Trajectory Replanning for Quadrotors Using Kinodynamic Search and Elastic Optimization

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

Abstract— We focus on a replanning scenario for quadrotors where considering time efficiency, non-static initial state and dynamical feasibility is of great significance. We propose a real-time B-spline based kinodynamic (RBK) search algorithm, which transforms a position-only shortest path search (such as A* and Dijkstra) into an efficient kinodynamic search, by exploring the properties of B-spline parameterization. The RBK search is greedy and produces a dynamically feasible time-parameterized trajectory efficiently, which facilitates non-static initial state of the quadrotor. To cope with the limitation of the greedy search and the discretization induced by a grid structure, we adopt an elastic optimization (EO) approach as a post-optimization process, to refine the control point placement provided by the RBK search. The EO approach finds the optimal control point placement inside an expanded elastic tube which represents the free space, by solving a Quadratically Constrained Quadratic Programming (QCQP) problem. We design a receding horizon replanner based on the local control property of B-spline. A systematic comparison of our method against two state-of-the-art methods is provided. We integrate our replanning system with a monocular vision-based quadrotor and validate our performance onboard.

I. INTRODUCTION

Micro aerial vehicles (MAVs), in particular quadrotors, have gained wide popularity in various inspection and exploration applications. To meet the need for fully autonomous navigation in unexplored environments, a real-time local replanner producing smooth, dynamically feasible trajectories is of great significance. Many existing planning methods [1] [2] [3] [4] follow a path planning and path parameterization two-step pipeline. The path planning part only produces unparameterized path, while the path parameterization part chooses a feasible dynamical profile for the path. The two-step pipeline is popular due to its efficiency, but is problematic in replanning scenario, since the path planning part is unaware of the vehicle's non-static initial states. For instance, the geometrically shortest path may diverge from the initial velocity direction, resulting in jerky trajectories or failure of the path parameterization.

Considering the replanning for quadrotor which has non-trivial dynamics, it is highly desirable to use kinodynamic motion planners, to facilitate non-static initial state and ensure dynamical feasibility. Sampling-based methods such as kinodynamic RRT* [6] are asymptotically optimal but

Fig. 1: Illustration of our experimental testbed which is equipped a monocular camera, an Intel i7 processor and a NVIDIA Jetson TX1. The localization module is based on our Monocular Visual Inertial Navigation System (VINS-Mono) [5], and mapping module is based on monocular dense mapping and TSDF fusion. A video of the experiment can be found at the attachment.

computationally expensive with an execution time of 10s to 100s. Allen *et al.* [7] proposed a real-time kinodynamic adaptation of FMT*, however, the computation time is still around half a second. Another issue of sampling-based methods is that the randomized behavior may result in unpredictable performance [3], especially when only limited sampling is permitted. Search-based method is suitable for replanning in the sense that its results are consistent given similar observations of the environment. To this end, Liu *et al.* [8] proposed a primitive-based resolution-complete (i.e., optimal in the induced lattice graph) search method using linear quadratic minimum time control. However, the running time is sensitive to both the discretization level and the order of control input. For the high-order control input (such as jerk-controlled) or large dynamic range which needs a higher discretization level, the running time is not stable and may be around one or two seconds.

In this paper, we explore a novel angle of kinodynamic search by searching for B-spline control point placement, to improve the time efficiency. Different from many existing works [9] [10] [11] which follow the two-step pipeline and use B-spline as the path parameterization method, we produce time-parameterized trajectory in the search process directly. Our contribution is that we explore three properties of B-spline to transform a low-complexity position-only search (such as A* and Dijkstra) into an efficient kinodynamic search. First, thanks to B-spline's *local control* property as elaborated in Sec. III, the evaluation of B-spline can be done locally, resulting in an efficient expansion of control points, without re-evaluation of whole trajectory. Second, based on the *convex hull* property of B-spline, the dynamical feasibility constraints can be rewritten in a linear form, avoiding the computation overhead brought by

feasibility checking. Third, the control cost can be evaluated in *closed-form*, without solving computationally expensive two-point boundary value problem BVP problem. The resulting real-time B-spline based kinodynamic (RBK) search shares a similar structure and complexity to the position-only search, but outputs dynamically feasible time-parameterized trajectory, which facilitates non-static initial state.

The RBK search is essentially a greedy adaptation of a full-scale B-spline based kinodynamic search. The full-scale search is optimal, but scales poorly with respect to the spline order and number of connections of the grid, as elaborated in Sec. IV. As verified in Sec. VI-A, the solution cost (the control cost and time cost) of the RBK seach is between the full-scale kinodynamic search and the position-only shortest path search, and the RBK search is at least two orders of magnitude faster than the full-scale search. To cope with the potential sub-optimality and grid discretization in the RBK search, we propose an elastic optimization (EO) approach as a post-optimization process, which finds the optimal control point placement inside the expanded elastic tube and ensures dynamical feasibility, by solving a Quadratically Constrained Quadratic Programming (QCQP) problem. Thanks to the enforcement of dynamical feasibility during the RBK search, the initial trajectory fed to elastic optimization is already feasible, enhancing the robustness of the post-optimization.

We complete the replanning system by further adopting a receding horizon planning (RHP) framework [3] with a B-spline specification using its local control property. A sliding window optimization strategy [12] is also introduced to bound the complexity of EO approach. We compare our replanning system with two state-of-the-art replanners [8] [13] in simulation, and validate our performance on a real monocular vision-based quadrotor as shown in Fig. 1.

II. System Overview

The overview of our autonomous planning system is shown in Fig. 2. We call RBK search the system *front-end*, and EO approach *back-end*. The RBK search provides an initial placement of control point, which will be refined during the optimization process. The elastic optimization approach consists of two steps, namely, the elastic tube expansion and elastic optimization. There are two modes of our front-end, i.e., passive mode and active mode. For the passive mode, the front-end is activated by collision checking, while in the active mode, the front-end is not only activated by collision checking, but also by a timer depending on the B-spline knot separation. The active mode is used to reject the effects of outliers, when the mapping is noisy in some circumstances. Basically, we work on a series of densely connected B-spline control points, which are constantly modified by the RBK search to avoid collision, and refined by the EO approach.

