TGK-Planner: An Efficient Topology Guided Kinodynamic Planner for Autonomous Quadrotors

Hongkai Ye, Xin Zhou, Zhepei Wang, *Graduate Student Member, IEEE*, Chao Xu, *Senior Member, IEEE*, Jian Chu, and Fei Gao, *Member, IEEE*

***Abstract*—In this letter, we propose a lightweight yet effective Topology Guided Kinodynamic planner (TGK-Planner) for quadrotor aggressive flights with limited onboard computing resources. The proposed system follows the traditional hierarchical planning workflow, with novel designs to improve the robustness and efficiency in both the pathfinding and trajectory optimization sub-modules. Firstly, we propose the *topology guided graph*, which roughly captures the topological structure of the environment and guides the state sampling of a sampling-based kinodynamic planner. In this way, we significantly improve the efficiency of finding a safe and dynamically feasible trajectory. Then, we refine the smoothness and continuity of the trajectory in an optimization framework, which incorporates the homotopy constraint to guarantee the safety of the trajectory. The optimization program is formulated as a sequence of quadratic programmings (QPs) and can be iteratively solved in a few milliseconds. Finally, the proposed system is integrated into a fully autonomous quadrotor and validated in various simulated and real-world scenarios. Benchmark comparisons show that our method outperforms state-of-the-art methods with regard to efficiency and trajectory quality. Moreover, we will release our code as an open-source package.[1]**

***Index Terms*—Aerial systems, applications, collision avoidance, motion and path planning.**

I. Introduction

IN RECENT years, although many works have been proposed toward online aerial planning, it is still challenging for quickly generating high-speed kinodynamic trajectories in a resource-limited quadrotor. Due to the complexity of the environment and system dynamics, generating an optimal and executable trajectory usually takes the price of high computational overhead. Moreover, for a quadrotor flying at high speed, re-planning has to be finished in a short time to react to unpredictable obstacles. For a cheap platform, especially the commercial quadrotor with a limited computing budget, the above two requirements are hard to be satisfied at the same time, making the high aggressiveness hard to achieve on the premise of safety guarantee. Some works [1], [2] integrate perception with planning for high-speed flight by using motion primitive libraries. However, the restricted primitive set guarantees no optimality, and the discretization makes long-term trajectories inconsistent.

In this letter, we investigate the above research gap and propose a systematic approach to bridge it. Our method follows the traditional hierarchical planning workflow, which consists of a kinodynamic planner that finds a trajectory according to a coarse system dynamics, and an optimizer that improves the smoothness and continuity of the trajectory. For kinodynamic planning in high-dimensional state spaces, sampling-based planners have great potential in efficiency by designing smart sampling strategies. Imagine this situation: a quadrotor flies along a corridor at high speed, states sampled towards the walls are most probably not useful, while a state with a velocity along the corridor certainly benefits the planning. Besides, many sampling-based methods have the *anytime* nature, which especially suits fast flight by improving the optimality of the plan while executing it [3].

Therefore, we adopt a sampling-based front-end and efficiently sample states with environmental awareness. Our front-end builds a topological structure capturing the free space's connectivity and then generates a high-quality feasible trajectory. Based on this trajectory, we design a lightweight optimization-based back-end to further improve its key attributes, smoothness and continuity, with a guarantee on its safety and dynamical feasibility. The proposed back-end fully exploits assets of our front-end, which are, reasonable homotopy residence and reasonable time allocation, by incorporating them into the objective. Furthermore, efficiency and optimality are retained by formulating the optimization as a sequence of QPs with closed-form solutions.

this letter highlights its efficiency in both the front-end and back-end, guarantees the asymptotic optimality, and retains the robustness and quality of the generated trajectory. We summarize the contributions as follow:

1) A sampling-based kinodynamic planning front-end, which significantly improves the efficiency of kinodynamic RRT* [4] algorithm by environment guided state sampling, and the cost converges in a few milliseconds.

2) A lightweight yet effective trajectory refinement back-end, which exploits the front-end assets to improve the smoothness

Manuscript received August 5, 2020; accepted December 15, 2020. Date of publication December 29, 2020; date of current version January 12, 2021. This letter was recommended for publication by Associate Editor P. Pounds and Editor J. Roberts upon evaluation of the reviewers' comments. This work was supported by National Natural Science Foundation of China under Grant 62003299 and Grant 62088101. *(Corresponding author: Fei Gao.)*

The authors are with the State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: hkye@zju.edu.cn; iszhouxin@zju.edu.cn; wangzhepei@zju.edu.cn; cxu@zju.edu.cn; chuj@iipc.zju.edu.cn; fgaoaa@zju.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2020.3047798, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3047798

[1]Code is released at https://github.com/ZJU-FAST-Lab/TGK-Planner.

and continuity of the trajectory by a sequence of least-square optimization.

3) Integrating the proposed methods which suits both global and local trajectory generation into a fully autonomous quadrotor system, presenting extensive benchmark and experimental validations, and releasing source code for the reference of the community.

II. Related Work

A. Kinodynamic Planning

Kinodynamic planning can be roughly divided into search-based and sampling-based. Search-based methods discretize the control space and use motion primitives to search for a solution with piece-wise constant controls. Recent typical works [5], [6] develop efficient heuristics by solving an unconstrained linear-quadratic energy-time minimization problem. However, in those methods, the resolution must be carefully chosen to make a trade-off between solution existence and search-space complexity. Besides, they leave apparent discontinuities in control inputs. For sampling-based methods, RRT-based algorithms are naturally extendable to kinodynamic systems by sampling in the state space. However, tree expansion can be extremely inefficient for complicated dynamics in high-dimensional state space. This is mainly caused by inefficient boundary value problem (BVP) solving and invalid state sampling. Webb *et al.* [4] derives the closed-form solutions to solve the BVP for linearized systems with a nilpotent dynamics matrix, which saves the computational overhead. Nevertheless, too much computation is wasted on connecting invalid samples, making it impossible for real-time usage on embedded platforms. To increase the probability of obtaining valid samples, it is necessary to design a strategy to bias/guide the sampling process. Some works [7]–[9] build sparse skeleton graphs of the environment and generate samples alongside edges of the graph. These works only consider cases in the $\mathbb{R}^3$ space, and extracting a complete topological graph is rather time-consuming as the scale and complexity of the environment grow. In this letter, we build our front-end upon [4] and propose a simple yet effective topology extraction method to guide the sampling.

