STD-Trees: Spatio-temporal Deformable Trees for Multirotors Kinodynamic Planning

Hongkai Ye, Chao Xu, and Fei Gao

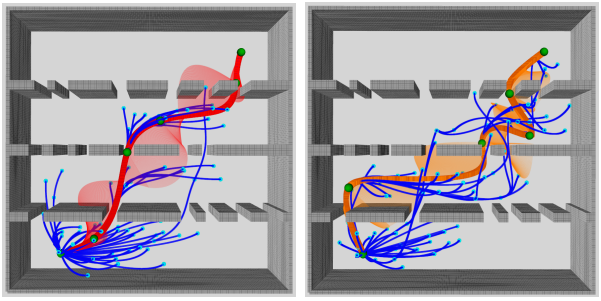
***Abstract*— In constrained solution spaces with a huge number of homotopy classes, stand-alone sampling-based kinodynamic planners suffer low efficiency in convergence. Local optimization is integrated to alleviate this problem.**

In this paper, we propose to thrive the trajectory tree growing by optimizing the tree in the forms of deformation units, and each unit contains one tree node and all the edges connecting it. The deforming proceeds both spatially and temporally by optimizing the node state and edge time durations efficiently. Deforming the unit only changes the tree locally yet improves the overall quality of a corresponding sub-tree. Further, to consider the computation burden and optimizing level, patterns to deform different tree parts in combination of different deformation units are studied and compared, all showing much faster convergence. The proposed deformation can be easily integrated into different RRT-based kinodynamic planning methods, and numerical experiments show that integrating the spatio-temporal deformation greatly accelerates the convergence and outperforms the spatial-only deformation. Compared with some hierarchical planners, the proposed one suits more in finding a better homotopy class.

I. Introduction

Minimum time and control trajectory generation [1] has long been developed for multirotors planning which considers the trajectory generation as an optimization problem under constraints. When collision avoidance is additionally regarded as a part of the constraints, many works [2]–[6] first consider the shortest path planning problem for a collision-free geometric path, and then generate kinodynamically-feasible trajectories in the free space nearby. In this way, locally optimal results can be obtained in a homotopy class solely conditioned by the prior geometric path. The optimal trajectory, however, may be excluded outside that specific homotopy class. Since a shorter path does not necessarily conclude a lower cost trajectory, the lack of objective coherence in the hierarchical planning process harms globally reasoning for an optimal trajectory.

This paper proposes an approach for global trajectory optimization and tries to bring a near-optimal collision-free trajectory. Unlike the hierarchical way, our approach breaks down the problem into smaller subproblems of the same objective and we respect directly the constraints in the process of globally exploring the entire constrained solution space. A trajectory tree is grown by connecting discretized state samples to search into spaces of different local minima and determine the best homotopy class on the fly. It is well known though that stand-alone sampling-based variants [7]–[9] converge slowly since the probability of sampling states exactly near the optimal solution is extremely low. Sufficient sampling helps organizing the trajectory tree however costs too much time. To improve, each sample is expected to make its best contribution to probe the solution space at a low cost. In addition for trajectory planning involving kinematics and system dynamics, we conceive the extra time dimension critical. As a result, we propose to deform the trajectory tree right after a sample node is added, in both space and time, that is, spatio-temporally (ST) to keep the tree better-organized, as shown in Fig. 1. The difficulties though, appear in designing practical objectives while retaining efficiency. Deforming the whole tree can be extravagant. We introduce the notion of a deformation unit as the collection of one node and the edges connected to it. One deformation unit only constitutes a very small part of the entire tree, yet deforming it improves the overall quality of the sub-tree rooted at the deformed node. In this way, we make the best use of each sample only at a low cost and thus facilitate the convergence.



(a) With ST deformation. (b) Without deformation.

Fig. 1: Current tree and best trajectory after adding a same number of nodes. Planning with deformation acquires a better-organized tree structure with overall higher quality and gets a smoother trajectory.

We summarize the main contributions as follows:

1) A novel sampling-based multirotors kinodynamic planner for global trajectory optimization. It thrives the trajectory tree growing by deforming some tree parts in not only shapes but also the time dimension.
2) The concept of deformation unit which contains only one tree node and the edges connecting it. Deforming in units can be easily integrated into different RRT-based kinodynamic planners, and integrating it shows a faster convergence and generates much better final trajectories compared with no deformation and space-only deformation.

All authors are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China and with Huzhou Institute of Zhejiang University, Huzhou 313000, China.

Corresponding author: Gao Fei

Email: (hkye, cxu, and fgaoaa)@zju.edu.cn

Code: https://github.com/ZJU-FAST-Lab/std-trees

3) Several patterns to deform the tree in different combinations of deformation units to balance the optimization level and the computation burden. Numerical results show that all the patterns converge quicker than the baseline and overtake some search-based planners.
4) Bbenchmarks with some hierarchical planning methods. The proposed method shows superiority in finding a better homotopy class for lower cost trajectories with less planning time.

II. Related Work

A. Multirotors Planning

Recently, many works adopt a hierarchical planning framework for multirotors trajectory generation. They first search for a collision-free geometric path, based on which Bry et al. [2] constrain the segment boundaries of the optimized trajectory in the path waypoints. If collide, additional waypoints are added and the optimization problem is reformulated and solved repeatedly. Many others [3]–[6] further extract free spaces, namely safe flight corridors (SFC) along the path, which consumes extra time, and then optimize a trajectory strictly inside the SFC. A SFC is usually represented by simple polyhedrons or spheres for efficiency. It however almost definitely leaves much free spaces uncovered, which may exclude the feasible spaces for the best trajectory. Our direct sampling requires no extraction of free spaces, and thus theoretically any potential free space is accessible. If a near-optimal solution is preferred, one needs to optimize a trajectory for each path in different homotopy classes and compare each, since a shorter path do not necessarily conclude a lower cost trajectory. Unlikely, our method enjoys the asymptotic optimality which enables the best homotopy class being determined during the searching process. Proof for the asymptotic optimality is not provided but rather addressed with numerical experiment.