III. B-spline Curve and Replanning

A. Local Control Property and Replanning

We use uniform B-spline as the trajectory parameterization method. The evaluation of B-spline of degree $k-1$ with uni-
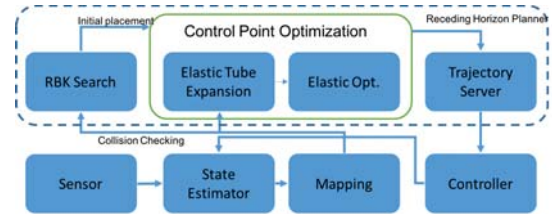

Fig. 2: A diagram of our autonomous replanning system.

form knot sequence $\{t_0, t_1, t_2, \ldots, t_n\}$ of fixed time interval Δt can be evaluated using the following equation:

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{v}_i N_{i,k}(u), \tag{1}$$

where u is the normalized parameter which can be computed according to $u = (t - t_i)/\Delta t$, for $t \in [t_i, t_{i+1}]$. $\mathbf{v}_i \in \mathbb{R}^3$ is the control point at time t_i and $N_{i,k}(t)$ is the blending function, which can be computed via the De Boor-Cox recursive formula [14]. Following the practice of B-spline which calls k consecutive knots as a *knot span*, we call the corresponding stacked control points as a *span*. For instance, for the knot span $\{t_0, t_1, t_2, \ldots, t_{k-1}\}$, the corresponding span is defined by $\mathbf{V}_0 := [\mathbf{v}_0 \, \mathbf{v}_1 \cdots \mathbf{v}_{k-1}]^{\mathsf{T}} \in \mathbb{R}^{k \times 3}$.

An important property of B-spline is the *local control*. Namely, a single span of a B-spline curve is controlled only by k control points, and any control point only affects k spans, as shown in Fig. 3 (a). Mathematically, for the i-th span covering k knots from t_i to t_{i+k-1}, the corresponding curve and its l-th order derivative can be evaluated in closed-form as follows:

$$\frac{d\mathbf{c}_i(u)}{d^l u} = \frac{1}{(\Delta t)^l} \frac{d\mathbf{b}^{\mathsf{T}}}{d^l u} \mathbf{M}_k \mathbf{V}_i, \tag{2}$$

where $\mathbf{b} = \begin{bmatrix} 1 \, u \, u^2 \cdots u^{k-1} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^k$ denotes the basis vector, $\mathbf{M}_k = (m_{i,j}) \in \mathbb{R}^{k \times k}$ denotes the basis matrix, where $m_{i,j} = \frac{1}{(k-1)!} \binom{k-1}{k-1-i} \sum_{s=j}^{k-1} (-1)^{s-j} \binom{k}{s-j} (k-s-1)^{k-l-i}$, and $\mathbf{V}_i = [\mathbf{v}_i \, \mathbf{v}_{i+1} \cdots \mathbf{v}_{i+k-1}]^{\mathsf{T}} \in \mathbb{R}^{k \times 3}$ stacks the k control points of the i-th span.

We incorporate the local control property into both a kinodynamic search process and a receding horizon planning framework. The usage of local control property in the kinodynamic search is elaborated in Sec. IV and its usage in the receding horizon planning is shown in Fig. 3 (b). The control points are divided into four types, namely, executed control points, committed control points, optimizing control points (the control points inside the sliding window) and unoptimized control points. The control point is unchangeable once committed. A stopping policy will be activated if a collision is detected for the committed trajectory. The RBK search and EO approach only affect the optimizing and unoptimized control points. As a result, the replacement caused by either the search or optimization will not cause any disturbance to the evaluation at the trajectory server.

B. Convex Hull Property and Dynamical Feasibility

Another important property of B-spline is the *convex hull* property. Given Eq. 2, the dynamical feasibility constraints
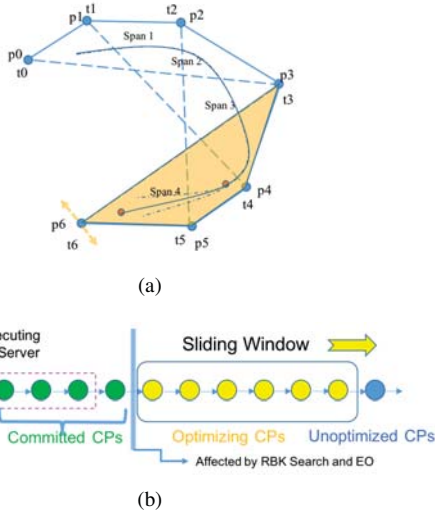
(a)

(b)

Fig. 3: (a) shows an example of cubic B-spline. $\{p_0, \ldots, p_6\}$ denote the six control points, while $\{t_0, \ldots, t_6\}$ denote the corresponding knots. Adjusting p_6 will only affect the trajectory corresponding to span 4. (b) shows how the local control property can be applied to the replanning system.

can be expressed by maximum derivative (velocity, acceleration, jerk, etc.) bounds in terms of control point placement. After applying the *convex hull* property, different order derivatives of the whole B-spline curve can be completely bounded using only linear constraints, as in the following Prop. 1. The limitation of Prop. 1 is its conservativeness, given that Prop. 1 is a sufficient but not necessary condition.

