ITC 3/52

Information Technology and Control
Vol. 52 / No. 3 / 2023
pp. 665-678
DOI 10.5755/j01.itc.52.3.33583

Kinodynamic RRT* Based UAV Optimal State Motion Planning with Collision Risk Awareness

Received 2023/03/08 | Accepted after revision 2023/05/08

HOW TO CITE: Yin, H., Li, B., Zhu, H., Shi, L. (2023). Kinodynamic RRT* Based UAV Optimal State Motion Planning with Collision Risk Awareness. *Information Technology and Control,* 52(3), 665-678. https://doi.org/10.5755/j01.itc.52.3.33583

Kinodynamic RRT* Based UAV Optimal State Motion Planning with Collision Risk Awareness

Haolin Yin, Baoquan Li, Hai Zhu, Lintao Shi

Key Laboratory of Intelligent Control of Electrical Equipment, School of Control Science and Engineering, Tiangong University, Tianjin 300387, China

Corresponding author: libq@tiangong.edu.cn

In this paper, an autonomous navigation strategy is proposed for unmanned aerial vehicles (UAVs) based on consideration of dynamic sampling and field of view (FOV). Compare to search-based motion planning, sampling-based kinodynamic planning schemes can often find feasible trajectories in complex environments. Specifically, a global trajectory is first generated with physical information, and an expansion algorithm is constructed regarding to kinodynamic rapidly-exploring random tree* (KRRT*). Then, a KRRT* expansion strategy is designed to find local collision-free trajectories. In trajectory optimization, bending radius, collision risk function, and yaw angle penalty term are defined by taking into account onboard sensor FOV and potential risk. Then, smooth and dynamic feasible terms are penalized based on initial trajectory generation. Trajectories are refined by time reallocation, and weights are solved by optimization. Effectiveness of the proposed strategy is demonstrated by both simulation and experiment.

KEYWORDS: UAVs, motion planning, collision risk function, bending radius, yaw angle penalty.

1. Introduction

With development of onboard sensor and onboard computer, UAVs gain capability for complex tasks and have been used broadly in various fields. For motion planning of UAVs, movement constraints such as kinematics and dynamics are taken into account to obtain optimal trajectories, which bene-fit for efficient and safe flight [9, 27]. RGB-D cameras are widely utilized as onboard sensors due to traits of informative, high precision, and low cost, but their confined sensing range brings constraints for UAV motion planning [23].

When flight in unknown environments, local maps are established due to perception-limited onboard sensors and real-time processing requirement [31]. Local maps are continuously updated during navigation to accomplish autonomous flight [21], and the pattern is usually adopted in UAV motion planning. Some methods based on greedy search are designed for UAV motion planning, where a trajectory is searched with satisfying constraints and then optimized. For example, a classic robust framework is proposed in [37], whose front-end focuses on applying search-based kinodynamic A* to obtain a feasible trajectory, and then an optimization problem is constructed by considering constraints to calculate a resultant trajectory.

As an alternative to search-based motion planning, sampling-based approaches can obtain initial trajectories faster and with a higher success rate for complex environments. On this basis, the proposed method is developed containing two aspects of fast collision-free trajectory generation and trajectory optimization in unknown environments. In this paper, a KRRT* expansion strategy is first designed to generate a robust initial trajectory for UAVs by taking into account dynamic feasibility and collision. By combining polynomial motion planning, a collision-free trajectory is generated for complex environments, where perception range of depth cameras is taken into account.

For optimization, variation of the UAV yaw angle is confined by limited sensor perception range and effectiveness of planned trajectories is weakened [22]. To deal with the problem, new penalty functions are constructed for different cases in the paper, where the local A* algorithm is improved. Bending radius and collision risk function are introduced for considering obstacle influence, and a collision penalty function is defined. Trajectory velocities are optimized into a trapezoid-like variation to avoid large fluctuation. The UAV yaw angle is penalized for collision avoidance, and a fitting penalty function is defined by hyperbolas in the trajectory replanning stage. In addition, cost weights are tuned by quadratic programming (QP) to enhance planning efficiency. The main contribution of the paper is that KRRT* expansion strategy is proposed to guide local trajectory generation with satisfying motion constraints, and collision and yaw replanning penalty functions are designed for trajectory optimization by considering potential risk during UAV flight.

2. Related Work

Path planning results are utilized as initial solutions for motion planning, and trajectory optimization and replanning can be conducted to make resultant trajectories satisfy constraints of energy optimal, time optimal, and so on [20]. The efficient local trajectory planner in [37] divides trajectory generation into two processes: combination of front-end kinematic and dynamical search, and back-end B-spline op-timization. Gao et al. propose a local sensing and replanning approach, which utilizes spatio-temporal optimization to obtain energy-efficient repetitive trajectories, and combines online sensing and replanning to ensure safety against environment variation [10]. Tordesillas et al. plan two paths with jump point search and employ Gurobi to solve the mixed integer QP problem so that robot trajectories are within polyhedrons [30].

Regarding to sampling-based trajectory planning, Karaman et al. demonstrate that online planning convergence during execution can be improved with RRT* [15]. In [18], Lai et al. propose the fast exploratory random discrete tree (RRdT*) method, which inherits probabilistic completeness and as-ymptotic optimality of RRT*. Gammell et al. propose the batch processing information tree (BIT*) method, which searches for minimum costs on the basis of heuristic schemes, and converges asymptotically to global optimum by processing multiple batches of samples [7]. Jaillet et al. combine elements of RG-RRT and RC-RRT into the new environment guided RRT framework to improve sampling efficiency [14]. Seventh-order Bezier is utilized by Neto et al. to connect vertices of generating trees on the basis of the RRT* algorithm [25]. To process high-dimensional problems with discrepancy constraints, the hierarchical rejection sampling method is proposed in [17] to improve efficiency of sampling-based planners. Sampling efficiency is guaranteed by above works when reducing sampling frequency.

