Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight

Boyu Zhou [ID], Fei Gao [ID], Luqi Wang [ID], Chuhao Liu [ID], and Shaojie Shen [ID]

Abstract—In this letter, we propose a robust and efficient quadrotor motion planning system for fast flight in three-dimensional complex environments. We adopt a kinodynamic path searching method to find a safe, kinodynamic feasible, and minimum-time initial trajectory in the discretized control space. We improve the smoothness and clearance of the trajectory by a B-spline optimization, which incorporates gradient information from a Euclidean distance field and dynamic constraints efficiently utilizing the convex hull property of B-spline. Finally, by representing the final trajectory as a non-uniform B-spline, an iterative time adjustment method is adopted to guarantee dynamically feasible and non-conservative trajectories. We validate our proposed method in various complex simulational environments. The competence of the method is also validated in challenging real-world tasks. We release our code as an open-source package.[1]

Index Terms—Motion and path planning, aerial systems: perception and autonomy, collision avoidance.

I. INTRODUCTION

UNMANNED aerial vehicles (UAVs) are recently involved in more and more applications, such as industrial inspection, search-and-rescue and package delivery. To achieve full autonomy in these scenarios, the motion planning module plays an essential role in generating safe and smooth motions.

Although plenty of works on quadrotor trajectory generation have been proposed, there are still two critical unsolved issues. Firstly, given limited time and onboard computing resources, no existing works guarantee to generate safe and kinodynamic feasible trajectory at a high success rate. However, the efficiency and robustness of the trajectory generation are essential. In many circumstances, such as a quadrotor flying at high speed in unknown environments, trajectories should be re-generated constantly in a very short time to avoid emergent threats. Secondly, to ensure the kinodynamic feasibility of the generated motions, constraints

[1]Open-source implementation is available at https://github.com/HKUST-Aerial-Robotics/Fast-Planner.

(a) Fast autonomous flight in an unknown environment. (b) Aggressive kinodynamic replans.

Fig. 1. Our proposed method tested on a fully autonomous quadrotor in (a), and on extremely challenging fast replanning with indoor external feedback in (b). Experimental details are given in Sect. VII. Video is available at https://www.youtube.com/watch?v=GIYGAjOeeI8&feature=youtu.be.

on velocity and acceleration are often enforced conservatively. As a result, the aggressiveness of the generated trajectories are often hard to be tuned to satisfy applications where a high flight speed is preferable.

In this letter, we propose a complete and robust online trajectory generation method to address these two issues systematically. A kinodynamic path searching based on heuristic search and linear quadratic minimum-time control is adopted. It searches efficiently for a safe, feasible and minimum-time initial path in the discretized control space. The initial path is then refined in a carefully designed B-spline optimization, which utilizes B-spline's convex hull property to incorporate gradient information and dynamic constraints. It improves the initial path and converges quickly to a smooth, safe and dynamically feasible trajectory. Finally, the trajectory is represented as a non-uniform B-spline, for which we investigate the relations between the control points of derivatives and time allocation. Based on the relations, an iterative time adjustment method is adopted to squeeze infeasible velocity and acceleration out from the profiles while avoiding constraining them conservatively.

Compared to existing works, our proposed method is able to generate high-quality trajectories in cluttered environments in a much shorter time with a higher success rate. It can generate aggressive motion under the premise of dynamic feasibility. We show the efficiency and robustness of our method in numerous simulational complex environments. We also demonstrate that our method is competent even for challenging fast flight when trajectories should be re-generated repeatedly in a very short time by real-world experiments. We summarize our contributions as follows:

1) We propose a robust and efficient systematic method, incorporating kinodynamic path searching, B-spline optimization and time adjustment, where safety, dynamic feasibility and aggressiveness are built from bottom-up.
2) We present an optimization formulation based on the convex hull property of B-splines that delicately incorporates

gradient information and dynamic constraints, which converges quickly to generate smooth, safe and dynamically feasible trajectories.

3) We investigate the relations between the control points of derivatives and the time allocation of non-uniform B-splines. A time adjustment method based on the relations is applied to guarantee feasible and non-conservative motion.

4) We present extensive simulation and real-world evaluation of our proposed method. The source code is released as a ros-package.

II. RELATED WORK

A. Hard-Constrained Methods

The problems of trajectory generation have been addressed by some work recently. Hard-constrained methods are pioneered by minimum-snap trajectory generation [1], in which trajectories are represented as piecewise polynomials and generated by solving a quadratic programming(QP) problem. [2] shows that minimum snap trajectories can be obtained in closed form, in which the safety of the trajectories is ensured by iteratively adding intermediate waypoints. Works [3]–[7] generate trajectories in a two-step pipeline. Free space represented by a sequence of cubes [3], [8], spheres [4], [9] or polyhedrons [5] is firstly extracted, which is followed by convex optimization, which generates smooth trajectory within the feasible space. [6], [7] proposed a B-spline-based kinodynamic search to find an initial trajectory which is then refined by an elastic band optimization approach. The use of uniform B-spline ensures dynamic feasibility but could generate conservative motion. One common drawback of these methods is that the time allocation of the trajectory is given by naive heuristics. However, a poorly chosen time allocation significantly reduce the quality of the trajectory. Besides, a feasible solution can only be obtained by iteratively adding more constraints and solving the quadratic programming problem, which is undesirable for real-time application. To address these problems, [8] proposed a method to search for a path with well-allocated time and guarantee the safety and kinodynamic feasibility of trajectory through optimization. Hard-constrained methods ensure global optimality by the convex formulation. However, distance to obstacles in the free space is ignored, which often results in trajectories being close to obstacles. Besides, the kinodynamic constraints are conservative, making the trajectory's speed deficient for fast flight.

