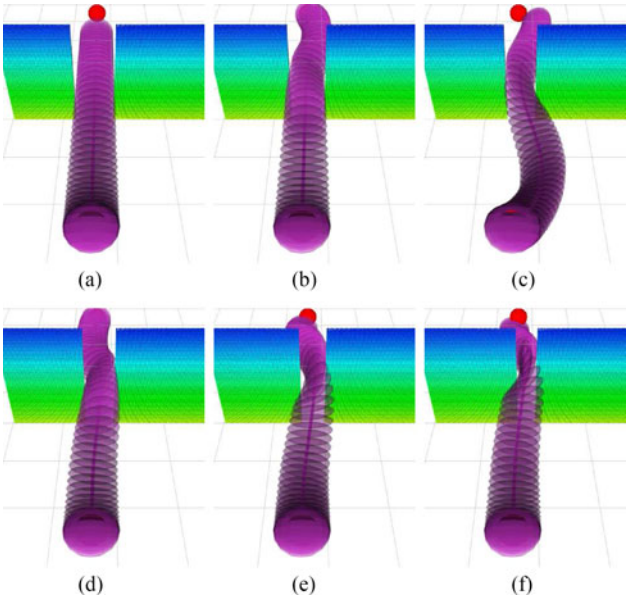

Fig. 7. Trajectories through gaps with different widths: $0.75, 0.65,$ 0.55 m from (a) to (c) (2-D planning) and $0.55, 0.45,$ 0.35 m from (d) to (f) (3-D planning). ϕ_{gap} indicates the maximum roll at the gap. Red dots show the start and goal. (a) $\phi_{gap} = 0°$. (b) $\phi_{gap} = 27°$. (c) $\phi_{gap} = 45°$. (d) $\phi_{gap} = 46°$. (e) $\phi_{gap} = 73°$. (f) $\phi_{gap} = 90°$.

since the desired acceleration in z-axis is zero. In other words, the smallest gap that the robot can pass through using 2-D planning is approximately equal to $2r \cos\theta$ (which is approximately equal to 0.525 m).

B. 3-D Planning

By adding control in the z-axis, we are able to plan in 3-D space and relax the constraint in (26) as follows:

$$-\arctan\frac{a_{\max}}{g - a_{\max}} \leq \phi \leq \arctan\frac{a_{\max}}{g - a_{\max}}. \quad (27)$$

When $a_{\max} \geq g$, $\phi \in (-\frac{\pi}{2}, \frac{\pi}{2}]$ can be arbitrary. Letting $a_{\max} = g$, we are able to reduce the gap width even more as shown in the following Fig. 7.

Another example of 3-D planning using a window with a rectangular hole in the middle is considered. By modifying the window's inclination ϕ_{win}, we are able to verify the planner's capability to generate trajectories as shown in Fig. 8.

C. Parameters

There are a few parameters that significantly affect the planning performance including computation time, resolution completeness, continuity and dynamics constraints. In this section, we analyze these relationships and provide a rough guidance on how to set the parameters in our planner. In the above examples of 2-D and 3-D planning, we used the following settings (here the control input is defined as jerk such that $u_{\max} = j_{\max}$):

ρ	τ	$v_{\max}$	$a_{\max}$	$u_{\max}$	du
10000	0.2 s	7 m/s	10 m/s^2	50 m/s^3	12.5 m/s^3

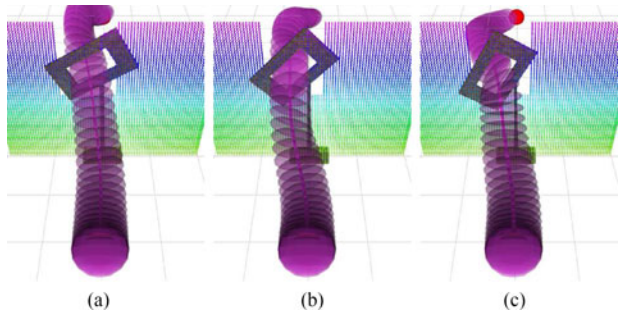

Fig. 8. Trajectories generated through a rectangular hole of size 0.4×0.8 m oriented at different angles. A robot with radius $r = 0.35$ m needs to fly through the hole with certain non-zero roll and pitch angles. The colored dots represent walls in the map that invalidate trajectories that go around the window. (a) $\phi_{win} = 30°$. (b) $\phi_{win} = 45°$. (c) $\phi_{win} = 60°$.