B. Trajectory Optimization

Trajectory optimization is essential in improving the path found by the front-end to meet the full system dynamics. The minimum-snap formulation [10] is widely adopted due to its simplicity and efficiency. In [11], the authors further convert it to an unconstrained quadratic programming (QP) problem and solve it in closed-form. The safety and dynamic feasibility of the trajectory is ensured by iteratively adding intermediate waypoints to the path and solving the QP. Some works [5] [12]–[14] extract obstacle-free corridors represented by a sequence of convex enclosed shapes, and then generate safe trajectories within the free space by convex optimization. Although these works enjoy the convexity in their formulations, too many hard-constraints impose intensive computational overhead, thus preventing them from being used in cheap platforms. Besides, no dynamics is considered in their front-end, making the optimization process always over-conservative. Gradient-based methods [15]–[17] formulate trajectory optimization as non-linear optimization problems with penalties on collision, control, and constraint violation. For ensuring safety, a costly Euclidean signed distance field (ESDF) has to be established, and the integration of cost terms is usually expensive. Some recent works [6], [18] mitigate these issues by parameterizing the trajectory as B-splines, and aggregate costs only on discrete control points. However, due to the underlying nonlinearity of the optimization program, it can not guarantee a good final solution and is sensitive to the initial guess. In this letter, we formulate our optimization problem as a sequence of QPs and utilize the topological information from the front-end, to design a fast and robust optimization pipeline.

Algorithm 1: Kinodynamic RRT*.

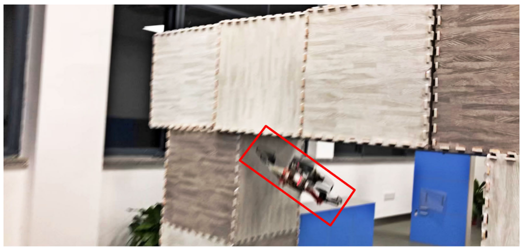
```
1:  Notation: Environment $\mathcal{E}$, Tree $\mathcal{T}$, State $\mathbf{x}$
2:  Initialize: $\mathcal{T} \leftarrow \emptyset \cup \{\mathbf{x}_{init}\}$
3:  for $i = 1$ to $n$ do
4:      $\mathbf{x}_{random} \leftarrow$ Sample($\mathcal{E}$)
5:      $\mathcal{X}_{backward} \leftarrow$ BackwardNear($\mathcal{T}$, $\mathbf{x}_{random}$)
6:      if $\mathcal{X}_{backward} \neq \emptyset$ then
7:          $\mathbf{x}_{\min} \leftarrow$ ChooseParent($\mathcal{X}_{backward}$, $\mathbf{x}_{random}$)
8:          $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_{random}\}$
9:          $\mathcal{X}_{forward} \leftarrow$ ForwardNear($\mathcal{T}$, $\mathbf{x}_{random}$)
10:         if $\mathcal{X}_{forward} \neq \emptyset$ then
11:             Rewire($\mathcal{T}$, $\mathcal{X}_{forward}$)
12:         end if
13:     end if
14: end for
15: return $\mathcal{T}$
```

III. Kinodynamic Trajectory Planning

We briefly review the Kinodynamic RRT* [4] algorithm, and then present our environment guided sampling strategy which significantly facilitates the efficiency.

A. Kinodynamic RRT* Framework

The main workflow of the Kinodynamic RRT* [4] is described in Algorithm 1, where a tree $\mathcal{T}$ grows from the initial state $\mathbf{x}_{init}$ towards the goal state $\mathbf{x}_{goal}$. Every time a valid state $\mathbf{x}_{random}$ is sampled, a subset of $\mathcal{T}$, $\mathcal{X}_{backward}$ whose elements can connect to $\mathbf{x}_{random}$ are found though **BackwardNear()**. If $\mathcal{X}_{backward}$ is not empty, then a node $\mathbf{x}_{\min}$ with the minimum transition cost is chosen as the parent node of $\mathbf{x}_{random}$ through **ChooseParent()**, and $\mathbf{x}_{random}$ is added to the tree $\mathcal{T}$. Moreover, **ForwardNear()** searches in $\mathcal{T}$ for a node set $\mathcal{X}_{forward}$ whose elements that $\mathbf{x}_{random}$ can connect to, and then **Rewire()** checks for every state in $\mathcal{X}_{forward}$ whether it can be reached by a lower cost route though $\mathbf{x}_{random}$. The loop terminates when either the maximum sampling number or the running time exceeds. Finally, the trajectory is obtained by tracing back from $\mathbf{x}_{goal}$ through its parent recursively, if $\mathbf{x}_{goal}$ is connected with any state node in the tree. A visualization is provided in Fig. 2.


(a) Dodge obstacles.

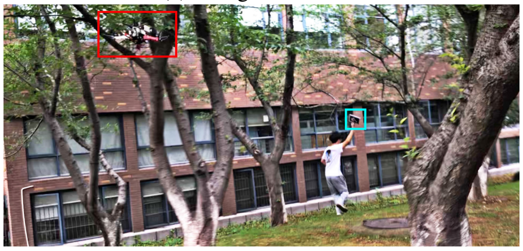

(b) Chase a fast-moving target.

Fig. 1. Fast autonomous flight in unknown in(out)-door environments. Video is at https://youtu.be/nNS0p8h5zAk.

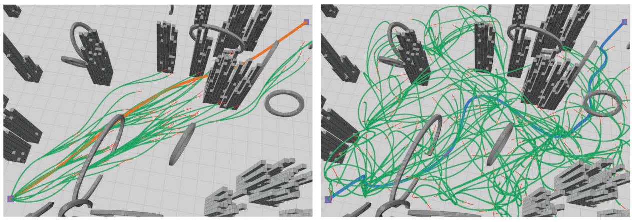

(a) Proposed sampling strategy (b) Uniformly random sampling

Fig. 2. (a) The generated trees (green) and the first feasible trajectory found by our guided sampling strategy (orange) in 3 ms; and (b) by uniformly random sampling (blue) in 100 ms.