B. Soft-Constrained Methods

There are also methods formulating trajectory generation as a non-linear optimization problem that takes smoothness and safety into account. [10] generates discrete-time trajectories by minimizing its smoothness and collision costs using gradient descent methods. [11] has similar problem formulation, but the optimization is solved by a gradient-free sampling method. [12] extended them to continuous-time polynomial trajectories. Since the time parameterization is continuous, it avoids numeric differentiation errors and is more accurate to represent the motions of quadrotors. However, it suffers from a low success rate. To solve this problem, [13] finds a collision-free initial path firstly using an informed sampling-based path searching method. This path serves as a higher quality initial guess of non-linear optimization and thus improve the success rate. In [14], the trajectory is parameterized as a uniform B-spline. Since
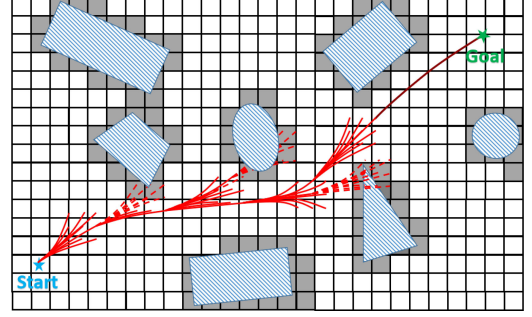

Fig. 2. An illustration of the mechanism of the kinodynamic path searching. Red curves indicate the motion primitives generated by Equ.3. The purple curve is the analytic expansion explained in Sect. III-C.

Algorithm 1: Kinodynamic Path Searching.

1 Initialize();
2 **while** $\neg \mathcal{P}$.**empty**() **do**
3      $n_c \leftarrow \mathcal{P}$.**pop**(), $\mathcal{C}$.**insert**(n_c) ;
4   **if** **ReachGoal**(n_c) $\vee$ **AnalyticExpand**(n_c) **then**
5   **return** **RetrievePath**();
6      $primitives \leftarrow$ **Expand**(n_c);
7      $nodes \leftarrow$ **Prune**($primitives$);
8   **for** n_i **in** $nodes$ **do**
9   **if** $\neg \mathcal{C}$.**contain**(n_i) $\wedge$ **CheckFeasible**(n_i) **then**
10             $g_{temp} \leftarrow n_c.g_c +$ **EdgeCost**(n_i) ;
11   **if** $\neg \mathcal{P}$.**contain**(n_i) **then**
12                 $\mathcal{P}$.**add**(n_i);
13   **else if** $g_{temp} \geq n_i.g_c$ **then**
14   continue;
15             $n_i.parent \leftarrow n_c$, $n_i.g_c \leftarrow g_{temp}$;
16             $n_i.f_c \leftarrow n_i.g_c +$ **Heuristic**(n_i);

---

a B-spline is continuous by nature, there is no need to enforce continuity explicitly, which reduce the number of constraints. It is also particularly useful for local replanning thanks to its property of locality. Soft-constrained methods utilize gradient information to push trajectory far from obstacles, but suffer from local minima and having no strong guarantee of success rate and kinodynamic feasibility. Our optimization method also utilizes gradient information to improve the safety of the trajectory. However, unlike previous methods in which computational expensive line integrals along the trajectory are calculated, the formulation is redesigned to be simpler based on the convex hull property of B-spline. It greatly improves the computation efficiency as well as the convergent rate.

III. KINODYNAMIC PATH SEARCHING

Our front-end kinodynamic path searching module is originated from the hybrid-state A* search first proposed for autonomous vehicle [15]. It searches for a safe and kinodynamic feasible trajectory that is minimal with respect to time duration and control cost in a voxel grid map. As shown in Alg.1 and in Fig. 2, the searching loop is similar to the standard A* algorithm, where $\mathcal{P}$ and $\mathcal{C}$ refer to the open and closed set. Instead of straight lines, motion primitives respecting the quadrotor dynamic are

used as graph edge. A structure *Node* is used to record a primitive, the voxel the primitives ends in and the g_c and f_c cost (Sect. III-B). Primitives **Expand**() the voxel grid map iteratively and those ending up in the same voxel except the one with the smallest f_c are pruned (**Prune**()). Then **CheckFeasible**() checks the safety and dynamic feasibility of the remained primitives. This process continues until any primitive reach goal or the **AnalyticExpand()** (Sect. III-C) succeeds.

A. Primitives Generation

We first discuss the generation of motion primitives used in **Expand**(). The differential flatness of quadrotor systems allows us to represent the trajectory by three independent 1-D time-parameterized polynomial functions [1]:

$$\mathbf{p}(t) := [p_x(t), p_y(t), p_z(t)]^\top, \quad p_\mu(t) = \sum_{k=0}^{K} a_k t^k \quad (1)$$

where $\mu \in \{x, y, z\}$. From the view of quadrotor systems, it corresponds to a linear time-invariant (LTI) system. Let $\mathbf{x}(t) := [\mathbf{p}(t)^\top, \dot{\mathbf{p}}(t)^\top, \ldots, \mathbf{p}^{(n-1)}(t)^\top]^\top \in \mathcal{X} \subset \mathbb{R}^{3n}$ be the state vector. Let $\mathbf{u}(t) := \mathbf{p}^{(n)}(t) \in \mathcal{U} := [-u_{\max}, u_{\max}]^3 \subset \mathbb{R}^3$ be the control input. The state space model can be defined as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{I}_3 \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} \quad (2)$$

The complete solution for the state equation is expressed as:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau \quad (3)$$

which gives the trajectory of the quadrotor system whose initial state is $\mathbf{x}(0)$ and control input is $\mathbf{u}(t)$.

In **Expand**(), given the current state of the quadrotor, a set of discretized control inputs $\mathcal{U}_D \subset \mathcal{U}$ is applied for duration τ. In practice, we select $n = 2$, which corresponds to a double integrator. Each axis $[-u_{\max}, u_{\max}]$ is discretized uniformly as $\{-u_{\max}, -\frac{r-1}{r}u_{\max}, \ldots, \frac{r-1}{r}u_{\max}, u_{\max}\}$, which results in $(2r+1)^3$ primitives.