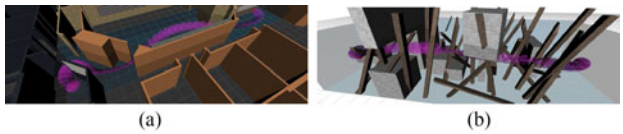

Fig. 9. Generated trajectories in two different environments. The robot radius is $r = 0.5$ m, making its diameter much larger than the door width in (a). If the obstacles in these environments are inflated by r, no feasible paths exist. (a) Office environment. (b) Unstructured environment.

A larger ρ results in faster trajectories. The scale of ρ should be comparable to the scale of the associated control effort. Here we use $\rho \approx 4u_{\max}^2$. The motion primitive duration τ decided the density of the lattices and computation time, for moderate flight speed (<10 m/s), we find $\tau = 0.2$ s to be a reasonable choice. A small τ makes the graph dense and requires more explorations to reach the goal, while a large τ may easily result in searching failure since the graph may be too sparse to cover the feasible region. The discretization in the control space $\mathcal{U}_M$ also affects the density of the graph as shown in Fig. 2. Its effect is similar to τ – finer discretization in $\mathcal{U}_M$ leads to a slower but more complete search and smoother trajectories and vice versa. The maximum velocity and acceleration are limited by the system's dynamics including thrust-to-weight ratio, max angular speed and air drag etc, but in many cases, we also want to limit the agility due to the space, state estimation and control limitations.

VI. Experiments

A. Simulation Results

The proposed planner is used to generate trajectories in complicated environments as shown in Fig. 9. A geometric model of the environment is converted into a point cloud and used to construct an obstacle *KD-tree* with 5 cm resolution.

In general, finding the optimal trajectories in complicated environments like Fig. 9 is slow (Table I gives the computation time of trajectory planning on a moderate fast computer with an Intel i7 processor with clock rate of 3.4 GHz.). As proposed in Section IV, we plan trajectories Φ^2 using acceleration control at first, based on which we plan the trajectory Φ_*^3 using jerk control. As shown in Table I, the computation time for hierarchical planning is much less than that for planning in the original 9 dimensional space with jerk input. We can also see in Fig. 10

TABLE I
EVALUATION: t REFERS TO THE COMPUTATION TIME, J IS THE TOTAL CONTROL (JERK) EFFORT AND T IS THE TOTAL TRAJECTORY EXECUTION TIME

	Office			Unstructured 3-D		
	t(s)	$J(\times 10^3)$	T(s)	t(s)	$J(\times 10^3)$	T(s)
Φ^3	89.42	8.9	4.6	129.58	5.6	3.0
Φ^2	9.34	0	4.4	21.64	0	3.6
Φ^3_*	2.03	11.1	5.0	24.02	15.1	4.8

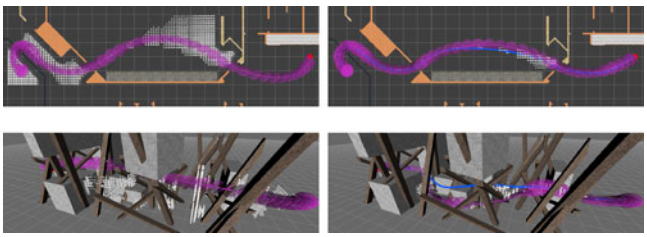

Fig. 10. Comparison between the optimal method (left) and refinement (right). The prior trajectory Φ^2 is plotted in blue, while the white dots indicate explored states. It is clear that the refinement explores fewer irrelevant regions but the generated trajectory is suboptimal.