B. Optimal States Transition

The cornerstone in the above Algorithm 1 is the optimal connection of two states. In [4], a general form of the connection for systems with a nilpotent dynamics matrix and a mixed time/energy cost criterion is derived. Specifically, we fit it to our model and derive optimal solutions using standard optimal control techniques. In this letter, the transition cost from state $\mathbf{x}_0$ to state $\mathbf{x}_1$ is defined as:

$$c(\mathbf{x}_0, \mathbf{x}_1) = \int_0^{\tau} (\rho + \frac{1}{2}\mathbf{u}(t)^{\mathsf{T}}\mathbf{u}(t))dt, \quad (1)$$

where τ is the time duration and ρ is the weight. Minimizing the cost is equivalent to solve a fixed-endpoint, free-time optimal control problem [19]:

$$\begin{aligned} \min \mathcal{J}(\mathbf{x}(t)) &= \int_0^{\tau} \mathcal{L}(t, \mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t))dt \\ \text{s.t.} \quad & f(t, \mathbf{x}, \mathbf{u}) - \dot{\mathbf{x}}(t) = \mathbf{0}, \\ & \mathbf{x}(0) = \mathbf{x}_0, \ \mathbf{x}(\tau) = \mathbf{x}_1, \\ & \mathbf{x}(t) \in \mathcal{X}^{free}, \ \mathbf{u}(t) \in \mathcal{U}^{free}, \end{aligned} \quad (2)$$

where Lagrangian $\mathcal{L}$ is the cost functional defined in Eq. 1, and $f(t, \mathbf{x}, \mathbf{u})$ is the differential constraint of the system:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (3)$$

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix},$$

$$\mathbf{u}(t) = \ddot{\mathbf{p}}(t), \quad \mathbf{p}(t) = \begin{bmatrix} p_x(t), \ p_y(t), \ p_z(t) \end{bmatrix}^{\mathsf{T}}, \quad (4)$$

which is modeled as a linear system according to the quadrotor's differential flatness property [10].

According to the calculus of variation, the Hamiltonian is written as $\mathcal{H}(t, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = \mathcal{L} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{f}$, where $\boldsymbol{\lambda}(t)$ is the costate vector. In our case, the optimal arriving time τ^* satisfies $\mathcal{H}(\tau^*, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = 0$, which is an equation of 4^{th} order polynomial whose coefficients are fully determined by boundary conditions. After solving this equation, τ^* is obtained and the problem becomes a fixed-endpoint, fixed-time problem.

Let $\mathbf{u}^*$ and $\mathbf{x}^*$ be the optimal control and state trajectory separately, we now apply Pontryagin Maximum Principle [19] to characterize $\mathbf{u}^*$. The state $\mathbf{x}^*$ and costate $\boldsymbol{\lambda}^*$ must satisfy the following canonical equations:

$$\begin{cases} \dot{\boldsymbol{\lambda}}^* = -\partial\mathcal{H}(t, \mathbf{x}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*)/\partial\mathbf{x}, \\ \dot{\mathbf{x}}^* = \partial\mathcal{H}(t, \mathbf{x}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*)/\partial\boldsymbol{\lambda}. \end{cases} \quad (5)$$

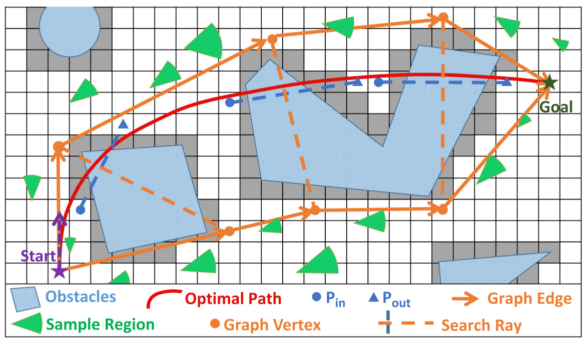
If assuming the control and state unbounded, the maximizer of $\mathcal{H}$ satisfies $\partial\mathcal{H}/\partial\mathbf{u}^* = \mathbf{0}$. Solving this equation along with the boundary and transversality conditions, we obtain the optimal solution pair $\{\mathbf{u}^*(t), \mathbf{p}^*(t)\}$ which is:

$$\begin{aligned} p_k^*(t) &= \frac{1}{6}c_{k,3}t^3 + \frac{1}{2}c_{k,2}t^2 + c_{k,1}t + c_{k,0}, \\ u_k^*(t) &= c_{k,3}t + c_{k,2}, \ k \in \{x, y, z\}. \end{aligned} \quad (6)$$

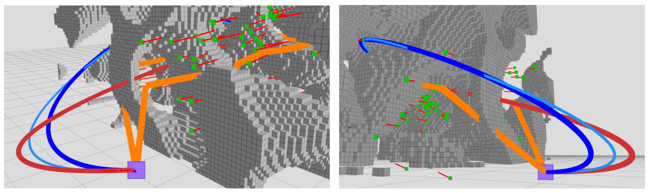
The corresponding optimal cost can be derived from Eq. 1. We then check the feasibility of the unconstrained optimal solution pair $\{\mathbf{u}^*(t), \ \mathbf{p}^*(t)\}$, and consider the connection failed if it violates any constraints. Note, although this simplification of the maximizer of $\mathcal{H}$ sacrifices some feasible samples, it greatly accelerates the BVP solving and facilities the overall efficiency.