B. Sampling-based Planning

Sampling-based geometric shortest path planning is studied for long. Basically, different algorithms differ in two procedures, the exploration of solution space and the exploitation of information brought by new samples, resulting in different convergence. The original RRT algorithm [10] simply connects a state node to the nearest node in the current tree and is very fast in exploration to get solutions. It is, however, proved to converge to a sub-optimal solution by Karaman and Frazzoli in [11] where the RRT* algorithms are introduced. They stand out in the process of choosing parent nodes and rewire local tree structure according to the best *cost-from-start* values and can converge to the optimal with probability one. The RRT$^{\#}$ [12] algorithm resolves the under-exploitation in RRT* and improves convergence by further rewiring in cascade to propagate the new information to other parts of the tree and maintain a well-grown tree of consistent nodes. To acquire higher convergence rate, recent works focus on improving sampling strategies by sampling directly in the Informed Set [13], the Relevant Region Set [14], or the Local Subset of GuILD [15]. However, these designs depend on the $L2$-Norm objective. For applications with varied desired objectives, analytic forms of the sampling set can not be obtained.

C. Integrating Optimization

Combining local optimization to boost convergence in the global searching process is studied by many for geometric shortest path planning. RABIT* [16] adopts CHOMP [17] to find feasible alternative connections for edges in collision and thus accelerate the search. However, the convergence rate highly depends on shrinking the $L2$-Norm based informed set in their BIT* [18]-like batch sampling and searching process. The rather heavy optimization can also hinder exploring different homotopy classes. DRRT [19] does not require any informed sampling set. It minimizes the overall tree cost by optimizing the locations of some existed nodes in the tree. A well-grown tree is maintained to cover the search space in each iteration, and thus paths obtained are of high quality and the convergence is improved. For kinodynamic planning, INSAT [20] combines trajectory optimization into discretized graph search. However, the solving time is still intractable because of the high-dimensional graph search, and long trajectories are being optimized many times in each node expanding step. Although weighted A* [21]–[23] can be used to accelerate searching, it eventually generates a sub-optimal solution since the admissibility criterion is relaxed. In our combining scheme, the sampling-based search globally reasons for the optimal solution asymptotically, and the optimization only deforms limited parts of the tree, retaining efficiency.

III. Problem Statement

The problem we address is to efficiently find a near-optimal trajectory for multirotors in complex environments. From a high level, it can be formulated as an optimization problem. Following [8, 9, 23, 24], the cost minimized is a trade-off between time and energy, which suits many applications. The constraints imposed include obstacle avoidance, system dynamics, kinematics and start and goal constraints. According to the multirotor systems' differential flatness property [1], we can use a linear model of chain integrator to represent its dynamics with four flat outputs p_x, p_y, p_z (position in each axis), ψ (yaw), and their derivatives being the state variables. Thus, the trajectory planning problem is formulated as a Linear Quadratic Minimum Time (LQMT) problem [8] in the area of optimal control. For each axis (yaw is excluded in this paper), it is formulated as follows:

$$\min_{u(t)} \mathcal{J} = \int_0^\tau (\rho + \frac{1}{2}u(t)^2)dt \tag{1a}$$

$$s.t. \quad \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) - \dot{\mathbf{x}}(t) = \mathbf{0}, \tag{1b}$$

$$\mathbf{x}(0) = \mathbf{x}_{start},\ \mathbf{x}(\tau) = \mathbf{x}_{goal}, \tag{1c}$$

$$\mathcal{G}(\mathbf{x}(t), u(t)) \preceq \mathbf{0},\ \forall t \in [0, \tau], \tag{1d}$$

where τ is the time duration of the trajectory, ρ the trade-off weight to penalize time against energy, $\mathbf{x}_{start}$ the initial state, $\mathbf{x}_{goal}$ the goal state, and $\mathcal{G} \preceq \mathbf{0}$ the obstacle avoidance and higher derivative limit constraints.

Algorithm 1 Spatio-temporal Deformable Trees

1: **Notation**: Tree $\mathcal{T}$, State $\mathbf{x}$, Deformation Units $\mathcal{U}$, Environment $\mathcal{E}$, Deform Type $\mathcal{L} \in \{$NODE, TRUNK, BRANCH, TREE$\}$
2: **Initialize**: $\mathcal{T} \leftarrow \emptyset \cup \{\mathbf{x}_{start}\}$
3: **while** Termination condition not met **do**
4: $\quad \mathbf{x}_{new} \leftarrow$ **Sampling**($\mathcal{E}$)
5: $\quad \mathcal{X}_{backward} \leftarrow$ **BackwardNear**($\mathcal{T}$, $\mathbf{x}_{new}$)
6: $\quad \mathbf{x}_n \leftarrow$ **ChooseParent**($\mathcal{X}_{backward}$, $\mathbf{x}_{new}$)
7: $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{x}_n, \mathbf{x}_{new}\}$
8: $\quad$ **if TryConnectGoal**($\mathbf{x}_{new}$, $\mathbf{x}_{goal}$) **then**
9: $\quad\quad$ One Solution Found.
10: $\quad$ **end if**
11: $\quad$ **if** A First Solution Is Found **then**
12: $\quad\quad \mathcal{U} \leftarrow$ **SelectDeformationUnits**($\mathbf{x}_n$, $\mathcal{L}$)
13: $\quad\quad$ **DeformInOrder**($\mathcal{U}$)
14: $\quad$ **end if**
15: $\quad$ **RewireInCascade**($\mathcal{T}$, $\mathbf{x}_{new}$)
16: **end while**
17: **return** $\mathcal{T}$

IV. Spatio-Temporal Deformable Trees

The nonlinear constraints especially the homotopy-rich environments make the entire solution space highly non-convex. Directly applying nonlinear programming will almost definitely fall into local minima. We thus propose a sampling-based approach as described in Alg. 1 to avoid the limits. Basically, the algorithm is based on kinodynamic RRT* [8] with features of RRT$^{\#}$ [12] and with an extra step to deform the shape and the time duration of the current trajectory tree (line 12, 13) in each iteration. The searching framework breaks down the problem into smaller subproblems by incrementally adding state samples to probe the solution space. Each subproblem is of the same form as Equ.1 with different boundary states, and we get a trajectory segment (a tree edge) by solving one.