Proposition 1. *Given uniform B-spline of order $k - 1$ and time step Δt, there exists a **constant** linear combination $\mathbf{S} = \mathbf{M}_k^{-1}\mathbf{C}_l\mathbf{M}_k/(\Delta t)^l \in \mathbb{R}^{k \times k}$, such that $|\mathbf{S}\mathbf{v}_i^D| \leq u_{l,D}^{max}\mathbf{1}_{k \times 1}$ is a sufficient condition for the derivative along coordinate D to be thoroughly bounded for the whole span, i.e., $\left(\frac{dc(u)}{d^l u}\right)_D \leq u_{l,D}^{max}, \forall u \in [0,1]$, where $\mathbf{v}_{l,D}^D \in \mathbb{R}^k$ is the stacked position vector of coordinate $D \in \{X, Y, Z\}$ in i-th span, and $\mathbf{C}_l \in \mathbb{R}^{k \times k}$ is a constant mapping matrix of the l-th derivative satisfying $\frac{d\mathbf{b}}{d^l u} = \mathbf{C}_l\mathbf{b}$.* [1]

Proof. Interested readers may refer to our report [15] for detailed proof. □

As described by Mellinger [2], the control cost of quadrotor trajectory can be expressed by the integral over squared derivatives (such as fourth derivative snap), which can also be evaluated in closed form in the case of uniform B-spline. The total control cost E_c of the i-th span can be expressed by a weighted sum of the integral over squared derivatives of different orders as follows:

$$E_i^c = \sum_{l=1}^{k-2} \int_0^1 w_l \left(\frac{dc_i(u)}{d^l u}\right)^2 du = \sum_{l=1}^{k-2} w_l \mathbf{V}_i^\mathsf{T} \mathbf{M}_k^\mathsf{T} \mathbf{Q}_l \mathbf{M}_k \mathbf{V}_i \tag{3}$$

where $\mathbf{Q}_l = \int_0^1 \left(\frac{d\mathbf{b}}{d^l u}\right)\left(\frac{d\mathbf{b}}{d^l u}\right)^\mathsf{T} du/(\Delta t)^{l+1}$ is the Hessian Matrix of the l-th squared derivative, which is constant for uniform B-spline, and w_l is the corresponding weight.

---

[1] $|\cdot|$ means the element-wise absolute value of a vector.

Without special mention, our replanning system uses quintic uniform B-spline ($k = 6$) to ensure the continuity up to snap for quadrotor systems.

IV. B-SPLINE BASED KINODYNAMIC SEARCH

A. Basics of B-spline Based Kinodynamic Search

In this section, we introduce the basics of B-spline based kinodynamic search. A uniform distributed M-connect spatial grid with N cells is chosen as the graph structure $\mathcal{G}$, with cell centers as vertices $\mathcal{V}$, and their connections as edges $\mathcal{E}$, i.e., $\mathcal{G} \coloneqq (\mathcal{V}, \mathcal{E})$. For instance, a 3-D grid with each cell connected to its neighbors is 26-connect. Two spans $\mathbf{V}_i$ and $\mathbf{V}_j$ are said to be neighboring spans if and only if the last $k - 1$ control points of $\mathbf{V}_i$ coincide with the first $k - 1$ control points of $\mathbf{V}_j$. We define the initial system state as a given span $\mathbf{V}_{\text{init}} \coloneqq [\mathbf{v}_0, \mathbf{v}_1, \ldots \mathbf{v}_{k-1}]^\mathsf{T}$ containing the first k control points. In the same way, goal state $\mathbf{V}_{\text{goal}}$ can be defined. Note that the initial and goal state we define are slightly different from those in traditional kinodynamic planners [6], given the fact that $\mathbf{V}_{\text{init}}$ and $\mathbf{V}_{\text{goal}}$ actually represent two short trajectories. The B-spline based kinodynamic search problem is finding a set of neighboring spans $\mathcal{S} \coloneqq \{\mathbf{V}_0, \ldots, \mathbf{V}_{n-k+1}\}$ that minimizes the following cost function,

$$\min_{\mathcal{S}} \quad J(\mathcal{S}) = \sum_{i=0}^{n-k+1} E_i^c + \lambda(n+1)\Delta t \tag{4}$$

where $\lambda \geq 0$ is the weight of total time cost. The *dynamical feasibility* constraints can be checked in terms of span. The *collision-free* constraints are enforced by choosing control point placement in unoccupied cells. Later we will discuss how to ensure the safety of resulting trajectory based on this.

To solve the graph search problem optimally, we need a *full-scale* search in terms of B-spline spans. The complexity of the full-scale search depends on the number of all possible B-spline span patterns. Typically, searching for $k - 1$-order B-spline spans on an M-connected grid will induce an M-connected span graph $\mathcal{G}'$ with a scaling factor $O(M^{k-1})$ of the number of vertices, if no pruning is applied. For instance, for 5-order B-spline search on a 26-connect 3-D grid, the vertex scaling factor is almost 10^7, which is unacceptable.

B. Real-time B-spline Based Kinodynamic Search

Given the high complexity of the full-scale search, we present the RBK search algorithm, which transforms the low-complexity position-only search into an efficient kinodynamic search, by exploring three properties of B-spline, namely, *local control* property, *convex hull* property and *closed-form evaluation* of control cost. The three properties exactly correspond to the three core operations in the search process, namely, span retrieving and expansion, dynamical feasibility checking and cost evaluation. The RBK algorithm shares a similar structure and complexity to position-only search algorithms such as A* and Dijkstra, but obtains the ability of evaluating the control cost and ensuring the dynamical feasibility. The result of the RBK search is dynamically feasible time-parameterized trajectory instead

Algorithm 1: RBK($\mathbf{V}_{\text{init}}, \mathbf{V}_{\text{goal}}, \mathcal{G}$)

---

1 Initializes: $\mathcal{O} = \mathcal{I} = \emptyset$;

2 Add($\mathcal{O}, p_{\text{new}}, \text{Cost}(\mathbf{V}_{\text{init}})$) ;

3 **while** size($\mathcal{O}$)$! = 0$ **do**

4 $p_{\text{cur}} \leftarrow \text{PopMin}(\mathcal{O})$, Add($\mathcal{I}, p_{\text{cur}}$) ;

