Optimal Trajectory Generation for Quadrotor Teach-and-Repeat

Fei Gao, Luqi Wang, Kaixuan Wang, William Wu, Boyu Zhou, Luxin Han, and Shaojie Shen

***Abstract*—In this letter, we propose a novel motion planning framework for quadrotor teach-and-repeat applications. Instead of controlling the drone to precisely follow the teaching path, our method converts an arbitrary jerky human-piloted trajectory to a topologically equivalent one, which is guaranteed to be safe, smooth, and kinodynamically feasible with an expected aggressiveness. Our proposed planning framework optimizes the trajectory in both spatial and temporal aspects. In the spatial layer, a flight corridor is found to represent the free space that is topologically equivalent with the teaching path. Then, a minimum-jerk piecewise trajectory is generated within the flight corridor. In the temporal layer, the trajectory is re-parameterized to obtain a minimum-time temporal trajectory under kinodynamic constraints. The spatial and temporal optimizations are both formulated as convex programs and are done iteratively. The proposed method is integrated into a complete quadrotor system and is validated to perform aggressive flights in challenging indoor and outdoor environments.**

***Index Terms*—Aerial systems: applications, motion and path planning, autonomous vehicle navigation.**

I. Introduction

Thanks to their mobility, agility, and flexibility, micro aerial vehicles (MAVs), especially quadrotors, have been applied to many aspects recently. Among all applications, teach-and-repeat has significant potentials in aerial videography, repetitive inspection, and human-robot interaction.

We start off by looking into whats the best way for a human to teach the drone to fly autonomously. Instead of asking the drone to exactly follow the human-piloted trajectory, which can be incredibly hard to fly in some environments, we only require the human pilot to provide a topologically equivalent trajectory. Such human trajectory can be slow or jerky, but it should capture the rough route that we expect the drone to fly. From this, we propose a complete motion planning framework which can generate a safe, smooth and dynamically feasible trajectory that captures human intention. With our method, the quadrotor can autonomously convert a poor (nonsmooth and jerky) teaching trajectory to an energy and time efficient one with an expected aggressiveness.

In this letter, the user's demonstrated path is recorded and the surrounding environment is reconstructed during the teaching. Afterward, our method finds a flight corridor which preserves the topological structure of the path. Then trajectory generation is decoupled into spatial and temporal layers. In the flight corridor, a safe and energy-optimal spatial trajectory, as well as a physically feasible and time-optimal temporal trajectory are generated iteratively. Under our planning framework, the aggressiveness of the generated trajectory is easily tunable. For aerial inspection/videography applications, by setting a preference on motion smoothness, users can generate slow trajectories with gentle transitions. And in some extreme cases such as a challenging drone race, an unskilled pilot who suffers a lot in controlling the drone can use her/his navie operations to teach the quadrotor the rough route of the expected movements, and the generated fully aggressive trajectories may even beat experienced drone racing pilots.

The motion planning method proposed in this letter is based on our previous works in safe and smooth trajectory generation [1] and optimal quadrotor trajectory time allocation [2], and has following new contributions:

1) A flight corridor generation method with local loop detection mechanism, for finding a topology-equivalent and loop-free corridor around the teaching path.
2) A coordinate descent framework for the spatial-temporal joint trajectory optimization problem, for generating time-energy optimal trajectories.
3) An integration of the proposed motion planning method with localization, mapping, and control modules into a quadrotor platform. Aggressive teach-and-repeat indoor and outdoor flights are presented.

II. Related Work

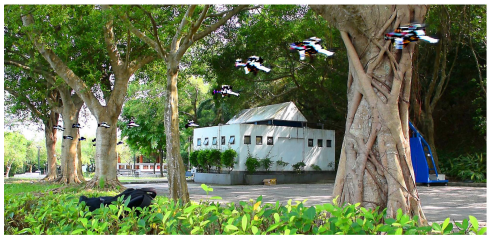
Robotics teach-and-repeat: Many robotics teach-and-repeat works have been published in recent years. In [3], a manifold map is built during the teaching phase of a ground vehicle and is then used for localization as the rover repeats the path autonomously. A multi-experience localization algorithm is proposed in [4], where the issue of appearance changes in environments is addressed by localizing against a number of past experiences. In [5] and [6], light and terrain changes are included in their proposed visual teach-and-repeat navigation system, for further improving the accuracy and robustness of

Manuscript received September 10, 2018; accepted January 9, 2019. Date of publication January 24, 2019; date of current version February 19, 2019. This letter was recommended for publication by Associate Editor F. Ruggiero and Editor J. Roberts upon evaluation of the reviewers' comments. This work was supported by Joint PG Program under HDJI Lab, Hong Kong University of Science and Technology. Grant/Award Number: project R9341. *(Corresponding author: Fei Gao.)*

The authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: fgaoaa@ust.hk; lwangax@ust.hk; kwangap@ust.hk; wwuar@ust.hk; bzhouai@ust.hk; luxin.han@ust.hk; eeshaojie@ust.hk).

Digital Object Identifier 10.1109/LRA.2019.2895110

(a) Composite image of the indoor quadrotor flight.

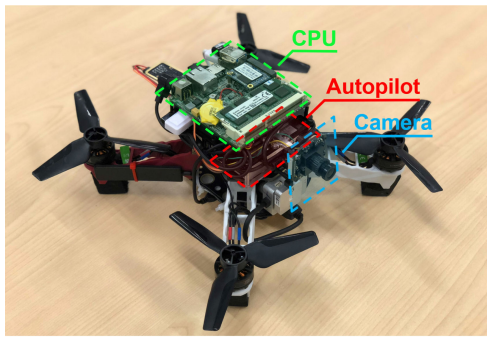


(b) Composite image of the outdoor quadrotor flight.