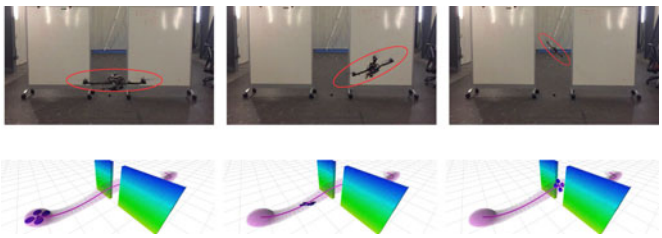

Fig. 11. Quadrotor tracks the planned trajectory to fly through a narrow gap. Top figures are the snapshots of the video, bottom figures are corresponding visualizations in ROS. Maximum roll angle at the gap is 40° as drawn in the top right figure.

that the refinement process tends to explore fewer states. As expected, the refined trajectory Φ^3_* has a higher cost compared to the optimal trajectory Φ^3.

B. Real World Experiments

The experiments is aiming to demonstrate the feasibility of planned trajectories on a real robot. We use AscTec Hummingbird as our quadrotor platform, we also use VICON motion capture system to localize the quadrotor and the obstacle map is obtained by depth sensor in advance to generate trajectories. The robot is able to avoid hitting obstacles by following the the control commands from extracting from the planned trajectory through wireless. Fig. 11 shows the flight when the quadrotor needs to roll aggressively in order to pass through the gap between white boards.

The control errors in velocity and roll are plotted in Fig. 12. The commanded roll includes the feedback attitude errors such that it is not as smooth as the desired roll from the planned trajectory. The existed lag in the attitude is due to the fact that the actual robot is not able to achieve specified angular velocity instantly, however for a moderate angular speed, this assumption still holds valid. A more accurate model for the quadrotor should be using snap as the control input instead of the jerk. The

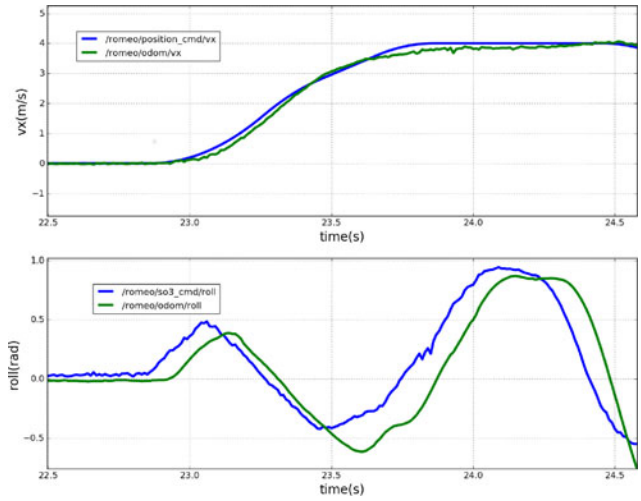

Fig. 12. Plots of control errors, the blue curve is the command value while the green curve shows the actual robot state. Top figure shows $v_x - t$, bottom figure shows $\phi - t$. The red verticle line indicates the time when the robot pass through the gap.

trajectory planned using the snap as the control input is straightforward to solve following the same pipeline as proposed in this letter, which has also been implemented in our open-sourced planner.

VII. CONCLUSION

In this work, we extend our previous motion-primitive-based planning algorithm [9] to enable aggressive flight with attitude constraints in cluttered environments for an under-actuated quadrotor system. We also presented a hierarchical refinement process that uses prior lower-dimensional trajectories to accelerate planning in higher dimensions. We believe that the proposed algorithm can be a foundation for future study of fast autonomous navigation of UAVs in cluttered environments.