5   ($\mathbf{V}_{\text{cur}}, c_{\text{cur}}$) $\leftarrow \text{RetrieveSpan}(p_{\text{cur}})$;

6   **if** NearEnd($\mathbf{V}_{cur}, \mathbf{V}_{goal}$) **then**

7   **return** success ;

8   **end**

9   **for** $\mathbf{V}_{nbr} \in \text{Neighbor}(\mathbf{V}_{cur}) \cap p_{nbr} \notin \mathcal{I}$ **do**

10   **if** CheckDynamics($\mathbf{V}_{nbr}$) **then**

11 $c' = \text{Cost}(\mathbf{V}_{nbr}) + c_{\text{cur}}$;

12   **if** $p_{nbr} \in \mathcal{O}$ **then**

13   **if** $c' < \text{RetrieveCost}(p_{nbr})$ **then**

14   Update($p_{\text{current}}, p_{nbr}$) ;

15   **end**

16   **else**

17   Add($\mathcal{O}, p_{nbr}, c' + \text{HeuristicCost}(\mathbf{V}_{nbr})$) ;

18   **end**

19   **end**

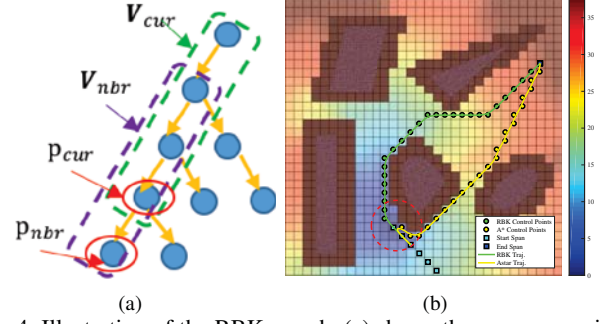**20**   **end**

21 **end**

Fig. 4: Illustration of the RBK search. (a) shows the span expansion process (Sec.IV-B), and (b) shows the result of the RBK search and the comparison with the position-only A* search.

of unparameterized path, thus being different from position-only search. Denote open set and closed set as $\mathcal{O}$ and $\mathcal{I}$, and current grid node and neighboring grid node as p_{cur} and p_{nbr}.

Span retrieving and expansion: the position-only search algorithms such as A* and Dijkstra implicitly maintain a tree structure via the predecessor data structure. Instead of retrieving one predecessor as in the position-only search, the RBK search retrieves $k - 1$ predecessors. By stacking these $k - 1$ predecessors with current control point, the current span is formed. The retrieving process is implemented in function RetrieveSpan(p_{cur}) in Alg. 1 and is shown in Fig. 4 (a). Thanks to the local control property, the expansion can be done locally just like position-only search, since there is no need to re-evaluate of the whole trajectory when expanding to new control points. Therefore, using local control property, the tree structure in A* and Dijkstra can be transformed into a tree of spans, enabling B-spline based search.

Dynamical feasibility checking: using the *convex hull* property, we derive a sufficient condition for dynamical feasibility as introduced in Sec. III, which is actually linear. By using this condition in function CheckDynamics($\mathbf{V}_{\text{nbr}}$), dynamical feasibility can be *guaranteed* without computation overhead, which speeds up the kinodynamic search process.

Cost evaluation: Traditional kinodynamic planners [6] rely on solving the two-step boundary value problem (BVP) to determine the cost of connecting two states, which is computationally expensive. By using B-spline parameterization, the cost of connecting to new expanded span can be evaluated using closed-form solution according to Eq. 3, which is implemented in function Cost($\mathbf{V}_{\text{nbr}}$) in Alg. 1. The limitation of our method is that the expansion of spans are restricted by the resolution and connection of the grid. For example, for 26-connect 3-D grid, the connection to the next control point is restricted to the 26 connections, resulting in

limited representations of B-spline trajectories.

The trajectory of the RBK search can be *guaranteed* to be collision free if moderate obstacle inflation is done. The analysis of how much inflation is needed is shown in our report [15], which means that the RBK search can be used as a standalone component without any post-processing, when computation is limited. The termination condition is implemented in NearEnd($\mathbf{V}_{\text{cur}}, \mathbf{V}_{\text{goal}}$) in Alg. 1. In practice, we terminate the search when the first control point of end span is reached and append the end span to it, since we observe that the part of dynamical profile can be automatically smoothed by the back-end. To speed up the search process, we use HeuristicCost($\mathbf{V}_{\text{nbr}}$) to focus on searching in the most promising direction. In practice, one admissible heuristic can be $\lambda \cdot \text{norm}(p_{\text{nbr}} - p_{\text{goal}})/v_{\text{max}}$, where p_{goal} is the first control point of end span. The design of more informative heuristic function is left as a future work.

As shown in Fig. 4 (b), based on the start span, the RBK search explores the low-cost region and generates dynamically feasible time-parameterized trajectory, while A* finds a shortest but unparameterized path, which does not contain any kinodynamic information. If we directly use the A* shortest path as B-spline control points placement, the resulting trajectory has large acceleration at the beginning, which is not good for replanning. As verified in Sec. VI-A, the RBK search generally finds better trajectories in terms of the control cost and time cost, compared to the position-only A*. Moreover, the dynamical feasibility is guaranteed for the RBK search. As verified in Sec. VI and Sec. VII, Alg. 1 only needs around 20 milliseconds to reach a good solution.