A. Tree Growing

In each iteration, after a state node is sampled, we seek its best parent node in the BackwardNear node set $\mathcal{X}_{backward}$ according to the *cost-from-start* values estimated by solving the LQMT subproblems. If a qualified best parent is found, the sampled node and a new edge are added to the tree. The node then tries to connect to the goal node for a possible solution. After that, the core part, the tree deformation is performed by optimizing some trajectory edges, both in profile and in time duration. This step reduces the overall tree cost without adding more nodes. Since spending more time on exploitation means less exploration, the deformation is only activated after finding a first feasible solution. Finally, since the structure of the implicit graph is changed while adding nodes, we rewire some nodes for potential improvement of their *cost-from-start* values. The rewire performs in cascade to propagate the new information brought by the new node to other parts of the tree. For detailed information of some functions (line 4, 5, 6) in Alg. 1, we recommend the readers to [8, 9] and we focus mainly on the tree deformation part in this paper.

The final trajectory tree consists of many trajectory segments as its edges, and the optimal trajectory between start and goal consists of a chain of trajectory segments between a series of successive states.

B. Tree Edge Representation

As seen, solving the LQMT subproblems makes up a fundamental part of the searching process and will be called many times in each iteration. Therefore, it needs to be solved as fast as possible. Suppose the integrator model is of s^{th}-order, which leads the state variable to be $\mathbf{x}(t) = [p(t), \cdots, p^{(s-1)}(t)]^\mathsf{T}$ and the control to be $u(t) = p^{(s)}(t)$. According to [8, 25], if all the inequality constraints (Equ. 1d) are temporally disregarded, the optimal control law for the unconstrained LQMT problems are parameterized by polynomials of degree $s-1$, and then each state variable in $\mathbf{x}(t)$ can be obtained by integration up to s times. Thus, we have $p(t) = \mathbf{c}^\mathsf{T}\beta(t), t \in [0, T]$, where $\mathbf{c} \in \mathbb{R}^{2s}$ is the polynomial coefficient vector, T the time duration, and $\beta(t) = (1, t, t^2, \cdots, t^{2s-1})^\mathsf{T}$ the natural basis.

This leads to a conclusion that given time durations T and boundary conditions $\mathbf{d} = [\mathbf{x}^\mathsf{T}(t)|_{t=0}, \mathbf{x}^\mathsf{T}(t)|_{t=T}]^\mathsf{T} \in \mathbb{R}^{2s}$, the optimal solutions for these specific LQMT problems are fully determined since no freedom is left. The boundary conditions can be sampled directly and the initial T is calculated according to [8].

As claimed by our previous work [26], a smooth bijection exists between the polynomial coefficient vector $\mathbf{c}$ and the boundary condition vector $\mathbf{d}$, that is, an analytic transformation exist between the two kinds of descriptions of polynomials $\{\mathbf{c}, T\}$ and $\{\mathbf{d}, T\}$:

$$\mathbf{d} = \mathbf{A}_f(T)\mathbf{c}, \ \mathbf{c} = \mathbf{A}_b(T)\mathbf{d}, \tag{2}$$

where $\mathbf{A}_f(T)$ and $\mathbf{A}_b(T)$ are forward and backward mapping matrices whose entries are analytically determined. The exact element value calculation can be found in [26].

The cost of one edge can then be calculated by $\{\mathbf{c}, T\}$ as

$$\begin{aligned} J_s(\mathbf{c}, T) &= \rho T + \int_0^T \frac{1}{2}\mathbf{c}^\mathsf{T}\beta^{(s)}(t)\beta^{(s)}(t)^\mathsf{T}\mathbf{c} \ dt \\ &= \rho T + \frac{1}{2}\mathbf{c}^\mathsf{T}\mathbf{Q}(T)\mathbf{c}, \end{aligned} \tag{3}$$

or by $\{\mathbf{d}, T\}$ as

$$\begin{aligned} J_s(\mathbf{d}, T) &= J_s(\mathbf{x}(t)|_{t=0}, \ \mathbf{x}(t)|_{t=T}, \ T) \\ &= \rho T + \frac{1}{2}\mathbf{d}^\mathsf{T}\mathbf{M}(T)\mathbf{d}, \\ \mathbf{d} = \begin{bmatrix} \mathbf{x}(t)|_{t=0} \\ \mathbf{x}(t)|_{t=T} \end{bmatrix}&, \ \mathbf{M}(T) = \mathbf{A}_b^\mathsf{T}(T)\mathbf{Q}(T)\mathbf{A}_b(T), \end{aligned} \tag{4}$$

both analytically. The previously ignored constraints (1d) are checked afterwards. If satisfied, a tree edge is grown, and the cost provides an estimation of the optimal transition cost between two state samples in the constrained solution space.

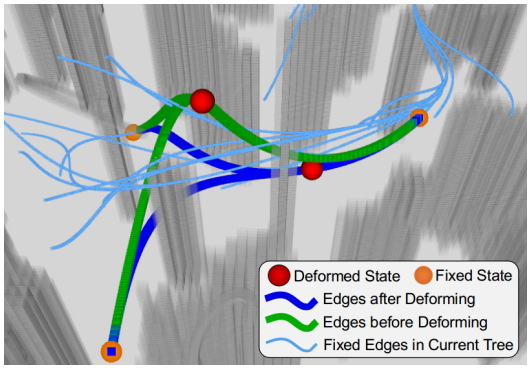


Fig. 2: The deforming of one deformation unit. In this case, the deformed node has two child nodes, which are fixed together with the parent node. Only the time durations of the deformed edges and the state of the deformed node are optimized.

The two kinds of representation of trajectory edges support the following tree deformation process.

C. Tree Deformation

When a new node is added to the tree in the tree growing process, some new information of the solution space is collected in the tree. However, the *cost-from-start* value of some nodes may not well estimate the actual value due to insufficient sampling, presenting that the tree does not grow well, especially in the time dimension. A deforming of the edges can improve its quality.