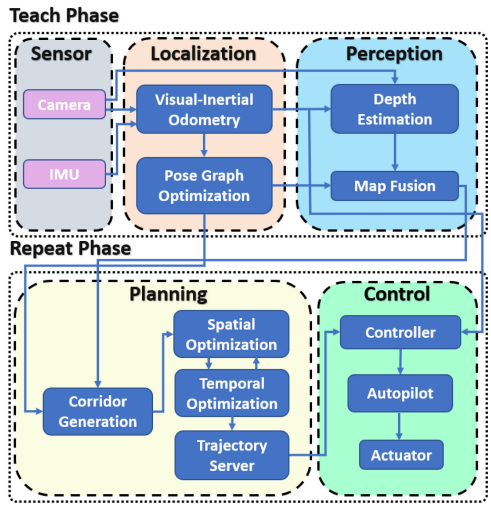
Fig. 1. Composite images of the experiments conducted in challenging indoor race and outdoor forest. The high-resolution video is available at https://www.youtube.com/watch?v=ehoJi4K_QKE.

the state estimation. Authors in [7] developed an iterative learning controller (ILC) to reduce the tracking error when the robot executes the teaching path. The ILC learns a feedforward control law to compensate unmodelled terrains, environments changes, and other control disturbance. Quadrotor teach-and-repeat work presented in [8] uses a vision-based drone to conduct a repetitive inspection by demonstrating the desired path and selecting some checkpoints. In the teaching phase, keyframes in the visual odometry are recorded. Then in the repeating phase, the quadrotor plans local trajectories to track the poses of keyframes by using minimum-snap polynomials [9]. To make this method work properly, the teaching path itself must be smooth and away from obstacles. On the contrast, our proposed method in this letter has no requirements on the teaching path and can convert an arbitrarily poor teaching path to a safe and efficient quadrotor trajectory.

Quadrotor trajectory optimization: Trajectory optimization is essential to generate a safe and executable trajectory from a poor jerky path. Gradient-based methods such as [10] formulate the problem as a nonlinear optimization over the penalty of safety and smoothness to obtain a locally optimal solution and has been successfully applied to aggressive quadrotor flight [11], [12]. Mellinger et al. [13] pioneered a minimum-snap trajectory generation algorithm, where the piecewise polynomial trajectory is optimized using the quadratic program (QP). Closed-form minimum snap trajectory generation which enables the aggressive quadrotor flight is proposed in [9], where the safety of the trajectory is achieved by iteratively adding intermediate waypoints to a discrete path. Our previous work [1], [14] carve a flight corridor consisting of some convex regions against the environment and confines a piecewise Bézier curve within the corridor by utilizing the convex hull property. In this way, the generated trajectory is guaranteed to be safe. Time optimization is used to generate minimum-time velocity/acceleration profile of a trajectory under the physical limits of the robot. Related methods can be divided into direct methods [15] which



(a) Overview of the hardware setting.



(b) The architecture of the software system.

Fig. 2. The hardware and software architecture of our quadrotor platform.

directly generate space-time optimal trajectory and indirect methods [16] which create a trajectory independent of time firstly and then find the relationship between the trajectory and the time. In [16], a mapping function from a virtual parametrization of the trajectory to time is optimized by nonlinear optimization. This method needs a kinodynamic feasible initial solution and it does not guarantee the global optimality. The approach proposed in [17] finds a near optimal mapping function by iteratively adding key points into the spatial trajectory to squeeze the kinodynamic feasibility of the time profile. However, this method also needs numerous numerical iterations. Our previous work [2] generates a piecewise polynomial-based kinodynamic feasible trajectory which is globally time-optimal with respect to an expected aggressiveness, by efficiently solving a convex program.

III. Teach-and-Repeat System

A. System Architecture

The overall hardware and software architecture of our quadrotor system is shown in Fig. 2. The perception module is running on an offboard laptop, while other processings are running on the onboard CPU. Our quadrotor localizes itself and reconstructs the environment by using a monocular camera and an IMU. The global map is built in the teaching phase and fed into the planning module. Based on the map, a flight corridor is generated by

inflating the teaching path meanwhile eliminating unnecessary loops. Then the spatial and temporal trajectories are optimized iteratively within the flight corridor under a coordinate descent scheme [18]. We use a geometric controller [19] to track the generated trajectory. And the attitude control is left to the autopilot.

B. Globally Consistent Localization and Mapping

The localization and perception functionalities build the foundation of our quadrotor system. In this letter we use the visual-inertial odometry (VIO) framework VINS [20] with loop closure to obtain accurate and globally corrected pose estimation. The perception module consists of a learning-based depth estimation [21] and a globally deformable map fusion [22]. In the teaching phase, when a loop closure is detected, the teaching path of the quadrotor is corrected by the global pose graph, and the map is deformed accordingly. In the repeating phase, the real-time pose estimation of the quadrotor is also corrected by the global pose graph to eliminate the drift of the pure visual-inertial odometry.

