An Efficient B-Spline-Based Kinodynamic Replanning Framework for Quadrotors

Wenchao Ding, Wenliang Gao, Kaixuan Wang, and Shaojie Shen

***Abstract*—Trajectory replanning for quadrotors is essential to enable fully autonomous flight in unknown environments. Hierarchical motion planning frameworks, which combine path planning with path parameterization, are popular due to their time efficiency. However, the path planning cannot properly deal with nonstatic initial states of the quadrotor, which may result in nonsmooth or even dynamically infeasible trajectories. In this article, we present an efficient kinodynamic replanning framework by exploiting the advantageous properties of the B-spline, which facilitates dealing with the nonstatic state and guarantees safety and dynamical feasibility. Our framework starts with an efficient B-spline-based kinodynamic (EBK) search algorithm, which finds a feasible trajectory with minimum control effort and time. To compensate for the discretization induced by the EBK search, an elastic optimization approach is proposed to refine the control point placement to the optimal location. Systematic comparisons against the state-of-the-art are conducted to validate the performance. Comprehensive onboard experiments using two different vision-based quadrotors are carried out showing the general applicability of the framework.**

***Index Terms*—Aerial systems: perception and autonomy, collision avoidance, motion and path planning, trajectory planning.**

I. Introduction

AUTONOMOUS navigation for quadrotors in unknown environments has gained significant interest for its practical usage in various inspection and exploration tasks. To fulfill the need of fully autonomous exploration in unknown environments, trajectory replanning is of great significance. Replanning requires a real-time response to unexpected obstacles to guarantee safety while satisfying the low-level feasibility constraints induced by the nontrivial dynamics.

Many existing methods [1]–[5] tackle this challenging problem using a hierarchical framework, which first finds a geometric path and then locally optimizes the path to a dynamically feasible trajectory with respect to a given time allocation. Although this framework is efficient, inadequacy exists between the path finding and the local path parameterization. Specifically, the parameterization may be restricted by the geometric path to a homotopy class that does not contain a globally optimal (or even feasible) solution, especially when faced with nonstatic initial states (nonzero velocities or any other higher order derivatives), as shown in Fig. 1.

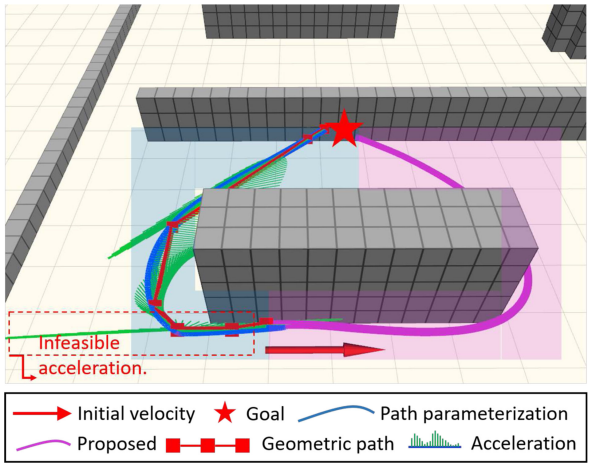


Fig. 1. Illustration of the motivating example. The initial state has nonzero velocity (*red arrow*). The traditional geometric planner finds the shortest path (red squares) and then parameterizes it using a piecewise polynomial (*blue*). However, the local path parameterization is restricted to a homotopy class (*blue* area), and the resultant trajectory is jerky (even infeasible) with respect to the given time allocation. In contrast, our framework produces a dynamically feasible trajectory (*purple line*) using kinodynamic planning.

To address the limitations of the hierarchical methods, it is essential to use kinodynamic motion planners, which directly find time-parameterized trajectories that are globally optimal with respect to control efforts and dynamical limits. The incorporation of kinodynamic planners into replanning facilitates dealing with nonstatic initial states and enhances replanning consistency.

Sampling-based motion planning algorithms, such as rapidly exploring random trees (RRTs) [6] and their variants [7]–[11], are popular in the kinematic/kinodynamic planning literature. Asymptotical optimality has been proved for some of them [7]–[10]. However, when applied to complex kinodynamic systems, they typically require solving a computationally expensive nonlinear two-point boundary value problem (BVP) [12], [13] and cannot run in real time. Liu *et al.* [14] explored a search-based counterpart and proposed a heuristic-guided resolution-complete (optimal with respect to discretization) search method using linear quadratic minimum time control. However, for high-order kinodynamic systems or a large dynamic range, the run-time efficiency is inadequate.

Manuscript received December 1, 2018; accepted June 4, 2019. Date of publication August 23, 2019; date of current version December 3, 2019. This article was recommended for publication by Associate Editor K. Hauser and Editor T. Murphey upon evaluation of the reviewers' comments. This work was supported by Hong Kong Ph.D. Fellowship Scheme, HKUST-DJI Joint Innovation Laboratory. *(Corresponding author: Wenchao Ding.)*

The authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: wdingae@connect.ust.hk; wenliang.gao@ust.hk; kwangap@ust.hk; eeshaojie@ust.hk).

This article has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2019.2926390

1552-3098 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

In this article, we present a kinodynamic replanning framework, which addresses the efficiency bottleneck. Our framework starts with an efficient B-spline-based kinodynamic (EBK) search algorithm, which finds B-spline control points on a spatial grid. We introduce the novel concept of vertex tuple to keep the search problem simple and analyzable, which enables a thorough theoretical characterization of the problem. Built on top of the structure of the optimal solution, a graph aggregation technique is proposed to minimize the computation time through a controllable discretization of the search space. An offline-computable minimum inflation is adopted to avoid unnecessary collision checking and further accelerate the online search. Compared to state-of-the-art methods (see Section VIII), our kinodynamic search finds the lowest-cost dynamically feasible trajectories in real time.

To compensate for the discretization in the search, an elastic optimization (EO) approach is proposed to refine the control point placement to the optimal location, by solving a convex quadratically constrained quadratic programming (QCQP) problem. The two components are integrated into a receding horizon replanner based on the local control property of the B-spline.

Our replanning framework is not only theoretically analyzed, but also validated in simulated and onboard experiments. The superior performance is shown through comprehensive comparisons against the state-of-the-art kinodynamic planning and trajectory optimization methods. Moreover, the practical impact of our framework is verified through onboard experiments using a monocular-vision-based quadrotor and a dual-fisheye vision-based quadrotor in unknown indoor and outdoor environments. We summarize our contributions as follows.

1) An EBK search algorithm that provides an initial-state-aware dynamically feasible time-parameterized trajectory.
2) An EO approach that refines the control point placement to the optimal location while preserving the safety and dynamical feasibility.
3) Systematic comparisons against the state-of-the-art showing the superior performance of the proposed framework.
4) Integration of our framework into a real monocular-vision-based quadrotor and a dual-fisheye vision-based quadrotor as well as extensive experiments demonstrating fully autonomous navigation in unknown, complex indoor, and outdoor environments.

A basic version of our framework was originally presented in [15], where we introduced the real-time B-spline-based kinodynamic (RBK) search. Although the RBK search achieves high computational efficiency, the absence of theoretical optimality analysis in [15] limits confidence in its solution quality and further limits its theoretical impact. In this article, instead of directly developing efficient methods, we tackle the kinodynamic search problem in a systematic way: 1) we first characterize the complexity and optimal solution of the search problem using a novel vertex tuple structure; and 2) we then establish the quality-efficiency tradeoff using a novel graph aggregation technique, which provides a user-specified parameter to control algorithm efficiency and solution quality. The abovementioned two theoretical additions render the more flexible and theoretically reliable EBK search. It also turns out that the preliminary version in [15] is perfectly contained in the EBK search. Apart from the theoretical additions, more comprehensive experimental analyses in a wide variety of environments are presented to support the new characteristics.

The relevant literature is discussed in Section II. An overview of the proposed replanning framework is provided in Section III. Mathematical background and advantageous properties of B-spline are introduced in Section IV. The problem formulation and algorithm detail of the EBK search are elaborated in Section V. The EO approach is presented in Section VI. Implementation details are given in Section VII. Systematic comparisons against the state-of-the-art methods are provided in Section VIII, and onboard experimental results are illustrated in Section IX. Finally, Section X concludes this article.

II. Related Works

There is an extensive literature on motion planning techniques for quadrotors from various perspectives, such as control-based methods [16]–[18], search-based methods [14], [19]–[21], sampling-based methods [7], [9], [22]–[24], and optimization-based methods [2], [5], [25], [26]. It is difficult to give a full literature review of all these techniques, so in this section, we choose the most relevant and organize them into two categories, namely, hierarchical motion planning techniques and kinodynamic motion planning techniques.

A. Hierarchical Motion Planning

Hierarchical motion planning refers to a high-level geometric path planner coupled with a low-level time parameterization scheme. The high-level geometric planner is concerned with finding an obstacle-free path, while the low-level parameterization scheme takes care of the vehicle dynamical constraints and generates a time-parameterized trajectory for execution. For quadrotors that have nontrivial dynamics, directly generating a trajectory in the high-dimensional state space is time consuming, while smoothing a given geometric path is computationally efficient (with a suitable relaxation) [27]. As such, the hierarchical framework is popular for quadrotors, and it enables a number of online methods [1]–[5].

Two pioneering works [1], [3] extract waypoints from the geometric path and formulate the trajectory generation problem as quadratic programming (QP) on polynomial coefficients. These methods are based on the differential flatness of the quadrotor [1]. Due to the deviation of the polynomial trajectory from the straight-line collision-free path, an iterative waypoint insertion scheme is adopted [3]. However, how many additional waypoints are needed is not quantified. Chen *et al.* [5] proposed a corridor-based geometric planner based on the octree-based map structure [28]. The control effort can be reduced by generating the trajectory in a series of connected cubes. Apart from that, they propose an iterative process of adding constraints on polynomial extremas to cope with the deviation from the corridor, and prove that a finite number of iterations is needed to guarantee safety. Liu *et al.* [4] further generalize the corridor representation to a series of connected convex polygons.

Although the hierarchical methods have made significant achievements, they suffer from the common problem that the

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

geometric planner is unaware of the vehicle dynamics, resulting in inadequacy between the path planning and path parameterization, especially when faced with nonstatic initial states. The example in Fig. 1 motivates us to explore the problem from the kinodynamic planning perspective, which is discussed in Section II-B.

B. Kinodynamic Motion Planning

The kinodynamic motion planner directly explores the high-dimensional state space, and outputs a time-parameterized trajectory, which fundamentally avoids the inadequacy between path planning and parameterization. RRTs [6] and their variants [7]–[11], [22] were originally designed for kinematic systems and can be easily extended to kinodynamic systems. These methods provide an efficient way of exploring the high-dimensional state space, and some of them possess asymptotical optimality [7]–[10]. However, for robots with nontrivial dynamics, the tree expansion typically involves solving the BVP, which is nonlinear and challenging. Webb and Van Den Berg [7] proposed a fixed-final-state-free-final-time optimal controller, which solves the BVP for linear (or linearized) controllable systems in closed form. Xie *et al.* [12] proposed an efficient BVP solver for general kinodynamic systems using sequential quadratic programming. Li *et al.* [13] worked in another direction, namely, expanding the tree using random control propagation, for cases where system models are complex and BVP solvers are not available.

Despite the fact that the efficiency of the kinodynamic planning techniques keeps improving [12], [13], it is still prohibitively expensive for replanning. Allen and Pavone [23] worked toward a real-time kinodynamic planning framework by combining FMT* [10] with a support vector machine for the classification of the reachable set. This framework [23] reduces the calling of the BVP solver to gain efficiency. However, the solution quality largely depends on the number of states presampled. On the other hand, Liu *et al.* [14] explored the search-based kinodynamic planning counterpart and develop efficient heuristics by solving a linear quadratic minimum time problem. Their solution is resolution-complete with respect to the discretization on the control input, and achieves near real-time performance. Note that both [14] and [23] used a simplified system model, i.e., a double or triple integrator, to reduce the computation complexity. However, the resultant trajectory only has limited continuity. To improve the smoothness, both [14] and [23] adopted trajectory reparameterization using the unconstrained QP formulation [3], which may break the dynamical feasibility and safety.

In contrast, the proposed EBK search adopts a high-order B-spline parameterization with continuity up to snap, which can be directly used to control the quadrotor. Moreover, the advantageous properties of the B-spline facilitate the kinodynamic replanning as follows.

1) Local control property for incrementally constructing the B-spline trajectory in the kinodynamic search and local refinement of the trajectory during replanning.
2) Convex hull property for enforcing collision-free constraints and providing a dynamical feasibility guarantee for the entire trajectory.

Given the same run-time budget according to the real-time requirement, the EBK search finds a lower-cost trajectory, as validated by the comprehensive comparisons against the state-of-the-art in Section VIII. Apart from this, the proposed refinement module, namely, the EO approach, can preserve the safety and dynamical feasibility by taking advantage of the convex hull property of the B-spline.

III. Overview

The structure of the proposed framework is shown in Fig. 2. The replanning framework is built on top of the state estimation and dense mapping module, which are discussed in Section VII. The frequency of the grid map update is 10 Hz. As such, the real-time requirement in this article refers to a run-time of less than 100 ms. The updated map and the initial state of the quadrotor are fed to the EBK search module (see Section II-B), and the replanning strategy is elaborated in Section VII. The control points are constantly refined by the proposed EO approach (see Section VI), which consists of an elastic tube expansion module (see Section VI-A) for free-space characterization and a convex optimization formulation (see Section VI-B) for trajectory refinement. The confirmed control points are evaluated, and position commands are generated accordingly.

IV. B-Spline Curve and Replanning

For the proposed kinodynamic planning framework, we adopt a B-spline parameterization for its advantageous properties, namely, local control and convex hull property, and we further adopt the uniform B-spline for its convenient closed-form evaluations. In this section, we elaborate these properties and explain how they can be applied to the replanning system.