Given current tree structure $\mathcal{T}$, each node $n \in \mathcal{T}$ contains the following information:

- $\mathbf{x}_n$: the corresponding state of n,
- p_n: the parent node of n in $\mathcal{T}$,
- T_n: the time duration of the edge from p_n to n,
- $\mathbf{c}_n$: the coefficient vector of the edge from p_n to n,
- $\mathcal{C}_n$: the direct children node set of n in $\mathcal{T}$,
- g_n: the *cost-from-start* value of n following $\mathcal{T}$.

We first introduce the deformation unit as depicted in Fig. 2.

1) Deformation Unit: If we reposition one of the nodes and fix all others, meanwhile keeping the connection between nodes unchanged, then only the edges that connect to the repositioned node will be affected. Thus, we set a deformation unit by the collection of one node, the edge from its parent, and the edges to its children. Within a unit, the deformation is performed by optimizing the node state and the time duration of these edges, while all other nodes' state and edge time durations are fixed. We denote the element collection in a deformation unit as

$$\{\mathbf{x}_n, \mathbf{T}_n\} = \{\mathbf{x}_n, T_n, T_i, |i \in \mathcal{C}_n\}, \tag{5}$$

where n is the node being deformed, and $\mathbf{T}_n = \{T_n, T_i, |i \in \mathcal{C}_n\}$ contains the time durations of all the edges connecting to n. Note that by setting the node state as decision variables instead of edge coefficients, the equality constraints imposed by the continuity requirement in-between tree nodes are implicitly eliminated, and the number of decision variables is reduced.

2) Objective Design: When optimizing the deformation unit, we need an appropriate objective. It should follow that after the deformation, the overall tree quality is improved and it stands a better chance to find a better solution or even the optimal solution. As revealed by Hauer et al. [19], the sum of the *cost-from-start* values of a fixed number of sample nodes stands for an estimation of the optimal value function for the solution space covered by the selected sample set. Considering also that optimizing the deformation unit will and only will change the *cost-from-start* values of node n and all its descendant nodes, leaving all other nodes unaffected. We therefore set the objective as

$$g_n + \sum_{i \in \mathcal{D}_n} g_i, \tag{6}$$

where $\mathcal{D}_n$ is the set of all n's descendant nodes. Optionally, we can assign a weight $w \in [0, 1]$ for each node suggesting the possibility of it constituting the final best trajectory. A heuristic estimation of the *cost-to-go* can be an alternative for this weight. The objective suggests an optimization of the sub-tree rooted at node n.

With the tree growing, we have

$$g_n = c(\mathbf{x}_{p_n}, \mathbf{x}_n) + g_{p_n}, \tag{7}$$

where $c(\mathbf{x}_i, \mathbf{x}_j)$ is the edge cost connecting node i to node j, analytically estimated by Equ. 4. We can then use a weighted sum of the cost of each tree edge to calculate the objective. The weight of an edge (i, j) is assigned by counting the total number of paths that start from the start node to any other node in the sub-tree using this edge.

The objective thus becomes

$$\begin{aligned} g_n + \sum_{i \in \mathcal{D}_n} g_i &= \sum_{i \in \mathcal{T}_n} \sum_{j \in \mathcal{C}_i} d_j c(\mathbf{x}_i, \mathbf{x}_j) \\ &= d_n c(\mathbf{x}_{p_n}, \mathbf{x}_n) + \sum_{i \in \mathcal{C}_n} d_i c(\mathbf{x}_n, \mathbf{x}_i) + \mathbf{C} \\ &= \sum_{i \in \{n\} \bigcup \mathcal{C}_n} d_i c(\mathbf{x}_{p_i}, \mathbf{x}_i) + \mathbf{C}, \end{aligned} \tag{8}$$

where $\mathcal{T}_n$ is the sub-tree rooted at node n, d_n equals $1 + nb\_des(n)$ with $nb\_des(n)$ the number of descendants of node n, and $\mathbf{C}$ a constant indicating the sum of all other edges' cost in the tree, which is irrelevant to the elements in the deformation unit. Therefore, the objective depends only on the edges in the deformation unit yet expresses the quality of a sub-tree that estimates the value function of a part of the solution space.

3) Unconstrained Formulation: In the process of tree growing, we ignore the constraints and check afterwards to achieve faster exploration of the entire solution space. When deforming, it is wished to focus more on the local space covered by the deformation unit and exploit the information already gathered in the tree. The constraints on obstacle avoidance and dynamical feasibility (Equ. 1d) should now be directly faced.

Denote $\mathcal{G}(p^{[s]}(t)) \in \mathbb{R}^{s+1} \preceq \mathbf{0}$ the functional-type constraints that consider obstacle avoidance and dynamical

limitations such as the velocity, acceleration and other higher order derivative constraints up to order s. To eliminate these inequality constraints, we construct penalty functions and turn them into soft ones. Since they consist of an infinite number of inequality constraints and can hardly be directly handled, we transform them into finite-dimensional ones via integral of constraint violations, and the integral is estimated by weighted sum of the sampled penalty functions. Note that in practice, these constraints are decoupled between edges, that is, $\mathcal{G}(p^{[s]}(t))$ with $t \in [0, T_i]$ are solely determined by $\mathbf{c}_i$ and T_i. Thus, for one trajectory edge parameterized by $\{\mathbf{c}_i, T_i\}$, we compute the penalty function as

$$J_f(\mathbf{c}_i, T_i, k_i) = \frac{T_i}{k_i} \sum_{j=0}^{k_i} \omega_j \mathcal{X}^\mathsf{T} max[\mathcal{G}(\mathbf{c}_i, T_i, t),\ \mathbf{0}], \quad (9)$$