C. Approximate Topological Graph Guided Sampling

As intuitively stated in Sect. I, a uniformly random sampling of the free states is inefficient. We here use a method to quickly construct a *topology guided graph*, which approximately captures the topological structure of the environment, as shown in Fig 3. The environment is represented as an occupancy grid map. To construct the *graph*, an optimal path directly connecting $\mathbf{x}_{init}$ and $\mathbf{x}_{goal}$ is firstly planned without considering any obstacles (Fig. 3, red curve). Along the path, we record positions where the path goes in and out of obstacles, denoting as $\mathbf{p}_{in}^i$ and $\mathbf{p}_{out}^i$. Connecting each pair of them forms *traversal lines* (Fig. 3, dashed blue line). Then, starting from the middle point of each *traversal line*, we do ray tracing (Fig. 3, dashed orange line) in the direction perpendicular to the *traversal line* and level to the horizontal plane. The tracing stops when an obstacle-free grid is found on both sides, and the stopping grids are taken as vertices of the *graph*, which are of the same height as the middle point of the corresponding *traversal line*. Positions of $\mathbf{x}_{init}$ and $\mathbf{x}_{goal}$ are also graph vertices. Finally, the *graph* is constructed by connecting the vertices from start to goal. Unlike

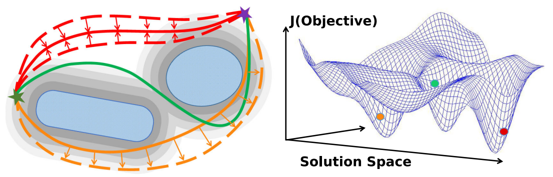


(a)



(b) (c)

Fig. 3. (a) Illustration of constructing a topological graph. Orange lines form the graph and green fans represent the probabilistic-based sampling region of states. (b) (c) Graph (Orange) and state samples (Green dots represent state position and red lines represent state velocity) in 3D environments. The initial velocity is nonzero.



(a) (b)

Fig. 4. Illustration of the homotopy constraint. (a) In our method, the initial path (solid red curve) attracts optimized paths (red dashed curves). In many others, an ESDF pushes path (orange curves) away from obstacles. (b) For a highly nonconvex optimization, solutions of different homotopy classes fall in different "pits" (nearby solution spaces of local minima) of the objective.

[20] and other methods that desire complete topological graphs in obstacle-free areas, our graph captures the partial topological structure of the environment in a much cheaper way. As a sacrifice, the graph edges (Fig. 3, solid orange line) are not guaranteed collision-free. However, this is acceptable since we sample state positions in free space in the vicinity of these edges with a normal distribution, as shown in Fig. 3. As for state velocity, its direction is sampled with a normal distribution to deviate from the direction of edges. Its magnitude is sampled according to speed limits.

IV. Fast Trajectory Refinement

As stated before, a trajectory obtained from the front-end (Sect. III) is based on a coarse dynamic model and, therefore, has relatively low fidelity despite it meets all constraints. In this section, we show how to efficiently improve the continuity and smoothness, by incorporating the homotopy structure of the front-end trajectory.

A. Problem Formulation

For each dimension, consider an m-segment, n^{th}-order polynomial trajectory $p_m(t) = c_0 + c_1 t + c_2 t^2 \cdots + c_n t^n$, and let $\mathbf{c}_m = [c_0, c_1, c_2, \ldots, c_n]^\mathsf{T}$ be the coefficient of the m^{th} segment, our goal is to find the optimal coefficient for each segment of the trajectory.

To optimize the trajectory, we investigate the proposed front-end, and build our back-end based on some special properties of its solution. Firstly, the quadrotor dynamics is roughly captured in the front-end, making the initial path be in a reasonable homotopy class (geometric region). As proved by [5], [21], a much better trajectory can be obtained starting from this initial trajectory and search in its nearby solution space. Secondly, the trajectory satisfies all the constraints imposed by the acceleration input model, including safety constraints and dynamical constraints. It is C^0 and C^1 continuous but only segment-wise C^2 continuous, that is, the acceleration changes abruptly in conjoined points between every two consecutive segments, although it is continuous within each segment. We define the acceleration differences between segments as an acceleration gap and aim to minimize it in the following optimization progress since the gap leads to quadrotor attitude jitters, which harm the control a lot.

Based on the above observations, we let the objective J make out of three terms, and the problem becomes:

$$\begin{aligned} \min\ J &= \lambda_s J_s + \lambda_h J_h + \lambda_c J_c \\ \text{s.t.}\ \ \mathbf{x}(t) &\in \mathcal{X}^{free},\ \mathbf{u}(t) \in \mathcal{U}^{free}, \end{aligned} \tag{7}$$

where J_s is the cost of overall smoothness, J_h the term that penalizes the difference in homotopy class compared with the front-end trajectory, J_c the term that penalizes acceleration discontinuity between segments, and λ_s, λ_h, λ_c the weights.

Here, the homotopy penalty J_h is essential, since it makes the online optimization with the above highly nonlinear constraints solvable. By adding this term, we turn the safety constraint from a collision rejecting one to a feasible solution attracting one, and avoid the expensive computation for an ESDF, as shown in Fig. 4(a). Besides, it significantly narrows the alternative solutions to a nearby solution space of the initial feasible solution, as shown in Fig. 4(b). Based on all these above, we design our optimization framework as a homotopy penalized, soft-constrained, iterative optimization problem. Fortunately, since all cost terms are quadratic, each iteration of the optimization has a closed-form optimal solution that is efficient and numerically stable.

B. Quadratic Objective Construction

1) Smoothness Cost: J_s is formulated as the integral of the squared derivative of the trajectory:

$$\begin{aligned} J_s &= \sum_{k\in\{x,y,z\}} \int_0^T [p_k^{(j)}(t)]^2 dt \\ &= \sum_{k\in\{x,y,z\}} \sum_{i=1}^{m} \mathbf{c}_{i,k}^\mathsf{T} \int_0^{t_i} \mathbf{t}^{(j)} (\mathbf{t}^{(j)})^\mathsf{T} dt\ \mathbf{c}_{i,k} \\ &= \sum_{k\in\{x,y,z\}} \mathbf{c}^\mathsf{T} \mathbf{Q}_s \mathbf{c}, \end{aligned} \tag{8}$$

where $T = t_1 + t_2 + \cdots + t_m$ is the total duration of the trajectory and t_i the duration for each segment. $\mathbf{t}^{(j)} = \mathrm{d}^j[1, t, t^2, \ldots, t^n]^\mathsf{T}/\mathrm{d}t^j$ is the j^{th}-order derivative vector of $\mathbf{t} = [1, t, t^2, \ldots, t^n]^\mathsf{T}$, and $\mathbf{c}^\mathsf{T} = [\mathbf{c}_{1,k}^\mathsf{T}, \mathbf{c}_{2,k}^\mathsf{T}, \ldots, \mathbf{c}_{m,k}^\mathsf{T}]$ is the coefficient vector of m segments.