Given $n+1$ control points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$ and knot vector $\{t_0, t_1, \ldots, t_m\}$, the B-spline curve $\mathbf{s}(t)$ of degree k is defined as follows:

$$\mathbf{s}(t) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,k}(t), \tag{1}$$

where $N_{i,k}(t)$ is the B-spline blending function of degree k, which can be evaluated recursively as follows:

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t). \tag{2}$$

The total number of knots should satisfy $m+1 = n+k+2$. The uniform B-spline is a special type of B-spline whose knot vector is uniformly distributed. Suppose the knot vector is separated with equidistance Δ_t. The half-open interval $[t_i, t_{i+1})$ is called the ith knot span. We normalize each knot span using $u = (t - t_i)/\Delta_t$, and for the ith knot span, only $k+1$ blending functions are nonzero, corresponding to $k+1$ control points $\mathbf{p}_{i-k}, \ldots, \mathbf{p}_i$. We stack the $k+1$ control points and call the stacked coordinate matrix a control point span $\mathbf{P}_{i-k} := [\mathbf{p}_{i-k}\, \mathbf{p}_{i-k+1} \cdots \mathbf{p}_i]^\intercal \in \mathbb{R}^{(k+1)\times 3}$. Since the blending functions $N_{i,k}(t)$ are shifted versions of each other for the uniform B-spline, we have closed-form matrix representations [29] for parametric evaluation. Let $j = i - k$, and the position and the derivatives of the B-spline curve corresponding to the jth

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

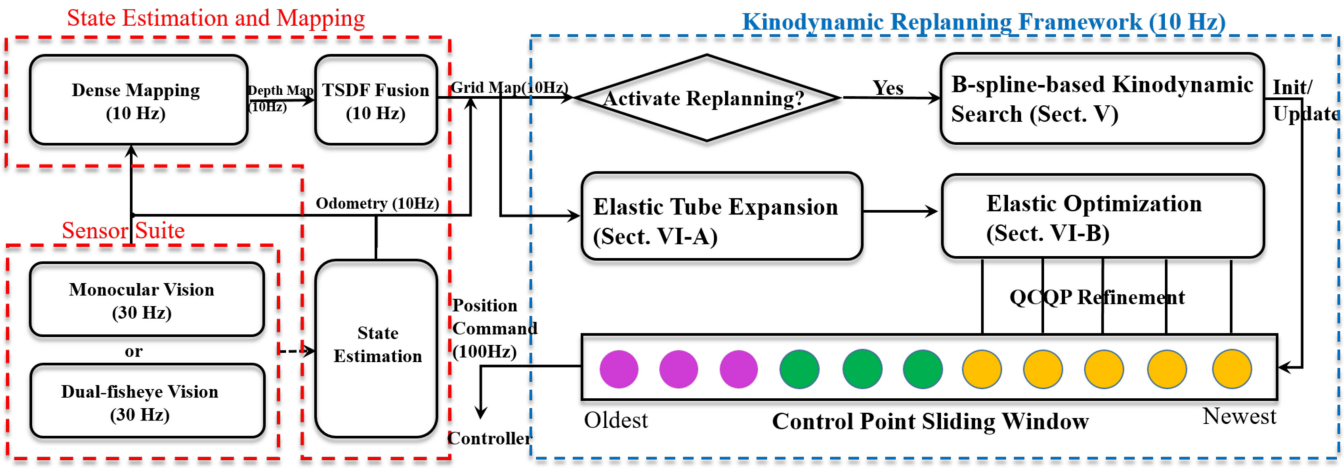


Fig. 2. Diagram of our kinodynamic replanning framework together with state estimation and mapping modules.

control point span can be evaluated as follows:

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \frac{1}{(\Delta t)^l}\frac{d\mathbf{b}^\intercal}{d^l u}\mathbf{M}_k\mathbf{P}_j, \tag{3}$$

where l denotes the order of the derivative ($l=0$ means the position), $\mathbf{b} = \begin{bmatrix}1\, u\, u^2\, \cdots\, u^k\end{bmatrix}^\intercal \in \mathbb{R}^{k+1}$ denotes the basis vector, and $\mathbf{M}_k = (m_{i,j}) \in \mathbb{R}^{(k+1)\times(k+1)}$ denotes the blending matrix, where $m_{i,j} = \frac{1}{k!}\binom{k}{k-i}\sum_{s=j}^{k}(-1)^{s-j}\binom{k+1}{s-j}(k-s)^{k-i}$. According to (3), the evaluation of the derivatives of the B-spline curve can be expressed by a linear matrix multiplication in terms of the control point span $\mathbf{P}_j$. The article uses a quintic uniform B-spline ($k=5$) to ensure the continuity up to snap for controlling quadrotors.

As described by Mellinger [1], the control cost of a quadrotor is closely related to the integral over squared derivatives of the planned trajectory, which can also be evaluated in closed form in the case of the uniform B-spline. The total control cost E_j^l of the jth control point span can be expressed by the integral over the squared derivatives of degree l (e.g., for the min-snap trajectory, $l=4$) as follows:

$$E_j^l = \int_0^1 \left(\frac{d\mathbf{s}_j(u)}{d^l u}\right)^2 du = \mathbf{P}_j^\intercal \mathbf{M}_k^\intercal \mathbf{Q}_l \mathbf{M}_k \mathbf{P}_j, \tag{4}$$

where $\mathbf{Q}_l = \int_0^1 (\frac{d\mathbf{b}}{d^l u})(\frac{d\mathbf{b}}{d^l u})^\intercal du/(\Delta_t)^{2l-1}$ is the Hessian matrix of the lth squared derivative, which is constant for the uniform B-spline. The control cost E_j^l is quadratic with respect to the control point span $\mathbf{P}_j$. Note that the cost evaluation of a span only depends on the stacked control point coordinates of this span.

A. Local Control Property and Replanning

The *local control* is one of the important properties of B-spline, making it suitable for replanning. Specifically, the evaluation of any point of the B-spline curve is controlled by a single control point span containing $k+1$ control points, and any control point only affects $k+1$ control point spans. We incorporate the local control property into a receding horizon (re-)planner, and we divide the planned trajectory into three types, namely, executed trajectory, executing trajectory, and optimizing trajectory. The executed trajectory means the part of the trajectory, which has already been executed, the executing trajectory means the part of the trajectory corresponding to the control point span being executed, and the optimizing trajectory means the part of the trajectory whose supporting control points are potentially under optimization.

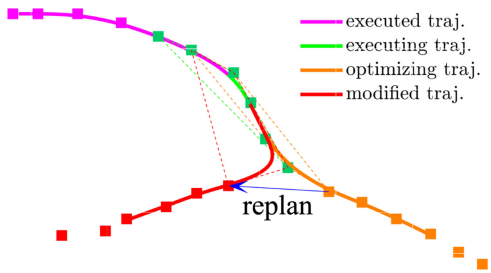


Fig. 3. Illustration of the B-spline local control property and its application to the replanning system. The control points are shown by squares. The control points corresponding to the executing trajectory are shown in *green*. The original locations of the control points ahead of the executing control point are in *orange*, and their subsequently modified positions are marked in *red*. Due to the locality of the B-spline, the replan causes no perturbation to the executing trajectory.

Thanks to the local control property, modification of the supporting control points of the optimizing trajectory will not affect the evaluation of the executing trajectory, as shown in Fig. 3. Unlike [30] and [31] where local reshaping may cause the violation of dynamical constraints, the dynamical feasibility of the executing trajectory can be preserved by leveraging the local control property. Moreover, the locality also makes it possible to optimize any subset of control points without regeneration of the whole trajectory, which is computationally efficient. The locality also helps to preserve a smooth trajectory since the next executing control point span always shares k control points with the current executing control point span, yielding kth-order continuity and a consistent trajectory.

B. Convex Hull Property and Dynamical Feasibility

Another important property of the B-spline is the *convex hull* property. A B-spline (or Bèzier [32]) trajectory is strictly bounded inside the convex hull supported by the corresponding control point span. Strictly speaking, dynamical feasibility

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

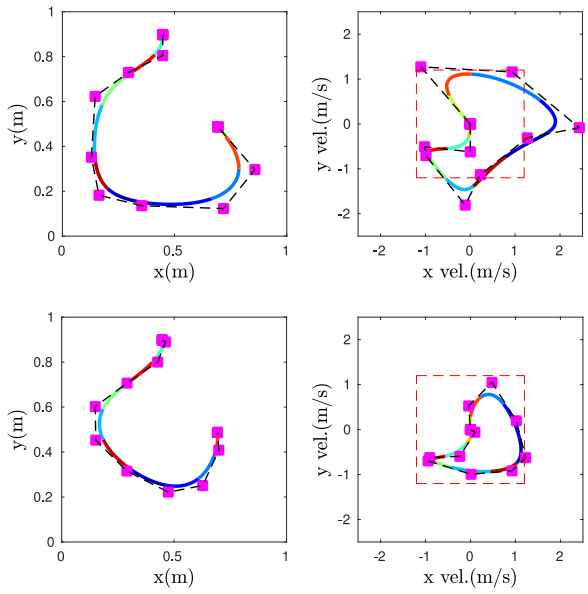


Fig. 4. Illustration of the convex hull property. The dashed red box shows the feasible velocity hull (1.2 m/s for each axis). Applying Proposition 1 under the objective of minimum change to control point positions (bottom left figure), the resulting velocity profile is shown at the bottom right, where the velocity profile is strictly bounded.

should be induced by the robot's kinematic and dynamical constraints. Given polynomial/spline parameterization, a common practice to enforce dynamical feasibility is to use maximum velocity and maximum acceleration bounds [4], [33], [34]. We follow this practice in this article. Note that for piecewise polynomial parameterization methods [2], [4], the dynamical feasibility constraints are enforced on a finite number of checkpoints. Denser checkpoints will enhance the robustness but yield higher computation complexity. In contrast, by using the convex hull property, the *entire* velocity and acceleration profile can be strictly bounded.

We utilize the fact that the derivative of the B-spline of degree k is a B-spline of degree $k-1$, which also enjoys the convex hull property. Therefore, if the supporting control points are bounded inside the convex hull expanded by the allowed maximum derivative, the derivative spline is subsequently bounded, as elaborated in Proposition 1. Note that Proposition 1 is a sufficient but not necessary condition. A toy example illustrating the relation between the convex hull property and dynamical feasibility is shown in Fig. 4.

Proposition 1: Given a uniform B-spline of degree k and knot separation Δ_t, there exists a constant linear combination $\mathbf{S}_l = \mathbf{M}_k^{-1}\mathbf{C}_l\mathbf{M}_k/(\Delta_t)^l \in \mathbb{R}^{(k+1)\times(k+1)}$ such that $u_{l,D}^{\min}\mathbf{1}_{(k+1)\times 1} \leq \mathbf{S}_l\mathbf{P}^D \leq u_{l,D}^{\max}\mathbf{1}_{(k+1)\times 1}$ is a sufficient condition for the derivative along coordinate D to be thoroughly bounded, i.e., $u_{l,D}^{\min} \leq \frac{d\mathbf{s}^D(u)}{d^l u} \leq u_{l,D}^{\max}$, $\forall u \in [0,1]$, where $\mathbf{P}^D \in \mathbb{R}^{k+1}$ is coordinate $D \in \{x,y,z\}$ of the control point span $\mathbf{P}$, and $\mathbf{C}_l \in \mathbb{R}^{(k+1)\times(k+1)}$ is a constant mapping matrix of the lth derivative satisfying $\frac{d\mathbf{b}}{d^l u} = \mathbf{C}_l\mathbf{b}$.[1]

Proof: Please refer to Appendix A for the detailed proof. ■

[1] $\mathbf{S}_l$ and $\mathbf{C}_l$ are fixed given the degree k and the derivative order l.

V. B-Spline-Based Kinodynamic Search

A. Motivating Example

As shown in the motivating example in Fig. 1, hierarchical motion planners may produce suboptimal or even dynamically infeasible trajectories given a nonstatic initial state of the quadrotor. The reason is that the geometric planner has no knowledge about the vehicle dynamics and restricts the solution space of path parameterization to a *homotopy class* of the geometric shortest path. The inadequacy motivates us to propose an efficient kinodynamic planning algorithm, which can work in real time. However, kinodynamic planning is typically time consuming [14], [23]. The major computation of the traditional kinodynamic planning lies in three tasks, namely, covering the large state space, solving the BVP and collision checking.

Given the advantageous properties of the B-spline introduced in Section IV, we propose using uniform B-spline parameterization in kinodynamic planning, which facilitates reducing the computation time for the abovementioned three tasks. Specifically, we propose a spatial-grid-based deterministic graph search to place B-spline control points. The proposed search algorithm has three major features as follows.

1) Controllable discretization of the state space: Due to the locality of the B-spline, it is possible to incrementally sample the B-spline control points during the search. A vertex tuple structure is proposed to recover the Markovian assumption and make the problem analyzable. A novel graph aggregation technique is proposed to control the discretization of the state space, which achieves a speed-quality tradeoff.
2) Closed-form evaluations of control cost and dynamical feasibility: The control cost and feasibility of the B-spline can be evaluated in closed forms efficiently.
3) Offline-computable inflation to avoid collision checking: The maximum deviation of uniform B-spline from the free-cells can be characterized offline and compensated for by workspace inflation.

The kinodynamic search accounts for the total control efforts and dynamical limits, which is a systematic way to deal with the nonstatic initial states in replanning.

B. Problem Formulation

In this section, we formally present the problem of the B-spline-based kinodynamic search on a spatial grid. The problem is formalized as a deterministic graph search where the action is the placement of the control points. Suppose the topological graph associated with the grid map is denoted as $\mathcal{G} := (V, E)$, where V is the set of vertices denoting the collection of free cells and E denotes the set of edges $(i,j) \subset V \times V$ between all adjacent vertices i and j. The adjacency of the vertices depends on the grid connectivity adopted. In this article, every cell in the three-dimensional (3-D) grid has 26 neighbors, which are cells connected to this cell at a Chebyshev distance of 1.[2]

Given uniform B-spline parameterization of degree k and knot separation Δ_t, the proposed search method finds a finite

[2] Note that the connectivity can also be defined based on a Chebyshev distance larger than one, which will result in a nonuniform control point placement and a higher computational complexity.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

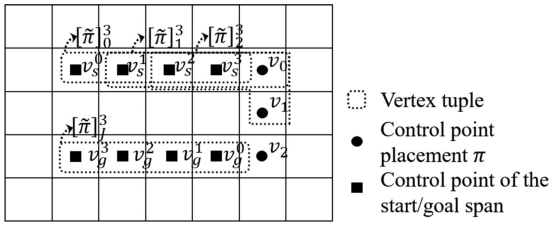


Fig. 5. Illustration of the construction process of three-degree vertex tuples from an admissible path. Each vertex tuple is formed by combining four consecutive control points. $[\tilde{\pi}]_0^3$ and $[\tilde{\pi}]_1^3$ are two neighboring vertex tuples since they overlap for three vertices.

sequence $\pi = (v_0, v_1, \ldots, v_T)$ of vertices representing an *admissible* control point placement, which satisfies $v_i \in V, (v_{i-1}, v_i) \in E$ for each $i = 1, \ldots, T$ and connects the given initial state $\pi_s = (v_s^0, v_s^1, \ldots, v_s^k)$ and goal state $\pi_g = (v_g^0, v_g^1, \ldots, v_g^k)$, i.e., $(v_s^k, v_0) \in E$ and $(v_T, v_g^0) \in E$. The sequence π possibly contains repetition since we allow placing the control points at the same cell. π_s and π_g are two tuples, both containing $k+1$ vertices, which form the control point span according to the definition of the B-spline in Section IV. Therefore, π_s and π_g actually represent two short trajectories, different from the initial and goal positions used in geometric planners and the position-velocity-acceleration state vector used in these kinodynamic planners [7], [14], [23].

Since the B-spline is evaluated in terms of the control point span, we reorganize π by combining neighboring $k+1$ vertices as one vertex tuple. Combining the sequence π with π_s and π_g, the overall sequence can be formalized as $\tilde{\pi} = (v_s^0, \ldots, v_s^k, v_0, \ldots, v_T, v_g^0, \ldots, v_g^k)$. We define the *ordered* subsequence containing consecutive $k+1$ vertices of $\tilde{\pi}$ as a *k-degree vertex tuple*, which is denoted by $[\tilde{\pi}]^k$. We provide a toy example in Fig. 5 showing how three-degree vertex tuples can be constructed.