Moreover, sampling time can also be shortened to improve sampling efficiency, with utilizing lazy check and greedy exploration. Salzman et al. propose the efficient lower bound tree RRT method, which combines solutions from fast but sub-optimal RRT algorithm and asymptotically optimal RRG algorithm [29]. Ko et al. propose the vector field RRT method by using upstream standard integers to construct RRT in state space, and generated random nodes have a prespecified bias to the direction indicated by the vector field [16]. Jaillet et al. design the continuous transition RRT method, which combines RRT exploratory traits with stochastic optimization [13]. Informed RRT* is constructed in [8] by utilizing directly informed sampling techniques to obtain linear convergent performance. Hauser et al. propose sampling-based motion planners of Lazy-PRM* and Lazy-RRG*, which use a lazy strategy to eliminate most collision checks [16]. However, above methods do not take into account UAV kinematic and dynamical models, and are not applicable for high speed flight. To solve the problem, Ye et al. propose an efficient KRRT* framework on the basis of topological sampling [36].

In terms of trajectory optimization, the process is coupled with UAV internal and workspace constraints. Leobardo et al. fuse sampling-based methods and QP-based optimization, so that dynamic constraints of a robot are fully considered for generating time varying trajectories [2]. A practical path planning method is designed in [3] on the basis of Bezier curves, so as to operate autonomous vehicles under waypoint and corridor constraints. UAV surrounding environments are decomposed into convex regions by Watterson et al. to form motion constrains, and a feasible trajectory is then generated in these regions with avoiding obstacles [33].

To improve safety of trajectory optimization, researchers propose flight corridor based methods. Preiss et al. find dynamically feasible trajectories in a series of connected convex polyhedra that represent free spaces of environments, showing optimized trajectories have more accurate naviga-tion performance [28]. Dhullipalla et al. select trajectories to satisfy given initial and final states, and then treat trajectory generation as an optimal control problem [5]. A streaming safe corridor approach is proposed in [32] to compute optimize trajectories, which is through a constrained optimization problem. Based on constraints of flight corridors, adequate safety of trajectories can be guaranteed in conventional environments.

Speed and time constraints also should be considered in trajectory optimization. Escamilla et al. generate a temporally parameterized smooth trajectory by splicing multiple Bezier curves to form an optimal 4D trajectory [6]. A sixth-order Bezier curve is introduced in [35] to reduce curvature, where output of the path planner is smoothed by adjusting rotation. Special spline planning can also be designed for generating smooth paths, as shown in [11]. Beul et al. design a specific control input pattern in optimal control, so as to determine a minimum time trajectory of two state transition [1]. Nieuwenhuisen et al. design a local optimization method for mesh-based path planning, making a collision-free path and ensuring smoothness of continuous curvature transition [26]. Delingette et al. design a continuous curvature trajectory generation strategy, which is based on trajectory deformation by energy minimization and solves general geometric constraints [4]. As a consequence, efficient optimization guarantees quality of trajectories by considering various constraints.

3. Sampling-based Kinodynamic Trajectory Generation

According to property of differential flatness, the UAV state space model is shown as follows [24]:

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}\boldsymbol{U}(t),$$
$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0}_{3\times3} \\ \boldsymbol{I}_3 \end{bmatrix}, \ \boldsymbol{U}(t) = \ddot{p}(t) \tag{1}$$

where $\boldsymbol{p}(t)$ represents the 3D position of the UAV, $\boldsymbol{X}(t)$ denote the UAV state vector that contains 3D position and velocity:

$$\boldsymbol{X}(t) = [\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t)]^T \tag{2}$$

and $\boldsymbol{U}(t)$ denotes the control input vector relates to translational accelerations:

$$\boldsymbol{U}(t) = [u_x(t), u_y(t), u_z(t)]^T. \tag{3}$$

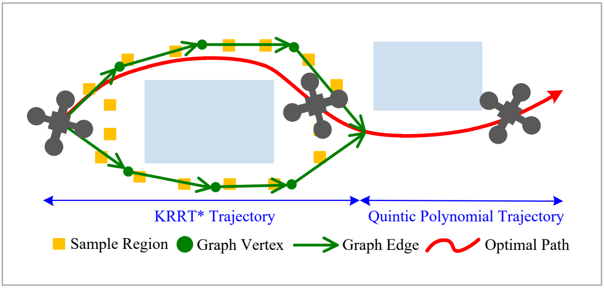
The linear system model brings much convenience for considering constraints of nonlinear terms and subsequent optimization calculation.