2) Homotopy Cost: The homotopy cost J_h is formulated as the integration over the squared difference between positions of the optimized trajectory and the original trajectory:

$$\begin{aligned} J_h &= \sum_{k\in\{x,y,z\}} \int_0^T [p_k(t) - p_k^*(t)]^2 dt \\ &= \sum_{k\in\{x,y,z\}} \sum_{i=1}^{m} (\mathbf{c}_{i,k} - \mathbf{c}_{i,k}^*)^\mathsf{T} \int_0^{t_i} \mathbf{t}\mathbf{t}^\mathsf{T} dt\, (\mathbf{c}_{i,k} - \mathbf{c}_{i,k}^*) \\ &= \sum_{k\in\{x,y,z\}} (\mathbf{c} - \mathbf{c}^*)^\mathsf{T} \mathbf{Q}_h (\mathbf{c} - \mathbf{c}^*), \end{aligned} \tag{9}$$

where $p^*(t)$ is the original trajectory with $\mathbf{c}^*$ its coefficient vector of m segments.

By adding this term, the optimizer will force the optimized trajectory to be close to the original one, thus more likely to be residing in free spaces of the same homotopy class.

3) Continuity Cost: We penalize the acceleration gap for approaching near C^2 continuity. The continuity cost is defined as:

$$\begin{aligned} J_c &= \sum_{k\in\{x,y,z\}} \sum_{i=1}^{m-1} [\ddot{p}_{i,k}(t_i) - \ddot{p}_{i+1,k}(0)]^2 \\ &= \sum_{k\in\{x,y,z\}} \sum_{i=1}^{m-1} [\mathbf{c}_{i,k}^\mathsf{T} \mathbf{t}^{(2)}|_{t=t_i} - \mathbf{c}_{i+1,k}^\mathsf{T} \mathbf{t}^{(2)}|_{t=0}]^2 \\ &= \sum_{k\in\{x,y,z\}} \mathbf{c}^\mathsf{T} \mathbf{Q}_c \mathbf{c}, \end{aligned} \tag{10}$$

where $\ddot{p}_{i,k}(t_i)$ is the terminal acceleration of the i^{th} segment and $\ddot{p}_{i+1,k}(0)$ is the beginning acceleration of the $(i+1)^{th}$ segment, both in the in k dimension.

Here, we formulate the acceleration gap penalty as a soft constraint, since imposing a hard constraint of overall C^2 continuity may prevent finding a feasible solution, especially among extremely cluttered obstacles. Considering safety as the top priority for planning, a minor acceleration gap is acceptable in exchange for higher possibilities to find trajectories with strict safety guarantees.

With the terms mentioned above, the overall objective function is written in a quadratic form:

$$\begin{aligned} \min J = &\ \lambda_s J_s + \lambda_h J_h + \lambda_c J_c \\ = &\sum_{k\in\{x,y,z\}} [\mathbf{c}^\mathsf{T}(\lambda_s \mathbf{Q}_s + \lambda_h \mathbf{Q}_h + \lambda_c \mathbf{Q}_c)\mathbf{c} \\ &- 2\lambda_h \mathbf{c}^\mathsf{T} \mathbf{Q}_h \mathbf{c}^* + \lambda_h (\mathbf{c}^*)^\mathsf{T} \mathbf{Q}_h \mathbf{c}^*] \\ \text{s.t.}\quad & \mathbf{A}\mathbf{c} = \mathbf{d}, \end{aligned} \tag{11}$$

where $\mathbf{c}$ is the decision variable, and $\mathbf{A}\mathbf{c} = \mathbf{d}$ is the boundary derivative constraints for each segments. The cost is independent of each axis and can be solved separately.

C. Closed-Form Solution for Each Iteration

As described in [11], a piecewise polynomial trajectory can be expressed in term of boundary derivatives instead of coefficients of each segment:

$$\mathbf{c} = \mathbf{K} \begin{bmatrix} \mathbf{d}_f \\ \mathbf{d}_p \end{bmatrix}, \ \mathbf{K} = \mathbf{A}^{-1}\mathbf{C}, \tag{12}$$

where matrix $\mathbf{K}$ maps the coefficients vector $\mathbf{c}$ to the derivatives vector which is reordered as fixed derivatives $\mathbf{d}_f$ and free derivatives $\mathbf{d}_p$ (the decision variables). Details about the construction of the mapping matrix are described in [11].

In this way, the objective can be rewritten in an unconstrained formulation in each dimension as:

$$\begin{aligned} J = &\begin{bmatrix} \mathbf{d}_f \\ \mathbf{d}_p \end{bmatrix}^\mathsf{T} \mathbf{K}^\mathsf{T}(\lambda_s \mathbf{Q}_s + \lambda_h \mathbf{Q}_h + \lambda_c \mathbf{Q}_c)\mathbf{K} \begin{bmatrix} \mathbf{d}_f \\ \mathbf{d}_p \end{bmatrix} \\ &- 2\lambda_h \begin{bmatrix} \mathbf{d}_f \\ \mathbf{d}_p \end{bmatrix}^\mathsf{T} \mathbf{K}^\mathsf{T} \mathbf{Q}_h \mathbf{c}^* + \lambda_h (\mathbf{c}^*)^\mathsf{T} \mathbf{Q}_h \mathbf{c}^*. \end{aligned} \tag{13}$$

Denote $\mathbf{K}^\mathsf{T}(\lambda_s \mathbf{Q}_s + \lambda_h \mathbf{Q}_h + \lambda_c \mathbf{Q}_c)\mathbf{K}$ as matrix $\mathbf{R}$, $\mathbf{K}^\mathsf{T}\mathbf{Q}_h\mathbf{c}^*$ as matrix $\mathbf{Z}$. Omit constants in J which do not affect the optimal solution, the Jacobian of J with respect to $\mathbf{d}_p$ in one axis is:

$$\frac{\partial J}{\partial \mathbf{d}_p} = 2\mathbf{R}_{pf}\mathbf{d}_f + 2\mathbf{R}_{pp}\mathbf{d}_p - 2\lambda_h \mathbf{Z}_p, \tag{14}$$

where $\mathbf{R}_{xx}$ and $\mathbf{Z}_x$ are block matrices of $\mathbf{R}$ and $\mathbf{Z}$. Let the Jacobian equal $\mathbf{0}$, and we get the closed-form solution of the decision variables:

$$\mathbf{d}_p = \mathbf{R}_{pp}^{-1}(\lambda_h \mathbf{Z}_p - \mathbf{R}_{pf}\mathbf{d}_f). \tag{15}$$