We denote by $[\tilde{\pi}]_j^k$ the jth k-degree tuple in $\tilde{\pi}$, and two neighboring tuples, $[\tilde{\pi}]_{j-1}^k$ and $[\tilde{\pi}]_j^k$, overlap for k vertices. According to this convention, $[\tilde{\pi}]_0^k = \pi_s$ and $[\tilde{\pi}]_J^k = \pi_g$, where $J + 1 = k + T + 3$. Each vertex tuple represents a short trajectory, and a sequence of neighboring vertex tuples represents a continuous trajectory. We associate with each k-degree tuple $[\tilde{\pi}]$ a *strictly positive* cost function $f_{k,\Delta_t} : [\tilde{\pi}]^k \to \mathbb{R}_+$. Note that the cost function has to be strictly positive to cope with the possible repetition in π. We state the problem as follows.

Problem 1: Given a uniform B-spline of degree k and knot separation Δ_t, initial state π_s and goal state π_g, find admissible control point placement $\pi = (v_0, v_1, \ldots, v_T)$ on $\mathcal{G}$ such that the following cost function is minimized:

$$\mathcal{J}_{k,\Delta_t}(\tilde{\pi}) = \sum_{j=0}^{J} f_{k,\Delta_t}([\tilde{\pi}]_j^k),$$

where $[\tilde{\pi}]_j^k$ is a k-degree vertex tuple, whose corresponding trajectory should satisfy collision-free and dynamical feasibility requirements.

We adopt a cost function following the idea of the linear quadratic minimum time control problem [14], [35], where the control cost is represented by (4) with a tradeoff penalty on the execution time Δ_t. Mathematically, the cost function f_{k,Δ_t} is represented as follows:

$$f_{k,\Delta_t}([\tilde{\pi}]_j^k) = \lambda \Delta_t + \int_0^1 \left(\frac{d\mathbf{s}_{[\tilde{\pi}]_j^k}(u)}{d^l u} \right)^2 du, \tag{5}$$

where $[\tilde{\pi}]_j^k$ can be rewritten into matrix form, as in Section IV; the integral can be evaluated according to (4); and λ is the weight for the trajectory execution time. The criterion to check the collision-free and dynamical feasibility of the vertex tuple is introduced in Section V-D.

C. Optimal B-Spline-Based Kinodynamic Search

The difficulty of solving Prob. 1 is that the placement of any control point depends on the placed k control points (not only the predecessor) within the vertex tuple. Regarding the placement of a single control point as an action, the Markovian assumption does not hold, which makes traditional graph search algorithms inapplicable. However, we find that Problem 1 can be transformed into an equivalent standard shortest path problem on a higher dimensional directed graph $\mathcal{G}_H = (V_H, E_H)$ induced by $\mathcal{G}$ by regarding the whole vertex tuple as a "vertex." Since the vertex tuple is the basic evaluation unit of the B-spline, the placement of the next vertex tuple will only depend on its predecessor, which follows the Markovian assumption. The transformation enables the usage of well-characterized shortest path search algorithms, such as Dijkstra's [36] and A* [37]. In the following, we elaborate the transformation.

Similar to the construction process of the vertex tuple in Section V-B, we construct $\mathcal{G}_H$ as follows: 1) each vertex tuple $\hat{v}_i = (v_i, v_{i+1}, \ldots, v_{i+k}) \in V_H$ is created by combining a sequence of adjacent vertices of $\mathcal{G}$ satisfying $(v_{j-1}, v_j) \in E$ for $j = i+1, \ldots, i+k$; and 2) two vertices $\hat{v}_i$ and $\hat{v}_j$ on V_H are adjacent if and only if the last k vertices of $\hat{v}_i$ overlap with the first k vertices of $\hat{v}_j$. The construction of $\mathcal{G}_H$ groups the associated control point coordinates into a high-dimensional state. Note that each dimension of the combined state has the same physical meaning, i.e., the spatial coordinates of the control point. This observation motivates us to come up with a low-dispersion search algorithm, as presented in Section V-E.

In Fig. 6(a), we show an example of how the trace of control points on $\mathcal{G}$ can be mapped to the path on $\mathcal{G}_H$. The initial vertex tuple $\hat{v}_s$ is shown by squares. Starting from $\hat{v}_s$, we consider two directions of the next-step placement, which form $\hat{v}_0$ and $\hat{v}_1$, respectively. In a similar way, starting from $\hat{v}_0$, two expansion directions are considered, forming $\hat{v}_3$ and $\hat{v}_4$, while for $\hat{v}_1$, one direction ($\hat{v}_5$) is considered. Note that although the last vertex of $\hat{v}_4$ and $\hat{v}_5$ are in the same spatial cell, in the induced high-dimensional graph $\mathcal{G}_H$, they are two distinct vertex tuples. Following the expansion of control points, we form a tree of vertex tuples on $\mathcal{G}_H$. From the expansion process, we observe that, given the initial and goal vertex tuple, the problem of finding the optimal control point placement is equivalent to finding the shortest path on the graph $\mathcal{G}_H$. We refer interested readers to Appendix B for further details.

Recall that in Section V-B, we allow the repetition of vertices since some necessary vertex tuples rely on repetition; for instance, the same vertex being repeated $k+1$ times actually represents a static state (for Δ_t). According to the definition of

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

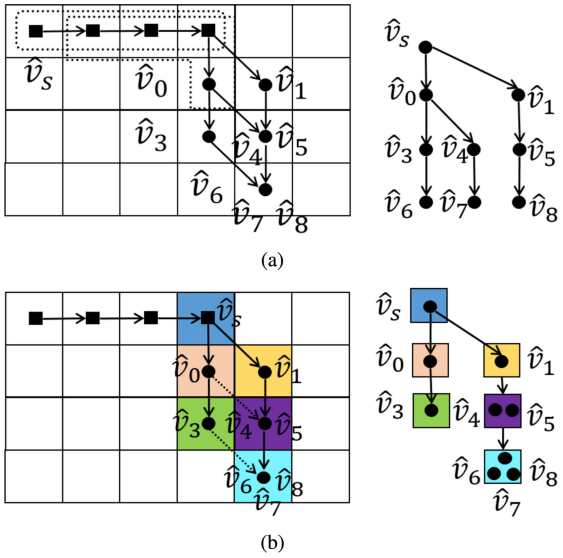


Fig. 6. Illustration of the mapping process to $\mathcal{G}_H$ and the graph aggregation process used by the EBK search. (a) Mapping from the traces on G to $\mathcal{G}_H$. (b) Graph aggregation.

TABLE I
ILLUSTRATION OF THE ON-DEMAND GRAPH CONSTRUCTION

Method	Run Time (s)	Actual # of Exp. Nodes	Total # of Nodes (Esti.)	Percentage (%)
EBK-D1	0.034	2,334	13,005	17.9
EBK-D2	0.383	36,818	351,135	10.5
EBK-D3	7.422	460,469	9,480,645	4.9
EBK-D4	173.710	9,096,338	255,977,415	3.6

$f_{k,\Delta_t}([\tilde{\pi}]^k)$, the cost of $\mathcal{G}_H$ is defined on vertices V_H instead of the edges. Although repetition is allowed, each vertex $\hat{v} \in V_H$ of $\mathcal{G}_H$ is associated with a *strictly positive* cost so that repetition is properly penalized.

Note that any path on $\mathcal{G}_H$ is a time-parameterized B-spline trajectory instead of a geometric path. The dynamical feasibility and control cost are taken into account by evaluating the corresponding short trajectories of the nodes of $\mathcal{G}_H$. Problem 1 can be solved optimally using traditional label-correcting algorithms such as Dijkstra's [36] and A* [37] on the induced graph $\mathcal{G}_H$. The optimal control point placement π^* can then be reconstructed. The optimal B-spline-based kinodynamic (OBK) search algorithm is outlined in Algorithm 1.

Typically, label-correcting algorithms maintain one or multiple sets of vertices as so-called *fringes* [38]. For example, A* [37] maintains two fringes (known as the *OPEN* set and the *CLOSED* set) to reduce the expansion of nodes and save computation. Similarly, Algorithm 1 maintains two fringes, namely, the *OPEN* set (as denoted by $\mathcal{O}$) and the *VISITED* set (as denoted by $\mathcal{L}$). The visited set $\mathcal{L}$ provides query and retrieving functions for vertices. We use INDEX$(\hat{v})$ (see Algorithm 2) to assign a unique integer index to each distinct vertex tuple. Specifically, Algorithm 2 collects the extracted coordinates $\mathcal{I}(\hat{v})$ (using the COORD(v) function) for each vertex v in the tuple $\hat{v}$, and uses the UNIQUEENCODE$(\cdot)$ function to generate a unique hash encoding for a series of integer coordinates $\mathcal{I}(\hat{v})$.

Algorithm 1: Optimal B-Spline-Based Kinodynamic Search.

function INITIALIZE$(\pi_s, \pi_g, k, \Delta_t)$
  $\mathcal{O} \leftarrow \varnothing$; $\mathcal{L} \leftarrow \varnothing$
  $n \leftarrow$ INDEX(π_s);
  $h(\pi_s) \leftarrow$ HEURISTIC(π_s, π_g); $g(\pi_s) \leftarrow f_{k,\Delta_t}(\pi_s)$;
  $\mathcal{O} \leftarrow$ INSERT$(\mathcal{O}, g(\pi_s) + h(\pi_s), \pi_s)$;
  $\mathcal{L} \leftarrow$ INSERT$(\mathcal{L}, n, \pi_s)$
function Main$(\pi_s, \pi_g, \mathcal{G}, k, \Delta_t)$
  $(\mathcal{O}, \mathcal{L}) \leftarrow$ INITIALIZE$(\pi_s, \pi_g, k, \Delta_t)$;
  while $\mathcal{O} \neq \varnothing$ **do**
    $(m, \hat{v}_i) \leftarrow$ POP$(\mathcal{O})$;
    if $m =$ INDEX(π_g) **then**
      return success
    for $\hat{v}_j \in$ FEASIBLESUCCS$(\hat{v}_i, \mathcal{G}, k, \Delta_t)$ **do**
      $n \leftarrow$ INDEX$(\hat{v}_j)$;
      if not VISITED$(n, \mathcal{L})$ **then**
        $g(\hat{v}_j) \leftarrow \infty$
      if $g(\hat{v}_j) > g(\hat{v}_i) + f_{k,\Delta_t}(\hat{v}_j)$ **then**
        $g(\hat{v}_j) \leftarrow g(\hat{v}_i) + f_{k,\Delta_t}(\hat{v}_j)$
        $h(\hat{v}_j) \leftarrow$ HEURISTIC$(\hat{v}_j, \pi_g)$
        $\mathcal{O} \leftarrow$ INSERT$(\mathcal{O}, g(\hat{v}_j) + h(\hat{v}_j), \hat{v}_j)$

Algorithm 2: Given a k-Degree Vertex Tuple $\hat{v} \in V_H$, Assign a Unique Index to Each Distinct Tuple.

function INDEX$(\hat{v})$
  $\mathcal{I}(\hat{v}) = \varnothing$;
  for all $v \in \hat{v}$ **do**
    $\mathcal{I}(\hat{v}) \leftarrow \mathcal{I}(\hat{v}) \cup$ COORD(v)
  return UNIQUEENCODE$(\mathcal{I}(\hat{v}))$

We construct the nodes of $\mathcal{G}_H$ only on demand during the search process as the graph size may be prohibitively large. At the beginning of the search, the whole $\mathcal{G}_H$ is not explicitly constructed, and the visited set $\mathcal{L}$ and the open set $\mathcal{O}$ are both initialized to empty. After the insertion of the initial state π_s, a tree of nodes is gradually expanded based on the FEASIBLESUCCS$(\cdot)$ function and the priority queue structure maintained by $\mathcal{O}$. Every time a new node is found (whether or not a successor node is a new node is identified by $\mathcal{L}$, which is implemented using a hash map structure), the node is added to $\mathcal{L}$ to trace its open/closed status. In practice, only a small proportion of the nodes of $\mathcal{G}_H$ are constructed before the search succeeds. A toy example[3] that compares the number of actual expanded nodes with the estimated graph size is shown in Table I. Note that the EBK method used in Table I is an efficient version of Algorithm 1, which is discussed in detail in Section V-E. The estimated graph size is computed based on the graph aggregation technique introduced in Section V-F.

Note that there are the following three functions making Algorithm 1 different from the traditional geometric path search.

[3]The setup of this example is the same as the qualitative experiment in Fig. 10, where the grid size is $51 \times 51 \times 5$ and the B-spline degree is five.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

1) The cost function $f_{k,\Delta_t}(\cdot)$ evaluates the control effort of the k-degree vertex tuple, instead of a simple path length measure.
2) The function INDEX$(\cdot)$ regards the k-degree vertex tuple as the "state," which expands the high-dimensional state-space supported by the control point coordinates, while the geometric path search typically regards positions as states.
3) The function FEASIBLESUCCS$(\cdot)$ will expand to the neighboring k-degree tuples with dynamical feasibility checking, while the geometric path search cannot check feasibility without parameterization.

As for the heuristic function HEURISTIC$(\cdot)$, we adopt the admissible minimum time heuristic in [14].

It is worth noting that the design of Algorithm 1 heavily relies on the properties of the B-spline. Thanks to the local control property, the cost evaluation and feasibility checking can be done locally based on the k-degree vertex tuple. Instead of solving the BVP, the proposed search method expands to new states on the high-dimensional graph $\mathcal{G}_H$ by expanding low-dimensional control point coordinates, which are associated with closed forms for cost evaluation. Moreover, checking for collision is time consuming in traditional kinodynamic planners [7], [14], [23]. By using B-spline parameterization, the process can be avoided by characterizing the B-spline deviation, as introduced in Section V-D. The limitation of our method is that the expansion of control points is restricted by the resolution and connection of the grid, which results in limited representations of B-spline trajectories.

D. Feasibility Condition

The dynamical feasibility of the k-degree vertex tuple can be validated via checking the extrema of the derivative of the B-spline. Since the derivative of the B-spline is another B-spline with decreasing degree, the velocity spline is of degree $k-1$ and the acceleration spline is of degree $k-2$. Considering that the convex hull property in Proposition 1 is a sufficient but not necessary condition, directly using Proposition 1 for feasibility checking may be conservative. Actually, there is a nonconservative approach for feasibility checking by using the closed-form solutions of the extremas of the uniform B-spline. Take the fifth-degree uniform B-spline as an example. The B-spline can be rewritten in a monomial basis according to (3). The velocity profile is a degree-4 polynomial whose extremas can be checked by finding the roots of its derivative (degree-3) in closed form.

For traditional kinodynamic planners [7], [14], [23], collision checking is an expensive process and may become the computation bottleneck of the algorithm [39]. Position-only shortest path search on the graph of the cell decompositions does not require collision checking since the piecewise linear connection between cell centers is restricted to collision-free cells. For the proposed method, given the B-spline parameterization of degree k and cell size of the decomposed environment, the B-spline may deviate from the piecewise linear connection due to the fact that it does not exactly pass through the control points. However, the maximum distance that the B-spline curve deviates from the piecewise linear collection can be characterized offline, which is compensable by moderate obstacle inflation. The inflation needed is characterized in Appendix C. In practice, since the degree of the B-spline is fixed and the cell size is not tuned frequently, the inflation can be calculated once and then used for many experiments.