3.1. KRRT* Trajectories

For UAVs flying in complex environments, kinodynamic motion planning is introduced to obtain ini-

Figure 1

The diagram for UAV flight processes by the designed KRRT* expansion algorithm



Algorithm 1. **KRRT* Expansion**

```
1: Notation: Environment $E$, State $\boldsymbol{X}$, Tree $\gamma$, Time $t$
2: Initialize: $t \leftarrow 0, \gamma \leftarrow \varnothing \cup \{\boldsymbol{X}_s\}$
3: if KRRT* ($\boldsymbol{X}_s, \boldsymbol{X}_{cg}$) then
4:   whlie $t \leq t_{\max}$ do
5:     $\psi \leftarrow$TopologyGraph ($\boldsymbol{X}_s, \boldsymbol{X}_{cg}, E$)
6:     $\boldsymbol{X}_{new}$ Sample ($\psi$)
7:     $\phi_{near}$ Nearest ($\boldsymbol{X}_{new}, r, \gamma$)
8:     if $\boldsymbol{X}_p$ Parent ($\boldsymbol{X}_{new}, \phi_{near}, \psi$) then
9:       if GoalConnect ($\boldsymbol{X}_{new}, \boldsymbol{X}_g$) then
10:        ReachGoal ()
11:      end if
12:    end if
13:    $t \leftarrow$ Time ()
14:  end while
15:  $t \leftarrow$ Time ()
16: else
17:   Repeat ($N$)
18: end if
19: $t_{\max} \leftarrow$Extend ($t_{\max}$)
20: Repeat (1)
21: $path \leftarrow$PathExtract ($\boldsymbol{X}_f, \gamma$)
22: if Judge ($T$) then
23:   $path \leftarrow$Quintic polynomial ($\boldsymbol{P}_f, \boldsymbol{P}_t$)
24: end if
25: return $path$
```

tial trajectories with collision-free and smoothness. Kinodynamic motion planning is mainly divided into two patterns of search-based and sampling-based, and the sampling-based pattern is adopted by considering distribution of obstacles in dense environ-ments.

As shown in [36], calculating an optimal trajectory between two nodes is equivalent to solving the two point boundary value problem (TPBVP) in optimal control by minimizing the objective function:

$$C(\boldsymbol{X}(t)) = \int_0^{T} M(t, \boldsymbol{X}(t), \dot{\boldsymbol{X}}(t), \boldsymbol{U}(t))d(t), \tag{4}$$

where $M(\cdot)$ represents the process cost function, $C(X(t))$ denotes the total process cost.

Constraints for are

$$\begin{cases} f(t, \boldsymbol{X}, \boldsymbol{U}) - \dot{\boldsymbol{X}}(t) = 0, \\ \boldsymbol{X}(0) = \boldsymbol{X}_0,\ \boldsymbol{X}(\tau) = \boldsymbol{X}_1, \\ \boldsymbol{X}(t) \in \Omega_{free},\ \boldsymbol{U}(t) \in \mathrm{O}_{free} \end{cases} \tag{5}$$

where $f(t, \boldsymbol{X}, \boldsymbol{U})$ is the differential constraint, and Ω_{free} and O_{free} relate to collision free states.

Let the Hamiltonian function be

$$H(t, \boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\lambda}) = M + \boldsymbol{\lambda}^T \boldsymbol{f}(\cdot) \tag{6}$$

and it can be obtained from (6) that [36]

$$p_k^*(t) = \frac{1}{6}c_{k,3}t^3 + \frac{1}{2}c_{k,2}t^2 + c_{k,1}t + c_{k,0}, \tag{7}$$

$$u_k^*(t) = c_{k,3}t + c_{k,2},\ k \in \{x, y, z\}, \tag{8}$$

where $p_k^*(t)$ is the position obtained by solving the optimal control problem, $u_k^*(t)$ is resultant inputs, and $c_{k,x}$ represents coefficients.

3.2. Polynomial Trajectories

For polynomial trajectories, low-order ones are with smoothness but may lead to discontinuous acceleration. To obtain a continuous acceleration trajectory while considering initial and terminal conditions, quintic polynomial trajectories are combined for UAV motion planning, as used for the pre-optimized trajectories in [37]. The trajectory in the x direction can be expressed as

$$\begin{aligned} x(t) &= a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \\ &= [t^5\ t^4\ t^3\ t^2\ t\ 1][a_5\ a_4\ a_3\ a_2\ a_1\ a_0]^T \end{aligned} \tag{9}$$

and trajectories in y and z directions are represented similarly to (9).

With (4)-(8), initial-final state and dynamics constraints are considered, and topological sampling is utilized to guide generation of the trajectory tree for reasonable path topology.

3.3. KRRT* Expansion

Based on [36], a KRRT* trajectory expansion algorithm is designed to find local trajectory. For complex environments, KRRT* is utilized to provide high quality trajectories with satisfying UAV kinematic and dynamical constraints, and KRRT* is also chosen for global guidance. On the other hand, quintic polynomial planning is used to complete collision-free trajectories, and it is also used for sparse environments. A trajectory tree can grow by a topology-guided sampling strategy, as described in [36].

Figure 1 shows UAV flight processes by the designed KRRT* expansion algorithm, which is described in Algorithm 1. Running time t and a trajectory tree γ expanding from the initial state $\boldsymbol{X}_s$ are initialized, and terminal condition is set as t exceeds preset time t_{max} (Lines 2-4). A topology guided graph is constructed based on environment E that starts at $\boldsymbol{X}_s$ and ends at the goal state $\boldsymbol{X}_g$(Lines 5-6). Then, new state point $\boldsymbol{X}_{new}$ is sampled according to structure of the topology guided graph. Moreover, a set of nearest neighbor nodes ϕ_{near} is searched with radius r from $\boldsymbol{X}_{new}$, and $\boldsymbol{X}_p$ is chosen from ϕ_{near} as the parent node of $\boldsymbol{X}_{new}$ (Lines 7-8). When $\boldsymbol{X}_{new}$ is less than a certain distance from $\boldsymbol{X}_g$, a collision-free trajectory from $\boldsymbol{X}_s$ to final node $\boldsymbol{X}_f$ is obtained by solving TPBVP to satisfy state constraints (Lines 9-10).