B. Actual Cost and Heuristic Cost

As we aim to find a trajectory that is optimal in time and control cost, we define the cost of a trajectory as:

$$\mathcal{J}(T) = \int_0^T \|\mathbf{u}(t)\|^2 dt + \rho T \quad (4)$$

Under this definition, **EdgeCost**() calculates the cost of a motion primitive generated with the discretized input $\mathbf{u}(t) = \mathbf{u}_d$ and duration τ as $e_c = (\|\mathbf{u}_d\|^2 + \rho)\tau$.

Following the terminology of A*, we use g_c to represent the actual cost of an optimal path from the start state $\mathbf{x}_s$ to the current state $\mathbf{x}_c$. Let this optimal path consists of J primitives, g_c is calculated as: $g_c = \sum_{j=1}^{J}(\|\mathbf{u}_{dj}\|^2 + \rho)\tau$.

An admissible and informative heuristic is essential to speed up the searching as in A*. Hence, we also design a **Heuristic**(). We compute a closed form trajectory that minimizes $\mathcal{J}(T)$ from $\mathbf{x}_c$ to the goal state $\mathbf{x}_g$ by applying the Pontryagins minimum principle [16]:

$$p_\mu^*(t) = \frac{1}{6}\alpha_\mu t^3 + \frac{1}{2}\beta_\mu t^2 + v_{\mu c} + p_{\mu c}$$

$$\begin{bmatrix} \alpha_\mu \\ \beta_\mu \end{bmatrix} = \frac{1}{T^3} \begin{bmatrix} -12 & 6T \\ 6T & -2T^2 \end{bmatrix} \begin{bmatrix} p_{\mu g} - p_{\mu c} - v_{\mu c}T \\ v_{\mu g} - v_{\mu c} \end{bmatrix}$$

$$\mathcal{J}^*(T) = \sum_{\mu \in \{x,y,z\}} \left(\frac{1}{3}\alpha_\mu^2 T^3 + \alpha_\mu \beta_\mu T^2 + \beta_\mu^2 T \right) \quad (5)$$

where $p_{\mu c}, v_{\mu c}, p_{\mu g}, v_{\mu g}$ are the current and goal position and velocity. $\mathcal{J}^*(T)$ is the cost defined by Equ.4. To find the optimal time T that minimize the cost, we substitute α_μ, β_μ into $\mathcal{J}^*(T)$ and find the roots of $\frac{\partial \mathcal{J}^*(T)}{\partial T} = 0$. The root making a minimum cost $min \, \mathcal{J}^*$ and feasible trajectory is selected and denoted as T_h. We use $\mathcal{J}^*(T_h)$ as the heuristic h_c. Finally, f_c is defined as: $f_c = g_c + h_c = g_c + \mathcal{J}^*(T_h)$.

C. Analytic Expansion

Due to the discretized control input, it is difficult to have a primitive end exactly in the goal state. To compensate for it and also to speed up the searching, we induce an analytic expansion scheme. When a node is popped from the open set, a trajectory from $\mathbf{x}_c$ to $\mathbf{x}_g$ is computed using the same approach in Sect. III-B. If it passes the safety and dynamic feasibility check, the searching is terminated in advance. This strategy is effective for improving efficiency especially in sparse environments, since it has a higher success rate and terminates the searching earlier.

D. Optimality and Completeness

Theoretically, we can not guarantee the optimality and completeness of the path searching. However, the practical performance is satisfactory. For the optimality, evaluation (Sect. VII-A1) shows that the sacrifice of optimality is acceptable and adjustable. Besides, provided the initial path lies near the optimum, our optimization (Sect. IV) will find that optimum. For the completeness, evaluation (Sect. VII-A1) indicates that in practice it can find a feasible solution in most case. Also, our method can be extended to support variable-duration primitives and a variable-resolution voxel grid as described by [15] to make stronger completeness guarantees.

IV. B-SPLINE TRAJECTORY OPTIMIZATION

As mentioned in Sect. III-D, the path produced by the path searching can be suboptimal. In addition, this path is often close to obstacle since distance information in the free space is ignored (Fig. 5). Therefore, we improve the smoothness and clearance of the path in the proposed B-spline optimization. The **convex hull** property of uniform B-splines are utilized to incorporate gradient information from the Euclidean distance field and dynamic constraints, for which it converges within a very short duration to generate smooth, safe and dynamically feasible trajectories.
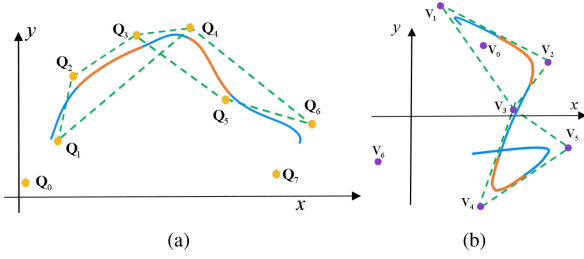
(a)   (b)

Fig. 3.   (a) A trajectory is represented by a B-spline ($p_b = 3$). Each segment is bounded by the corresponding convex hull of the control points (example convex hulls and segments are shown in green and orange). (b) The first order derivative (velocity) is also a B-spline, thus it has the same property. The control points of the derivatives can be calculated by Equ.7.