where k_i is the sample number on this edge, $\mathcal{X} \in \mathbb{R}^{s+1}_{\geq 0}$ is a vector of penalty weights for each entry of $\mathcal{G}$, $(\omega_0, \omega_1, \cdots, \omega_{k_i-1}, \omega_{k_i}) = (1/2, 1, \cdots, 1, 1/2)$ are the quadrature coefficients following the trapezoidal rule, $max[\,]$ is an element-wise comparison, and $t = j/k_i \cdot T_i$ is the sampled time stamp. Integrating the penalty into the objective by each edge and the unconstrained optimization of one deformation unit is formulated as

$$\begin{aligned} &\min_{\mathbf{x}_n, \mathbf{T}_n} \sum_{i \in \{n\} \bigcup \mathcal{C}_n} d_i (J_s(\mathbf{c}_i, T_i) + J_f(\mathbf{c}_i, T_i, k_i)), \\ &\mathbf{c}_i = \begin{cases} \mathbf{A}_b(T_i) \left[\mathbf{x}_{p_n}^\mathsf{T},\ \mathbf{x}_n^\mathsf{T}\right]^\mathsf{T}, & i = n \\ \mathbf{A}_b(T_i) \left[\mathbf{x}_n^\mathsf{T},\ \mathbf{x}_i^\mathsf{T}\right]^\mathsf{T}, & i \in \mathcal{C}_n \end{cases} \end{aligned} \quad (10)$$

with $\{\mathbf{x}_n, \mathbf{T}_n\}$ being the decision variables.

4) Spatio-temporal Optimization: For one edge $\{\mathbf{c}_i, T_i\}$, $i \in \{n\} \bigcup \mathcal{C}_n$ in the deformation unit, we derive the gradient of the decoupled objective w.r.t $\{\mathbf{x}_n, \mathbf{T}_n\}$ by chain rule as follows:

$$\frac{\partial J_s}{\partial \mathbf{x}_n} = \frac{\partial J_s}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{x}_n} = \mathbf{Q}(T)\mathbf{c}_i, \frac{\partial J_f}{\partial \mathbf{x}_n} = \frac{\partial J_f}{\partial \mathcal{G}} \frac{\partial \mathcal{G}}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{x}_n}, \quad (11)$$

$$\frac{\partial J_s}{\partial T_i} = \rho + \frac{1}{2} \mathbf{c}_i^\mathsf{T} \dot{\mathbf{Q}}(T_i) \mathbf{c}_i, \frac{\partial J_f}{\partial T_i} = \frac{J_f}{T_i} + \frac{\partial J_f}{\partial \mathcal{G}} \frac{\partial \mathcal{G}}{\partial t} \frac{j}{k_i}, \quad (12)$$

$$\frac{\partial J_f}{\partial \mathcal{G}} = \frac{T_i}{k_i} \sum_{j=0}^{k_i} \omega_j \mathcal{X} \odot max[Sign[\mathcal{G}(\mathbf{c}_i, T_i, \frac{j}{k_i})],\ \mathbf{0}], \quad (13)$$

$$\frac{\partial \mathbf{c}_i}{\partial \mathbf{x}_n} = \begin{cases} \left[\mathbf{A}_b^{01}(T_i)^\mathsf{T}\ \mathbf{A}_b^{11}(T_i)^\mathsf{T}\right]^\mathsf{T}, & i = n \\ \left[\mathbf{A}_b^{00}(T_i)^\mathsf{T}\ \mathbf{A}_b^{10}(T_i)^\mathsf{T}\right]^\mathsf{T}, & i \in \mathcal{C}_n, \end{cases} \quad (14)$$

where $\odot$ is the Hadamard product, $Sign[\cdot]$ is element-wise indication of the sign of a number, and $\mathbf{A}_b^{mn}(T)$ contains block elements of $\mathbf{A}_b(T)$ partitioned by s. The gradients of an edge w.r.t. the time durations of other edges are all 0. The conditional computation implies different cases that $\mathbf{x}_n$ being tail or head state of an edge.

The only parts undefined in the above equations are $\partial \mathcal{G}/\partial \mathbf{c}_i$ and $\partial \mathcal{G}/\partial t$. For obstacle avoidance, we wish the edge positions have certain clearance and thus define

$$\mathcal{G}_o = r - \mathcal{F}(p_i(t)), \quad (15)$$

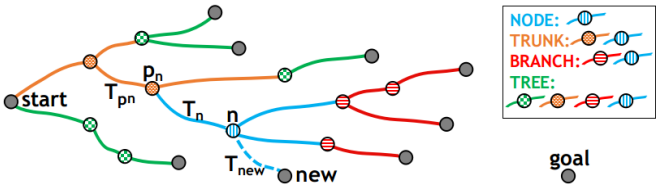


Fig. 3: Deformation patterns depiction. The start node and all the leaf nodes stay fixed.

where $r > 0$ is the preferred distance away from obstacles, and $\mathcal{F}(p(t))$ computes a minimum distance to the closest obstacle given a position in the edge, which can be pre-computed with a distance field built incrementally [27, 28] or in batch [29]. For dynamical feasibility, we limit the amplitude of higher-order derivatives and define

$$\mathcal{G}_k = p_i^{(k)}(t)^2 - m_k^2, \quad k \in \{1, 2, \cdots, s\} \quad (16)$$

where m_k is the maximum allowed value of the k^{th} derivative. The corresponding gradients are

$$\begin{aligned} &\frac{\partial \mathcal{G}_k}{\partial \mathbf{c}_i} = 2\beta^{(k)}(t) p_i^{(k)}(t)^\mathsf{T}, \quad \frac{\partial \mathcal{G}_k}{\partial t} = 2\beta^{(k+1)}(t)^\mathsf{T} \mathbf{c}_i p_i^{(k)}(t), \\ &\frac{\partial \mathcal{G}_o}{\partial \mathbf{c}_i} = -\beta(t) \frac{\partial \mathcal{F}}{\partial p_i(t)}, \quad \frac{\partial \mathcal{G}_o}{\partial t} = -\dot{\beta}(t)^\mathsf{T} \mathbf{c}_i \frac{\partial \mathcal{F}}{\partial p_i(t)}, \end{aligned} \quad (17)$$

where $\partial \mathcal{F}/\partial p(t)$ is computed with interpolation of the distance field. The analytic transformation between $\{\mathbf{c}_i, T_i\}$ and $\{\mathbf{x}_{p_i}, \mathbf{x}_i, T_i\}$ provides efficiency for calculating the objective and the gradients since no matrix inversion is required. With the gradients at hand, numerical optimization is applied then to solve this optimization problem. Considering that the interpolation of the distance fields $\mathcal{F}$ introduces non-smoothness, Newton-type methods like BFGS can be inaccurate or fail sometimes. We thus adopt bundle methods to address it. In this work, the Limited Memory Bundle Method (LMBM) [30] is used.