Moreover, if above process fails, it is repeated by N times, and then the trajectory is generated if it succeeds (Line 17). If the trajectory still cannot be obtained, flight time is extended twice to repeat the process again (Lines 19-21). Since distance exists between final position $\boldsymbol{P}_f$ of the KRRT* trajectory and desired position $\boldsymbol{P}_g$ of $\boldsymbol{X}_g$, a quintic polynomial trajectory is generated to bring UAV to $\boldsymbol{P}_g$ (Lines 22-24).

3.4. Optimization Framework with KRRT*

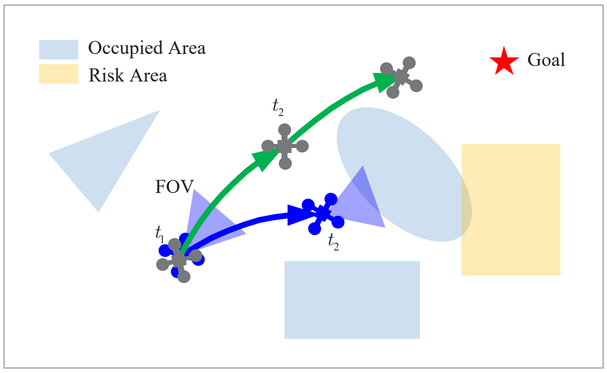
The designed optimization framework is described in Algorithm 2, and Figure 2 shows corresponding trajectory adjustment. KRRT* is first used to calculate a global guided trajectory $\boldsymbol{\Gamma}$ from starting position $\boldsymbol{P}_s$ to the global target position $\boldsymbol{G}^*$, and initial state information ξ_0 of the trajectory is extracted (Lines 2-3). Then,

Algorithm 2. State Optimization with KRRT*

```
1: Notation: Global Goal $G^*$, Local Goal $G$, Control Points $Q$,Time $t_0$, $t_1$, $t_2$
2: Initialize: $\Gamma \leftarrow$ FindGlobalTrajectory ($P_s$, $G^*$)
3: $\xi_0 \leftarrow$ StateExtract ($t_0$, $\Gamma$)
4: $Q \leftarrow$ FindLocalIntial ($P_s$, $G$)
5: while Collitioncheck ($\varepsilon$, $Q$) do
6:     $Q \leftarrow$ TopologyOptimization ($\varepsilon$, $Q$)
7: end while
8: if CheckFeasible ($Q$)
9:     $\Delta t \leftarrow$ AllocateTime ($Q$)
10:    $Q \leftarrow$ Reoptimization ($Q$)
11: end if
12: $\xi_1 \leftarrow$Update ($t_1$, $Q$)
13: $Q \leftarrow$ YawAngleOptimization ($Q$)
14: $\xi_2 \leftarrow$Update ($t_2$, $Q$)
15: return $Q$
```

Figure 2

The diagram for flight trajectory adjustment before and after Algorithm 2



Algorithm 1 is used to generate a local trajectory from $\boldsymbol{P}_s$ to local target position $\boldsymbol{G}$, and initial control points Q are calculated. The trajectory planning method [40] is used to optimize two topological trajectories in different threads separately to select low cost trajectory (Lines 4-7). If control points Q satisfy feasibility constraints, they are time reallocated and reoptimized, and state information ξ_1 of the trajectory is obtained (Lines 8-12). Yaw angle optimization is utilized for Q, and latest state information $\xi_2$of the trajectory is obtained (Lines 13-14).

Although Algorithm 2 takes more time to generate an initial trajectory compared with EGO [39] and considers less for optimality of generated trajectories, it takes less time for optimization. Moreover, and initial trajectory by Algorithm 2 is close to the optimal one, thus the optimizer in the paper is faster to find best results, as shown in Section 6.1.

4. Trajectory Optimization

In the section, an initial trajectory is parameterized as a uniform B-spline trajectory. Then, an unconstrained trajectory optimization problem is developed, and a final execution trajectory is thus solved.

4.1. Objective Functions for B-spline

A B-spline trajectory is determined by order k, n +1 control points $\{\boldsymbol{Q}_0, \boldsymbol{Q}_1, \boldsymbol{Q}_2, ..., \boldsymbol{Q}_n)$, and a knot vector $[t_0, t_1, ... t_M$, where $\boldsymbol{Q}_i \in \mathbb{R}^3$, $t_m \in \mathbb{R}$, and $M = n + k + 1$. The knot span of a uniform B-spline trajectory is denoted as

$$\Delta t_m = t_m - t_{m-1}. \tag{10}$$

Control points of velocity, acceleration, and jerk can be expressed as

$$\boldsymbol{V}_i = \frac{\boldsymbol{Q}_{i+1} - \boldsymbol{Q}_i}{\Delta t}, \boldsymbol{A}_i = \frac{\boldsymbol{V}_{i+1} - \boldsymbol{V}_i}{\Delta t}, \boldsymbol{J}_i = \frac{\boldsymbol{A}_{i+1} - \boldsymbol{A}_i}{\Delta t} \tag{11}$$

An objective function for trajectory optimization is denoted as

$$\min_{Q} E = \lambda_c E_c + \lambda_s E_s + \lambda_d E_d, \tag{12}$$

where E_c, E_s, and E_d are penalty functions for collision, smoothness, and feasibility, respectively, and λ_c, λ_s, and λ_d are weights.