Algorithm 3: Given a k-Degree Vertex Tuple $\hat{v} \in V_H$, Assign an Integer Index Based on the Selected Coordinates of the Tuple.

```
function INDEX(v̂, d)
    I(v̂) = ∅;
    for all i ∈ {k − d + 1, . . . , k} do
        I(v̂) ← I(v̂) ∪ COORD(v̂[i])
    return UNIQUEENCODE(I(v̂))
```

E. Efficient Low-Dispersion Search

Before proposing the efficient methods, we present a complexity analysis of the OBK search. According to the definition of the k-degree vertex tuple, the total number of vertices V_H of the graph $\mathcal{G}_H$ grows exponentially w.r.t. the degree k of B-spline parameterization, i.e., $|V_H| = O(|V|^{k+1})$. Note that according to Proposition 2, Algorithm 1 shares a similar complexity with the execution of Dijkstra's algorithm on the graph $\mathcal{G}_H$ if the heuristic is set to zero. According to the known result that each vertex is expanded at most once for Dijkstra's algorithm (see [40] and [41]), the maximum number of iterations of Algorithm 1 is upper bounded by $|V_H| = O(|V|^{k+1})$, which characterizes the worst-case execution time of the OBK search. Actually, since $|V_H|$ and $|E_H|$ grow exponentially with k, the worst-case execution time of *any* algorithm that optimally solves Problem 1 scales exponentially with k.

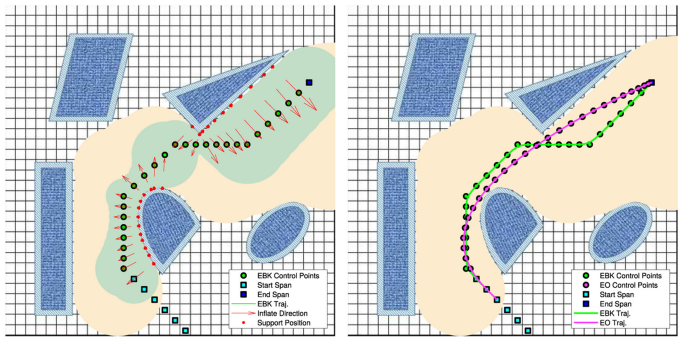
Given the observation that the state in the OBK search is homogeneous (i.e., all the dimensions are control point coordinates), we can aggregate the nodes of $\mathcal{G}_H$ based on the proximity of the coordinates to gain efficiency. Specifically, according to Algorithm 2, each vertex $\hat{v} \in V_H$ is marked with a unique integer index. In the modified INDEX$(\cdot)$ function in Algorithm 3, we encode the vertex $\hat{v}$ only based on the coordinates for the last d vertices such that the vertices, which share the same partial coordinates will be regarded as the same node. By aggregating the nodes of $\mathcal{G}_H$, the dimension of the search space is directly controlled by the user-specified parameter d. The modification essentially conducts a low-dispersion search on the high-dimensional graph $\mathcal{G}_H$ with a control on the number of expanded nodes.

Different d values will determine how the vertex tuples are aggregated, which in turn affects the solution quality and algorithm efficiency. Note that although the search space is reduced, the continuity and smoothness of the resultant trajectory is maintained since the modification preserves the sharing of the B-spline coordinates. The resultant search method is called EBK search and a formal analysis of the EBK search is provided in Section V-F.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

Algorithm 4: Expand Elastic Tube in Configuration Space $\mathcal{C}^{\text{ELAS}}$.

function TUBEEXPANSION($\mathcal{P}, \mathcal{C}^{\text{ELAS}}$)
  Initializes: $d_{\text{infl}}^{\min}, d_{\text{infl}}^{\max}, d_{\text{thres}}$. $\mathcal{R} = \mathcal{R}' = \mathcal{Q} = \varnothing$;
  for all $\mathbf{p}_i \in \mathcal{P}$ **do**
    $(\mathbf{n}_i, r_i) \leftarrow \text{NNSEARCH}(\mathbf{p}_i, \mathcal{C}^{\text{ELAS}})$;
    $\overrightarrow{n} = (\mathbf{p}_i - \mathbf{n}_i)/\text{NORM}(\mathbf{p}_i - \mathbf{n}_i)$;
    while $d_{\text{infl}}^{\max} - d_{\text{infl}}^{\min} > d_{\text{infl}}^{\text{tol}}$ **do**
      $d \leftarrow \left(d_{\text{infl}}^{\max} + d_{\text{infl}}^{\min}\right)/2$,
      $\mathbf{p}_{i,\text{infl}} \leftarrow \mathbf{p}_i + d \cdot \overrightarrow{n}$;
      $(\mathbf{n}_i', r_i') \leftarrow \text{NNSEARCH}(\mathbf{p}_{i,\text{infl}}, \mathcal{C}^{\text{ELAS}})$;
      if $\text{ABS}(r_i' - d - r_i) > d_{\text{thres}}$ **then**
        $d_{\text{infl}}^{\max} \leftarrow d$
      else
        $d_{\text{infl}}^{\min} \leftarrow d$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathbf{p}_{i,\text{infl}}, \mathcal{R}' \leftarrow \mathcal{R}' \cup r_i'$
  return $\mathcal{Q}, \mathcal{R}'$



(a) (b)

Fig. 7. Illustration of the EO approach. (a) Elastic tube expansion process (see Section VI-A). (b) Optimization process (see Section VI-B). In (a), the original tube is marked in *green*, while the inflated tube is marked in *yellow*.

F. Analysis of the EBK Search

As introduced in Section V-E, the user-specified parameter d determines how vertex tuples in the graph $\mathcal{G}_H$ are aggregated. We provide a toy example of $d = 1$ in Fig. 6(b) to understand the graph aggregation. When choosing $d = 1$, as in Fig. 6(b), vertex tuples are aggregated based on the last vertex of the tuple. For instance, $\hat{v}_4$ and $\hat{v}_5$ share the same last vertex and are aggregated into the same node, as marked in *purple*. In the same way, the three distinct nodes $\hat{v}_6$, $\hat{v}_7$, and $\hat{v}_8$ are aggregated into the same *cyan* node.

The edges of $\mathcal{G}_H$ are also reduced accordingly. For example, the edges $(\hat{v}_4, \hat{v}_7)$ and $(\hat{v}_5, \hat{v}_8)$ are aggregated since they are connecting the same two aggregated nodes. Note that once a path to the aggregated node is determined, such as the path *blue-yellow-purple-cyan*, the vertex tuple associated with each aggregated node is determined. In this example, the *purple* node is associated with $\hat{v}_5$ and the *cyan* node is associated with $\hat{v}_8$. Therefore, given the initial vertex tuple, any path on the aggregated graph, can be uniquely transformed to a path on the graph $\mathcal{G}_H$. And, apparently, a path on the graph $\mathcal{G}_H$ can be transformed to a path of aggregated nodes. The equivalence states that we are actually conducting a graph search on the aggregated graph with a controllable number of vertices, i.e., a low-dispersion search on the original high dimensional graph $\mathcal{G}_H$. The resultant path on the aggregated graph can be reconstructed as an admissible path on $\mathcal{G}_H$.

For the simple case of $d = 1$, as illustrated in Fig. 6(b), the size of the aggregated graph is the same as the original graph $\mathcal{G}$. Therefore, the EBK search can be as efficient as a shortest path search on the spatial grid by choosing a small d. And the advantage of the EBK search is that it directly outputs a time-parameterized dynamically feasible trajectory. It turns out that the EBK search is resolution complete with respect to the aggregated graph, and we refer interested readers to a detailed analysis of the EBK search in Appendix D.

By choosing a small $d < k + 1$, a large number of vertex tuples are aggregated into one group, and the "resolution" of the graph becomes large. Due to the aggregation, the representation of the trajectory is limited. An intuitive example is that, when choosing $d = 1$, the search process will never choose to place the same control point in the same grid cell due to the strictly positive cost. As a result, the trajectory obtained may fail to reach the exact end state, such as a static state. However, the issue can be addressed by choosing a larger d and sacrificing efficiency.

Compared to the preliminary version, i.e., the RBK search in [15], the EBK search is more flexible and allows for control of the algorithm efficiency and solution quality. The connection is that the RBK search is essentially the EBK search using $d = 1$.

VI. Elastic Optimization

To compensate for the discretization introduced by the EBK search and further improve the trajectory quality, we present the EO approach, which refines the control point placement to the optimal location w.r.t. the free space. Our approach is motivated by the seminal work [42], in which a collision-free "tube" around the initial path is identified and the path is "stretched" within the tube so that the shape is optimized. Mathematically, the tube is defined as a series of balls, with the ball centers denoted as $\mathcal{P} := \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_T\}$ and corresponding radiuses denoted as $\mathcal{R} := \{r_0, r_1, \ldots, r_T\}$, where $\mathbf{p}_i$ denotes the ball center and r_i denotes the radius. The tube is defined to be "well-connected" if and only if $\|\mathbf{p}_i - \mathbf{p}_{i+i}\|_2 \leq r_i + r_{i+1}$, $\forall i \in \{0, \ldots, T-1\}$. Compared to [42], which cannot handle dynamical feasibility constraints for complex kinodynamic systems such as quadrotors, we propose a convex optimization formulation based on B-spline parameterization, which uses the convex hull property to enforce feasibility.

Note that Zhu *et al.* [43] also proposed a convex elastic smoothing formulation for car-like robots. There are two major differences: 1) The formulation in [43] is based on the dynamics of car-like robots and cannot be applied to complex dynamic systems, while our formulation uses high-order B-spline parameterization, which can be directly used to control quadrotors. 2) In [43], the smoothed trajectory may collide with obstacles due to the geometric incompleteness of the tube constraint as shown in Fig. 8(a) and only a heuristic waypoint insertion/obstacle inflation scheme is provided, while the EO approach has a theoretical safety guarantee, which is achieved by a two-level inflation scheme to ensure the connectivity of the tube and a finite iterative control point insertion process.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

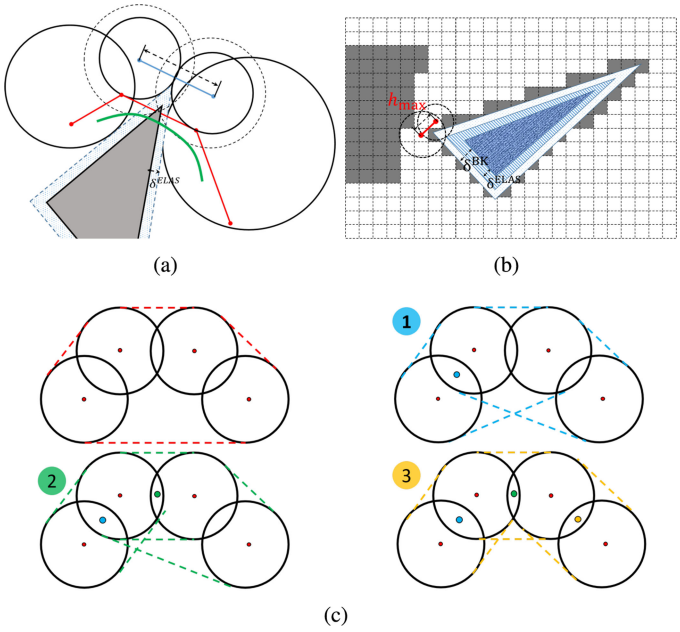


Fig. 8. Illustration of (a) geometric incompleteness of the ball constraint, (b) two-level inflation scheme, and (c) iterative convex hull shrinking process for enforcing the safety guarantee. It can be observed in (a) that even the straight-line segments between the balls may collide with the obstacle.

A. Elastic Tube Expansion

In [42] and [43], the elastic tube is a series of connected balls, which are centered at the waypoints of the reference path. Intuitively, the tube generated in this way cannot fully utilize the free space around it, as shown in Fig. 7(a). We, therefore, propose a lightweight tube expansion algorithm so that the tube can roughly represent the locally largest free space. Given the initial control point placement $\pi = (v_0, \ldots, v_T)$ provided by Algorithm 1, we first extract the coordinates of π, and denote the collection of coordinates as $\mathcal{P} := \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_T\}$ following the notation used in Section IV.

The elastic tube expansion algorithm (see Algorithm 4) can be divided into two steps: First, we construct the initial tube, by conducting a radius search for the initial placement $\mathcal{P}$, and obtain the nearest obstacle position $\mathbf{n}_i$. Second, we push the center of the bubbles in the direction $\overrightarrow{n}$ (away from the nearest obstacle) while satisfying the criterion that the new bubble contains the original bubble, as required by condition $\textsc{Abs}(r_i' - d - r_i) \leq d_{\text{thres}}$, as shown in Fig. 7(a). The inflation process is implemented in a binary search manner. Algorithm 4 will finally find a series of local maximum volume bubble centers $\mathcal{Q} := \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T\}$ based on the initial tube $\mathcal{P}$. For the parameter settings, $d_{\text{infl}}^{\max}$ and $d_{\text{infl}}^{\min}$ are the maximum and minimum inflation distance, respectively; d_{thres} is the threshold for checking whether the new bubble contains the original one, and should be set to a small value, e.g., less than the map resolution; and $d_{\text{infl}}^{\text{tol}}$ is the binary search end condition, which can be set to the resolution of the map. The function NNSEARCH is the nearest neighborhood search, which can be done efficiently if a k-dimensional (k-d) tree is maintained. The efficiency of Algorithm 4 is verified in Section. VIII.

B. EO Formulation

Given the inflated ball centers $\mathcal{Q} = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_T\}$ and corresponding radiuses $\mathcal{R}' = \{r_0', r_1', \ldots r_T'\}$, the EO formulation minimizes the total control effort by finding the optimal placement $\mathcal{P}^* = \{\mathbf{p}_0^*, \mathbf{p}_1^*, \ldots, \mathbf{p}_T^*\}$ while satisfying the safety and dynamical feasibility constraints. The safety constraints are enforced by constraining the control point position inside the 3-D balls, and in Section VI-C, we will discuss how to theoretically guarantee the safety of the resultant trajectory. The dynamical feasibility constraints are enforced using Proposition 1.