C. Coordinate Descent Optimization

The most essential step in converting a poor teaching path to an optimal trajectory is to determine the relationship between its geometric shape and time profile. Although it's hard to concurrently optimize the spatial and temporal properties of a trajectory, generating a safe saptial trajectory under a given time allocation (Sect. IV) and optimizing the time allocation of a given spatial trajectory (Sect. V) are both solvable. Therefore, we design a coordinate descent [18] framework to optimize the trajectory in the space-time joint solution space. We use a naive time allocation based on the Euclidean distance of the flight corridor to initialize the optimization. After generating a minimum-jerk spatial trajectory in the corridor, we use the time optimizer to obtain the optimal time allocation for this particular trajectory. Then the optimal time allocation is used to re-generate a spatial trajectory again. We define an overall objective function with weighting energy cost and time duration. And the spatial-temporal coordinate descent optimization is done iteratively until total cost cannot be reduced any more. The coordinate descent framework used in this letter combines our previous two works [1], [2] and enables the aggressiveness of the trajectory controllable by users. However, our previous method [1] is only applicable for generating relative slow motions under a conservative time allocation, and would easily be infeasible when the expected aggressiveness is high.

IV. Spatial Trajectory Optimization

A. Flight Corridor Generation

Having the mapping results from the perception layer, our planner maintains an occupancy grid map. We firstly extract free space around the teaching path to build a flight corridor, which provides a large solution space for the following trajectory optimization. The flight corridor consists of a series of axis-aligned cubes. Each cube is initialized as a voxel (3D grid) on the teaching path and is then inflated on x, y, z directions iteratively until its largest free volume. For a face of a cube in the μ-direction ($\mu \in \{x^+, x^-, y^+, y^-, z^+, z^-\}$), we query all the neighbor voxels attached to this face from μ-direction. If all these neighbor voxels are obstacle-free, we inflate the face to the μ-direction one step (voxel). More details about the cube inflation are illustrated in [1].

Algorithm 1: Flight Corridor Generation..

```
Notation: Path P, Corridor C, Point p, Cube c
Initialize :
p ← P.pop_front()
c = Inflate(p)
C.push_back(c)
while p ≠ p_target do
    if Outside(p, C[-1]) then
        if Inside(p, C[-2]) then
            C.pop_back()
        else
            c = Inflate(p)
            C.push_back(c)
        end if
    end if
    p ← P.pop_front()
end while
return C
```

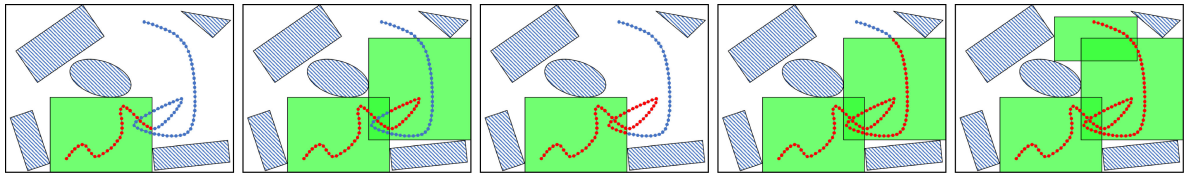
Here we highlight the mechanism we use to eliminate repeating loops existing in the human's teaching trajectory, as shown in Alg. 1. The flight corridor is initialized by inflating the first point (starting point) of the path. Then, all the points in the path are checked sequentially until the last one (target point p_{target} in Alg. 1). For a point, if it is outside the last cube ($\mathcal{C}$[-1]), we further check whether this point goes back to the previous corridor or discovers new space. If the point is inside the second last cube ($\mathcal{C}$[-2]), it means that the path returns to the previous corridor again. Therefore the last cube is considered repeating and is deleted from the corridor. Otherwise, the point is considered as discovering new free space and a new cube is generated and added to the corridor. The procedure is also illustrated in Fig. 3. Using this mechanism, the flight corridor that finally found can cover the topological free space of the teaching path without unnecessary loops. This property is essential to our framework since, in this way, our quadrotor can repeat a navie path using the most efficient motions. Experiments presented in Sect. VI prove that our method can convert extremely swerving and nonsmooth teaching trajectories to straight curves.

B. Piecewise Bézier Curve Optimization

In this letter, we use the Bernstein basis polynomials to represent the spatial trajectory, since they can be easily constrained in the flight corridor. The i^{th}-order Bernstein polynomial basis is:

$$b_n^i(t) = \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i}, \tag{1}$$

where n is the degree of the Bernstein polynomial basis, $\binom{n}{i}$ is the binomial coefficient and t is the variable parameterizing the trajectory. A polynomial consists of Bernstein bases is called a Bézier curve. For a piecewise Bézier curve, the m^{th} piece of



(a) Initialize. (b) Discover a new cube. (c) Delete the current cube. (d) Discover a new cube. (e) Finish.

Fig. 3. An illustration of the corridor generation. Red dots are points that have been checked while blue dots have not. Points on the teaching path are popped up and checked sequentially. A new cube is generated and added to the corridor when the current point leaves the last cube and enters undiscovered space, as in (b). Moreover, the last cube is deleted from the corridor when the current point leaves it and back to the second to last cube, as in (c).

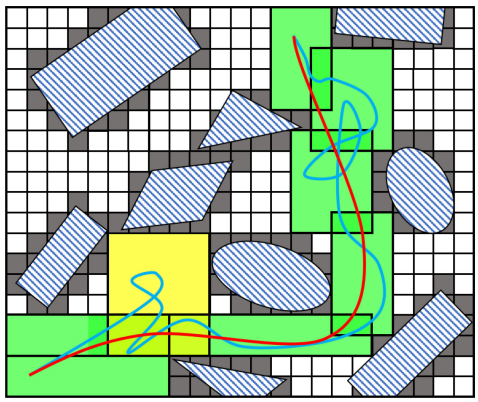


Fig. 4. The flight corridor and spatial trajectory generated in the grid map. Grids in grey are occupied by obstacles, while white cubes are free. The flight corridor is shown in green and the cube being deleted from the corridor is in yellow. The teaching path of the human and the generated trajectory are shown in the blue and red curve, respectively.

the curve at each dimension is written as:

$$f_m(t) = \sum_{i=0}^{n} c_m^i b_n^i \left(\frac{t}{T_m}\right), t \in [0, T_m], \tag{2}$$

where c_m^i is the control point of the i^{th} basis, and T_m is the time duration of this curve. Note that a normal Bézier Curve is defined on the interval [0,1], so we scale up t by T_m.