4.2. Improvement of Local A*

Inspired by [39], a local A* algorithm is used to provide distance information for trajectory cost functions, and thus control points of the trajectory are adjusted by optimization. Cost function of an A* node position $\boldsymbol{p}_n(x_n, y_n, z_n)$ can be expressed as

$$f_n = g_n + h_n. \tag{13}$$

Height variation of A* path is minimized by redesigning actual cost function g_n :

$$g_n = g_{n-1} + d(p_n, p_{n-1}) + k_n \tag{14}$$

where $d(\boldsymbol{p}_n, \boldsymbol{p}_{n-1})$ represents the distance between neighbor nodes, and k_n is

$$k_n = \begin{cases} k_{n-1} + \rho\varphi(n), & n > 0, \\ 0, & n = 0. \end{cases} \tag{15}$$

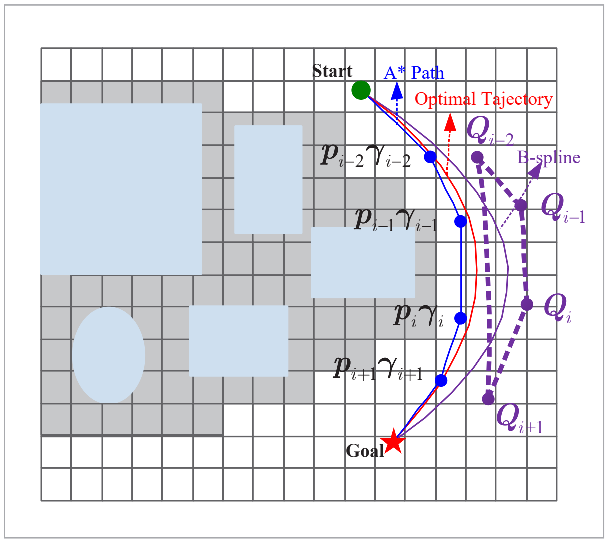
where

$$\varphi(n) = \rho\left(\frac{h}{|z_n - z_{n-1}|} + \frac{|z_n - z_{n-1}|}{h}\right), \tag{16}$$

where ρ is a height factor, and h is maximum height variation of A* nodes.

Figure 3 illustrates control point adjustment by local A*, where blue areas are obstacles and gray areas are occupied areas after obstacle expanding. Purple points are initial control points corresponding to a B-spline trajectory, and blue points are collision free path points on obstacle surfaces searched by A*. The red curve is an optimal trajectory after following optimization.

Figure 3
The diagram of trajectory adjustment with a local A* algorithm



4.3. Bending Radius

A UAV needs to stay within a certain distance from obstacles to ensure safety, and distance $(\boldsymbol{Q}_i - \boldsymbol{p}_i)\gamma_i$ is defined between a B-spline control point $\boldsymbol{Q}_i$ and a collision-free A* path point $\boldsymbol{p}_i$ as

$$L_{\min} < (\boldsymbol{Q}_i - \boldsymbol{p}_i)\gamma_i < L_{\max}, \tag{17}$$

where L_{min} and L_{max} are minimum and maximum values of allowed distance between UAV and obstacle surface, respectively, and γ_i is a unit vector from $\boldsymbol{Q}_i$ to $\boldsymbol{p}_i$ that points to obstacle surface.

As shown in Figure 4, bending radius r is introduced to make a UAV consider potential oversteer risk for safe turning. Bending radius is closely related to speed and acceleration of a UAV, and following equation can be derived:

$$r_{\max} = \frac{v_m^2}{2a_m}. \tag{18}$$

Moreover, a time constraint is introduced to sufficiently ensure UAV safety:

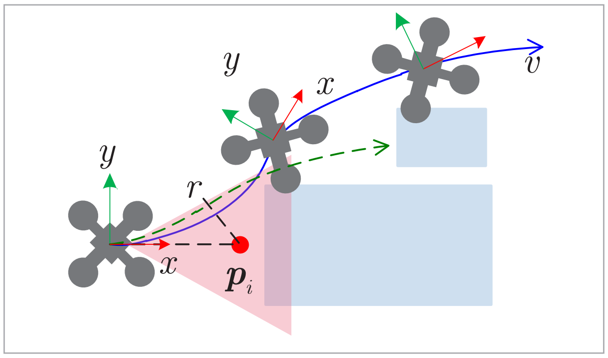
$$t_{\max} - \frac{||\boldsymbol{v}||}{||\boldsymbol{a}||} \geq \sqrt{\frac{2(\boldsymbol{Q}_i - \boldsymbol{p}_i)^T \boldsymbol{\gamma}_i}{||\boldsymbol{a}||\cos(\pi/4)}}. \tag{19}$$

Thus, the bending radius r is obtained:

$$r = \frac{1}{2}||\boldsymbol{v}||(t_{\max} - \sqrt{\frac{2(\boldsymbol{Q}_i - \boldsymbol{p}_i)^T \boldsymbol{\gamma}_i}{||\boldsymbol{a}||\cos(\pi/4)}}). \tag{20}$$

Figure 4

Schematic diagram considering potential unknown obstacles



which contains time term, velocity term, and realtime distance for better collision avoidance.