Note that like the EBK search, we consider the initial and goal state of the quadrotor. Denote the coordinates of π_s and π_g as $\mathcal{P}^s = \{\mathbf{p}_s^0, \mathbf{p}_s^1, \ldots, \mathbf{p}_s^k\}$ and $\mathcal{P}^g = \{\mathbf{p}_g^0, \mathbf{p}_g^1, \ldots, \mathbf{p}_g^k\}$, respectively. These coordinates are fixed during optimization. Similar to the construction of the k-degree vertex tuple in Section V-B, we concatenate $\mathcal{P}^s$, $\mathcal{P}$, and $\mathcal{P}^g$ and formalize the concatenation in terms of the control point spans. Following the similar notation in Section V-B, we denote by $[\tilde{\mathcal{P}}]_j^k$, the jth control point span of the concatenated coordinates $\tilde{\mathcal{P}}$. It follows that $[\tilde{\mathcal{P}}]_0^k = \mathcal{P}^s$ and $[\tilde{\mathcal{P}}]_J^k = \mathcal{P}^g$, where $J = k + T + 2$. We denote by $\mathbf{P}_j$ the stacked coordinates matrix of $[\tilde{\mathcal{P}}]_j^k$, with $\mathbf{P}_j^D$ denoting the $D \in \{x, y, z\}$ axis. The optimization problem can be expressed as follows:

$$\begin{aligned} \min \quad & \sum_{j=0}^{J} f_{k,\Delta_t}([\tilde{\mathcal{P}}]_j^k) \\ \text{s.t} \quad & [\tilde{\mathcal{P}}]_0^k = \mathcal{P}^s, [\tilde{\mathcal{P}}]_J^k = \mathcal{P}^g \\ & \|\mathbf{p}_i - \mathbf{q}_i\|_2 \leq r_i' \qquad \forall i \in \{0, \ldots, T\} \\ & |\mathbf{S}\mathbf{P}_j^D| \leq u_{l,D}^{\max} \mathbf{1}_{(k+1)\times 1} \qquad \forall j, D \in \{x, y, z\}, \end{aligned} \tag{6}$$

where the first constraint expresses that the initial and goal states need to be fixed. And the second constraint (quadratic) restricts the control points inside the expanded tube. The third constraint (linear) is to ensure the dynamical feasibility using the sufficient condition in Proposition 1. Since the control points can be adjusted in continuous free space, the potential conservativeness brought by Proposition 1 is minor in practice. The overall formulation is a QCQP, which can be solved efficiently using off-the-shelf convex solvers. The time cost is fixed given the initial placement $\mathcal{P}$, so it is not included in the objective.

C. Enforcing Safety Guarantee

Restricting control points inside the balls is not a sufficient condition for safety, for the following two reasons: 1) the balls are placed with a finite density and constraining the control points in the balls cannot preserve the collision-free characteristic even for straight-line segments between control points [see Fig. 8(a)], and 2) the B-spline does not exactly pass the control points and deviates from the straight-line segments. The first issue is also observed in [43], which proposes using waypoint insertion/obstacle inflation to handle the problem. However, there is no quantification of how much inflation or how many insertions are needed and only a heuristic is given. The second issue is inherently similar to one common issue faced by piecewise polynomial parameterization [2]–[5]: the

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

polynomial may deviate from the collision-free straight-line segments between waypoints or exceed the safe flight corridor. Chen *et al.* [5] proposed a finite iterative process by adding constraints on polynomial extremas based on a cube corridor (linear constraints), but this is not directly applicable to B-spline parameterization with quadratic constraints.

In our case, the issues can be resolved using a *two-level inflation scheme* and an iterative *convex hull shrinking process*. The two-level inflation scheme is to ensure the connectivity of the elastic tube, and furthermore, the inflation needed is quantified in the scheme. For simplicity, we first consider the original tube before applying the tube expansion algorithm (see Algorithm 4). The two-level obstacle inflation scheme is as follows: the configuration space in which we conduct kinodynamic search with larger obstacle inflation δ^{BK} is $\mathcal{C}^{\mathrm{BK}}$, while we generate the elastic tube and optimize the control points in the configuration space $\mathcal{C}^{\mathrm{ELAS}}$ with smaller obstacle inflation δ^{ELAS}, as shown in Fig. 8(b). The difference adds additional clearance to any point in the configuration space $\mathcal{C}^{\mathrm{BK}}$. Without the difference, the minimum radius of the ball at the grid center is $\min(c_x, c_y, c_z)$, where c_x, c_y, and c_z are the grid size. Denote the maximum separation distance of two neighboring control points in the 3-D grid as $h_{\max}$. It follows that, if the difference $\delta^{\mathrm{BK}} - \delta^{\mathrm{ELAS}} > h_{\max}/2 - \min(c_x, c_y, c_z)$ holds, the additional clearance will ensure that the two neighboring balls overlap, thus ensuring the connectivity of the elastic tube. Note that Algorithm 4 maintains the connectivity of the tube by ensuring that the inflated ball contains the original ball while keeping the same support point on the obstacle. For the low-level inflation, given $h_{\max}$, $\delta^{\mathrm{ELAS}} \geq \frac{\sqrt{2}-1}{2} h_{\max}$ is sufficient for the straight-line segments to be contained in the free space [15].

Based on the two-level inflation scheme, we propose an iterative *convex hull shrinking* process, which pulls the B-spline trajectory back to the free space in the case of collision. The idea of the process is to iteratively add control points to the original B-spline control point sequence. The newly added control points are constrained in the intersection of two consecutive balls.[4] The process is built upon the convex hull property of the B-spline and constrains the B-spline trajectory by shrinking the convex hull. We highlight that only a finite number of control points are needed to enforce the safety of the B-spline trajectory. This could save a significant amount of computation power compared to the methods, which apply dense constraint points based on a conservative heuristic [1], [43]. We conclude this feature in Theorem 1.

Theorem 1: Given a well-connected elastic tube, a B-spline trajectory that fits within the tube can be generated by iteratively adding constrained control points to the original B-spline control point sequence. The newly added control point is constrained inside the intersection of neighboring balls. The iterative process succeeds in a finite number of iterations, or infeasibility is reported when the dynamical feasibility cannot be satisfied, given the current tube and control point sequence.

Proof: Please refer to Appendix E for the detailed proof. ■

In Fig. 8(c), we provide a toy example of the convex hull shrinking process. The initial placement is constrained in four balls, and the convex hull envelope exceeds the free space. If collision is detected, we add one additional control point (*blue dot*), which is constrained to be inside the intersection of the first and second ball. The extended EO is expected to be executed again, and the convex hull shrinks. Similarly, if the collision is still not resolved, we iteratively add control points to the intersection space. It can be shown that at most nine iterations are needed to resolve the collision in this case.

VII. Implementation Details

A. Receding Horizon Replanner Using Local Control

As shown in Fig. 3, B-spline has the local control property, which facilitates the receding horizon (re)planning. Specifically, all the control points are organized in a sliding window. The control points corresponding to the executed and executing trajectory are committed and fixed. The disturbance caused by the optimization will not affect the feasibility of the executing trajectory due to the local control. A stopping policy will be activated if no feasible solution is found before the end of the executing trajectory.

When replanning is activated, the EBK search is called to update the placement for the control points under optimization. Note that the initial and goal state of the EBK search can be determined by the control points inside the window according to the local planning range. Note that the control points from the sliding window are in the continuous space after reshaping. However, the EBK search should use the discretized control points as the reference initial/goal state to preserve optimality. The strategy of getting the reference states for the EBK search are to find the closest span pattern in terms of position and velocity error while matching the last control point to the grid cell. We constantly gather a fixed number of control points (e.g., twelve) for EO as the window moves forward.

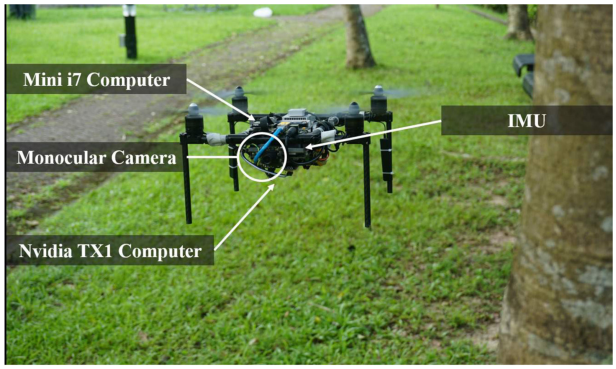
There are two modes for the activation of replanning, namely, active mode and passive mode. For the passive mode, the EBK search is only called when collision is detected, while for the active mode, the EBK search is constantly activated as the sliding window moves forwards. Since the active mode can constantly improve the trajectory quality, it is more robust when the mapping quality is limited, but it is more computationally expensive.

B. Monocular-Vision-Based Testbed

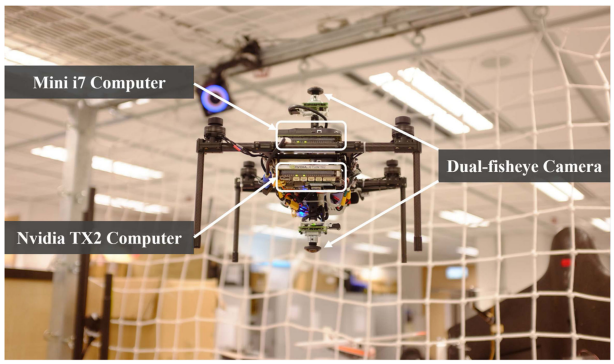
The monocular quadrotor testbed is equipped with a monocular camera (30 Hz), one IMU (100 Hz), an Intel i7 processor, and an NVIDIA Jetson TX1 [see Fig. 9(a)]. The localization, mapping, and planning modules are all running onboard. The localization module is based on our Monocular Visual Inertial Navigation System (VINS-Mono) [44], and the mapping module is based on our monocular dense mapping method [45] and

[4] According to the introduction in Section IV, the total number of knots should satisfy $m + 1 = (n + 1) + k + 1$. Since uniform B-spline is used, when a new control point is inserted, the number of knots is increased by one, while the knot separation Δ_t remains the same. The new control point sequence still matches the new knot vector.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.



(a)



(b)

Fig. 9. Illustration of our (a) monocular-vision-based quadrotor testbed and (b) dual-fisheye vision-based quadrotor testbed.

truncated signed distance field fusion. No prior knowledge of the environment is required.

C. Dual-Fisheye Vision-Based Testbed

The dual-fisheye quadrotor testbed is equipped with two fisheye cameras (30 Hz), one IMU (100 Hz), an Intel i7 processor, and an NVIDIA Jetson TX2 [see Fig. 9(b)]. All modules are running onboard. It is worth noting that by using the two fisheye cameras, the system can provide omnidirectional perception and the quadrotor is able to fly a round-trip with a fixed yaw angle. The mapping module is based on our dual-fisheye omnidirectional stereo system [46]. Also no prior knowledge of the environment is required.

VIII. Analysis

In this section, we present an analysis of the proposed kinodynamic planning framework. We begin with two individual analyses for the EBK search and the EO approach, respectively. To analyze the EBK search, we compare the proposed method with two kinodynamic planning algorithms, namely, search-based motion primitive (SMP) [14] and kinodynamic RRT* (kRRT*) [7], representing both the search-based method and the sampling-based method, respectively. For the EO, we compare the EO approach with two state-of-the-art trajectory optimization techniques, namely, continuous trajectory (CT) optimization [25] and gradient-based safe (GS) trajectory optimization [33], which are popular nonlinear optimization techniques for trajectory refinement. After the individual tests, we analyze the run-time efficiency of the proposed replanning framework, and compare the whole replanning system with the SMP [14] method. We run all the simulations on a desktop computer equipped with an Intel I7-8700K CPU.

A. Analysis of the B-Spline-Based Kinodynamic Search

Recall that the motivation for introducing kinodynamic search to replanning is to facilitate dealing with the nonstatic initial states of the quadrotors. To this end, we analyze the performance of the kinodynamic search by planning from a given nonstatic initial state to varying goal states in a $10 \times 10 \times 2$ m test field. We compare our results with SMP [14] and kRRT* [7] in terms of the trajectory quality and time efficiency, under the same planning setup. At the same time, we also illustrate the results of a hierarchical geometric planner as a common baseline. Specifically, the geometric planner first uses A* under the Euclidean distance measure to find the shortest path, and then parameterizes the path using the unconstrained QP formulation introduced in [3].

The kinodynamic planners from [7], [14] and our method all have a tuning parameter to control the algorithm complexity and solution quality. For instance, SMP [14] can control the discretization resolution for the control input. To further investigate the optimality-efficiency tradeoff achieved by each algorithm, we also demonstrate the results for these algorithms under different parameter setups. Note that we focus on the real-time replanning scenario, so we are concerned with the operation region where the run-time efficiency is close to real time. Specifically, we compare the following methods.

1) *Geometric†:* A path finder using A* and a fifth-degree polynomial parameterization using the unconstrained QP formulation [3] by minimizing the average integral of acceleration.
2) *kRRT*-T100†:* The kRRT* planner [7] using a 3-D acceleration-controlled double integrator system under a 100 ms termination condition for the sampling.
3) *kRRT*-T600*: The kRRT* planner under a 600 ms termination condition.
4) *SMP-U3†:* The SMP planner [14] using a 3-D acceleration-controlled double integrator system with three discrete control inputs for each axis.
5) *SMP-U5:* The SMP planner with five discrete control inputs for each axis.
6) *EBK-D1†:* Our EBK planner using fifth-degree B-spline parameterization and $d = 1$.
7) *EBK-D2:* The EBK planner using $d = 2$.

The reason for using the acceleration-controlled double integrator system for kRRT* and SMP is that if a higher order system were used, the run time would be too large, making it inapplicable to replanning. For a similar reason, a termination time larger than 600 ms for kRRT* and number of control inputs larger than five for SMP are not considered. The methods marked with † are amenable to real time, and the other methods serve as showcases illustrating how the trajectory quality can be improved given a larger computation time budget.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

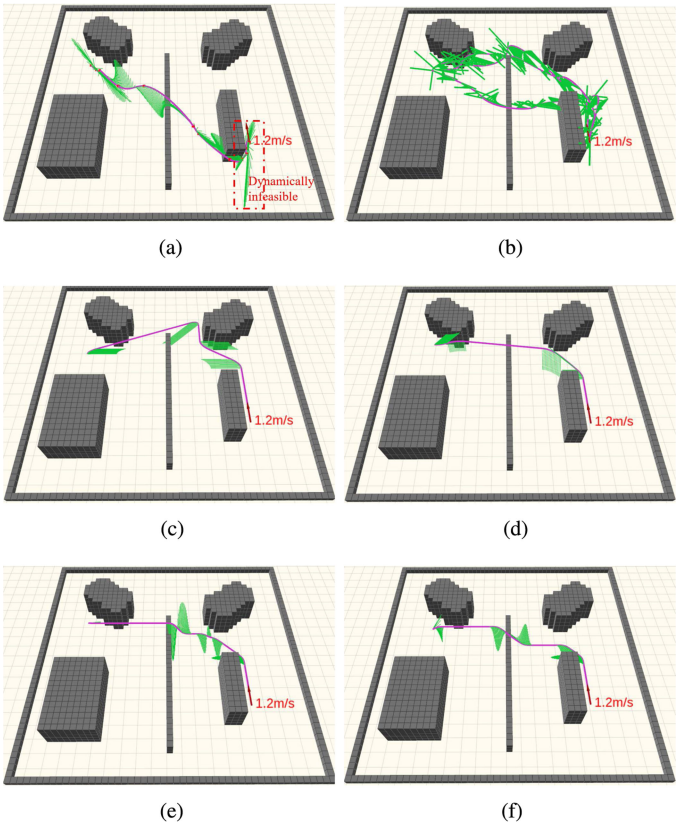


Fig. 10. Comparisons of different kinodynamic planning approaches. (a) Geometric method. (b) Kinodynamic RRT*-T100. (c) SMP-U3. (d) SMP-U5. (e) EBK-D1. (f) EBK-D2.