As is noted above, we temporarily ignore all the inequality constraints and derive the formulation as an unconstrained QP. Given time durations of trajectory segments and the weights of cost terms, the solution that minimizes the overall cost can be obtained efficiently in closed-form. To ensure the feasibility of the final solution, after solving Eq. 15, we check whether safety and dynamical feasibility constraints are violated in each iteration, as shown in Algorithm 2. This is done by an extremely efficient continuous-time feasibility checker proposed in [22]. With our kinodynamic front-end providing initial trajectories with proper time allocation, the feasibility constraints are prone to be satisfied, as shown in our experimental tests.

D. Optimization Process

The optimization process is shown in Algorithm2. Denote $r_c = \lambda_c/(\lambda_s + \lambda_h + \lambda_c)$ and $r_h = \lambda_h/(\lambda_s + \lambda_h)$. The initial value of r_c is set close to 1 to prefer continuous acceleration between segments, and the initial value of r_h is also set close to 1 to make the solution similar to the original feasible one. As shown above, in each iteration of the first loop, r_c is decreased by $d_{r,c}$ while r_h is fixed, and a temporary trajectory is obtained by **ClosedFormSolve()** with Eq.15, once this temporary trajectory is checked infeasible, the iteration stops and r_c is fixed. In this way, the segment-wise acceleration discontinuity is heavily penalized, and we will obtain a solution with near C^2 continuity.

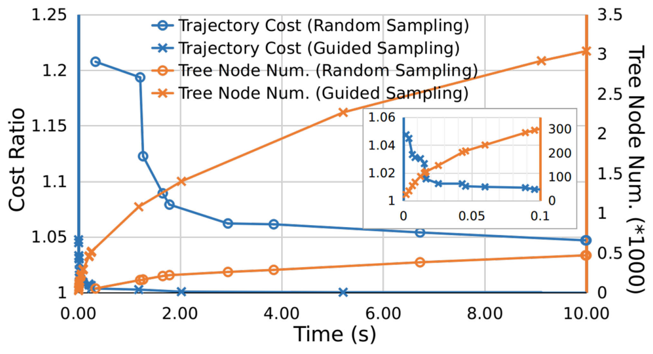


Fig. 5. Comparison of our guided sampling strategy and random sampling method. The inner figure shows the detailed result between 0 to 0.1 s.

```
Algorithm 2: Iterative Optimization.
 1: Notation: Environment $\mathcal{E}$, Trajectory $\pi$
 2: Initialize: $r_c \leftarrow r_{c,init}$, $r_h \leftarrow r_{h,init}$, $\pi \leftarrow \pi_{init}$
 3: while $r_c > 0$ do
 4:     $\pi_{temp} \leftarrow$ ClosedFormSolve($r_c$, $r_h$)
 5:     $\pi \leftarrow \pi_{temp}$
 6:     if $\neg$ CheckFeasible($\pi$, $\mathcal{E}$) then break;
 7:     end if
 8:     $r_c \leftarrow r_c - d_{r,c}$
 9: end while
10: while $r_h > 0$ do
11:     $\pi_{temp} \leftarrow$ ClosedFormSolve($r_c$, $r_h$)
12:     $\pi \leftarrow \pi_{temp}$
13:     if $\neg$ CheckFeasible($\pi$, $\mathcal{E}$) then break;
14:     end if
15:     $r_h \leftarrow r_h - d_{r,h}$
16: end while
17: return $\pi$
```

In the second loop, r_c is fixed and r_h is decreased by $d_{r,h}$ in each iteration. As r_h continues to decrease, the importance of the smoothness term increase. Thus it seeks a smoother trajectory in a relatively small solution space around the solution of the same homotopy class provided by the original trajectory, meanwhile satisfying the feasibility checking.

V. Benchmark Comparisons

A. Sampling Strategy

We compare our proposed topology guided sampling strategy with typical uniform sampling. We conduct a random simulation in a $40 \times 40 \times 3m$ environment with 100 randomly deployed obstacles and starting and goal positions. All the benchmark computations are done with a 2.2 GHz Intel i7-4702HQ processor. We limit the maximum planning time to 10 s and take the trajectory cost of our method as a baseline. The optimality ratio against planning time is shown in Fig. 5.

As shown in Fig. 5, using our guided sampling strategy, the cost decreases rapidly after the first solution found within a few milliseconds. In contrast, it takes hundreds of milliseconds to find the first trajectory with uniformly random sampling, and the cost takes much longer time to approach the optimum. As also validated in this figure, given a time budget, our method generates more feasible samples due to the reasonable state distribution. Therefore, our method has a higher possibility of accessing a better solution and converges faster. Fig. 2 presents an illustrative sample of the comparison.

TABLE I
Front-End Comparison Results of 10-15 m Goals in 150 Obstacles Environment

Method	Comp. Time (ms)	Seg. Num.	Ctrl. Cost (m^2/s^3)	Traj. Dura. (s)	Traj. Len. (m)	Succ. Rate (%)
Proposed First Traj.	4.76	**4.05**	**24.97**	5.49	**13.21**	**96.01**
Zhou's	**4.58**	6.50	40.16	**5.33**	13.45	94.07

B. Quadrotor Planning System

We conduct benchmark comparisons against the state-of-the-art quadrotor online planning methods in three-folds: the front-end kinodynamic planning, the back-end trajectory optimization, and the integrated systematic results. Simulations are conducted in environments with different obstacle densities and starting-goal distances. The velocity and acceleration limits are set as 5 m/s and 6 m/s^2.

1) Comparisons of the Kinodynamic Planning: For the front-end, the first feasible trajectory found by our method is compared against Zhou's. As shown in Table I, our method finds trajectories with much lower control costs, shorter trajectory length, higher success rate, and comparable computing time. Since our method generates properly distributed state samples and explores the environment according to the topological structure, it finds a solution with fewer states expanded. Besides, it better exploits the results of BVPs and generates piece-wise linear inputs instead of the piece-wise constant ones of method [6], thus improves the smoothness. Although our method has slightly higher computing time, it provides a much better initial trajectory and thus significantly alleviates the computational burden of the back-end, as validated below.