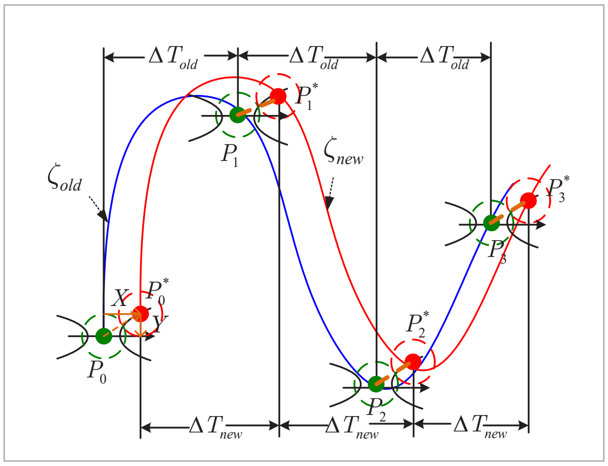
4.4. Collision Penalty

To switch among collision penalty functions, a collision risk function Φ is defined by neglecting perpendicular velocity component v_z as

$$\Phi = \frac{r}{(r/r_{\max})v_x + v_y}. \tag{21}$$

Figure 5

Diagram for time reallocation and refinement



Then, by considering normal flight and turning, a penalty function for single control point is designed as

$$e_{ci} = \begin{cases} (\frac{1}{2}v_v t_{\max} - L)^3, & 0 \leq \Phi \leq \frac{1}{4}\Phi_{\max}, \\ L^3, & \frac{1}{4}\Phi_{\max} < \Phi < \frac{3}{4}\Phi_{\max}, \\ w_1 L^3 + w_2(\frac{1}{2}v_v t_{\max} - L)^3, & \frac{3}{4}\Phi_{\max} \leq \Phi \leq \Phi_{\max} \end{cases} \tag{22}$$

where L: = $(\boldsymbol{Q}_i - \boldsymbol{p}_i)\gamma_i$. Thus, total cost of collision is obtained as

$$E_c = \sum_{i=1}^{N_c} e_c(\boldsymbol{Q}_i) = \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} e_{ci}. \tag{23}$$

where N_p is the pair number of each $\boldsymbol{Q}_i$ matching with $\boldsymbol{p}_i$, and N_c is the number of control points for the local trajectory.

4.5. Smoothness and Feasibility Penalty

Based on [39], smoothness penalty functions for both initial trajectory and yaw angle optimization are designed in this part to optimize trajectories progressively. Firstly, there are jerk control points corresponding to the initial trajectory from Algorithm 1, which are minimized by

$$E_s = \sum_{i=1}^{N-2} ||\boldsymbol{J}_i||^2. \tag{24}$$

Then, when optimizing the yaw angle, both acceleration control points and jerk control points are minimized, guaranteeing smoothness of the trajectory:

$$E_s = \sum_{i=1}^{N-1} ||\boldsymbol{A}_i||^2 + \sum_{i=1}^{N-2} ||\boldsymbol{J}_i||^2. \tag{25}$$

Moreover, for feasibility objective, velocity and acceleration of a trajectory should be less than maximum values, and feasibility penalty functions of two sequential stages are constructed. For optimizing initial kinodynamic trajectory, as in Algorithm 1, a feasibility penalty function is first designed from both velocity and acceleration as

$$E_d = w_v \sum_{i=1}^{N} F(\boldsymbol{V}_i) + w_A \sum_{i=1}^{N-1} F(\boldsymbol{A}_i) \tag{26}$$

where $F(\cdot)$ is a continuous derivable function similar to that in [39]. Then, for optimizing yaw angle, the constraint on the velocity aspect is needed to prevent UAV from danger due to large velocities:

$$E_d = \sum_{i=1}^{N} F(\boldsymbol{V}_i). \tag{27}$$

4.6. Yaw Angle Penalty

Inspired by [38], the yaw angle $\phi(t)$ is optimized to be smooth and dynamic feasible, which is parameterized as a uniform B-spline curve with control points $\Phi := \{\phi_1, \phi_2, \phi_3, ..., \phi_M\}$ and knot span Δt_ϕ. To flight with maximum bending radius, the cost function can be designed as

$$e_{\theta c} = (r_{\max} - r)^3. \tag{28}$$

The cost function for the i th control point ϕ_i can be written as

$$e_{\theta c}(\phi_i) = \sum\nolimits_{j=1}^{N_p} e_{\theta ci}, \tag{29}$$

where N_p is the pair number of each ϕ_i matching with $\boldsymbol{p}_i$. Thus, total cost for the yaw angle is

$$E_{\theta c} = \sum\nolimits_{i=1}^{M} e_{\theta c}(\phi_i), \tag{30}$$

where M is the number of control points for the local trajectory.

$$\min_{Q} E_\theta = \lambda_{\theta c} E_{\theta c} + \lambda_{\theta s} E_{\theta s} + \lambda_{\theta d} E_{\theta d}, \tag{31}$$

where $\lambda_{\theta c}$, $\lambda_{\theta s}$, and $\lambda_{\theta d}$ are weights, and it is noted that $E_{\theta s}$ and $E_{\theta d}$ are smoothness and feasibility functions, respectively, which are derived the same with E_s and E_d in Section 4.4.

5. Trajectory Refinement with Weight Solving

5.1. Time Reallocation and Trajectory Refinement

As shown in [39], exceeding ratio σ of trajectory velocity regarding to maximum permissible values needs to be calculated. It is expanded in the proposed method for time reallocation to make the trajectory safer. From the trajectory velocity $\boldsymbol{V}_{i,r}$, the speed restriction value $v_{restric}$ is set as

$$v_{restrict} = \left| \boldsymbol{V}_{i,r} + \frac{r_{safe}(v_{\max} - \boldsymbol{V}_{i,r})}{r_{\max}} \right|, \tag{32}$$

where $i \in \{1, \ldots, N_q - 1\}$, $r \in \{x, y, z\}$. Safety distance r_{safe} is designed as

$$r_{safe} = \frac{1}{2} ||\boldsymbol{v}||_2 (t_{\max} - \sqrt{\frac{2d_{safe}}{||\boldsymbol{a}|| \cos(\pi/4)}}), \tag{33}$$

where d_{safe} is a safe distance from an obstacle, and then exceeding ratio σ is calculated as

$$\sigma = \max\{ \left| \frac{v_{restrict}}{v_{\max}} \right|, \sqrt{\left| \frac{\mathbf{A}_{j,r}}{\mathrm{a}_{\max}} \right|} \}. \tag{34}$$