All the kinodynamic planners have a similar form of the cost function according to (4). Since both kRRT* and SMP adopt the acceleration-controlled double integrator system, the order of the derivative l we can take is two for all the experiments. The weight λ of the trajectory time is set to 20. We consider two additional metrics, namely, average acceleration and maximum acceleration, which represent the smoothness and feasibility of the resultant trajectory.

For the methods where cell decomposition is needed, such as A* for the geometric method and EBK, the environment is decomposed into cells with a fixed size of 0.2 m for each dimension. The geometric planner requires a *time allocation* module since the path does not contain any time information. Generating the optimal time profile for a path is actually not a trivial problem [47], and the common practice is based on a heuristic allocation method, such as a trapezoid velocity profile [4], [25]. We follow this practice and set the average speed to achieve a similar trajectory duration to the kinodynamic planners. Time allocation is not needed for kinodynamic planners since they directly produce time-profiled trajectories. The initial state is set to a fixed position with nonzero velocity 1.2 m/s and zero acceleration, as shown in Fig. 10. The goal state is static, and its position is regularly sampled with a distance separation of 0.7 m. In total, there are 136 collision-free goals available. The velocity and acceleration limit are set to 2 m/s and 4.7 m/s^2 for each axis. Δ_t for the EBK method is set to 0.17 s. The statistics

TABLE II
COMPARISON OF DIFFERENT KINODYNAMIC PLANNING APPROACHES

Method	Run Time (s)	Traj. Dur. (s)	Acc. Cost (m^2/s^3)	Ave. Acc. (m/s^2)	Max. Acc. (m/s^2)
kRRT*-T600	0.661	4.04	30.6	2.42	4.18
SMP-U5	2.878	**3.86**	**12.8**	1.36	2.97
EBK-D2	**0.383**	4.52	14.4	0.95	4.60
Geometric†	0.018	4.34	108.8	4.60	9.48
kRRT*-T100†	0.110	4.60	38.8	2.54	4.28
SMP-U3†	0.111	4.47	32.3	1.70	4.70
EBK-D1†	**0.034**	**4.31**	**15.2**	1.10	4.56

averaged over the 136 rounds of planning are shown in Table II, and the qualitative results for the planning to the same goal state are shown in Fig. 10.

According to the qualitative results in Fig. 10, there is a significant difference in the performance. For the geometric method [see Fig. 10(a)], since the shortest path diverges from the initial velocity direction, the parameterization is jerky at the beginning. Moreover, as the unconstrained QP cannot enforce the dynamical feasibility, the generated trajectory is infeasible due to the nonstatic initial state. As for the kRRT* with a 100 ms time budget [see Fig. 10(b)], it can respect the initial state of the quadrotor since the control effort is directly considered in the sampling process. It also guarantees the dynamical feasibility of the resultant trajectory by constraining the control input and states along the tree edges. However, the trajectory quality is unsatisfactory due to the limited sampling. We also observe some unpredictable randomized behavior as shown by the three rounds of planning with exactly the same initial state and goal state in Fig. 10(b). Meanwhile, for the SMP method, the shape of the trajectory of SMP-U3 is not natural since the resolution of the control input is large.[5] Some maneuvers such as flying over the little step in the middle are not included in the solution space of SMP-U3. SMP-U5 performs better than SMP-U3 due to finer discretization. Finally, EBK-D1 and EBK-D2 both generate an initial-state-aware smooth trajectory. EBK-D2 finds a slightly better trajectory than EBK-D1 according to the acceleration profile.

It is notable that the trajectory provided by the EBK method has continuity up to snap, which is good for controlling quadrotors. We can observe from Fig. 10 that the kRRT* trajectory only has continuity up to acceleration and that of the SMP has continuity up to velocity, due to the restriction of computation complexity and order of the system model.

From the quantitative results in Table II, among the real-time methods (with †), EBK-D1 finds the lowest cost trajectory, achieving a 15.2 m^2/s^3 total cost within 0.034 s. Our method shows the superior performance given the real-time requirement. It is of interest to examine the situation where we have a time budget in the range of seconds. In that case, SMP-U5 achieves a lower cost than EBK-D1 and EBK-D2. The reason is that the trajectory duration of EBK-D2 cannot be efficiently reduced since the control points have to be expanded step by step on

[5]The acceleration bound we use is larger than that in [14].

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISON OF DIFFERENT TRAJECTORY OPTIMIZATION APPROACHES

Map	Method	Density (pillars/m^2)	Success. Frac. (%)	Run-Time (s)	Traj. Dura. (s)	Jerk Cost (m^2/s^5)	Ave. Vel. (m/s)	Ave. Acc. (m/s^2)	Max. Acc. (m/s^2)	Traj. Len. (m)
Random Map	CT [25]	0.1	95	0.030	8.1	70.2	1.73	0.93	2.63	14.3
		0.2	68	0.082	8.0	91.6	1.72	1.01	2.90	13.9
		0.4	46	0.073	7.9	105.6	1.69	1.05	3.10	13.7
	GS [33]	0.1	**100**	0.062	11.1	186.8	1.39	0.87	2.99	15.6
	EO (proposed)	0.1	**100**	**0.012**	**10.6**	**174.0**	1.43	0.38	1.35	14.1
	GS [33]	0.2	**100**	0.075	12.8	205.9	1.28	0.85	2.89	15.9
	EO (proposed)	0.2	**100**	**0.012**	**11.3**	**181.2**	1.38	0.49	1.85	14.5
	GS [33]	0.4	**100**	0.206	12.9	190.8	1.33	0.89	3.10	17.4
	EO (proposed)	0.4	**100**	**0.013**	**12.4**	**132.4**	1.36	0.55	1.34	16.2

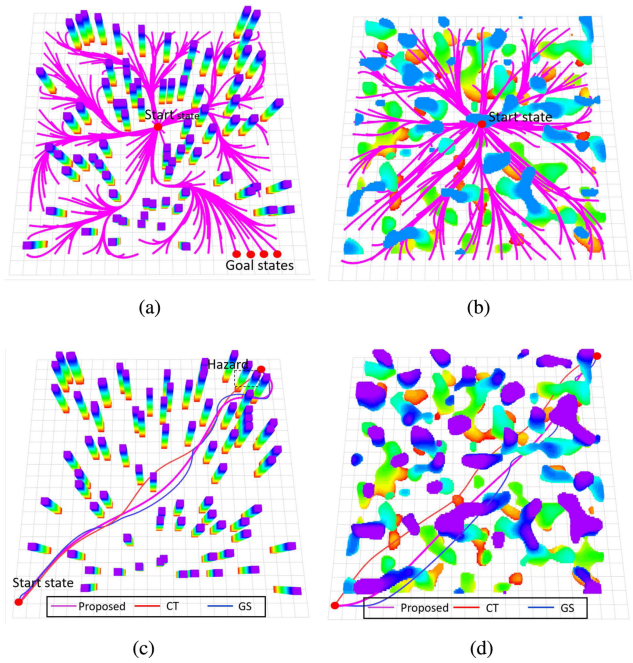


Fig. 11. Comparisons of different trajectory optimization methods on two different maps. The EO method is shown in *purple*, the CT method is shown in *red* and the GS method is shown in *blue*. (a) Random map (0.2 pillars/m^2). (b) Perlin noise map. (c) Random map (0.2 pillars/m^2). (d) Perlin noise map.

the discrete grid. This illustrates the limitation induced by the discretization for the EBK method. However, the difference between the SMP method and EBK method is minor, meaning the EBK method can provide competitive solutions given a run-time budget of seconds.

B. Analysis of the EO

To evaluate the performance of EO, we compare our method with two state-of-the-art trajectory optimization methods: the CT method [25] and the GS method [33]. Two test environments are provided, namely, a random map shown in Fig. 11(c) and a Perlin noise map[6] shown in Fig. 11(d). The map size is fixed to 20 m $\times$ 20 m $\times$ 4 m for both maps. The start location is fixed to the center of the test field for the qualitative experiments in Fig. 11(a) and (b), and is fixed to the left bottom corner for all the experiments in Table III to test the performance for long trajectories. The goal location is regularly sampled in the test field with a distance separation of 1 m. We vary the obstacle density of the random map from 0.1 to 0.4 pillars/m^2, where the pillar size is set to 0.5 $\times$ 0.5 m. The qualitative results are provided in Fig. 11 and the quantitative statistics of the optimization performance are organized in Table III. Note that we omit the statistics for the Perlin noise map in Table III since the trend is similar to that of the random map. All the optimization methods can handle high-order parameterization and they are set to minimizing the integral of the squared jerk. According to (4), the jerk cost listed in Table III has unit m^2/s^5 and represents the accumulated integral of the squared change rate of the acceleration, which represents the total control efforts.

The CT method uses a gradient-based nonlinear optimization process to optimize the polynomial coefficients such that the resultant trajectory is collision free. It requires a Euclidean signed distance field to evaluate the collision cost. Following the practice in [25], we provide an initial polynomial trajectory as an initial guess, which is parameterized by a straight-line guiding path. The number of segments is determined by a 3-m distance separation. The GS method shares a similar formulation to the CT method, with the difference being that the GS method starts from a collision-free initial guess, which is provided by A* search in the experiment. Both the CT method and the GS method require *time allocation*, and in the experiment, we use the trapezoid velocity profile and scale the total allocated time such that different optimization methods achieve a similar average velocity, as shown in Table III. The CT method has a larger average velocity due to the cases where it fails to resolve collision and the trajectory does not contain necessary deceleration. The success fraction is calculated by counting the collision-free and dynamically feasible trajectories among the total number of rounds.

First, we examine the optimization reliability, i.e., the success fraction for the different methods. As shown in Table III, the CT method is sensitive to the obstacle density. For a density of 0.1 pillars/m^2, the success fraction of the CT method is 95%, which means that in obstacle-sparse environments, the CT method can resolve collision efficiently. However, when the density increases to 0.4 pillars/m^2, we observe that the success fraction

[6][Online]. Available: https://github.com/HKUST-Aerial-Robotics/mockamap.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

drops significantly to 46%. The reason is that the CT method easily gets stuck in the infeasible local minimum when the trajectory is inside the cluttered obstacle. As shown in Fig. 11(c), the trajectory of the CT method gets stuck between two pillars where there is not enough clearance considering the quadrotor size. On the other hand, we do not observe a drop in the success fraction for either the GS method or EO method. The reason is that the GS method starts from a collision-free initial guess and has a dominant collision penalty compared to its smoothness cost. The EO method has a high success fraction according to its theoretical guarantee.

Second, we focus on the two reliable methods, namely, the GS method and EO method, and further investigate the trajectory statistics. As shown in Table III, the average trajectory durations of the two methods are close, so the comparison of the jerk cost is fair. For an obstacle density of 0.2 pillars/m^2, averaged over the 135 rounds, the jerk cost of the EO method is 181.2 m^2/s^5, which is smaller than that of the GS method. Similar results are also observed for different densities. The reason is that the GS method is sensitive to the time allocation, since it also starts with an unparamterized path. However, for unknown environments, the shape of the initial path may vary and there is no systematic way to allocate the time. The heuristic trapezoid velocity profile may fail to provide a good initial guess when the initial path is not in a regular shape. For the EO method, since it starts from a time-parameterized B-spline trajectory, there is no need for time allocation.

From the efficiency perspective, for the GS method to achieve a similar jerk cost to the EO method, it needs run-times of 0.062 s, 0.075 s, and 0.206 s on average for the three densities, which are significantly longer than the EO method. Note that the GS method is based on a nonlinear optimization formulation. In the experiments, the nonlinear optimization of the GS method is terminated when the nonlinear solver (NLOPT [48]) reports convergence. For the GS method to converge to a compatible solution to that of our EO method, it requires a significantly longer run time. Moreover, the run time of the GS method is sensitive to the density, since for cluttered environments, more segments of the polynomial are needed, which may affect the convergence rate of the GS method. However, the EO method has lower run times for the three different densities. The efficiency of the EO method is affected by the total number of control points, but we observe that the efficiency difference is minor for the different densities.

C. Analysis of the Run-Time Efficiency

In this section, we test our replanning system, combining the EBK search with EO refinement. For the EBK search, we adopt EBK-D1 since it is the most efficient and the trajectory quality is satisfactory as verified in Section VIII-A. We provide two different maps, namely, the random map and Perlin noise map. We evaluate the run-time efficiency of our replanning system, and list the statistics of all components in Table IV. The overall trajectory illustrating the whole round trip is shown in Fig. 12. As shown in Table IV, on the random map, the EBK-D1 method consumes an average computing time of 0.017 s with a standard deviation of 0.01 s. The elastic tube expansion method can be finished in 0.002 s, and the optimization can be done in 0.021 s. On the Perlin noise map, which contains unstructured 3-D obstacles, our method has a similar performance, showing that our method works well in complex 3-D environments.

TABLE IV
RUN-TIME ANALYSIS ON DIFFERENT MAPS

Maps	# Replans	Time (s)	EBK Search	# Opt.	Time (s)	Tube Expan.	Traj. Opt.	Total Opt.
Random map (0.25 pillars/m^2)	76	Avg	**0.017**	993	Avg	**0.002**	**0.021**	**0.023**
		Max	0.049		Max	0.009	0.043	0.044
		Std	0.010		Std	0.001	0.010	0.010
Perlin Map	19	Avg	**0.014**	1044	Avg	**0.002**	**0.028**	**0.030**
		Max	0.026		Max	0.008	0.058	0.061
		Std	0.006		Std	0.001	0.010	0.011

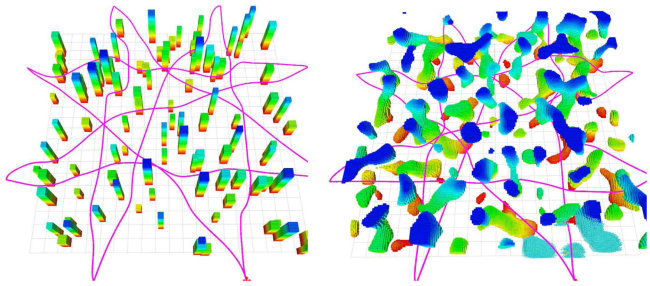


Fig. 12. Illustration of our replanning system on different maps. (a) Replan on the random map. (b) Replan on the Perlin noise map.

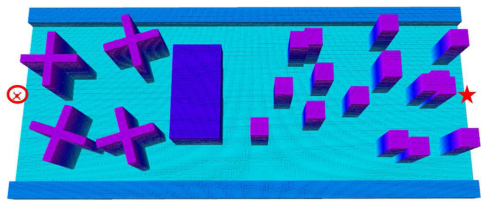


Fig. 13. Illustration of the simulated environment for benchmarking.

D. Comparison of the Replanning Framework

In this section, we conduct a system comparison with the SMP method [14]. We use SMP-U3 since SMP-U5 cannot work in real time. Moreover, we conduct an ablation test, which excludes the initial-state aware EBK search and adopts the naive position-only A* search combined with EO local reshaping. We call the method for the ablation test A*-EO. The ablation test validates the critical role of the kinodynamic search in replanning.