A. Uniform B-Splines

A B-spline is a piecewise polynomial uniquely determined by its degree p_b, a set of $N + 1$ **control points** $\{\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_N\}$ and a **knot vector** $[t_0, t_1, \dots, t_M]$, in which $\mathbf{Q}_i \in \mathbb{R}^3$, $t_m \in \mathbb{R}$ and $M = N + p_b + 1$. A B-spline trajectory is parameterized by time t, where $t \in [t_{p_b}, t_{M-p_b}]$. For a **uniform** B-spline, each **knot span** $\Delta t_m = t_{m+1} - t_m$ has identical value Δt. To evaluate the position at time $t \in [t_m, t_{m+1}] \subset [t_{p_b}, t_{M-p_b}]$, we first normalize t as $s(t) = (t - t_m)/\Delta t$. Then the position can be evaluated using the matrix representation [17]:

$$\mathbf{p}(s(t)) = \mathbf{s}(t)^\top \mathbf{M}_{p_b+1} \mathbf{q}_m$$

$$\mathbf{s}(t) = \begin{bmatrix} 1 & s(t) & s^2(t) & \cdots & s^{p_b}(t) \end{bmatrix}^\top$$

$$\mathbf{q}_m = \begin{bmatrix} \mathbf{Q}_{m-p_b} & \mathbf{Q}_{m-p_b+1} & \mathbf{Q}_{m-p_b+2} & \cdots & \mathbf{Q}_m \end{bmatrix}^\top \quad (6)$$

here $\mathbf{M}_{p_b+1}$ is a constant matrix determined by p_b. In our implementation, p_b is set as 3. The evaluation of the derivatives is exactly the same, since the derivative of a B-spline is also a B-spline.

The **convex hull** property of B-splines (Fig. 3) is essential for designing our optimization formulation. We show in Sect.IV-B that it is extremely useful for ensuring the dynamic feasibility and safety of the entire trajectory.

B. Convex Hull Property

The convex hull property (Fig. 3) is used extensively in our method to ensure both dynamic feasibility and safety.

For the dynamic feasibility, it suffices to constrain all velocity and acceleration control points $\{\mathbf{V}_0, \mathbf{V}_1, \dots, \mathbf{V}_{N-1}\}$ and $\{\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{N-2}\}$ so that $\mathbf{V}_i \in [-v_{\max}, v_{\max}]^3$ and $\mathbf{A}_i \in [-a_{\max}, a_{\max}]^3$. $\mathbf{V}_i$ and $\mathbf{A}_i$ are calculated by Equ.7, where Δt is the knot span:

$$\mathbf{V}_i = \frac{1}{\Delta t}(\mathbf{Q}_{i+1} - \mathbf{Q}_i), \quad \mathbf{A}_i = \frac{1}{\Delta t}(\mathbf{V}_{i+1} - \mathbf{V}_i) \quad (7)$$

For the safety of the B-spline, we need to ensure that all its convex hulls are collision-free. Equivalently, we need to ensure that $d_h > 0$, where d_h is the distance between any one occupied voxel and any one point Q_h in the convex hull (Fig. 4). By the triangle inequality, we have $d_h > d_c - r_h$, where d_c is the distance between the voxel and any one control point. We also have $r_h \leq r_{12} + r_{23} + r_{34}$, since Q_h is inside the convex hull. Combining them, $d_h > d_c - (r_{12} + r_{23} + r_{34})$ is always valid.
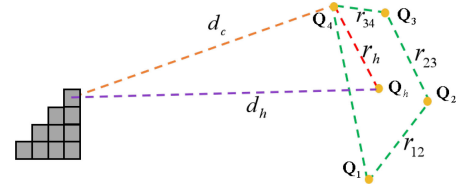

Fig. 4.   Illustration of ensuring that a convex hull of the B-spline ($p_b = 3$) is collision-free.
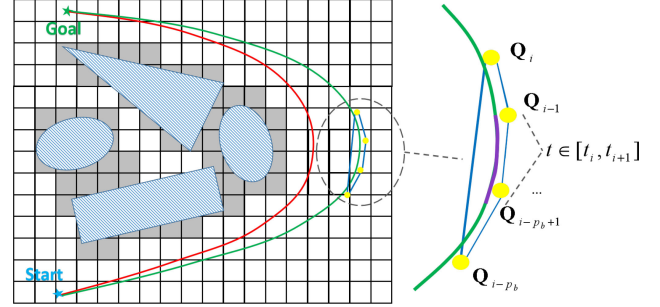

Fig. 5.   Using gradient-based numeric optimization to deform the trajectory. The red and the green curves are the initial path and the B-spline after the optimization. Yellow dots stand for the control points of the B-spline. The initial path is close to the obstacles since distance information is ignored, while the B-spline is pushed away by the gradient-based optimization.

Therefore, if we ensure:

$$d_c > 0, \quad r_{j,j+1} < d_c/3 \ (j \in \{1,2,3\}) \quad (8)$$

then the convex hull is guaranteed to be collision-free.

C. Problem Formulation

For a p_b degree B-spline trajectory defined by $N + 1$ control points $\{\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_N\}$, we optimize the subset of $N + 1 - 2p_b$ control points $\{\mathbf{Q}_{p_b}, \mathbf{Q}_{p_b+1}, \dots, \mathbf{Q}_{N-p_b}\}$. The first and last p_b control points should not be changed because they determine the boundary state. The overall cost function is defined as:

$$f_{total} = \lambda_1 f_s + \lambda_2 f_c + \lambda_3 (f_v + f_a) \quad (9)$$

where f_s and f_c are smoothness and collision cost. f_v and f_a are soft limits on velocity and acceleration. λ_1, λ_2 and λ_3 trade off the smoothness, safety and dynamic feasibility.

We define the smoothness cost f_s by a function that captures the geometric information of the trajectory and does not depend on time allocation, unlike many recent works that use integral of the squared snap or jerk. The reason is that after optimization the time allocation may be adjusted (Sect. V). This will change the derivatives of the trajectory and make the optimized snap (jerk) less meaningful. We use an elastic band cost function[2] [18], [19]:

$$f_s = \sum_{i=p_b-1}^{N-p_b+1} \| \underbrace{(\mathbf{Q}_{i+1} - \mathbf{Q}_i)}_{\mathbf{F}_{i+1,i}} + \underbrace{(\mathbf{Q}_{i-1} - \mathbf{Q}_i)}_{\mathbf{F}_{i-1,i}} \|^2 \quad (10)$$

From a physical standpoint, this formulation view a trajectory as an elastic band, where each term $\mathbf{F}_{i+1,i} = \mathbf{Q}_{i+1} - \mathbf{Q}_i$ and

---

[2]The control points $\mathbf{Q}_{p_b-2}, \mathbf{Q}_{p_b-1}, \mathbf{Q}_{N-p_b+1}$ and $\mathbf{Q}_{N-p_b+2}$ are not optimized but needed to evaluate the overall smoothness. Similarly, some of them are included in Equ.14 as constant values to evaluate the derivatives.