APPENDIX
LINEAR QUADRATIC MINIMUM TIME FOR JERK CONTROL

The heuristic function $H(\mathbf{s}, \mathbf{s}_g)$ for graph search is an underestimation of actual cost from the state $\mathbf{s}$ to the goal $\mathbf{s}_g$ by relaxing the dynamics and obstacles constraints. We try to find a state-to-state optimal trajectory of Problem 2, whose cost serves as the heuristic H. The explicit solution for the optimal cost for velocity, acceleration control has been shown in [9], here we show the explicit solution for jerk control.

Problem 2: Given a current state $\mathbf{s}$, the goal state $\mathbf{s}_g$, find the optimal trajectory according to the cost function

$$c \min_{\mathbf{j},T} \int_0^T \|\mathbf{j}\|^2 dt + \rho T \tag{28}$$

Assume the initial state is given as $\mathbf{s} = [\mathbf{p}_0^\mathsf{T}, \mathbf{v}_0^\mathsf{T}, \mathbf{a}_0^\mathsf{T}]^\mathsf{T}$, the formulation of position of the optimal trajectory for (28) is given from the Pontryagin's minimum principle [18] as

$$\mathbf{p} = \frac{\mathbf{d}_5}{120}t^5 + \frac{\mathbf{d}_4}{24}t^4 + \frac{\mathbf{d}_3}{6}t^3 + \frac{\mathbf{a}_0}{2}t^2 + \mathbf{v}_0 t + \mathbf{p}_0 \tag{29}$$

The coefficients $\{\mathbf{d}_5, \mathbf{d}_4, \mathbf{d}_3\}$ are defined in [18] by $\mathbf{s}, \mathbf{s}_g$ and T. As a result, the total cost of (28) can be written as a function of time T as

$$\begin{aligned}\mathcal{C}(T) &= \int_0^T \left(\frac{\mathbf{d}_5}{2}t^2 + \mathbf{d}_4 t + \mathbf{d}_3\right)^2 dt + \rho T \\ &= \frac{\mathbf{d}_5^2}{20}T^5 + \frac{\mathbf{d}_4^\mathsf{T}\mathbf{d}_5}{4}T^4 + \left(\frac{\mathbf{d}_4^\mathsf{T}\mathbf{d}_4}{3} + \frac{\mathbf{d}_3^\mathsf{T}\mathbf{d}_5}{3}\right)T^3 \\ &\quad + \mathbf{d}_3^\mathsf{T}\mathbf{d}_4 T^2 + \mathbf{d}_3^2 T + \rho T \end{aligned} \tag{30}$$

The minimum of $\mathcal{C}(T)$ can be derived by taking the derivative with respect to T and finding the root T^* of

$$\frac{d\mathcal{C}}{dT} = c_0 + \ldots + c_6 T^{-6} = 0,\ T \in [0, \infty) \tag{31}$$

Therefore, $H(\mathbf{s}, \mathbf{s}_g) = \mathcal{C}(T^*)$. The coefficients in (31) are derived as follows:

(1) Fully Defined $\mathbf{s}_g = [\mathbf{p}_1^\mathsf{T}, \mathbf{v}_1^\mathsf{T}, \mathbf{a}_1^\mathsf{T}]^\mathsf{T}$

$$\begin{aligned} c_0 &= \rho,\ c_1 = 0,\ c_2 = -9\mathbf{a}_0^2 + 6\mathbf{a}_0^\mathsf{T}\mathbf{a}_1 - 9\mathbf{a}_1^2, \\ c_3 &= -144\mathbf{a}_0^\mathsf{T}\mathbf{v}_0 - 96\mathbf{a}_0^\mathsf{T}\mathbf{v}_1 + 96\mathbf{a}_1^\mathsf{T}\mathbf{v}_0 + 144\mathbf{a}_1^\mathsf{T}\mathbf{v}_1, \\ c_4 &= 360(\mathbf{a}_0 - \mathbf{a}_1)^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1) - 576\mathbf{v}_0^2 \\ &\quad - 1008\mathbf{v}_0^\mathsf{T}\mathbf{v}_1 - 576\mathbf{v}_1^2, \\ c_5 &= 2880(\mathbf{v}_0 + \mathbf{v}_1)^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1), \\ c_6 &= -3600(\mathbf{p}_0 - \mathbf{p}_1)^2. \end{aligned} \tag{32}$$