Table 1

Comparison of this method with ego's trajectory generation, trajectory optimization, and total time

Density(obs./m²)		T_{init} (ms)		T_{opt} (ms)		T_{all} (ms)	
		Avg	S.D.	Avg	S.D.	Avg	S.D.
Proposed	0.4	1.8	0.772	0.659	0.216	2.46	0.887
	0.32`	1.656	0.42	0.861	0.534	2.517	0.908
	0.24	1.707	0.702	0.913	0.495	2.458	1.138
EGO	0.4	0.416	0.245	7.336	6.505	7.71	6.478
	0.32	0.224	0.063	2.766	2.275	2.999	2.321
	0.24	0.265	0.135	5.297	2.964	5.554	2.964

Thus, the knot span of the new trajectory ζ_{new} can be written as

$$\Delta T_{new} = \sigma \Delta T_{old}. \tag{35}$$

After adjusting knot span, the initial trajectory is obtained by solving the least squares problem in closed form.

Then, to refine the trajectory, a penalty function is introduced to describe difference between two curves:

$$E_h = \int_0^1 (\frac{X^2}{a^2} - \frac{Y^2}{b^2}) d\rho, \tag{36}$$

where X and Y represents horizontal and vertical displacement of points on two curves, respectively. Thus, objective function of the trajectory after time reallocation is

$$\min_Q E_{new} = \lambda_s E_s + \lambda_h E_h + \lambda_d E_d, \tag{37}$$

where λ_h is a weight for E_h. Geometric effectiveness of new cost function and objective function is shown in Figure 5, where the blue trajectory ζ_{old} is before refinement and the red one ζ_{new} is after adjusting knot span.

5.2. Weights Solving by Optimization

To solve the problem that weights in unconstrained optimization are usually determined empirically in previous methods, they are calculated based on the QP model in the paper to obtain high quality trajectories.

To slightly tune weights first, values of cost functions E_s, E_c, E_d are selected corresponding to high quality trajectories, which are used to calculate expectation HE_s, HE_d, HE_c of the cost function. Then, a covariance matrix for building the QP model can be derived as

$$COV = \begin{pmatrix} \mathrm{cov}(E_s, E_s) & \mathrm{cov}(E_s, E_d) & \mathrm{cov}(E_s, E_c) \\ \mathrm{cov}(E_d, E_s) & \mathrm{cov}(E_d, E_d) & \mathrm{cov}(E_d, E_c) \\ \mathrm{cov}(E_c, E_s) & \mathrm{cov}(E_c, E_d) & \mathrm{cov}(E_c, E_c) \end{pmatrix} \tag{38}$$

Regarding to the unknown weights $\lambda_s, \lambda_d, \lambda_c$ of E_s, E_c, E_d an objective function is established as

$$\begin{aligned} DE_\Sigma &= D(E_s\lambda_s + E_d\lambda_d + E_c\lambda_c) \\ &= D(E_s\lambda_s) + D(E_d\lambda_d) + D(E_c\lambda_c) + 2\,\mathrm{cov}(E_s\lambda_s, E_d\lambda_d) \\ &\quad + 2\,\mathrm{cov}(E_s\lambda_s, E_c\lambda_c) + 2\,\mathrm{cov}(E_d\lambda_d, E_c\lambda_c) \\ &= \sum_{i=s,d,c} \sum_{j=s,d,c} \lambda_i \lambda_j \,\mathrm{cov}(E_i, E_j) \end{aligned} \tag{39}$$

which is used to minimize variance of the total cost of a trajectory. Then, a QP model specially for solving weights can be written as

$$\begin{aligned} \min \quad & DE_\Sigma = \sum_{i=s,d,c} \sum_{j=s,d,c} \lambda_i \lambda_j \,\mathrm{cov}(E_i, E_j) \\ s.t. \quad & \lambda_s + \lambda_d + \lambda_c = A,\ \lambda_s, \lambda_d, \lambda_c \geq 0, \\ & \lambda_s HE_s + \lambda_d HE_d + \lambda_c HE_c \geq \min(E_s, E_d, E_c) \end{aligned} \tag{40}$$

where A is a constant of maximum weight sum with relatively high quality trajectories. The lingo optimization software is used to solve optimal weights.

6. Simulation and Experiment Results

6.1. Simulation Tests

The proposed method is tested by both simulation and realworld experiments. In simulation, comparison with the method [39] is carried out in different obstacle density environments, with fixed start and end points. Simulations run on a computer with an Intel i7-7700HQ CPU and a GeForce GTX 1050 GPU. The L-BFGS method [19] is used to solve the trajectory optimization problem, and the method is implemented on [39] under occupancy grid maps. Maximum velocity and acceleration are set as 2m/s and 2m/s^2, respectively.