$\mathbf{F}_{i-1,i} = \mathbf{Q}_{i-1} - \mathbf{Q}_i$ is the joint force of two springs connecting the nodes $\mathbf{Q}_{i+1}, \mathbf{Q}_i$ and $\mathbf{Q}_{i-1}, \mathbf{Q}_i$ respectively. If all terms equal to zero, all the control points would uniformly distribute in a straight line, which is ideally smooth.

Similarly, the collision cost is formulated as the repulsive force of the obstacles acting on each control point:

$$f_c = \sum_{i=p_b}^{N-p_b} F_c(d(\mathbf{Q}_i)) \tag{11}$$

where $d(\mathbf{Q}_i)$ is the distance between $\mathbf{Q}_i$ and the closet obstacle. F_c is a differentiable potential cost function with d_{thr} specifying the threshold of obstacle clearance:

$$F_c(d(\mathbf{Q}_i)) = \begin{cases} (d(\mathbf{Q}_i) - d_{thr})^2 & d(\mathbf{Q}_i) \leq d_{thr} \\ 0 & d(\mathbf{Q}_i) > d_{thr} \end{cases} \tag{12}$$

As shown in Sect.IV-B, Equ.8 must be satisfied so that the trajectory is collision-free. Since the collision cost pushes the control points away from obstacles, $d_c > 0$ is apparent. Also, $r_{j,j+1}$ are tunable parameters depend solely on the parameterization of the B-spline. In practice, as long as we select $r_{j,j+1}, (j \in \{0, 1, \ldots, N\})$ that are significantly small (in our implementation $r_{j,j+1} < 0.2$), the trajectory is safe in most cases. This may be invalid in extreme cases, for instance, the environment is very cluttered. Even so, we can re-parameterize the B-spline to select smaller $r_{j,j+1}$, after which Equ.8 will still be satisfied.

We penalize velocity or acceleration along the trajectory exceeding maximum allowable value $v_{\max}$ and $a_{\max}$ with a cost similar to Equ.12. The penalty for 1-D velocity v_μ is:

$$F_v(v_\mu) = \begin{cases} (v_\mu^2 - v_{\max}^2)^2 & v_\mu^2 > v_{\max}^2 \\ 0 & v_\mu^2 \leq v_{\max}^2 \end{cases} \tag{13}$$

where $\mu \in \{x, y, z\}$. The acceleration penalty has identical form. Applying the convex hull property (Fig. 3), we define f_v and f_a so that infeasible velocity and acceleration control points are penalized:

$$f_v = \sum_{\substack{\mu \in \\ \{x,y,z\}}} \sum_{i=p_b-1}^{N-p_b} F_v(V_{i\mu}), \quad f_a = \sum_{\substack{\mu \in \\ \{x,y,z\}}} \sum_{i=p_b-2}^{N-p_b} F_a(A_{i\mu}) \tag{14}$$

V. TIME ADJUSTMENT

Although we constrain kinodynamic feasibility in the path searching and optimization, sometimes we get infeasible trajectories. The basic reason is that gradient information tends to lengthen the overall trajectory while pushing it far from obstacles. Consequently, the quadrotor has to fly more aggressively in order to travel longer distance within the same time, which unavoidably causes over aggressive motion if the original motion is already near to the physical limits.

To guarantee dynamic feasibility, we adopt a time adjustment method based on the relations between the derivatives control points and the time allocation (knot spans) of the non-uniform B-spline. Thanks to the relations, we can change the flight aggressiveness as we expected by adjusting the associated time allocation. Thus dynamic feasibility can be ensured without over-conservative constraints.

Algorithm 2: Iterative Time Adjustment.

1 **repeat**
2      $\mathcal{V}, \mathcal{A} = \textbf{findInfeasible}()$;
3   **for** $\mathbf{V}_i'$ *in* $\mathcal{V}$ **do**
4          $v_m \leftarrow \max_{\mu \in \{x,y,z\}} | V_{i,\mu}' |$;
5          $\mu_v' \leftarrow \min\{\alpha_v, \frac{v_m}{v_{max}}\}$;
6   **AdjustKnotSpansVel**$(\mathbf{V}_i', \mu_v')$;
7   **for** $\mathbf{A}_i'$ *in* $\mathcal{A}$ **do**
8          $a_m \leftarrow \max_{\mu \in \{x,y,z\}} | A_{i,\mu}' |$;
9          $\mu_a' \leftarrow \min\{\alpha_a, (\frac{a_m}{a_{max}})^{\frac{1}{2}}\}$;
10   **AdjustKnotSpansAcc**$(\mathbf{A}_i', \mu_a')$;
11 **until** $\mathcal{V}.\textbf{empty}() \wedge \mathcal{A}.\textbf{empty}()$;

We first introduce the mathematic fundament of the time adjustment. Then the Alg. 2 is presented to tackle over-aggressive trajectories.

A. Non-Uniform B-Spline

Non-uniform B-spline is a more general kind of B-spline. The only difference to uniform B-spline is that each of its knot span $\Delta t_m = t_{m+1} - t_m$ is independent to others. The control points of a non-uniform B-spline's first and second order derivatives $\mathbf{V}_i'$ and $\mathbf{A}_i'$ can be computed by:

$$\mathbf{V}_i' = \frac{p_b(\mathbf{Q}_{i+1} - \mathbf{Q}_i)}{t_{i+p_b+1} - t_{i+1}}, \quad \mathbf{A}_i' = \frac{(p_b - 1)(\mathbf{V}_{i+1}' - \mathbf{V}_i')}{t_{i+p_b+1} - t_{i+2}} \tag{15}$$

By the convex hull property (Fig. 3), to enforce the dynamic feasibility of a trajectory represented by a non-uniform B-spline, it suffices to enforce all control points of the first and second order derivatives within the feasible domain. We show that this can be achieved by changing the corresponding knot spans of the infeasible control points in Sect. V-B.