(2) Partially Defined $\mathbf{s}_g = [\mathbf{p}_1^\mathsf{T}, \mathbf{v}_1^\mathsf{T}]^\mathsf{T}$

$$\begin{aligned} c_0 &= \rho,\ c_1 = 0,\ c_2 = -8\mathbf{a}_0^2, \\ c_3 &= -112\mathbf{a}_0^\mathsf{T}\mathbf{v}_0 - 48\mathbf{a}_0^\mathsf{T}\mathbf{v}_1, \\ c_4 &= 240\mathbf{a}_0^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1) - 384\mathbf{v}_0^2 - 432\mathbf{v}_0^\mathsf{T}\mathbf{v}_1 - 144\mathbf{v}_1^2, \\ c_5 &= (1600\mathbf{v}_0 + 960\mathbf{v}_1)^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1), \\ c_6 &= -1600(\mathbf{p}_0 - \mathbf{p}_1)^2. \end{aligned} \tag{33}$$

(3) Partially Defined $\mathbf{s}_g = \mathbf{p}_1$

$$\begin{aligned} c_0 &= \rho,\ c_1 = 0,\ c_2 = -5\mathbf{a}_0^2, \\ c_3 &= -40\mathbf{a}_0^\mathsf{T}\mathbf{v}_0, \\ c_4 &= 60\mathbf{a}_0^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1) - 60\mathbf{v}_0^2, \\ c_5 &= 160\mathbf{v}_0^\mathsf{T}(\mathbf{p}_0 - \mathbf{p}_1), \\ c_6 &= -100(\mathbf{p}_0 - \mathbf{p}_1)^2. \end{aligned} \tag{34}$$

REFERENCES

[1] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1484–1491.

[2] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Berlin, Germany: Springer, 2016, pp. 649–666.

[3] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 5332–5339.

[4] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 42–49.

[5] J. Canny, B. Donald, J. Reif, and P. Xavier, "On the complexity of kinodynamic planning," in *Proc. 29th Annu. Symp. Found. Comput. Sci.*, 1988, pp. 306–316.

[6] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing," arXiv:1612.00291, 2016.

[7] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.

[8] T. Hirata and M. Kumon, "Optimal path planning method with attitude constraints for quadrotor helicopters," in *Proc. Int. Conf. Adv. Mechatronic Syst.*, 2014, pp. 377–381.

[9] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2872–2879.

[10] M. J. Van Nieuwstadt and R. M. Murray, "Real time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Control*, vol. 8, no. 11, pp. 995–1020, 1998.

[11] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.

[12] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.

[13] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1476–1483.

[14] J. Ichnowski and R. Alterovitz, "Fast nearest neighbor search in SE (3) for sampling-based motion planning," in *Algorithmic Foundations of Robotics XI*. Berlin, Germany: Springer, 2015, pp. 197–214.

[15] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 489–494.

[16] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, vol. 26, no. 3, pp. 308–333, 2009.

[17] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *Proc. IEEE Int. Conf. Robot. Autom*, 2013, pp. 3933–3940.

[18] M. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[20] E. A. Hansen and R. Zhou, "Anytime heuristic search," *J. Artif. Intell. Res.*, vol. 28, pp. 267–297, 2007.

[21] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 767–774.

[22] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 224–243, 2016.

[23] C. M. Wilt and W. Ruml, "When does weighted A* fail?" in *Proc. Annu. Symp. Combinatorial Search*, 2012, pp. 137–144.

[24] K. Gochev, B. Cohen, J. Butzke, A. Safonova, and M. Likhachev, "Path planning with adaptive dimensionality," in *Proc. 4th Annu. Symp. Combinatorial Search*, 2011.

[25] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *Proc. 49th IEEE Conf. Decision Control*, 2010, pp. 5420–5425.

[26] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[27] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011.