We minimize the squared jerk of the curve as it corresponds to angular velocity therefore benefits visual tracking. The objective function for the m^{th} piece of the curve is:

$$J = \int_0^{T_m} \left(\frac{d^3 f_m(t)}{dt^3}\right)^2 dt. \tag{3}$$

which is in a quadratic form. The complete objective function is the sum of all pieces of the curve in *x, y, z* dimensions and is denoted as $\mathbf{c}^T \mathbf{Q}_o \mathbf{c}$. Here $\mathbf{c}$ consists of all control points in *x, y, z* dimensions, and $\mathbf{Q}_o$ is the Hessian matrix.

The safety of the trajectory is guaranteed by enforcing control points of each piece to be inside the corresponding cube. Moreover, the piecewise smoothness, and the initial/final boundary constraints (start and terminal states) are enforced by setting affine equality constraints. Finally, the trajectory generation problem is formulated as a QP. Details about the formulation of the convex optimization can be checked in our previous publication [1], and an illustrative result is given in Fig. 4. Note that unlike [1], in this letter the kinodynamic feasibility of the trajectory is enforced by the following temporal optimization (Sect. V) instead of adding high-order constraints. For a trajectory with starting and ending states that are both static, the QP is always mathematically feasible.

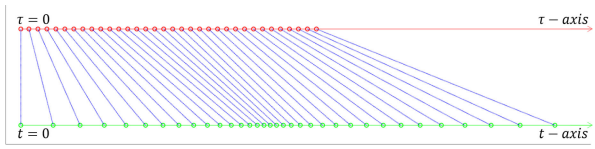


Fig. 5. The timeline of t and τ for a given spatial trajectory $f(t)$. Corresponding times are connected by blue dotted lines in this figure.

V. Temporal Trajectory Optimization

A. Piecewise Re-Timing Trajectory

After the spatial optimization, geometric properties of a corridor-constrained trajectory are fixed at a given time allocation. In this section, our objective is to design a piecewise re-timing function $\{t(\tau) : \tau \to t\}$ which maps t to a new time variable τ to generate motions along the spatial trajectory as fast as possible without violating the kinodynamic limits of the quadrotor. This procedure is called temporal optimization in this letter, and we denote the re-timing function as the temporal trajectory. A figure shows the relation between t and τ is given in Fig. 5

Suppose we have already solved the spatial trajectory $f(t)$. The piecewise re-timing function $t(\tau)$ is written as:

$$t(\tau) = \begin{cases} t_0(\tau), & t_1(0) = 0, t_0(\mathcal{T}_0^*) = T_0, \\ t_1(\tau), & t_1(0) = 0, t_1(\mathcal{T}_1^*) = T_1, \\ \vdots & \vdots \\ t_m(\tau), & t_m(0) = 0, t_m(\mathcal{T}_m^*) = T_m, \end{cases} \tag{4}$$

where $T_0, T_1, ...T_m$ are the original piecewise time durations of the curve and $\mathcal{T}_0^*, \mathcal{T}_1^*, ...\mathcal{T}_m^*$ are the optimized time durations after re-timing. Obviously $t(\tau)$ must be a monotonically increasing function since physically time can only increase. Therefore $\dot{t}(\tau) \geq 0$. By substituting t with $t(\tau)$ and using chain rule, we re-write the velocity and acceleration as:

$$\begin{aligned} v &= \dot{f}(t(\tau)) = f'(t) \cdot \dot{t}, \\ a &= \ddot{f}(t(\tau)) = f'(t) \cdot \ddot{t} + f''(t) \cdot \dot{t}^2, \end{aligned} \tag{5}$$

which are also piecewise functions. Here we use the notation $\dot{c} = dc/d\tau, \ddot{c} = d^2c/d\tau^2$ for taking derivatives with respect to the new time variable τ, and use $c' = dc/dt, c'' = d^2c/dt^2$ for derivatives with respect to t.

B. Convex Minimum-Time Optimization

1) Objective: With the fact $\dot{t} = dt/d\tau$, the total time of the trajectory is written as:

$$\mathcal{T} = \int_0^{\mathcal{T}} 1 d\tau = \sum_{i=0}^{m} \int_0^{T_i} \frac{1}{\dot{t}_i} dt. \tag{6}$$

We can introduce the regularization on changing rate of s in our objective, to trade-off the time optimality and control extremeness (expected aggressiveness):

$$\mathcal{J} = \sum_{i=0}^{m} \int_0^1 \left(\frac{1}{\dot{s}_i} + \rho \cdot \ddot{s}_i^{\,2} \right) ds, \tag{7}$$

where ρ is the weighting parameter. Increasing ρ leads to more gentle motions along the trajectory.