B. Knot Spans Adjustment

Let $\mathbf{V}_i' = [V_{i,x}', V_{i,y}', V_{i,z}']^\top$ be an infeasible control point of velocity. Let $V_{i,\mu}'$ be the largest infeasible component and $| V_{i,\mu}' | = v_m$. From Equ.15 we know that $V_{i,\mu}'$ is influenced by the duration $t_{i+p_b+1} - t_{i+1}$. If we change this duration to $\hat{t}_{i+p_b+1} - \hat{t}_{i+1} = \mu_v(t_{i+p_b+1} - t_{i+1})$, then $V_{i,\mu}'$ changes to:

$$\hat{V}_{i,\mu} = \frac{p_b}{\hat{t}_{i+p_b+1} - \hat{t}_{i+1}}(Q_{i+1,\mu} - Q_{i,\mu})$$

$$= \frac{1}{\mu_v} \frac{p_b}{t_{i+p_b+1} - t_{i+1}}(Q_{i+1,\mu} - Q_{i,\mu}) = \frac{1}{\mu_v} V_{i,\mu}' \tag{16}$$

Therefore, if we set $\mu_v = \frac{v_m}{v_{\max}}$, then the velocity is feasible, because $| \hat{V}_{i,\mu} | = \frac{v_{\max}}{v_m} | V_{i,\mu}' | = v_{\max} \in [-v_{\max}, v_{\max}]$.

The enforcement of acceleration feasibility is similar.[3] We know that $A_{i,\mu}'$ is actually influenced by $t_{i+p_b+2} - t_{i+1}$ since it is coupled with $V_{i,\mu}'$ and $V_{i+1,\mu}'$. We change $\Delta t_m = t_{m+1} - t_m$

---

[3]We use the same notation as those in the last paragraph and do not define them explicitly for brevity.
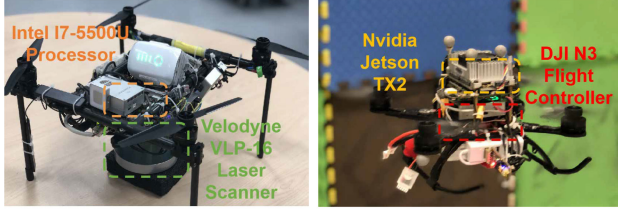
(a) Quadrotor platform used in the autonomous flight experiment.

(b) Quadrotor platform used in the aggressive replanning experiment.

Fig. 6.   Quadrotor platforms used in (a) fully autonomous flight and (b) aggressive kinodynamic replanning.
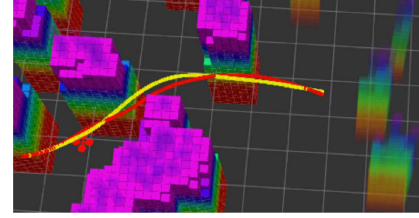

Fig. 7.   The local planning strategy for a limited sensing range. The red curve and the yellow curve are the trajectories before and after the optimization. Opaque and transparent obstacles are known and unknown ones.

to $\Delta \hat{t}_m = \mu_a \Delta t_m$ for $m \in \{i+1, i+2, \ldots, i+p_b+1\}$ and we get:

$$
\begin{aligned}
\hat{A}_{i,\mu} &= \frac{p_b - 1}{\hat{t}_{i+p_b+1} - \hat{t}_{i+2}} (\hat{V}_{i+1,\mu} - \hat{V}_{i,\mu}) \\
&= \frac{1}{\mu_a} \frac{p_b - 1}{t_{i+p_b+1} - t_{i+2}} \left(\frac{1}{\mu_a} V'_{i+1,\mu} - \frac{1}{\mu_a} V'_{i,\mu} \right) \\
&= \frac{1}{\mu_a^2} \frac{p_b - 1}{t_{i+p_b+1} - t_{i+2}} (V'_{i+1,\mu} - V'_{i,\mu}) = \frac{1}{\mu_a^2} A'_{i,\mu} \quad (17)
\end{aligned}
$$

Similarly, let $\mu_a = (\frac{a_m}{a_{\max}})^{\frac{1}{2}}$, then $| \hat{A}_{i,\mu} | = \frac{a_{\max}}{a_m} | A'_{i,\mu} | = a_{\max} \in [-a_{\max}, a_{\max}]$.

C. Iterative Time Adjustment

Based on the derivation in Sect.V-B, Alg. 2 is adopted to enforce dynamic feasibility. It iteratively finds the infeasible velocity and acceleration control points $\mathcal{V}$ and $\mathcal{A}$ of the trajectory (Line 2) and adjust the corresponding knot spans (Lines 3-10). Because a knot span Δt_m influences a few control points and vice versa, bounding μ'_v and μ'_a with two constant α_v and α_a slightly larger than 1 (Line 5, 9) prevents any time span from being extended excessively.

VI. IMPLEMENTATION DETAILS

A. Experiment Settings

The motion planning method proposed in this letter is implemented in C++11 with a general non-linear optimization solver NLopt.[4] We set $r = 2, \tau = 0.5$ for the path searching, $\lambda_1 = 10.0, \lambda_2 = 0.8, \lambda_3 = 0.01$ for the optimization and $\alpha_a = \alpha_v = 1.1$ for the time adjustment in all experiments. We present two sets of real-world experiments to validate our proposed planning method.

Firstly, we conduct fast autonomous flight experiments in unknown cluttered environments (Sect.VII-B). We use a self-developed quadrotor platform (Fig. 6(a)) equipped with a Velodyne VLP-16 3-D LiDAR. LOAM [20] is adopted to estimate the pose of the quadrotor and generate a dense point cloud map. To obtain high-rate state estimation for feedback control, we fused the laser-based estimation with IMU and sonar measurements by the extended Kalman filter (EKF). All modules including motion planning, state estimation, mapping and control run on a

dual-core 3.00 GHz Intel i7-5500U processor, which has 8 GB RAM and 256 GB SSD.