V. ELASTIC OPTIMIZATION

To compensate for the potential sub-optimality of the RBK search and the limitation induced by the discretization of the grid, we present the post-optimization process, i.e., the EO approach, to fully utilize free space and further reduce the trajectory cost. Since the initial trajectory provided by the RBK search is already dynamically feasible and time-parameterized, the post-optimization is equipped with at least one feasible solution, which makes the optimization process robust. The EO approach can be divided into two components, namely, an elastic tube expansion algorithm

Algorithm 2: $(\mathcal{Q}, \mathcal{R}') \leftarrow \texttt{ElasticTube}(\mathcal{P}, \mathcal{C}^{\text{ELAS}})$

---

1 Initializes: $d_{\text{infl}}^{\min}, d_{\text{infl}}^{\max}, d_{\text{thres}}. \mathcal{R} = \mathcal{R}' = \mathcal{Q} = \emptyset$;

2 **for** $\mathbf{p}_i \in \mathcal{P}$ **do**

3 $\quad$ $(\mathbf{n}_i, r_i) \leftarrow \texttt{NNSearch}(\mathbf{p}_i, \mathcal{C}^{\text{ELAS}})$;

4 $\quad$ $\overrightarrow{f} = (\mathbf{p}_i - \mathbf{n}_i)/\texttt{norm}(\mathbf{p}_i - \mathbf{n}_i)$;

5 $\quad$ **while** $d_{\text{infl}}^{\max} - d_{\text{infl}}^{\min} > d_{\text{infl}}^{tol}$ **do**

6 $\quad\quad$ $d \leftarrow \left(d_{\text{infl}}^{\max} + d_{\text{infl}}^{\min}\right)/2, \mathbf{p}_{i,\text{infl}} \leftarrow \mathbf{p}_i + d \cdot \overrightarrow{f}$;

7 $\quad\quad$ $(\mathbf{n}_i', r_i') \leftarrow \texttt{NNSearch}(\mathbf{p}_{i,\text{infl}}, \mathcal{C}^{\text{ELAS}})$;

8 $\quad\quad$ **if** $\texttt{abs}(r_i' - d - r_i) > d_{\text{thres}}$ **then**

9 $\quad\quad\quad$ $d_{\text{infl}}^{\max} \leftarrow d$;

10 $\quad\quad$ **else**

11 $\quad\quad\quad$ $d_{\text{infl}}^{\min} \leftarrow d$;

12 $\quad\quad$ **end**

13 $\quad$ **end**

14 $\quad$ $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathbf{p}_{i,\text{infl}}, \mathcal{R}' \leftarrow \mathcal{R}' \cup r_i'$;

15 **end**
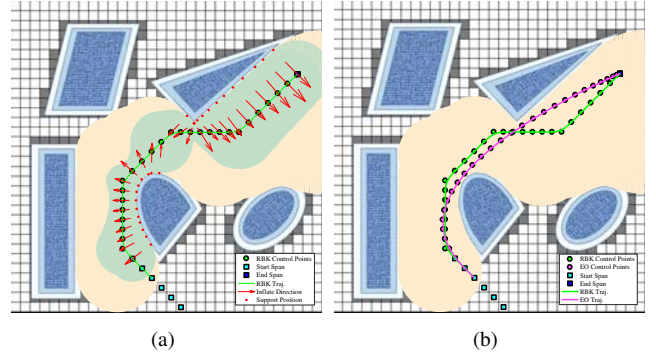

(a) $\qquad\qquad$ (b)

Fig. 5: Illustration of elastic optimization approach. (a) shows the elastic tube expansion process (Sec. V-A). (b) shows the elastic optimization process (Sec. V-B).

and an elastic optimization formulation. The elastic tube expansion algorithm generates a series of connected local maximum volume "bubbles", i.e., an elastic tube. The elastic optimization is to stretch the trajectory (so-called "elastic band") inside the tube, so that the objective is minimized.

A. Elastic Tube Expansion

We denote the set of control point placement provide by Alg. 1 as $\mathcal{P} := \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_m\}$ as the initial placement, excluding the first $k - 1$ control points of the start span, and the last $k - 1$ control points of the end span. As shown in Alg. 2, the elastic tube expansion algorithm can be divided into two steps: first, we construct the initial tube, by doing radius search for the initial placement $\mathcal{P}$ and get the nearest obstacle position $\mathbf{n}_i$. Second, we push the center of the bubbles in the direction $\overrightarrow{f}$ (away from the nearest obstacle), while satisfying the criterion that the new bubble contains the original bubble, as required by condition $\texttt{abs}(r_i' - d - r_i) \leq d_{\text{thres}}$, as shown in Fig. 5 (a). The inflation process is implemented in a binary search manner to reduce calling $\texttt{NNSearch}$. Alg. 2 will finally find a series of local maximum volume bubble centers $\mathcal{Q} := \{\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_m\}$ based on the initial tube $\mathcal{P}$. As 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 $\texttt{NNSearch}$ is the nearest neighborhood search which can be done efficiently if a KD-tree is maintained. The actual run time of Alg. 2 is around 2 millisecond, as verified in Section. VI.

B. Elastic Optimization Formulation

We are inspired by the elastic smoothing method for car-like robots in [16]. The key insight of [16] is the convex representation of geometric smoothness and enforcement of speed feasibility. However, the formulation in [16] is completely based on car model. In contrast to [16] which

worked on a tube limited by the initial path, we use Alg. 2 to generate a local maximum volume elastic tube to enhance the optimization performance. Furthermore, we represents the control cost and dynamical feasibility constraints in closed-form using the high-order B-spline parameterization for controlling quadrotors.

We denote $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_m\}$ as the control point placement (optimization variable). We denote the stacked spans as $\{\mathbf{V}_i | i \in \{-k+1, -k+2, \dots m\}\}$ which also contain the hybrid spans which consist of optimization variable and fixed start span or end span control points. The optimization problem can be expressed as follows:

$$\min \quad \sum_{i=-k+1}^{m} E_i^c$$
$$\text{s.t} \quad \mathbf{x}_0 = \mathbf{p}_0, \mathbf{x}_m = \mathbf{p}_m \tag{5}$$
$$\|\mathbf{x}_i - \mathbf{q}_m\|_2 \leq r_i', \qquad \forall i \in \{2, \dots, m-1\}$$
$$|\mathbf{S}\mathbf{v}_i^D| \leq u_{l,D}^{\max} \mathbf{1}_{k \times 1}, \quad \forall i, D, l \in \{1, \dots k-2\}$$

where the first constraint is the position constraint of the first and last control point to maintain the connectivity. The second constraint is to restrict the control point inside the safety space, which is quadratic. The third constraint is to ensure the dynamical feasibility using the sufficient condition in Prop. 1, which is linear. As a whole, the formulation is a QCQP. As verified in VI, the EO approach can be completed in around 30 milliseconds.