2) Comparisons of the Trajectory Optimization: For fair comparisons, we use the same path returned by our front-end planner as the initial value for Zhou's [6] that adopts a B-spline formulation and optimize control points in a distance field with gradient descent to ensure safety, Richter's [11] that optimizes derivatives on waypoints through an unconstrained QP while adjusting time allocation by gradient descent and scaling, and Mellinger's [10] that optimizes time allocation with total duration fixed and use backtracking gradient descent. For Tordesillas's [14], front-end paths are found by informed-RRT*. For waypoints-based methods Richter's and Mellinger's, both trapezoidal time initialization and the time allocation produced by our kinodynamic front-end are used and compared (denoted as T and K, respectively). When collisions occur in a particular segment, a point from the collision-free front-end path is added as an additional waypoint. The trajectory is then re-optimized, and the process is repeated until the whole trajectory is collision-free. The stopping criterion for each iteration is set as 5 ms running time. The results are shown in Table II and Fig. 6. Our proposed method generates much smoother and shorter trajectories in much less time. This is because the compared ones adopt an underlying non-convex gradient-based formulation and require expensive computations for a general nonlinear optimization

TABLE II
BACK-END COMPARISON RESULTS

Method	Comp. Time (ms)	Inte. of Acc. (m^2/s^3)	Inte. of Jerk (m^2/s^5)	Traj. Dura. (s)	Traj. Len. (m)
Proposed	**2.82**	**18.72**	**36.17**	5.42	12.96
Zhou's	6.20	19.16	92.80	5.42	13.17
Richter's + T	24.79	77.94	505.95	5.74	13.22
Richter's + K	17.39	34.17	197.06	5.42	13.18
Mellinger's + T	26.92	34.53	78.77	6.01	13.19
Mellinger's + K	32.12	29.60	98.60	5,13	13.18
Tordesillas's	183.03	24.52	45.51	**4.97**	**12.57**

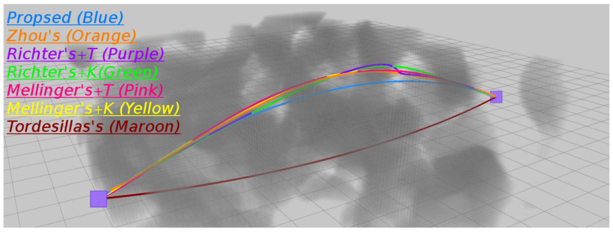
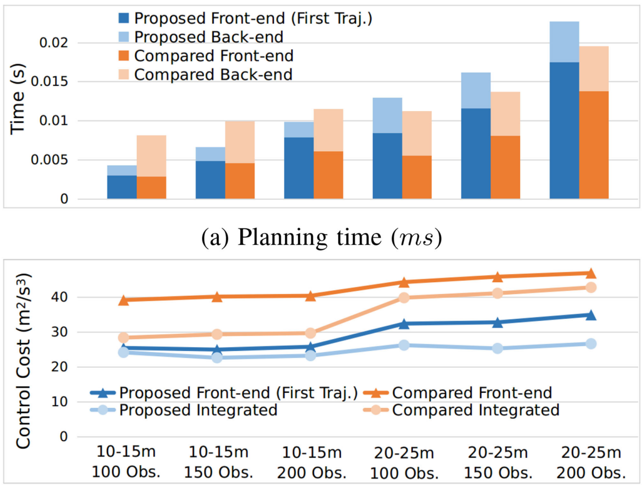


Fig. 6. An instance of trajectories generated by different methods. The 3D obstacles are set transparent to provide better views.



(b) Control effort cost (m^2/s^3)

Fig. 7. Comparison of the integrated results in environment of different obstacle densities and different distance goals.

solver to converge. Our method, however, enjoys the convex formulation to find the optimal solutions in its every iteration. [6] checks collision in an ESDF which requires extra computation (50 ms in the testing case). [14] builds a free corridor and perform MIQP with it. [11] and [10] do not account for collision in optimization thus may require many iterations to find a collision-free trajectory. Our method, however, avoids these by incorporating the collision-free front-end trajectory into the objective. Note that although the proposed method does not optimize time allocation, our final trajectory duration is less than Richter's and Mellinger's that use trapezoidal initialization, and is comparable to the ones that use the time allocation of our kinodynamic front-end.

3) Comparisons of the Integrated Results: For the integrated comparison, results of different scenarios are shown in Fig. 7. As an entire planning pipeline, our system generates trajectories with much lower control cost in each scenario and less time used in relatively short distances. However, as the goal distance and obstacle density increase, our method requires a bit more time than Zhou's method. For a planning problem with a large scale, samples near the goal are inferior to grow the tree since they are less likely to safely connect to an existing state, especially in complex environments. However, this is not critical since for common real-world applications, the sensing range and planning horizon of a lightweight drone are usually within 10 m, or even 5 m. It is verified in our real-world tests in Sec. VI.

VI. Flight Experiments

A. Experiment Settings

We conduct autonomous flight experiments in both indoor and outdoor unknown cluttered environments. The flight platform we use is a customized quadrotor equipped with a forward-facing RealSense D435i[2] and an N3 flight controller[3] for depth sensing and flight control. Collision checking is done with occupancy grid maps [23] fused by the depths and the poses estimated. Unknown space is treated as free. Replan is conducted when obstacles are newly perceived or new goals are set. All the computations are done online with an onboard computer Manifold2-C.[4]

B. Waypoints Navigation

The quadrotor, with limited sensing range (3 m) and field of view (60°), navigates to a goal of about 50 m and 15 m away and then come back in the outdoor and indoor flight tests, respectively. The executed trajectories are depicted in Fig. 8 and Fig. 9. In the outdoor flight, the quadrotor operates in previously unknown dense and unstructured woods. In the indoor environment, the obstacles are more massive and cause more occlusions. Thus some obstacles are more likely to appear suddenly. In these experiments, our planner shows its capability to facilitate autonomous navigation while avoiding obstacles. More details are available in the video.