Following the direct transcription method in [23], we introduce two piecewise functions $a(t)$ and $b(t)$ which satisfy:

$$a_i(t) = \ddot{t}_i, \quad b_i(t) = \dot{t}_i^2. \quad i = 0, 1, ..., m. \tag{8}$$

We can easily derive that for each piece:

$$b_i(t) \geq 0, \quad b_i'(t) = 2 \cdot a_i(t). \tag{9}$$

Then the objective is derived to minimize:

$$\mathcal{J} = \sum_{i=0}^{m} \int_0^{T_i} \left(\frac{1}{\sqrt{b_i(t)}} + \rho \cdot a_i(t)^2 \right) dt, \tag{10}$$

2) Constraints: The continuities of the temporal trajectory are enforced by equality constraints between two consecutive pieces of the trajectory. In each dimension we have:

$$f_i'(T_i) \cdot \sqrt{b_i(T_i)} = f_{i+1}'(0) \cdot \sqrt{b_{i+1}(0)}, \tag{11}$$

$$f_i'(T_i) \cdot a_i(T_i) + f_i''(T_i) \cdot b_i(T_i) = f_{i+1}'(0) \cdot a_{i+1}(0) + f_{i+1}''(0) \cdot b_{i+1}(0) \tag{12}$$

Then, boundary constraints are set to meet the initial and the terminal velocity and acceleration a^0, v^0, a^f, v^f:

$$f_0'(0) \cdot \sqrt{b_0(0)} = v^0, \tag{13}$$

$$f_m'(T_m) \cdot \sqrt{b_m(T_m)} = v^f, \tag{14}$$

$$f_0'(0) \cdot a_0(0) + f_0''(0) \cdot b_0(0) = a^0, \tag{15}$$

$$f_m'(T_m) \cdot a_m(T_m) + f_{m,}''(T_m) \cdot b_m(T_m) = a^f, \tag{16}$$

Kinodynamic feasibilities are guaranteed by enforcing:

$$-v_{\max} \leq f_i'(t) \cdot \sqrt{b_i(t)} \leq v_{\max}, \tag{17}$$

$$-a_{\max} \leq f_i'(t) \cdot a_i(t) + f_i''(t) \cdot b_i(t) \leq a_{\max}, \tag{18}$$

where $v_{\max}$ and $a_{\max}$ are the kinodynamic limits.

3) SOCP Formulation: To make the above optimization problem tractable, the time variable $t_i \in [0, T_i]$ for each piece of the trajectory is discretized to $t_i^0, t_i^1, ...t_i^{K_i}$. Here $t_i^k - t_i^{k-1} = \delta t$, δt is a given resolution and $K_i = \lceil T_i/\delta t \rceil + 1$. $a_i(t)$ is set to be piecewisely constant at each δt and therefore $b_i(t)$ is piecewise linear according to Eq. 9. Then $a_i(t)$ and $b_i(t)$ are modeled by introducing discrete decision variables a_i^k and b_i^k, for which b_i^k is assigned at t_i^k and a_i^k is assigned at the middle of t_i^k and t_i^{k+1}, as shown in Fig. 6.

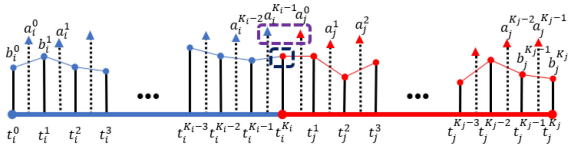


Fig. 6. b^k and a^k assigned in the i^{th} (in blue) and j^{th} (in red) pieces of the temporal trajectory, $j = i + 1$. b^k and a^k are indicated by circles and triangles. The continuity of velocity is enforced at the joint point between two segments and is shown in the black rectangle. And the continuity of acceleration is enforced in the purple rectangle between $a_i^{K_i-1}$ and a_j^0.

By discretization, the objective in Eq. 7 is derived as:

$$\mathcal{J} = \sum_{i=0}^{m} \sum_{k=0}^{K_i-1} \left(\frac{2}{\sqrt{b_i^{k+1}} + \sqrt{b_i^k}} + \rho \cdot a_i^{k\,2} \right) \cdot \delta t, \tag{19}$$

which is equivalent to the affine function:

$$\sum_{i=0}^{m} \sum_{k=0}^{K_i-1} \left(2 \cdot d_i^k + \rho \cdot a_i^{k\,2} \right) \cdot \delta t, \tag{20}$$

by introducing slack variables d_i^k, plus additional constraints:

$$\frac{1}{\sqrt{b_i^{k+1}} + \sqrt{b_i^k}} \leq d_i^k, \quad k = 0, ...K_i - 1; i = 0, ...m. \tag{21}$$

Eq. 21 is transformed to a quadratic form as

$$\frac{1}{c_i^{k+1} + c_i^k} \leq d_i^k, \quad k = 0, ...K_i - 1; i = 0, ...m. \tag{22}$$

$$c_i^k \leq \sqrt{b_i^k}, \quad k = 0, ...K_i; i = 0, ...m. \tag{23}$$

by introducing slack variables c_i^k in each segment.

Eq. 22 is equivalent to the typical formulation of a rotated quadratic cone:

$$2 \cdot d_i^k \cdot \left(c_i^{k+1} + c_i^k \right) \geq \sqrt{2}^2, \tag{24}$$

which is denoted as

$$(d_i^k, c_i^{k+1} + c_i^k, \sqrt{2}) \in Q_r^3. \tag{25}$$

And Eq. 23 can be written in the typical formulation of a (non-rotated) quadratic cone:

$$\left(b_i^k + 1 \right)^2 \geq \left(b_i^k - 1 \right)^2 + \left(2 \cdot c_i^k \right)^2, \tag{26}$$

and can be denoted as

$$(b_i^k + 1, b_i^k - 1, 2c_i^k) \in Q^3. \tag{27}$$

Finally, an additional slack variable t is added to convert the quadratic objective in Equ. 20 to an affine form as

$$\sum_{i=0}^{m} \sum_{k=0}^{K_i-1} 2\Delta s \cdot d_i^k + \rho \cdot \Delta s \cdot t, \tag{28}$$

with a rotated quadratic cone:

$$2 \cdot t \cdot 1 \geq \sum_{i=0}^{m} \sum_{k=0}^{K_i-1} (a_i^k)^2, \tag{29}$$

i.e.

$$(t, 1, \mathbf{a}) \in Q_r^{2+\sum_{i=0}^{m}(K_i)}, \tag{30}$$

where $\mathbf{a}$ is the vectorized variable consisting of all a_i^k.