C. Enforcing Safety Guarantee

One issue of the elastic optimization formulation is that restricting control points inside the bubbles is not a sufficient condition for safety, since 1) bubbles are placed with finite density and 2) B-spline does not exactly pass the control points. The issue can be resolved in two steps: first we show that piece-wise linear segments connecting the control points can be completely contained in collision-free space using a two-level obstacle inflation strategy; and second, the B-spline trajectory can be pulled arbitrarily closed to the piece-wise linear segments using an iterative process of adding control points. The two-level obstacle inflation strategy is as follows: we model the robot as a ball with radius δ_r, and we search in configuration space $\mathcal{C}^{\text{RBK}}$ with a larger minimum clearance $c_{\min}^{\text{RBK}}$, while we generate the elastic tube and optimize the

control points in the configuration space $\mathcal{C}^{\mathrm{ELAS}}$ with a smaller minimum clearance $c_{\min}^{\mathrm{ELAS}}$ as shown in Fig. 5. Based on the given grid size of RBK search, the maximum distance between control points $d_{\max}$ can be computed. By the two-level inflation, we could have $2(c_{\min}^{\mathrm{RBK}} - c_{\min}^{\mathrm{ELAS}}) > d_{\max}$, which is used to ensure the connectivity of the initial tube. Based on this, a sufficient condition for the piece-wise linear segments to be completely contained inside the elastic tube is that $c_{\min}^{\mathrm{ELAS}} - \delta_r > \frac{\sqrt{2}-1}{2}d_{\max}$ [2]. Then, if the collision is caused by the deviation from the piece-wise linear segments, control points are added till the collision is resolved. Thanks to the local control property, this can be done with high efficiency without re-evaluation of the whole trajectory. In the case of collision, we can always *guarantee* the success of iterative process by realizing that if the same $k-1$ control points are added to one control point, the robot will exactly pass through that control point. The linear segment can be closely followed by adding control points on both sides. [3]

VI. ANALYSIS

We begin with an individual test for the RBK search, then we test the replanning system. A set of default parameters is used for all the simulation: 1) Maximum velocity and acceleration are set to $3.5m/s$ and $5.2m/s^2$ respectively. Accordingly, $u_{1,D}^{\max} = 2.0m/s$ (velocity bound for each axis) and $u_{2,D}^{\max} = 3.0m/s^2$. 2)The B-spline order is set to 5, and the time step Δt is set to $0.15s$. 3) The control cost in Eq.3 is set to minimizing snap (only w_4 being non-zero). The weight of total time cost λ in Eq. 4 is set based on the criterion that the time cost and the control cost are at the same order for each span. 4) For the replanning system, the RBK search is implemented on a $60 \times 60 \times 20$ uniform grid, with an actual scale of $10m \times 10m \times 3m$. The sliding window size is set to 12, and the local sensing range is set to $4m$.

A. Kinodynamic Search Performance

To verify the performance of the RBK search, we compare it with the full-scale B-spline based kinodynamic search and position-only A* path search. Since the full-scale B-spline based kinodynamic search is of high complexity, the comparisons are done on a relatively small grid of 14×14. We can not directly compare our method with the position-only A* search according to the cost function defined in Eq. 4, since the path returned by the position-only search does not contain any kinodynamic information. Hence, we assume that the position-only search results are directly parameterized using B-spline, to illustrate the performance if a position-only search is used as the system front-end.

As shown in Fig. 6 (a), if the position-only A* shortest path is used as the control point placement, there are dynamically infeasible parts as marked in red. Our RBK search and the full-scale B-spline based search both *guarantee* dynamical feasibility, and in this case our RBK method actually
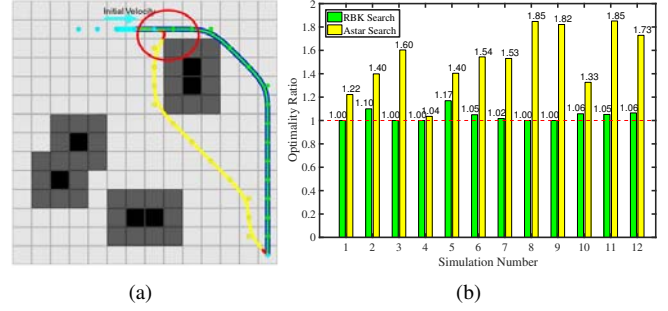

(a)   (b)

Fig. 6: Illustration of the RBK search performance and comparisons. In (a), the start span (in cyan) has a non-zero initial velocity pointing to the right. The end span (in cyan) is at the same grid representing a static state. The path of the position-only A* search (in yellow) and dynamically infeasible parts (in red) are shown. The results of the full-scale B-spline based search and the RBK search are marked in blue and green respectively.

finds the optimal solution. We further conduct Monte-Carlo testing by randomly choosing the start state (random location and random pattern of the span) and the obstacle location. As shown in Fig. 6 (b), we demonstrate the optimality ratio statistics (i.e., total cost divided by the optimal cost given by the full-scale search). The RBK search finds a lower cost solution than position-only A* search for all 12 rounds and reaches the optimal solution for five rounds. As for the time efficiency, the average computing time for the position-only A*, the RBK and full-scale search are $0.0003s$, $0.0035s$ and $7.29s$, respectively. Therefore, the RBK search is about three orders of magnitude faster than the full-scale search even on the small grid. Note that more efficient heuristics can be developed to improve the RBK performance and characterize the optimality gap, and this will be left as a future work.