We set up a challenging obstacle-cluttered 3-D complex simulation environment containing walls, 3-D steps (free space below), and pillars, as shown in Fig. 13. The replanning strategy is choosing a local goal state on a given straight-line guiding path with a local replanning range of 5 m. The simulated quadrotor is equipped with a depth camera, which has a sensing range of 4 m. The maximum velocity and acceleration bound are set to 2 m/s and 3.2 m/s^2 for each axis. For SMP-U3, we use a conservative bound with maximum acceleration 1.0 m/s^2 for each axis since SMP-U3 can only work with a narrow velocity and acceleration range. Using a large dynamic range for SMP-U3 will result in a very sparse primitive graph, which does not even contain

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

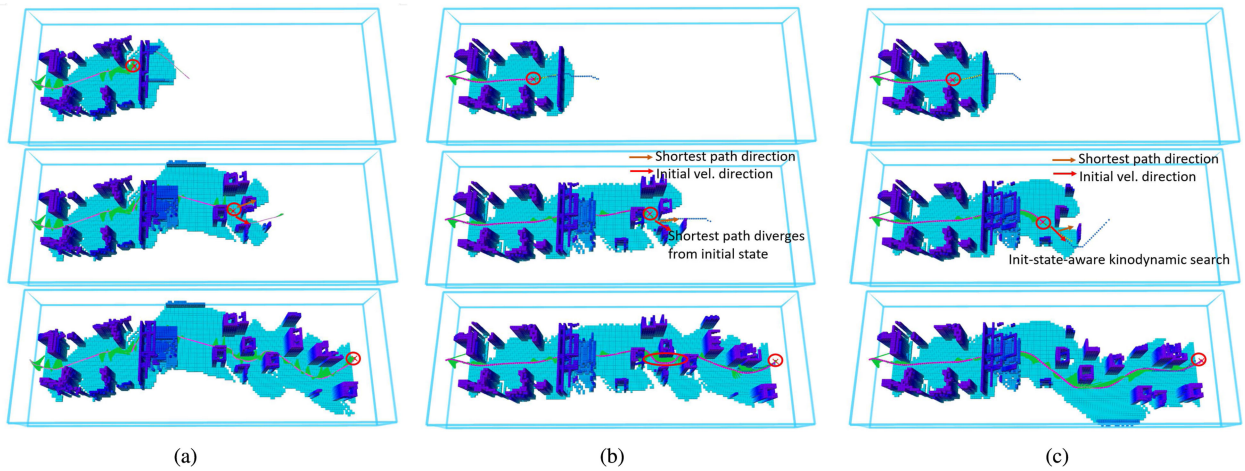


Fig. 14. Illustration of different replanning methods in the same simulated environment. The trajectory is shown in *purple*, and the acceleration profile is marked in *green*. For the replanning cases where the shortest path direction is inconsistent with the initial state, we mark the initial velocity with a *red arrow* and the shortest path direction in *brown*. (a) SMP-U3 (Conservative). (b) A*-EO. (c) Our method.

TABLE V
PERFORMANCE OF DIFFERENT REPLANNING SYSTEMS

Method	Trajectory Statistics				Time Efficiency(s)		
	Traj. Dura.(s)	Jerk Cost (m^2/s^5)	Mean Vel. (m/s)	Max Acc. (m/s^2)	Ave	Max	Std
SMP-U3 (Conservative)	33.3	956.3	1.14	**1.3**	**0.010**	0.027	0.004
A*-EO	25.2	723.0	1.31	3.18	0.014	0.062	0.011
Our method	**24.4**	**648.1**	**1.37**	3.18	0.019	0.080	0.012

one feasible solution. For the EBK search of our method, we use EBK-D1 with a $60 \times 60 \times 20$ uniform grid. For the EO approach, 12 control points are refined as the window moves forward. We evaluate the replanning system from the trajectory statistics and time efficiency perspectives, as shown in Table V.

Critical snapshots of the three methods are shown in Fig. 14. For SMP-U3 (Conservative) in Fig. 14(a), when the quadrotor observes the 3-D step, it chooses to circle around instead of directly flying through the space below since the latter action requires a large acceleration range. This phenomenon demonstrates that SMP-U3 (Conservative) sacrifices maneuverability due to the restriction of the dynamic range. SMP-U3 takes the initial state into account, as shown in the middle snapshot in Fig. 14(a). The necessity of using the kinodynamic search instead of the position-only A* search is identified by the totally different maneuvers in Fig. 14(b) and (c). When the quadrotor enters the region of the pillars, it has two distinct choices, namely, "pass on the left" or "pass on the right." For the A*-EO method, as shown in the middle snapshot in Fig. 14(b), the quadrotor tends to choose the direction purely based on the shortest path. So there are cases where the quadrotor is passing in one direction and suddenly switches to the opposite direction due to finding a new shortest path, which results in an inconsistent and nonsmooth replanning trajectory. Compared to A*-EO, the kinodynamic EBK search provides an initial-state-aware trajectory for the local reshaping, as shown in Fig. 14(c). With the EBK search, the overall trajectory is clearly more natural and the replanning is more consistent.

As shown in Table V, SMP-U3 (Conservative) is efficient and has an average computation time of 0.010 s. However, since the acceleration bound is conservative, the maneuverability is sacrificed and the trajectory duration is 33.3 s, longer than the other two methods. Note that we use acceleration-controlled SMP [14] with unconstrained QP [3] reparameterization. Since the unconstrained QP has no dynamical feasibility guarantee, the maximum acceleration is 1.3 m/s^2, which exceeds its designed maximum acceleration (1 m/s^2). Compared to SMP-U3 and A*-EO, our method has the lowest total jerk cost and the improvement is achieved by incorporating the kinodynamic search. The average velocity of our method is 1.37 m/s, slightly higher than the A*-EO, due to the fact that the kinodynamic search can reduce sharp decelerations. The maximum acceleration of our method is 3.18 m/s^2, which obeys the dynamical feasibility constraint. Compared to the position-only A*-EO method, the jerk cost is reduced by 10% by using the kinodynamic search. Considering that the advantages of the EBK search are outstanding for part of the trajectory where potential inconsistency exists, this quantitative improvement still faithfully identifies the gain of using the kinodynamic search. Note that navigating through this challenging environment with enough agility already requires considerable control efforts, and the 10% cost reduction represents a reasonable overall improvement.

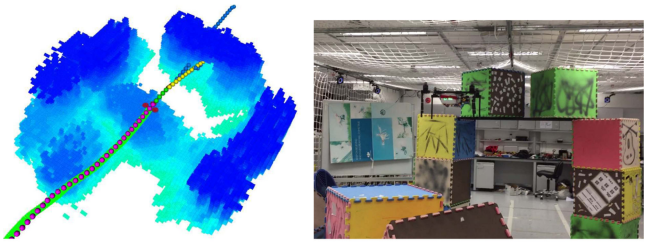
IX. EXPERIMENTAL RESULTS

We conduct onboard experiments[7] with the two vision-based testbeds to show the general applicability of the proposed framework. For onboard testing, the parameters are as follows: the time step Δ_t is set to 0.35 s; the maximum velocity and maximum acceleration are set to 1.2 m/s and 2.0 m/s^2, respectively,[8] and the local planning range is set to $10 \times 6 \times 1.1$ m. (The corresponding grid size is $55 \times 35 \times 6$.)

[7][Online]. Available: https://www.youtube.com/watch?v=sg46XT9-o1k.

[8]The speed limit and acceleration limit are set to be slightly conservative considering the perception delay in onboard experiments.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 15. Illustration of the snapshot of the indoor replanning with the monocular vision. (a) Control points found by EBK (*blue*), executed control points (*pink*), committed control points (*green*), and control points under optimization (*yellow*) are marked. (b) Corresponding indoor environment.

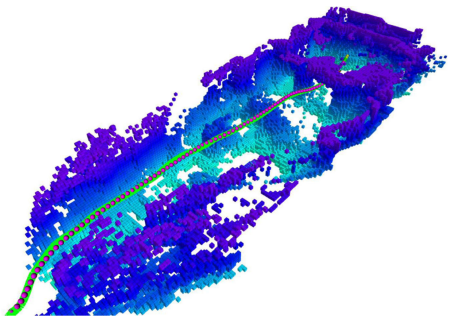


Fig. 16. Illustration of the whole trajectory and final accumulated map for the indoor replanning using the monocular perception system.

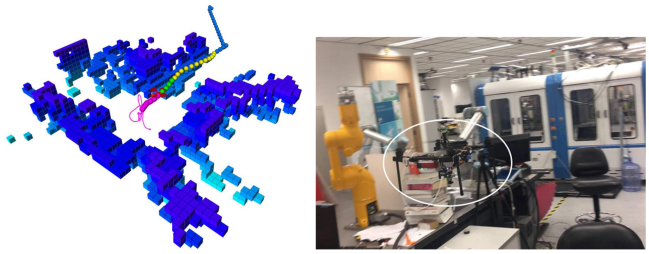


Fig. 17. Illustration of the snapshot of the indoor replanning with the omnidirectional vision. With a fixed yaw angle, the obstacles around the quadrotor can be mapped.

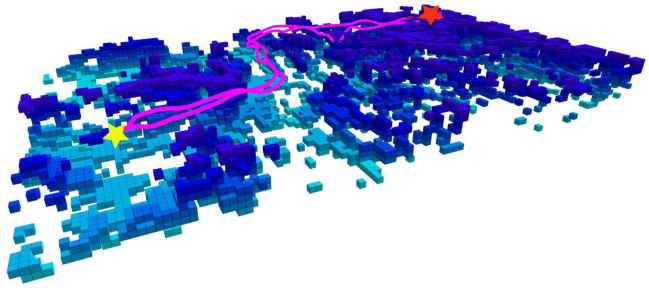
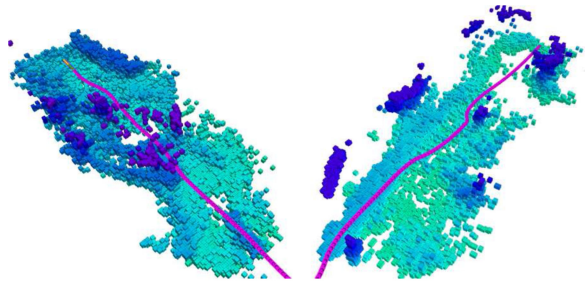


Fig. 18. Illustration of the whole trajectory for the replanning using the dual-fisheye omnidirectional perception system. Unlike the experiments using the monocular testbed, we can achieve round-trip flight in an unknown complex indoor environment.



(a) (b)

Fig. 19. Illustration of the outdoor experiments using the monocular vision. (a) Flight through a pavilion (part of the map on the top is cut for visualization purposes). (b) Flight avoiding trees.

A. Indoor Replanning Performance

1) Monocular-Vision-Based Indoor Navigation: As shown in Fig. 15, our replanning system works in complex 3-D environments with only a local map. The quadrotor is commanded to navigate to a 3-D position where the environment is previously unknown. The whole trajectory and the final accumulated map is shown in Fig. 16. The trajectory length of the final trajectory is 18.6 m and total trajectory execution time is 43.4 s. The average velocity of the quadrotor is 0.45 m/s with a maximum velocity of 0.79 m/s. The maximum acceleration of the trajectory is 0.58 m/s^2, the whole trajectory is dynamically feasible, and there are a total 125 calls of the EBK search (active mode), with an average computation time of 0.010 s. There are 105 calls of EO. The average computation times of the elastic tube expansion and the optimization are 0.001 s and 0.031 s, respectively.

2) Dual-Fisheye-Based Indoor Round-Trip Navigation: As shown in Figs. 17 and 18, with omnidirectional perception, our quadrotor testbed is able to fly a round-trip without controlling the yaw angle. Online mapping using the dual-fisheye cameras is challenging due to the high distortion of the images acquired from the fisheye cameras. Although the uncertainty of the map is larger than the monocular case, our replanning system is still able to robustly avoid the unexpected obstacles and navigate in the unstructured cluttered environment.

B. Outdoor Replanning Performance

As shown in Fig. 19, we demonstrate outdoor experiments using the monocular vision testbed. For Fig. 19(a), the trajectory length is 19.6 m and total execution time is 41.3 s. The average velocity of the quadrotor is 0.49 m/s. The maximum acceleration of the trajectory is 1.06 m/s^2, and the whole trajectory is dynamically feasible. There are a total 35 calls of EBK search (passive mode) with an average computation time of 0.025 s. The average computation times of the elastic tube expansion and optimization are 0.001 s and 0.030 s, respectively. For the experiment shown in Fig. 19(b), the performance and trajectory statistics are similar.

X. Conclusion

In this article, we presented an efficient kinodynamic replanning framework by exploiting the advantageous properties of the B-spline. The proposed EBK search algorithm was flexible and

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

provided a user-specified parameter, which can be used to control the algorithm efficiency and solution quality. The problem of the B-spline-based kinodynamic search on a spatial grid was characterized in detail, and the theoretical performance of the EBK search was analyzed. To compensate for the discretization, we proposed an EO process. We combined the two components into a receding horizon framework. Detailed analysis and comprehensive experiments were carried out to validate the performance. Systematic comparisons against the state of the art were provided to verify the claims. The replanning framework was efficient and complete, and can be used in various kinds of exploration tasks and different kinds of quadrotor testbeds. The current limitation of the framework lies in the fact that for EBK search, we were using the uniform B-spline on the spatial grid, which will result in limited B-spline patterns. In the future, we expect to explore NURBS for kinodynamic search, which will allow for various motion patterns in the kinodynamic search.

APPENDIX

A. Proof of Proposition 1

The correctness of the proposition follows from the fact that the derivative of the B-spline of degree k is another B-spline of degree $k-1$, which enjoys the convex hull property. Specifically, for the lth derivative, let $\mathbf{C}_l$ map the basis $\mathbf{b}$ to the derivatives, i.e., $\frac{d\mathbf{b}}{d^l u} = \mathbf{C}_l \mathbf{b}$. It follows that

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \frac{1}{(\Delta_t)^l}\frac{d\mathbf{b}^\intercal}{d^l u}\mathbf{M}_k\mathbf{P}_j = \frac{1}{(\Delta t)^l}\mathbf{b}^\intercal\mathbf{C}_l^\intercal\mathbf{M}_k\mathbf{P}_j. \tag{7}$$

Plug in $\mathbf{S}_l = \mathbf{M}_k^{-1}\mathbf{C}_l\mathbf{M}_k/(\Delta_t)^l$. It follows that

$$\frac{d\mathbf{s}_j(u)}{d^l u} = \mathbf{b}^\intercal\mathbf{M}_k(\mathbf{S}_j\mathbf{P}_j), \tag{8}$$

where $\mathbf{S}_j\mathbf{P}_j$ is the control point span of the derivative spline. By applying the convex hull property, we complete the proof.

B. Relationship Between $\mathcal{G}$ and $\mathcal{G}_H$

Lemma 1: Given the initial state π_s and goal state π_g, let $\pi = (v_0, v_1, \ldots, v_T)$ be an admissible control point placement of Problem 1. The extended sequence $\tilde{\pi}$ uniquely corresponds to an admissible path $\Phi = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_Q)$ on the graph $\mathcal{G}_H$, with $\hat{v}_0 = \pi_s$, $\hat{v}_Q = \pi_g$, and $Q = K + T + 2$.