5) Deformation Patterns: The optimization of one deformation unit can explore its surrounding local solution space more thoroughly while other parts of the tree are left unchanged. In each iteration of Alg. 1, it is noticed that the current tree topology can be changed by adding new nodes and by *RewireInCascade* (line 15). Applying deformation to more tree parts can improve the overall tree quality but may at the cost of more computation loads. Weighing the optimization levels and the computation burden, we propose four kinds of patterns to deform different parts of the tree in different combinations of deformation units. As depicted in Fig. 3, after one node new is newly added to the tree, denoting its parent node in the tree by n, the tree deformation is performed in one of the following patterns.

1) **NODE**: optimizes only one deformation unit which contains n and the edges connecting to it.
2) **TRUNK**: optimizes several deformation units, and the units are selected by following parent pointers from n up to a direct child of $start$. The deformation is then

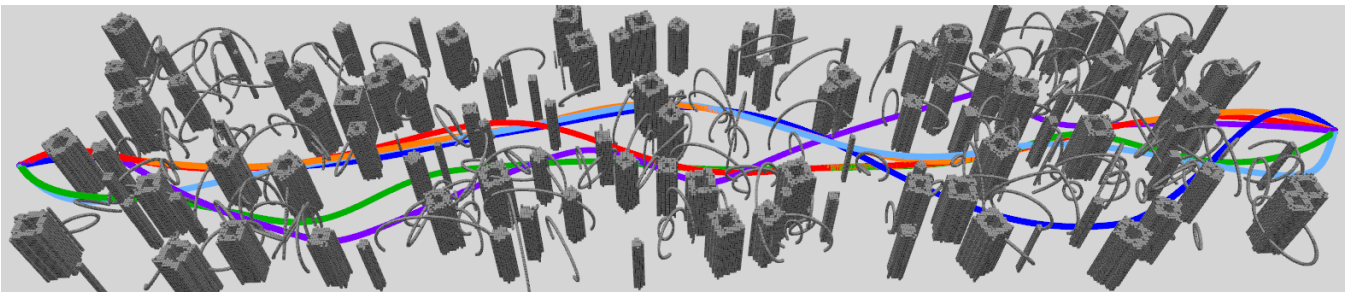


Fig. 4: Environment for the pattern comparison and final trajectories of one trial. The color indications are light blue for NODE, orange for TRUNK, red for BRANCH, green for TREE, purple for w/o, and dark blue for search-based with a weight 1.7. The proposed patterns generate smoother trajectories with lower costs, especially pattern BRANCH (red).

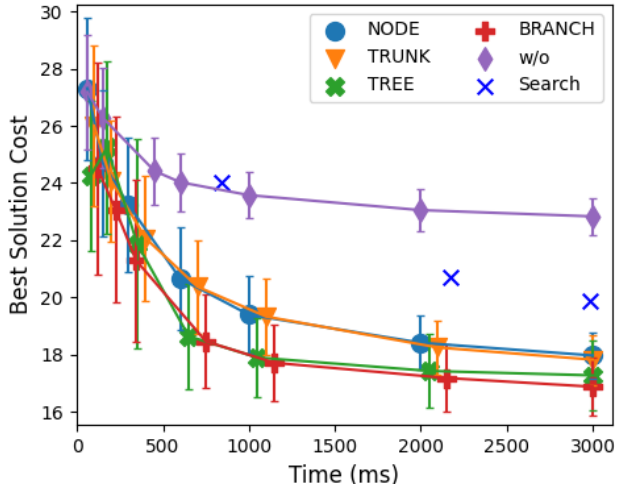


Fig. 5: Convergence comparisons of deformation patterns and search-based method [23]. Short lines indicate standard deviations. No heuristic amplification costs tens of seconds (not shown).

performed in an order from the child node to n. This is the strategy DRRT [19] adopts.

3) **BRANCH**: optimizes every nodes and edges in the sub-tree rooted at n. The order follows breadth-first search starting from n. All the leaf nodes are excluded.
4) **TREE**: optimizes every node and edge in the entire tree except the start node and all the leaf nodes. The order follows breadth-first search starting from the direct children nodes of $start$.

Effectiveness and efficiency of deforming in each pattern will be studied in Sec. V-B.

V. Numerical Results

A. Experiment Settings

For numerical comparisons, we set $s = 3$, which means a third-order integrator is used to model our multirotor system. The weight of time ρ is set 100. The dynamical limitations are set as $5m/s$ for velocity, $7m/s^2$ for acceleration, and $15m/s^3$ for jerk. For collision checking, the maps are pre-built and transformed into occupancy grids of $0.1m$ resolution with obstacles inflated by $0.2m$. Exp. V-B is conducted with a desktop computer of a 3.4GHz Intel i7-6700 processor while Exp. V-B and V-D are conducted with a computer of a 2.6GHz Intel i7-10750H processor.

B. Pattern Comparison

In this experiment, the proposed planning method w/o deformation or with one of the four deformation patterns introduced in Sec. IV-C.5 are benchmarked with each other and with search-based kinodynamic planners. For search-based ones, the weighted version of A* [22] is used to accelerate the search. Three different weights are used to amplify the heuristic: $1.7, 2.3$, and 2.8. Each variant runs for 100 trials with a 3 second time budget. The environment and final trajectories of one trial are shown in Fig. 4. As Fig. 5 shows, the proposed methods find the first solution within milliseconds and then quickly converge. Compared to the baseline (w/o), the convergence rate all improves remarkably by planning in any patterns of the proposed spatio-temporal deformation. Among the patterns, BRANCH slightly beats TREE and much outperforms NODE and TRUNK. We think it is because a new sample brings potential improvements mostly on the sub-tree while other tree parts are less likely influenced, and thus deforming just the sub-tree is a good balance on computation cost and tree quality improvement. The search-based ones with different weights, however, take much longer time to get a solution with even higher cost.