B. Run Time Analysis of Replanning System

In this section, we test our replanning system on two different maps, including the random map and Perlin noise map[4]. The Perlin noise map is a complex 3-D world, as shown in Fig. 7(b). We evaluate the run time efficiency of our replanning system and list the statistics of all components as shown in Table. I. The overall trajectory illustrating the whole round trip is shown in Fig. 7.

TABLE I: Run Time Analysis on Different Maps

Maps	# Replans	Time(s)	RBK 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

As we can see from Tab. I, on the random map, the RBK method consumes an average computing time of $0.017s$ with a standard deviation of $0.01s$. The elastic tube expansion method can be done in $0.002s$, which is much faster than traditional free space segmentation methods such as IRIS [17], which may take up to $0.1s$ to find a solution. The elastic optimization can be done in $0.021s$. On the Perlin noise map,

---

[2]Interested readers may also refer to our report [15] for detailed proof.

[3]The strategy is to guarantee collision-free for the post-processing.

[4]https://github.com/HKUST-Aerial-Robotics/mockamap
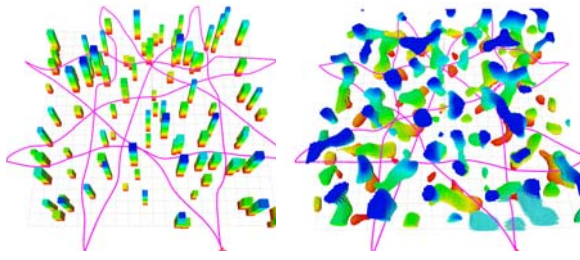
(a) Replan on the random map (b) Replan on the perlin noise map

Fig. 7: Illustration of our replanning system on different maps.

TABLE II: Performance of Different Replanning Methods

Method	Trajectory Statistics		Time Efficiency(s)		
	Mean Vel. (m/s)	Max Acc.(m/s^2)	Ave	Max	Std
SMP	1.46	1.73	0.064	0.621	0.086
SMP (Conservative)	1.20	1.56	0.010	0.080	0.012
Our Method	1.17	2.69	0.035	0.064	0.012

which contains unstructured 3D obstacles, our method has a similar performance as on the random map, showing that our method works well in complex 3-D environments.

C. System Comparison With State-of-the-Art Methods

In this section, we conduct a system comparison with two state-of-the-art methods, including the Continuous Trajectory (CT) [13] method and Searched-based Motion Primitive (SMP) method by Liu *et al.* [8]. The SMP method [8] constructs motion primitives online based on linear quadratic minimum time control, and conducts heuristic-guided search to generate dynamically feasible trajectories. We set up a challenging obstacle-cluttered 3-D complex simulation environment containing walls and 3-D steps, as shown in Fig. 8. The replanning strategy is choosing a local target on a given straight-line guiding path. For CT method, the initial guiding path is parameterized using polynomial, following their practice. The CT method tries to use gradient-based trajectory optimization to push a short local trajectory out of the Euclidean Signed Distance Field (ESDF) formed by the obstacles. For SMP method, we use receding horizon planning in their paper [8], namely, for every time slot, we plan for the next time slot while executing current committed trajectory. The replanning strategy for our method is the passive mode, as introduced in Sec.II. We evaluate the replanning system from the trajectory statistics and time efficiency perspectives (as shown in Tab. II, CT is excluded in the table due to its low success rate).

As shown in Fig. 8 (a), the CT method uses the ESDF to push the initial trajectory out of the obstacles, but suffers from a low success rate due to local minimum issue. The SMP method is not real-time when the full dynamic range is applied, since a large dynamic range requires a higher level of discretization of the control input, which will result in a dramatically growing state lattice. As shown in Tab. II, the SMP under a full dynamic bound may need up to $0.621s$ (with a $0.086s$ standard deviation), which is unacceptable for real-time replanning. If we limit the dynamic bounds to be conservative, the SMP method (so-called SMP Conservative) is efficient but scarifies the maneuverability. Note that only acceleration-controlled SMP can give a real-time performance (10Hz) in 3-D environments [8]. To obtain a smooth enough control input, SMP uses the unconstrained QP formulation in [1] to reparameterize the trajectory, but this post-processing has no safety or dynamical feasibility guarantee.

The RBK method can be done efficiently under full dynamic bounds, since grid-based B-spline search actually induces a discretization of the state space. The RBK search outputs a snap-continued trajectory, while the real-time SMP is only velocity-continued. The limitation of the RBK method is its greedy nature and limited representation of trajectories due to the discretization of the grid. The advantage of using RBK search is to obtain a dynamically-feasible time-parameterized trajectory efficiently. Thanks to this property, the post-optimization process has a feasible initial solution and rarely fails. Moreover, the EO refinement provides both safety and dynamical feasibility guarantee.

VII. EXPERIMENTAL RESULTS

For onboard testing, the parameters are as follows: the time step Δt is set to $0.35s$; the maximum velocity and maximum acceleration are set to $1.2m/s$ and $2.0m/s^2$ repectively; and the local planning range is set to $10m \times 6m \times 1.1m$ (the corresponding gird size is $55 \times 35 \times 6$).

A. Indoor Replanning Performance

As shown in Fig. 9, our replanning system works in complex 3-D environment with only local map. The whole trajectory length is $18.6m$ and total trajectory execution time is $43.4s$. The average velocity of the quadrotor is $0.45m/s$ with a maximum velocity of $0.79m/s$. The maximum acceleration of the trajectory is $0.58m/s^2$ and the whole trajectory is dynamically feasible. There are totally 125 calls of RBK search (active mode) with average computation time $0.010s$, since the grid size is smaller compared to that in simulation. There are 105 calls of EO. The average computation time of elastic tube expansion and elastic optimization is $0.001s$ and $0.031s$, respectively.