Lemma 2: Any admissible path $\Phi = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_Q)$ on the graph $\mathcal{G}_H$, which satisfies $\hat{v}_0 = \pi_s$, $\hat{v}_Q = \pi_g$, and $(\hat{v}_{j-1}, \hat{v}_j) \in E_H$ for $j = 1, \ldots, Q$ uniquely corresponds to an admissible control point placement π of Problem 1.

Proposition 2: Given a strictly positive cost function $f_{k,\Delta_t} : [\tilde{\pi}]^k \to \mathbb{R}_+$, Problem 1 is equivalent to the shortest path problem on the graph $\mathcal{G}_H$, where the cost is defined on the vertices according to the function $f_{k,\Delta_t}(\cdot)$.

C. Characterization of the Inflation for the B-Spline-Based Kinodynamic Search

We denote by $\mathcal{C}^{\mathrm{BK}}$ the configuration space in which we conduct the B-spline-based kinodynamic search. The configuration space $\mathcal{C}^{\mathrm{BK}}$ is generated by inflating all the obstacles by δ^{BK} in the workspace. We take a 26-connected 3-D grid with fixed cell size $d_{\mathrm{x}} \times d_{\mathrm{y}} \times d_{\mathrm{z}}$ and a fifth-degree B-spline as an example. There is a finite number of possible span patterns (27^5 in total). The minimum clearance $c_{\min}^{\mathrm{BK}}$ of the configuration space $\mathcal{C}^{\mathrm{BK}}$ can be expressed by the cell size and obstacle inflation δ_{BK} according to $c_{\min}^{\mathrm{BK}} = \min\left(d_{\mathrm{x}}/2 + \delta^{\mathrm{BK}}, d_{\mathrm{y}}/2 + \delta^{\mathrm{BK}}, d_{\mathrm{z}}/2 + \delta^{\mathrm{BK}}\right)$. The problem of finding the sufficient condition such that the overall trajectory is collision free is equivalent to finding the minimum inflation δ^{BK} such that the trajectories for all the B-spline patterns are completely bounded inside the inflated cells. Since the total number of patterns is finite, δ^{BK} can be found by enumerating all the possible span patterns and picking out the one with the largest deviation. The script is available.[9] Note that the process of finding the sufficient inflation is one-time work prior to the planning process. Therefore, it does not affect the EBK search efficiency. Typically, for a 26-connected 3-D grid with fixed cell size $0.16 \times 0.16 \times 0.16$ m, the inflation needed is less than 0.03 m, which is easy to satisfy in practice.

D. Performance Analysis of the EBK Search

To analyze the performance of the EBK search, we show than the modified INDEX function in Algorithm 3 induces another search graph. The characterization of the search graph unveils the complexity of the EBK search. In the following, we give a formal definition of the search graph given by the modified INDEX function.

We begin with the definition of the nodes, which are called *virtual nodes*. Specifically, given the encoding level d and encoding index e, we denote by $\mathcal{H}_e^d := \{\hat{v} \in V_H | \textsc{Index}\ (\hat{v}, d) = e\}$ the virtual node aggregating all the vertex tuples, which share the same encoding, i.e., the same last d coordinates. It follows that each vertex tuple $\hat{v} \in V_H$ belongs to exactly one virtual node due to the uniqueness induced by the function UNIQUEENCODE$(\cdot)$. For each vertex tuple $\hat{v}_i \in \mathcal{H}_{e_i}^d$, we can obtain the set of neighboring virtual nodes $\{\mathcal{H}_{e_j}^d | e_j = \textsc{Index}\ \hat{v}_j, d, \forall(\hat{v}_i, \hat{v}_j) \in E_H\}$. The interesting part is that every vertex tuple of $\mathcal{H}_{e_i}^d$ has exactly the same set of neighboring virtual nodes. We denote by $\mathcal{E}$ the set of edges between virtual nodes, and we denote by $\mathcal{G}_D = (\mathcal{H}^d, \mathcal{E})$ the *virtual graph* formed by the virtual nodes, where $\mathcal{H}^d$ denotes the set of all virtual nodes.

There are several unique transformations between $\mathcal{G}_D$ and $\mathcal{G}_H$. Given the initial state $\pi_s \in \mathcal{H}_{e_s}^d$ and the goal state $\pi_g \in \mathcal{H}_{e_g}^d$, an *admissible* path $\Theta = (\mathcal{H}_{e_0}^d, \mathcal{H}_{e_1}^d, \ldots, \mathcal{H}_{e_Q}^d)$ on the virtual graph $\mathcal{G}_D$ should satisfy $\mathcal{H}_{e_0}^d = \mathcal{H}_{e_s}^d, \mathcal{H}_{e_Q}^d = \mathcal{H}_{e_g}^d$, and $(\mathcal{H}_{e_{j-1}}^d, \mathcal{H}_{e_j}^d) \in \mathcal{E}$ for $j = 1, \ldots, Q$. By the construction, any admissible path Φ on $\mathcal{G}_H$ uniquely corresponds to an admissible path Θ on $\mathcal{G}_D$ due to the uniqueness of encoding. Given a known initial state π_s, any admissible path Θ on $\mathcal{G}_D$ also uniquely corresponds to an admissible path Φ on $\mathcal{G}_H$.[10] The reason is that the encoding is based on the last $d > 1$ coordinates, and with the known initial virtual node, the original vertex tuple can be reconstructed.

[9] The corresponding script can be found at [Online]. Available: https://github.com/WenchaoDing/kinodynamic_replanning.git.

[10] Without π_s, the uniqueness no longer holds.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

We define the cost to the virtual node to be $c[\mathcal{H}_e^d] = \min\{c[\pi_s, \hat{v}] | \forall \hat{v} \in \mathcal{H}_e^d\}$, where $c[\pi_s, \hat{v}]$ is the minimum cost from the start vertex tuple π_s to $\hat{v}$ according to (5). The EBK search cannot reach the exact goal state π_g, and instead it can only reach the virtual node $\mathcal{H}_{e_g}^d$. In other words, the EBK search can only reach the relaxed goal state. Given the start and goal virtual nodes, the EBK search is complete (finds the optimal admissible path if one exists) with respect to the aggregated graph $\mathcal{G}_D$, as stated in the following theorem.

Theorem 2: Given the graph $\mathcal{G}_D = (\mathcal{H}^d, \mathcal{E})$, the start virtual node $\mathcal{H}_{e_s}^d \in \mathcal{H}^d$ and the goal virtual node $\mathcal{H}_{e_g}^d \in \mathcal{H}^d$, the EBK search finds the optimal admissible path $\Theta = (\mathcal{H}_{e_s}^d, \mathcal{H}_{e_1}^d, \ldots, \mathcal{H}_{e_g}^d)$ on $\mathcal{G}_D$ if one exists.

Proof: We prove by induction and contradiction. The proof follows a similar reasoning process to proving the correctness of Dijkstra's algorithm [36]. And the difference is that for one virtual node, there is one vertex tuple picked out to associate with the virtual node, and the association will be updated during the search process. For the expanded virtual node, the association will be fixed and is supposed to yield the minimum cost to the virtual node among all the aggregated vertex tuples.

Suppose the following hypothesis holds: for each expanded virtual node $\mathcal{H}_v^d$, $c[\mathcal{H}_v^d]$ is the lowest cost from the source virtual node to $\mathcal{H}_v^d$; and for unexpanded virtual node $\mathcal{H}_u^d$, $c[\mathcal{H}_u^d]$ is the lowest cost from the source virtual node to $\mathcal{H}_u^d$ via expanded virtual nodes only. The base case is that there is just the initial virtual node, and the hypothesis holds obviously.

Assume there are $n-1$ expanded virtual nodes, and the hypothesis holds, in which case, the lowest costs to the $n-1$ virtual nodes are known, and given the source virtual node, the vertex tuple associations for the $n-1$ virtual nodes are fixed accordingly. We choose $\mathcal{H}_v^d$ from the $n-1$ nodes such that it has the least $c[\mathcal{H}_u^d] = c[\mathcal{H}_v^d] + f[\mathcal{H}_u^d]$ while satisfying $(\mathcal{H}_v^d, \mathcal{H}_u^d) \in \mathcal{E}_d$, where $f[\mathcal{H}_u^d]$ is the cost of the virtual node.

First, $c[\mathcal{H}_u^d]$ should be the shortest path from the source node to $\mathcal{H}_u^d$. Since if there is a shorter path reaching $\mathcal{H}_u^d$ via the nodes other than the $n-1$ expanded nodes, and $\mathcal{H}_w^d$ is the first unexpanded node on that path, it follows that $c[\mathcal{H}_w^d] > c[\mathcal{H}_u^d]$, which yields a contradiction.

Second, for any of the remaining unvisited nodes $\mathcal{H}_w^d$, $c[\mathcal{H}_w^d]$ should still be the shortest path via the expanded nodes. Since if a lower cost of $c[\mathcal{H}_w^d]$ is found by adding $\mathcal{H}_u^d$ to the expanded nodes, Line 17 of Algorithm 1 will have updated it. Note that the update of $c[\mathcal{H}_w^d]$ will update the association with $\mathcal{H}_w^d$, so the cost and association are consistent. ■

E. Proof of Theorem 1

The correctness of the theorem follows from the convex hull property of the B-spline. For brevity, we only consider one control point span, which consists of $k+1$ control points, but can be generalized to a long control point sequence without any difficulty. The original tube of one control point span consists of $k+1$ balls, and the connectivity is already guaranteed by the two-level inflation scheme. As such, there are k intersection areas for the sequence of $k+1$ control points. Note that here we only consider the intersection between the two balls associated with two neighboring control points. In the extreme case, we add k control points to each of the intersection areas, and the overall control point sequence consists of $k+1+k^2$ control points, i.e., k^2+1 control point spans. By applying the convex hull property to each of the spans, the trajectory for each control point span is bounded inside one of the balls. As such, the iterative process can succeed in k^2 iterations if no violation of the dynamical feasibility is reported.

REFERENCES

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

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

[3] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. Int. J. Robot. Res.*, 2016, pp. 649–666.

[4] 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.

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

[6] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[7] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5054–5061.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 3067–3074.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, pp. 846–894, 2011.

[10] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, pp. 883–921, 2015.

[11] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *Proc. AIAA Guidance, Navig. Control Conf. Exhibit*, 2008, p. 7166.

[12] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4187–4194.

[13] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 528–564, 2016.

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

[15] 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.

[16] J. Van Den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "LQG-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 346–353.

[17] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 6567–6572.

[18] D. Bareiss, J. Van Den Berg, and K. K. Leang, "Stochastic automatic collision avoidance for tele-operated unmanned aerial vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 4818–4825.

[19] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.

[20] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res.*, vol. 28, pp. 933–945, 2009.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.

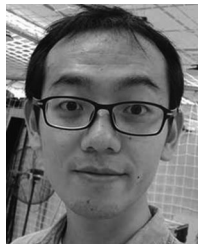
[21] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *Int. J. Robot. Res.*, vol. 35, pp. 224–243, 2016.
[22] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Proc. Robot., Sci. Syst.*, vol. 104, 2010. [Online]. Available: http://www.roboticsproceedings.org/rss06/p34.html
[23] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," Ph.D. dissertation, Dept. Aeronaut. Astronaut., Stanford University, Stanford, CA, USA, 2016.
[24] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 2452–2458.
[25] 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.*, 2016, pp. 5332–5339.
[26] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 42–49.
[27] S. K. Kannan *et al.*, "Close proximity obstacle avoidance using sampling-based planners," in *Proc. AHS Specialists' Meeting Unmanned Rotorcraft Netw. Centric Operations*, 2013.
[28] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3656–3663.
[29] K. Qin, "General matrix representations for B-splines," *Vis. Comput.*, vol. 16, no. 3, pp. 177–186, 2000.
[30] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 561–568, Jun. 2010.
[31] L. Yang, D. Song, J. Xiao, J. Han, L. Yang, and Y. Cao, "Generation of dynamically feasible and collision free trajectory by applying six-order Bezier curve and local optimal reshaping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 643–648.
[32] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2997–3004, Jul. 2019.
[33] F. Gao, Y. Lin, and S. Shen, "Gradient-based online quadrotor safe trajectory planning in 3d complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3681–3688.
[34] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform B-splines and 3D circular buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.
[35] E. Verriest and F. Lewis, "On the linear quadratic minimum-time problem," *IEEE Trans. Autom. Control*, vol. 36, no. 7, pp. 859–863, Jul. 1991.
[36] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[38] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Kuala Lumpur, Malaysia: Pearson Education Limited, 2016.
[39] M. Kleinbort, O. Salzman, and D. Halperin, "Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning," 2016, *arXiv:1607.04800*.
[40] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, no. 2. Belmont, MA, USA: Athena Scientific, 1995.
[41] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT, 2009.
[42] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1993, pp. 802–807.
[43] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. IEEE Control Decis. Conf.*, 2015, pp. 835–842.
[44] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[45] K. Wang, W. Ding, and S. Shen, "Quadtree-accelerated real-time monocular dense mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.
[46] W. Gao and S. Shen, "Dual-fisheye omnidirectional stereo," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6715–6722.
[47] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen, "Optimal time allocation for quadrotor trajectory generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4715–4722.
[48] S. G. Johnson, *The NLopt Nonlinear-Optimization Package*, 2011. [Online]. Available: http://ab-initio.mit.edu/nlopt

Wenchao Ding received the B.Eng. degree in electronic and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently working toward the Ph.D. degree in robotics with the Hong Kong University of Science and Technology under the supervision of Prof. Shaojie Shen.

His research interests include decision making, prediction, motion planning, and autonomous navigation for aerial robots and autonomous vehicles.

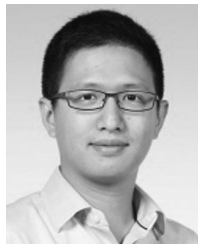
Wenliang Gao received the B.Eng. degree in optical engineering from the Beijing Institute of Technology, Beijing, China, in 2016, and the M.Phil. degree in robotics from the Hong Kong University of Science and Technology, Hong Kong, in 2018.

He is currently an Algorithm Engineer with the DJI, Shenzhen, China. His research interests include state estimation, navigation, and mapping with the visual-inertial system of autonomous robots.



Kaixuan Wang received the B.Eng. degree in automation from Southeast University, Nanjing, China, in 2016.

He joined HKUST Aerial Robotics Group, Hong Kong University of Science and Technology, Hong Kong, under the supervision of Prof. Shaojie Shen. His research interests include real-time three-dimensional perception and mapping for robotics navigation, especially for UAV autonomous flight, and autonomous vehicles.



Shaojie Shen received the B.Eng. degree in electronic engineering from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2009, and the M.S. degree in robotics and the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2011 and 2014, respectively.

He joined the Department of Electronic and Computer Engineering, HKUST, in 2014 as an Assistant Professor. His research interests include robotics and unmanned aerial vehicles, with focus on state estimation, sensor fusion, localization and mapping, and autonomous navigation in complex environments. Dr. Shen is currently an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS and *Autonomous Robots*.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on September 26,2022 at 06:51:35 UTC from IEEE Xplore. Restrictions apply.