We also apply the discretization of $a_i(t)$ and $b_i(t)$ to all constraints in Sect. V-B2 and abstract them as affine equality and in-equality constraints ($\mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq}$ and $\mathbf{A}_{ie} \cdot \mathbf{x} \leq \mathbf{b}_{ie}$). Details are omitted here. Besides, although we assume a_k is piecewisely constant, we would like to bound the changing rate of a_k considering the response time of the actuators of a quadrotor. We write the affine in-equality constraints for bounding the transition of acceleration:

$$-\delta a \leq (a_i^k - a_i^{k-1})/\delta t \leq -\delta a, \tag{31}$$

where δa (not jerk) is a given parameter to limit the changing rate of acceleration. Since the difference of τ between t_i^k and t_{i+1}^k cannot be determined during the optimization, we can only constrain the changing rate of a_k in t timeline.

Finally, the temporal optimization is formulated as a typical convex Second Order Cone Program (SOCP):

$$\begin{aligned}
\min \quad & \mathbf{h}^T \mathbf{d} + \rho \cdot t, \\
\text{s.t.} \quad & \mathbf{A}_{eq} \cdot \mathbf{x} = \mathbf{b}_{eq}, \\
& \mathbf{A}_{ie} \cdot \mathbf{x} \leq \mathbf{b}_{ie}, \\
& (t, 1, \mathbf{a}) \in Q_r^{2+\sum_{i=0}^{m}(K_i)}, \\
& (d_i^k, c_i^{k+1} + c_i^k, \sqrt{2}) \in Q_r^3, k = 0, ..., K_i - 1, \\
& (b_i^k + 1, b_i^k - 1, 2c_i^k) \in Q^3, k = 0, ..., K_i, i = 0, ...m.
\end{aligned} \tag{32}$$

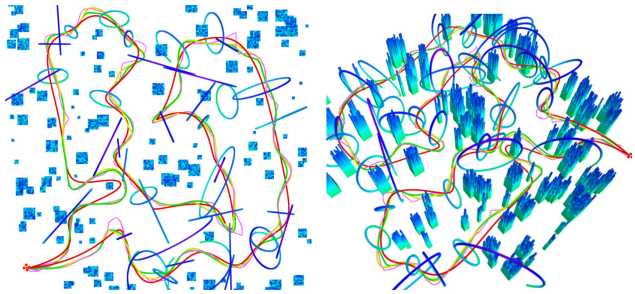
Here $\mathbf{d}$ and $\mathbf{x}$ consist of all d^k and a^k, b^k, c^k, d^k. δt controls the discretization of t and can be viewed as the resolution of the problem. We set δt to 0.025 in our following experiments. The effect of setting differnet ρ and δt and details about the derivation of SOCP can be viewed in [2].

Note that for a repeating trajectory with static initial and final states, the SOCP is always mathematically feasible regardless the geometric shape of the spatial trajectory. Because the optimization program can always find a feasible solution by enlarging the time duration. Combined with the mathematical feasibility of the spatial optimization (Sect. IV), once the flight corridor is found, an optimized spatial-temporal traejctory can always be found.

VI. Results

A. Implementation Details

The motion planning method proposed in this letter[1] is implemented in C++11 with a QP solver OOQP[2] and a SOCP solver Mosek.[3] The visual-inertial SLAM [20] and the deformable dense reconstruction [21] are also open sourced[4]. Readers can easily replicate the results presented in this letter. Note that the drift[5] of the VIO may cause unexpected crashes in the autonomous flight. Therefore we adopt the loop closure detection mechanism with global pose graph optimization in our localization module. Our dense map is globally deformable to attach to the pose graph, and all planning is done in the global frame. During the repeating phase, relative transformations from the global frame to the VIO frame are calculated in real-time and are used to project the control commands to VIO frame to compensate the drift.



(a) Bird view. (b) Side view.

Fig. 7. The comparison of the trajectory optimization based on the maunal teaching. The manual flight trajectory is shown as the purple cueve. Red, green and yellow trajectories are generated by our proposed method, gradient-based method [11] and waypoints-based method [9], respectively.

B. Benchmark Comparison

We compare the proposed method against other representative optimization-based trajectory generation methods, waypoint-based method [9], and gradient-based method [11]. We firstly convert the teaching trajectory to a piecewise path by recursively finding the collision-free straight line along the trajectory. Then we use the path to provide seeds for the waypoint-based [9] method and also to initialize the gradient-based method [11]. For a fair comparison, benchmarked methods are also integrated into the coordinate descent framework. We randomly generate 20 maps with dense obstacles and conduct 5 teach-and-repeat trials in each map. A sample result is shown in Fig. 7.

As shown in Table I, our proposed method outperforms in all length, time and energy aspects. Main reasons are as what follows. Firstly, compared to the waypoint-based method, our method is initialization-free and guarantees to find the best solution in its surrounding free space (the flight corridor). While the waypoint-based method is dominated by the initial waypoints which are fixed on the teaching path. Secondly, the gradient-based method is non-convex since it includes a collision cost term in objective function for the safety of the trajectory. Therefore, the gradient-based method always finds a locally optimal solution around its initial guess (the teaching path). However, our method enjoys the convexity in its formulation to obtain the global energy (smoothness)-optimal solution in the whole flight corridor. In turn, a smoother trajectory facilitates a better time optimization.