C. Fast Replan Tasks

To further challenge our planner and test the replan performance, we conduct tasks with continually changing goals for the quadrotor to chase in unknown cluttered woods. Replan happens whenever the goal changes or the current tracking path is blocked by a newly detected obstacle. In the first task, the quadrotor is made to chase after a fast-moving target, a QR code board, which determines the goal position (See Fig. 1(b)). In the second task, the goals are set and changed arbitrarily and abruptly at any time during flight by an operator. Our drone keeps a speed over 3 m/s while planning new trajectories as soon as newly observed obstacles block the current flight trajectory. Higher speed can be achieved with longer confidence sensing range and less latency, which is mainly caused by map fusion. We refer readers to the video for more flight tests.

[2][Online]. Available: https://www.intelrealsense.com/depth-camera-d435i/
[3][Online]. Available: https://www.dji.com/cn/n3
[4][Online]. Available: https://www.dji.com/cn/manifold-2

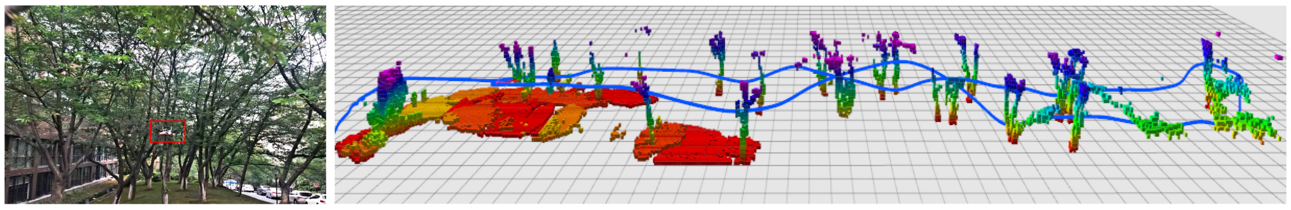


Fig. 8. Outdoor flights. The quadrotor flies about 100 m with an average speed of about 3 m/s. Velocity profiles are plotted in the video.

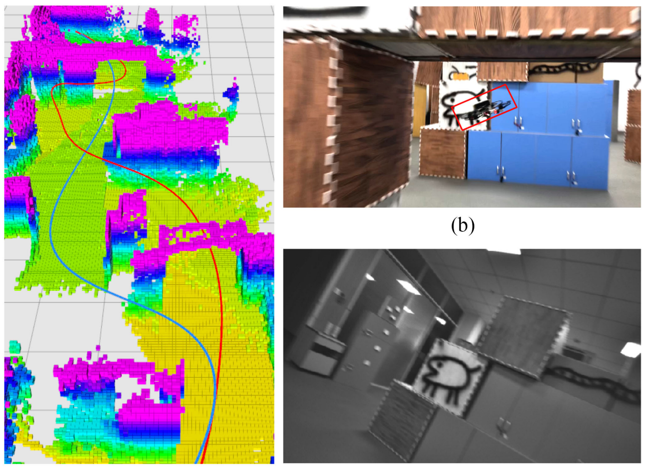


Fig. 9. Indoor flights. (a) The trajectory planned back (blue) is smoother than the departure trajectory (red) since it has seen part of the environment in the previous flight. (b) The quadrotor makes a turn when facing a wall right after flying through a gate. The average speed is about 2.5 m/s. (c) The first person view.

VII. Conclusion

In this letter, a novel online motion planning framework for quadrotor fast flight is proposed. The method is composed of 1) a guided sampling-based kinodynamic planner for finding an initial safe, kinodynamiclly feasible and time-energy optimal trajectory and 2) a homotopy penalized, soft constrained, iterative optimizer to further improve the smoothness and continuity of the trajectory. Benchmark comparisons show that our method outperforms the state-of-the-art methods in both efficiency and optimality. Moreover, we validate our method in simulated and real-world challenging tasks. In the future, we plan to further improve the obstacle clearance of the refined trajectory and challenge our method for large-scale problems.

References

[1] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *Proc. IEEE Intl. Conf. Robot. Autom.*, 2017, pp. 5759–5765.
[2] M. Ryll, J. Ware, J. Carter, and N. Roy, "Efficient trajectory planning for high speed flight in unknown environments," in *Proc. IEEE Intl. Conf. Robot. Autom.*, 2019, pp. 732–738.
[3] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proc. IEEE Intl. Conf. Robot. Autom.*, 2011, pp. 1478–1483.
[4] D. J. Webb and J. van denBerg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE Intl. Conf. Robot. Autom.*, May 2013, pp. 5054–5061.
[5] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robot. Automat. Lett.* vol. 2, no. 3, pp. 1688–1695, Jul. 2017.
[6] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.
[7] H. Cover, S. Choudhury, S. Scherer, and S. Singh, "Sparse tangential network (spartan): Motion planning for micro aerial vehicles," in *Proc. IEEE Intl. Conf. Robot. Autom.*, May 2013, pp. 2820–2825.
[8] F. Blöchliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual slam maps," in *Proc. IEEE Int. Conf. Robot. Automat.*, Brisbane, QLD, 2018, pp. 3818–3825, doi: 10.1109/ICRA.2018.8460641.
[9] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3 d topological graphs for micro-aerial vehicle planning," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.
[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Intl. Conf. Robot. Autom.*, Shanghai, China, May. 2011, pp. 2520–2525.
[11] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. Intl. Sym. Robot. Research*, Dec. 2013, pp. 649–666.
[12] 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.
[13] F. Gao, L. Wang, B. Zhou, L. Han, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," 2019, *arXiv:abs/1907.00520*.
[14] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, 2019, pp. 1934–1940.
[15] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Intl. Conf. Robot. Autom.*, May 2009, pp. 489–494.
[16] 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 Intl. Conf. Intell. Robots Syst.*, Daejeon, Korea, Oct. 2016, pp. 5332–5339.
[17] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 3681–3688.
[18] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and a 3 d circular buffer," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 215–222.
[19] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*, Princeton University Press, 2012.
[20] T. Simon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Adv. Robot.*, vol. 14, no. 6, pp. 477–493, 2000.
[21] 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.
[22] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4836–4843, Jul. 2020.
[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.