C. Deformation Comparison

The proposed deformation is compatible with different sampling-based kinodynamic planners, and we integrate it into three RRT-based methods, kRRT, kRRT*, and kRRT$^{\#}$. We compare the proposed spatio-temporal deformation with deforming only spatially and with no deformation, denoting a suffix of -ST, -S, and no suffix, respectively. For each deformation, pattern BRANCH is adopted. Each method runs for 100 trials with a 7 second time budget. The environment and final trajectories of one trial are shown in Fig. 6. Fig. 7 shows that for all the RRT-based methods, both deformations accelerate the convergence evidently. The statistics in Table. I show that the proposed spatio-temporal deformation further generates trajectories with less execution time and cost than the spatial-only deforming, showing the effectiveness to deform in the time dimension.

D. Comparison with Hierarchical Methods

To test the ability of searching for a better homotopy class (expressed as lower trajectory cost), we compare the

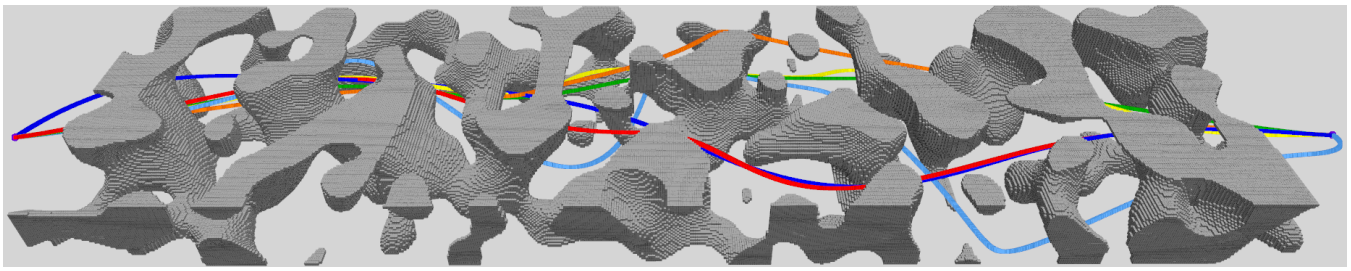


Fig. 6: Environment for the experiment in Sec. V-D and final trajectories of one trial. The color indications are light blue for kRRT, dark blue for kRRT-ST, yellow for kRRT*, green for kRRT*-ST, orange for kRRT$^{\#}$, and red for kRRT$^{\#}$-ST. Planning with ST deformation generates smoother trajectories with lower cost (red, green, and dark blue compared with orange, yellow, and light blue, respectively).

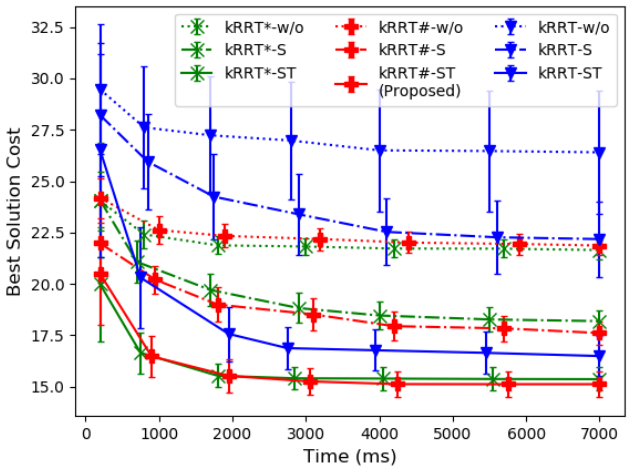


Fig. 7: Convergence of different searching schemes with the proposed spatio-temporal deformation (-ST) and with the spatial-only deforming (-S) compared with stand-alone sampling-based kinodynamic planners. Short lines indicate standard deviations.

proposed sampling-based planner with some hierarchical planners, Bry's [2] and Wang's [6]. All methods adopt the same time-energy objective and trajectories are all expressed as segment-wise polynomials. A prior path is needed by both compared methods. In [2], the path implicitly encodes nearby free space information. By fixing segment boundary positions at each path node, the optimized trajectory is restricted to be near the path (free space). Other free variables and each segment's time duration are optimized. In [6], the path further guides the generation of a SFC, and a trajectory is generated strictly inside the SFC with the segment boundary conditions and the time durations being optimization variables. To the authors' best knowledge, this [6] is the most efficient trajectory planner based on SFCs. We use RRT* for the geometric path planning and a prominent method described in [4] for the SFC generation. We take several RRT* path results of increasing planning time budget (2, 5, 10, 30, 50, and 100 ms, resulting in decreasing path length) as prior paths for Bry's and Wang's methods. Each result trajectory is then respectively compared with our method's results of increasing time budget (10, 50, 150, 200, 250, and 300 ms. Each comparison runs for 100 trials. The test environment and trajectory results of one trial are shown in Fig. 8. By looking at the cost trend in Fig. 9a, we find that for

TABLE I: Trajectory durations and cost of different deformation methods after the same computation time.

Method		Traj. Dura. (s) (Avg / Std)	Traj. Cost (Avg / Std)
kRRT	-w/o	21.65 / 2.39	26.40 / 2.98
	-S	21.42 / 1.83	22.18 / 1.84
	-ST	**14.49 / 0.74**	**16.49 / 0.99**
kRRT*	-w/o	18.12 / 0.34	21.65 / 0.44
	-S	17.47 / 0.63	18.18 / 0.53
	-ST	**13.75 / 0.45**	**15.37 / 0.57**
kRRT#	-w/o	18.37 / 0.49	21.87 / 0.47
	-S	16.94 / 0.73	17.61 / 0.58
	-ST	**13.47 / 0.56**	**15.12 / 0.64**

hierarchical planners, their cost increases sometimes even though the prior path length decreases, meaning a shorter path does not necessarily conclude a better homotopy class with a lower cost trajectory. By comparing the corresponding groups in Fig. 9a and Fig. 9b , we can see that our method plans the least cost trajectory with the littlest time usage.