B. Outdoor Replanning Performance

As shown in Fig. 10, we demonstrate the outdoor experiments. For Fig. 10 (a), the trajectory length is $19.6m$ and total execution time is $41.3s$. The average velocity of the quadrotor is $0.49m/s$. The maximum acceleration of the trajectory is $1.06m/s^2$ and the whole trajectory is dynamically feasible. There are totally 35 calls of RBK search (passive mode) with average computation time $0.025s$. The average computation time of elastic tube expansion and elastic optimization is $0.001s$ and $0.030s$, respectively. For Fig. 10 (b), the performance is similar and the detailed statistics are shown in the video attachment.

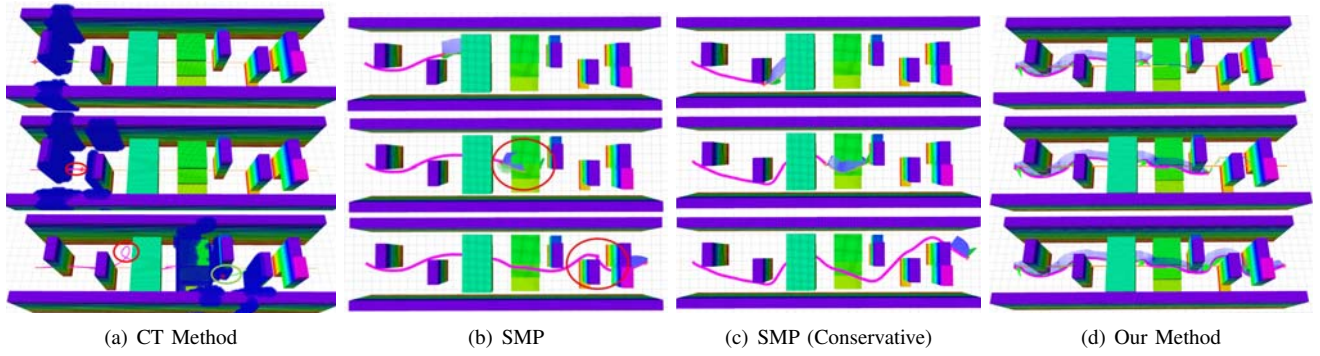|			
(a) CT Method	(b) SMP	(c) SMP (Conservative)	(d) Our Method

Fig. 8: For the CT method in (a), ESDF is shown in blue. The red circle in (a) marks the part that is either infeasible or jerky due to the local minimum issue. For SMP with a full dynamic bound in (b), the primitive duration is set to $1s$ and level of discretization is set to 3. The red circle marks the part which does not satisfy the real-time requirement (10Hz). For SMP (Conservative) in (c), the maximum velocity is set to $2m/s$, the maximum acceleration is set $2m/s^2$, and the level of discretization is set to 1. For our method in (d), we use the default parameters in Sec. VI.
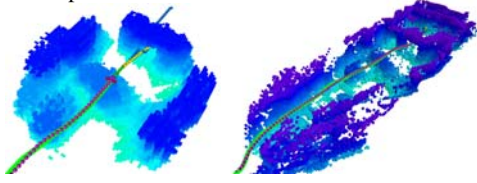

Fig. 9: In (a), only local map is available for replanning. The control points found by RBK (in blue), executed control points (in pink), committed control points (in green), and optimizing control points (in yellow) are marked. The whole trajectory is shown in (b) and the trajectory following result is shown in green.
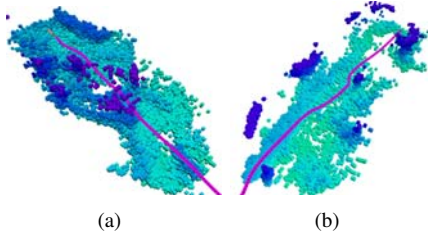

(a)	(b)

Fig. 10: Illustration of the outdoor experiments. (a) shows a flight through a pavilion (part of the map on the top is cut for visualization purpose). (b) shows the flight avoiding trees.

VIII. CONCLUSION AND FUTURE WORK

We focus on the replanning scenario for quadrotors. We present the RBK search method, which transforms the position-only search (such as A* and Dijkstra) into a kinodynamic search, by exploring the properties of B-spline. The RBK method facilitates the quadrotor's non-static initial state and produces dynamically feasible time-parameterized trajectory instead of unparameterized path. We present an EO approach as the post-optimization process to refine the trajectory, while the safety and dynamical feasibility are guaranteed. The dynamical feasibility is guaranteed for both the front-end and back-end to ensure the robustness of the optimization process. We design a receding horizon replanner using sliding window optimization strategy and local control property of B-spline. System comparisons and onboard experiments are provided to validate the superior performance of our replanning system. Extensions to this work may include developing informative heuristics for the RBK search to enhance the performance.

REFERENCES

[1] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Intl. J. Robot. Research*. Springer, 2016, pp. 649–666.

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

[3] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, 2017.

[4] F. Gao, Y. Lin, and S. Shen, "Gradient-based online quadrotor safe trajectory planning in 3d complex environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2017.

[5] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[6] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2013, pp. 5054–5061.

[7] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," Ph.D. dissertation, Stanford University, 2016.

[8] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2017.

[9] M. Elbanhawi, M. Simic, and R. N. Jazar, "Continuous path smoothing for car-like robots using b-spline curves," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 23–56, 2015.

[10] E. Koyuncu and G. Inalhan, "A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2008, pp. 815–821.

[11] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K. ichiro Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Computer-Aided Design*, vol. 42, 2010.

[12] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and 3d circular buffer," *arXiv preprint arXiv:1703.01416*, 2017.

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

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

[15] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," http://uav.ust.hk/icra2018wenchao.

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

[17] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2015, pp. 42–49.