Then, in Sect.VII-C, we focus on testing the fast-replanning capability of our proposed method in aggressive flight, for which we use a more light-weight and agile quadrotor platform (Fig. 6(b)). To eliminate uncertainties introduced by onboard sensings, accurate pose feedback is provided by the motion capture system *OptiTrack*[5] and the map of the environment is pre-built. The motion planning and control modules run onboard on an Nvidia TX2 computer.

B. Re-Planning Strategy

1) Receding-Horizon Local Planning: When the quadrotor flies in an unknown environment, it has to re-plan its trajectory frequently due to the limited sensing range. To improve efficiency, we adopt a receding-horizon planning scheme, in which trajectories are generated only within the known space (Fig. 7). The path searching is terminated once a motion primitive ends outside this range and is followed by the optimization and time adjustment. Planning in the unknown space is often useless, thus such efforts can be saved.

2) Re-Planning Triggering Mechanism: The re-planning is triggered in two situations. Firstly, it is triggered if the current trajectory collides with newly emergent obstacles,[6] which ensures that a new safe trajectory is available as soon as any collision is detected. Secondly, the planner is called at fixed intervals of time. It updates the trajectory periodically using the most up-to-date environmental information.

C. Euclidean Distance Field

We maintain an EDF of the voxel grid map for our optimization, which is computed by an efficient $O(n)$ algorithm [21], where $n = N^3$ is the number of updated voxel grids. To compensate for the discretized error of the EDF introduced by the voxel grid map and benefit the numeric optimization, trilinear interpolation is used to improve the accuracy of the distance and gradient information [14]. Global update of the EDF is very costly and can block the planning module that is crucial for fast autonomous flight. To address this issue, we only update the voxel grids within the sensing range using an incremental update strategy [22].

---

[4]https://nlopt.readthedocs.io/en/latest/

[5]https://optitrack.com/

[6]The collision checking is conducted in the configuration space (C-space), where the quadrotor is modeled as a point and obstacles are inflated by a radius. The computation cost of such checking is negligible.

TABLE I
COMPARISON OF PATH SEARCHING

|		Comp. Time(s)	Traj. Time(s)	Ctrl. Cost(m^2/s^3)	Succ. Rate(%)
method [23]	Mean	0.0592	8.439	**11.42**	
	Max	0.8740	20.000	28.00	100.0
	Std	0.1060	2.970	4.07	
Proposed (Res.= 0.04m)	Mean	**0.0047**	**7.719**	**14.42**	
	Max	0.1837	10.500	24.50	100.0
	Std	0.0113	0.909	3.60	
Proposed (Res.= 0.2m)	Mean	**0.0018**	**7.696**	**15.08**	
	Max	0.0287	10.500	27.50	100.0
	Std	0.0017	0.917	3.78	
Proposed (Res.= 1.0m)	Mean	**0.0017**	**7.645**	**16.20**	
	Max	0.0059	15.500	56.00	**77.8**
	Std	0.0007	1.295	6.21	

VII. RESULTS

A. Analysis and Comparisons

1) Comparison of Path Searching: We compare our path searching with method [23], both of which use the time-optimal control formulation to generate primitives. The comparison is done on a $40 \times 40 \times 5$ m map randomly deployed with 100 obstacles and the maximum velocity and acceleration limits are set as 3 m/s and 2m/s^2 respectively. Since the resolution of voxel grids is a critical factor for the performance of our proposed method, different resolutions are used for comprehensive evaluation (Table I, column 1, rows 3-5). For a fair comparison, we use the open source implementation of [23]. Results are listed in Table I.

As is shown in statics, both methods generate kinodynamic feasible trajectories. Our method is faster with one order of magnitude and tends to generate a path with a shorter duration. However, the control cost for it is slightly higher. As the voxel girds get coarser, the efficiency of our method increases at the expense of higher control cost and lower success rate. This trend is expected because pruning primitives with coarser voxel grids results in lower searching complexity, whereas more feasible (and maybe superior) paths are lost.

2) Comparison of Optimization: For the back-end trajectory optimization, we conduct a comparison against our previous work [13]. Both of our previous method and the proposed method utilize the EDF for non-linear optimization. For fairness, we use the same path given by our path searching as the initial value. Firstly, we compare the costs of the objective function with respect to time for both methods (Fig. 8(a)). Obviously, the cost of the proposed method drops rapidly within the first few milliseconds, while the other one decreases much slower. Secondly, comparison of smoothness (integral of the squared jerk) is conducted as shown in Fig. 8(b) and Table II. Even though less time is given for the proposed method, the resulting trajectories are smoother.

B. Onboard Autonomous Flight

We conducted fully autonomous fast flight experiments in a challenging unknown environment (Fig. 9(a)). To further challenge our method, we prune the global map using a sphere with a radius of 5 m that centered on the quadrotor and only use the map within this sphere for trajectory generation (Fig. 9(b)–9(d)) which is much smaller than our real perception range. The unstructured environment, limited perception range as well as the high flying speed pose a challenge to the motion
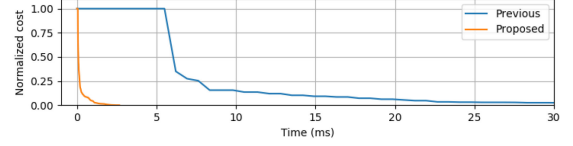
(a) The convergent performance is compared through the cost profiles of the objective function. The cost of the proposed method decreases rapidly to zero within $3ms$, while the other takes a significantly longer time. The cost for both methods are normalized to [0,1] for comparison. Computation time is limited to $30ms$ for both.
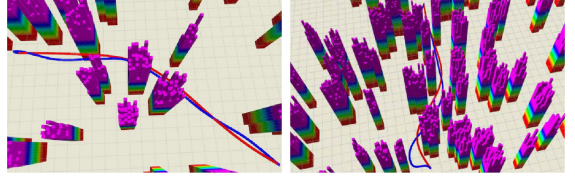
(b) Trajectories generated by the proposed (red) and previous (blue) method. Note that computation time for the proposed method is only $1ms$, while that for the previous method is $10ms$.
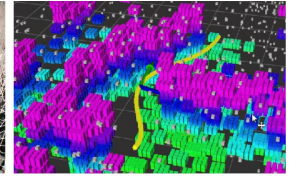
Fig. 8.   Comparing our proposed optimization method with gradient-based optimization method [13] in random cluttered environments.
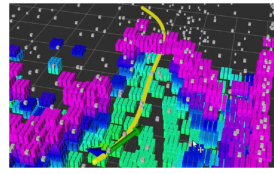
TABLE II
COMPARISON OF TRAJECTORY OPTIMIZATION