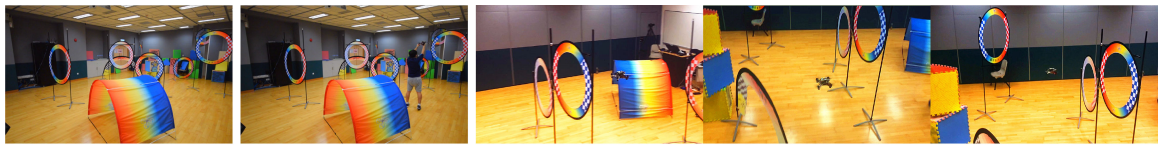
[1] Source code will be released in https://github.com/HKUST-Aerial-Robotics/stTraj after the publish of this letter.

[2] http://pages.cs.wisc.edu/~swright/ooqp/

[3] https://www.mosek.com

[4] https://github.com/HKUST-Aerial-Robotics/

[5] The drift is mainly caused by the error accumulation of random-walked measurements noise and the marginalization of past estimated robotics states. Details are in [24].



(a) Race scenario. (b) Teaching. (c) Repeating.

Fig. 8. The procedure of the teach-and-repeat experiment in an indoor environment. An overview of the test scenario is in (a). The quadrotor is handheld by a human operator to teach the path and scan obstacles in (b). Finally, the quadrotor tracks the generated trajectory in the repeating phase as in (c).

TABLE I
COMPARISON OF TRAJECTORY OPTIMIZATION

Method	Length (m)	Time (s)	Energy $((m/s^3)^2)$
Proposed Method	**91.604**	**42.50**	**448.21**
Gradient-based [11]	98.815	83.09	1091.31
Waypoint-based [9]	98.957	74.17	2002.12

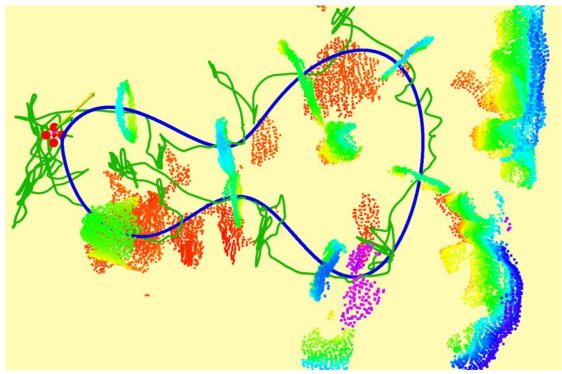
C. Indoor Flight Test

1) Drone Race Application: We conduct flight tests in a cluttered drone racing site, to validate the robustness of the proposed method, and also push the boundary of the quadrotor aggressive flight. In this test, a number of racing obstacles such as circles, tunnels, and arches are randomly deployed, as shown in Fig. 1(a). The maximum velocity and acceleration for the quadrotor are set as $3m/s$ and $2m/s^2$. And ρ in Equ. 7 is set as 0, which means we expect the quadrotor to fly as fast as possible given the kinodynamic limits. The entire teach-and-repeat pipeline is shown in Fig. 8. During the teaching phase, the quadrotor is handheld by a person to move amid obstacles and scan surroundings to build the map. Then the drone autonomously converts this teaching path to an efficient trajectory and tracks it. The teaching path and the generated trajectory which has a total length of $23.56m$ and time duration of $17.94s$. The teaching and planned trajectories are visualized in Fig 9. The desired and estimated positions and velocities of the drone are given in Fig. 10, showing neglectable tracking errors in the flight.

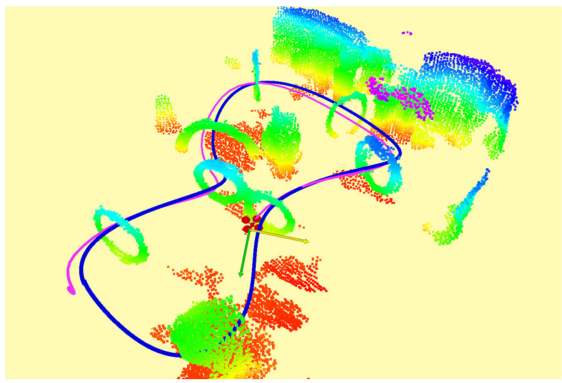
2) Drone Inspection Application: Here we also present flight test for applications where slow motions are more preferable, such as the aerial inspection or surveillance. In our proposed method, the expected aggressiveness of the final trajectory is easily tunable by setting the weighting between the time duration and the control cost. Therefore, a slow trajectory can be obtained by setting a large ρ (Sect. V-A) or conservative kinodynamic limits. Here we set the maximum velocity and acceleration as $0.5m/s$ and $0.5m/s^2$, and choose a relatively large ρ as 5, to generate slow motions which mimic the movements in indoor inspection scenarios. The teaching and repeating trajectory are shown in Fig. 11.

D. Outdoor Flight Test

Finally, we conduct quadrotor flight experiments in an outdoor natural environment, as in Fig. 1(b). No external positioning devices such as GPS are used. In this experiment, the quadrotor is commanded to repeat the path three times after



(a) Teaching phase.



(b) Repeating phase.

Fig. 9. Drone race experiment. The teaching, planed and actual tracking trajectory are shown as green, blue and purple curves, respectively. Color code in the map indicate heights of obstacles. The green and yellow arrows are the desired velocity and acceleration. The gap between the tracking and desired trajectory is caused by the drift of VIO, and is compensated by the global pose graph when controlling the quadrotor.