VI. Conclusion

In this paper, we propose a sampling-based kinodynamic planning method for multirotors combining local optimization by deforming some selected tree edges spatially and temporally. The optimization is performed efficiently within deformation units which contain the state of one node and the time durations of the edges connecting it. Though only limited parts are deformed, the overall quality of a sub-tree is improved and a well-grown tree is maintained after each node is added to the tree. By deforming the state of some selected nodes and related edge durations without adding more nodes, the convergence is improved. Benchmark results show that integrating the proposed deformation achieves a much faster convergence rate. We open source our code for the reference of the community.

References

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

[2] A. Bry, C. Richter, A. Bachrach, and N. Roy, "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments," *Intl. J. Robot. Research (IJRR)*, vol. 34, no. 7, pp. 969–1002, 2015.

[3] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2019.

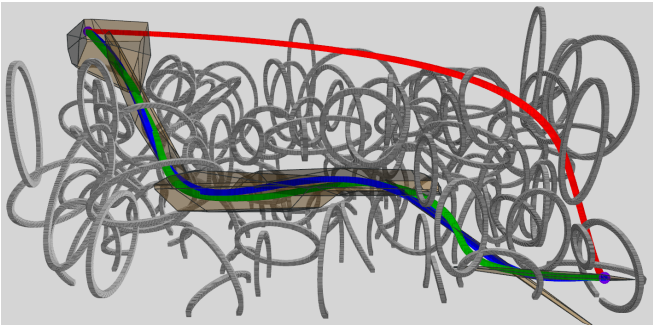
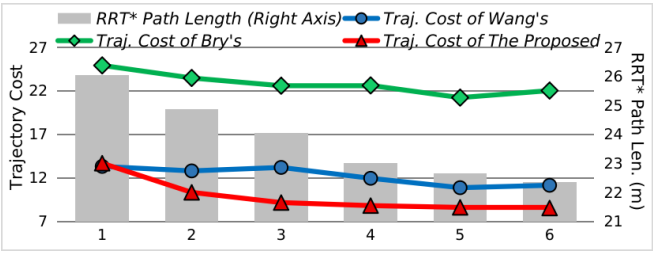
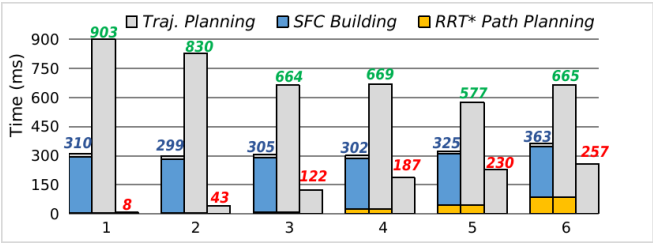


Fig. 8: Environment for Exp. V-D. Result trajectories are with 150 ms planning time budget of the proposed planner (red), and 10 ms for RRT*. Except from the planning time for RRT*, Wang's [6] (blue) needs extra time of more than 300 ms to build a SFC, and Bry's [2] (green) takes more than 500 ms to optimize.



(a) Average trajectory cost of Wang's and Bry's based on decreasing RRT path length. Their cost increases sometimes even though the prior path length decreases. Ours plan a trajectory of the lowest cost in each group.



(b) Average total planning time and each procedure's time usage in each method. In each group, the left bar stands for Wang's, the middle for Bry's, and the right for ours.

Fig. 9: The planning time and trajectory cost comparison based on different time budgets and prior path length. For the two plots, each group correlate with the one in the other plot.

[4] 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 (RA-L)*, pp. 1688–1695, 2017.

[5] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[6] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *arXiv preprint arXiv:2103.00190*, 2021.

[7] E. Schmerling, L. Janson, and M. Pavone, "Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2574–2581.

[8] 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.*, May 2013, pp. 5054–5061.

[9] H. Ye, X. Zhou, Z. Wang, C. Xu, J. Chu, and F. Gao, "Tgk-planner: An efficient topology guided kinodynamic planner for autonomous quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 494–501, 2021.

[10] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[12] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 2421–2428.

[13] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Sept 2014, pp. 2997–3004.

[14] S. S. Joshi and P. Tsiotras, "Relevant region exploration on general cost-maps for sampling-based motion planning," *arXiv preprint arXiv:1910.05361*, 2021.

[15] M. Aditya, S. Rosario, H. Brian, C. Sanjiban, and S. Siddhartha, S., "Guided incremental local densification for accelerated sampling-based motion planning," *arXiv preprint arXiv:2104.05037*, 2021.

[16] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4207–4214.

[17] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[18] 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 *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[19] F. Hauer and P. Tsiotras, "Deformable rapidly-exploring random trees," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[20] R. Natarajan, H. Choset, and M. Likhachev, "Interleaving graph search and trajectory optimization for aggressive quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5357–5364, 2021.

[21] I. Pohl, "First results on the effect of error on heuristic search," *machine intelligence*, 1970.

[22] E. Rdiger and D. Rolf, "Weighted a* search unifying view and application," *Artificial Intelligence*, vol. 173, no. 14, pp. 1310–1342, 2009.

[23] 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.*, Sept 2017, pp. 2872–2879.

[24] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[25] E. Verriest and F. Lewis, "On the linear quadratic minimum-time problem," *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 859–863, 1991.

[26] Z. Wang, H. Ye, C. Xu, and F. Gao, "Generating large-scale trajectories efficiently using double descriptions of polynomials," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, vol. in press, no. in press, 2021, p. in press.

[27] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[28] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4423–4430.

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, no. 19, pp. 415–428, 2012.

[30] N. Haarala, K. Miettinen, and M. Mäkelä, "Globally convergent limited memory bundle method for large-scale nonsmooth optimization," *Mathematical Programming*, vol. 109, pp. 181–205, 2007.