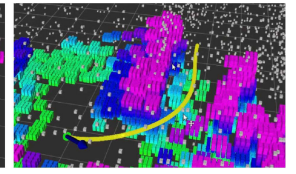|	Integral of Jerk^2 (m^2/s^5)			Comp. Time(s)
	Mean	Max	Std	
Previous [13]	43.913	181.495	18.394	0.010
Proposed	**35.932**	**131.913**	**13.118**	0.001

(a) The environment setup of the autonomous flight experiment.

(b) Flight 1, bypassing a vertical wall.

(c) Flight 2, avoiding vertical obstacles.

(d) Flight 3, flying through a narrow hole.

Fig. 9.   Fully autonomous flight in an unknown cluttered environment. Only the colored map is known by the motion planning module. In this confined environment, the maximum and average speed of flight 1-3 reach up to 1.7 m/s and 1.3 m/s^2 respectively.

planning module, as it should re-generate trajectory continually and rapidly upon sudden appearing of new threats. We refer the readers to the video attachment for more detailed information.

C. Aggressive Flight

The aggressive flight experiment is done in the environment depicted in Fig. 10. In the experiment, the goals of the quadrotor are changed constantly and arbitrarily by a human. As soon as a new goal is set, a new trajectory is re-planned and executed immediately. The maximum velocity and acceleration are set as 2.5 m/s and 1.5 m/s^2 respectively. This task is challenging in
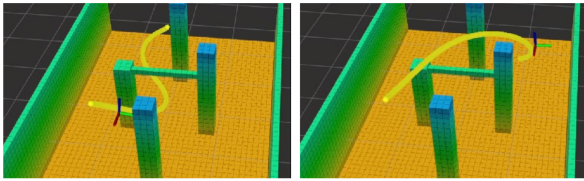
Fig. 10. Aggressive flight test. The goals are changed arbitrarily and new trajectories are replanned during the aggressive flight.

several aspects. Since the flight is aggressive and the changes in the goals are abrupt, the motion planning module should generate new trajectories in considerably short time to quickly react to the changes, so that the motion of the quadrotor is continuous and smooth. Also, as the environment is confined and cluttered, it is difficult to generate smooth, safe and dynamically feasible trajectories in a very short time. This experiment validates that our method can generate aggressive motion under the premise of feasibility. It also shows that our method can quickly generate a new trajectory in complex environments even if the goal is changed suddenly during the aggressive flight. More details are also included in the video.

VIII. CONCLUSION

In this letter, we propose a novel online motion planning method for quadrotor autonomous navigation. We decouple the online fast motion planning problem as a front-end kinodynamic path searching and a back-end nonlinear trajectory optimization. We adopt a kinodynamic path searching to find a safe, kinodynamic feasible and minimum-time initial path, which is further improved in smoothness and clearance by a gradient-based optimization. By utilizing the convex hull property of B-spline, we significantly improve the efficiency and convergent rate of the optimization compared to previous gradient-based planning methods. Finally, by representing the trajectory as a non-uniform B-spline, we adjust the time allocation according to a given expected flight aggressiveness. We validate our proposed method in various complex environments and the simulation. The competence of the method is also validated in challenging real-world tasks.

In the future, we plan to challenge our quadrotor system in extreme situations such as large-scale or dynamic environments. Furthermore, we will extend our trajectory optimization method to swarm problems.

REFERENCES

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

[2] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. Int. Symp. Robot. Res.*, Dec. 2013, pp. 649–666.

[3] J. Chen, K. Su, and S. Shen, "Real-time safe trajectory generation for quadrotor flight in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Zhuhai, China, Aug. 2015, pp. 1678–1685.

[4] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *Proc. IEEE Int. Symp. Saf., Secur. Rescue Robot.*, Lausanne, Switzerland, 2016, pp. 139–146.

[5] 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.

[6] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 7595–7602.

[7] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," 2019, *arXiv:1906.09785*.

[8] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, Australia, May 2018, pp. 344–351.

[9] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *J. Field Robot.*, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21842

[10] M. Zucker *et al.* "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9/10, pp. 1164–1193, 2013.

[11] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4569–4574.

[12] 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.*, Daejeon, South Korea, Oct. 2016, pp. 5332–5339.

[13] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 3681–3688.

[14] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform b-splines and a 3d circular buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.

[15] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.

[16] M. W. 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.

[17] K. Qin, "General matrix representations for b-splines," *Visual Comput.*, vol. 16, no. 3, pp. 177–186, 2000.

[18] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, pp. 802–807.

[19] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. 54th IEEE Conf. Decis. Control*, 2015, pp. 835–842.

[20] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot., Sci. Syst.*, Jul. 2014, pp. 109–111.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2012.

[22] T. Schouten and E. L. van den Broek, "Incremental distance transforms (IDT)," in *Proc. 20th Int. Conf. Pattern Recognit.*, 2010, pp. 237–240.

[23] 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.*, Sep. 2017, pp. 2872–2879.