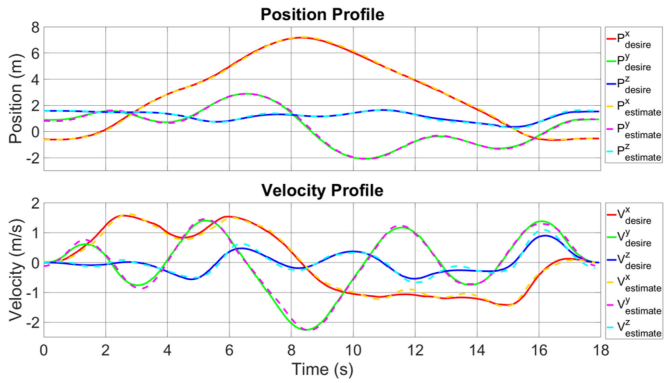


Fig. 10. The desired and estimated position and velocity profile. The estimated positions and velocities are obtained from our localization module. The tracking error is neglectable as shown in the figure.

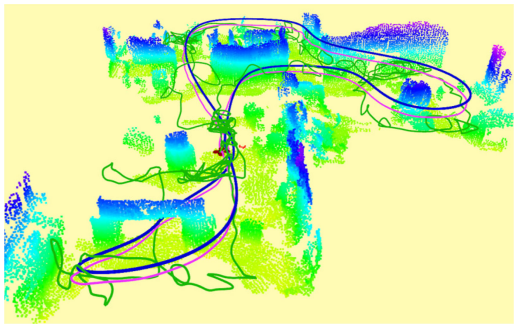
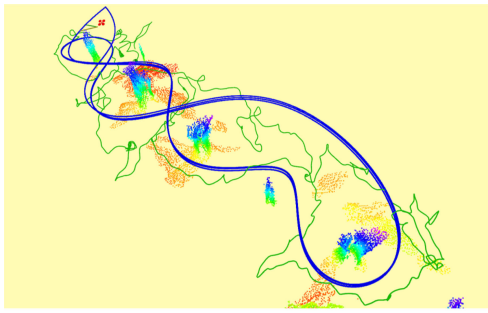
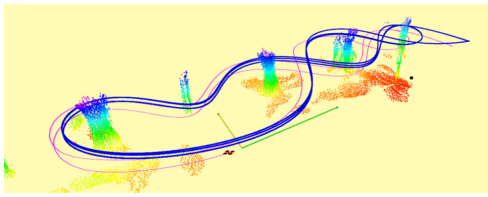


Fig. 11. Indoor drone inspection experiment. Markers are interpreted the same as in Fig. 9. The total length and time duration of the generated trajectory are $52.5194m$ and $103.02s$.



(a) Teaching phase.



(b) Repeating phase.

Fig. 12. Outdoor experiment. Markers are interpreted the same as previous.

the teaching. The maximum velocity and acceleration are set to $5m/s$ and $3m/s^2$. And ρ is set as 0. The total length and time duration of the generated trajectory are $173.806m$ and $72.167s$. Results are visualized in Fig. 12. Details about the experiments can be found in the video.

VII. Conclusion

In this letter, we propose a novel motion planning framework for quadrotor teach-and-repeat. Since no information about the time profile of an optimal trajectory is known, it is hard to obtain an optimal space-time trajectory by simply smoothing the path spatially. Instead, we decouple the optimal trajectory generation problem to spatial and temporal sub-problems and iteratively optimize them under coordinate descent optimization framework. In this work, we assume the environment is static from the teaching to the repeating and the quadrotor is commanded to execute the trajectory once it is generated. In the future, we plan to add reactive local (re)planning into our current framework to deal with dynamic environments. In this way, the generated repeating trajectory will work as a global reference path for the quadrotor. Then the quadrotor will track the reference path and avoid collisions with moving objects by local motion plans.

References

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

[2] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen, "Optimal time allocation for quadrotor trajectory generation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Madrid, Spain, Oct. 2018, pp. 4715–4722.

[3] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, no. 5, 2010, pp. 534–560.

[4] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, 2016, pp. 1918–1925.

[5] M. Paton, K. MacTavish, C. J. Ostafew, and T. D. Barfoot, "It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1519–1526.

[6] L.-P. Berczi and T. D. Barfoot, "It's like déjà vu all over again: Learning place-dependent terrain assessment for visual teach and repeat," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3973–3980.

[7] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 176–181.

[8] M. Fehr, T. Schneider, M. Dymczyk, J. Sturm, and R. Siegwart, "Visual-inertial teach and repeat for aerial inspection," 2018, *arXiv: 1803.09650.*

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

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

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

[12] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, 2018.

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

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

[15] H. M. Choset *et al. Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA, USA: MIT Press, 2005.

[16] M. Roberts and P. Hanrahan, "Generating dynamically feasible trajectories for quadrotor cameras," *ACM Trans. Graph.*, vol. 35, no. 4, 2016, Art. no. 61.

[17] J. Jamieson and J. Biggs, "Near minimum-time trajectories for quadrotor UAVs in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1550–1555.

[18] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.

[19] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *Proc. IEEE Control Decis. Conf.*, Atlanta, GA, USA, Dec. 2010, pp. 5420–5425.

[20] T. Qin, P. Li, and S. Shen, "VINS-MONO: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[21] K. Wang and S. Shen, "MVdepthNet: Real-time multiview depth estimation neural network," in *Proc. Int. Conf. 3D Vis.*, 2018, pp. 248–257.

[22] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[23] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[24] R. Mur-Artal and and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 796–803, Apr. 2017.