Left pictures of Figure 6 show trajectories by Ego [39] and right ones show trajectories by the proposed method. It is seen that the proposed method plans smoother trajectories then Ego, where green lines form the topology guided graph. Figure 7 shows distribution of yaw angles in the two density environments, and it is seen that variation of the yaw angle by the proposed method is small than that of Ego. Figures 8-9 show UAV posi-

Figure 6

Visualized pictures for simulation comparison between Ego [39] (left) and the proposed method (right) in different density environments

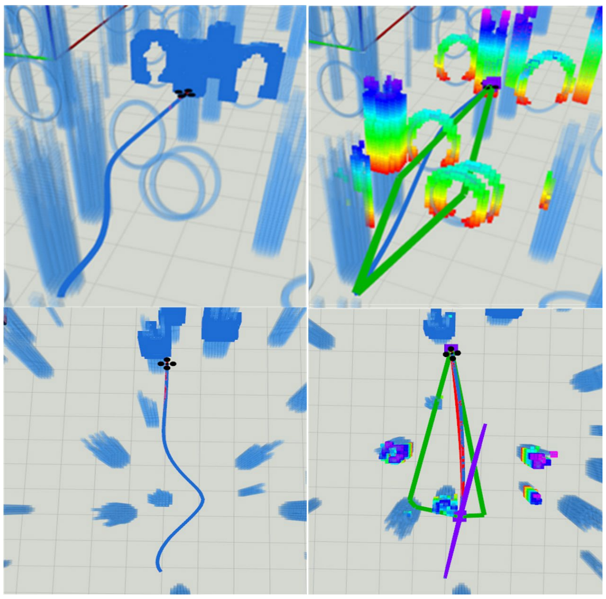


Figure 7

Distribution of yaw angles in the two density environments (left: in sparse obstacles, right: in dense obstacles)

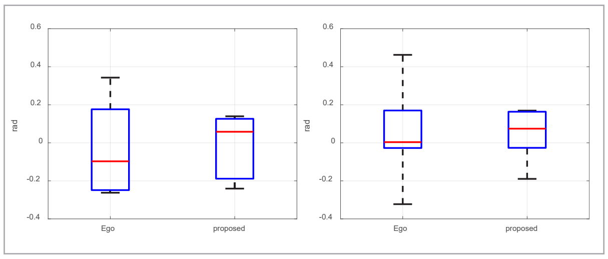


Figure 8

UAV position and velocity in dense obstacle environment, the (dashed lines: Ego, solid lines: the proposed method)

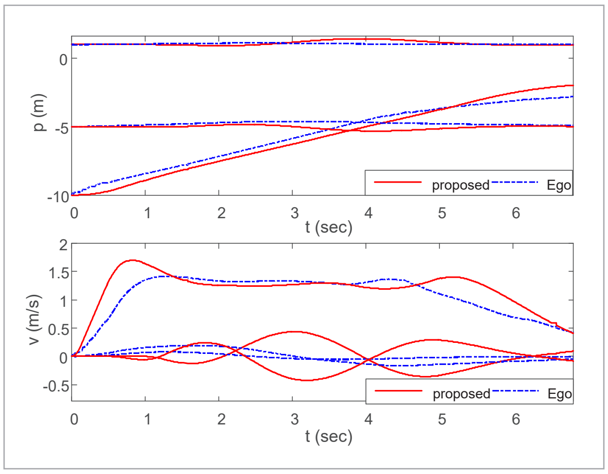
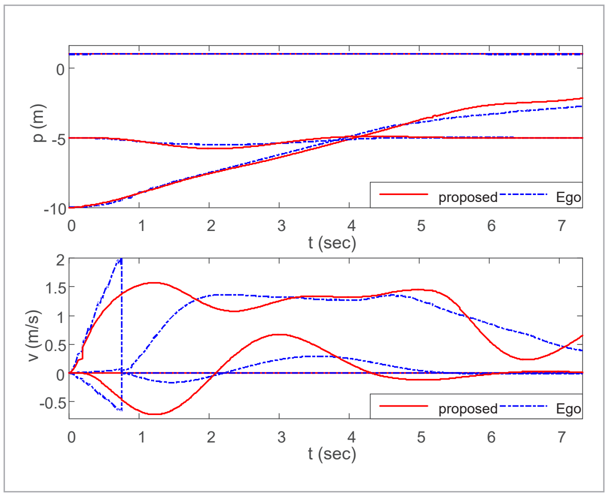


Figure 9

UAV position and velocity in sparse obstacle environment, the (dashed lines: Ego, solid lines: the proposed method)



tion and velocity in the different density environments, where it is known velocities by the proposed method approximate to trapezoidal variation in the forward direction and varies steadily in the other two directions.

Time cost is cost is shown in Table 1, for initial trajectory generation, since the proposed method considers kinodynamic constraints, it spends more time than Ego. Time taken by the method increases slightly as obstacle density increases to 0.4obs./m^2. For backend trajectory optimization, the proposed method takes much less time than Ego.

6.2. Experiments

As shown in Figure 10, the UAV is equipped with and Intel RealSense D435i depth camera and an onboard computer NVIDIA Xavier NX. Required trajectories are iterated faster by GPU acceleration. Figure 11 shows software architecture of the UAV system, and the local map setup, trajectory planning, localization, and control run on the onboard computer, while visualization runs on the ground station. An occupancy grid map is applied for map setting up, which converts depth information observed by D435i, and VINS-FUSION is employed for UAV localization.

Moreover, experiments are carried out in a $7.6 \times 4.9 \times 2.8\text{m}^3$ field, where obstacles are deployed arbitrarily. Maximum speed and acceleration are set as 1m/s and 1.5m/s^2, respectively. The proposed method and Ego are run under the same environment with fixed starting and ending points. Comparative results are shown in Figure 12, where the left graph is trajec-

Figure 10

The UAV platform for experiment flight

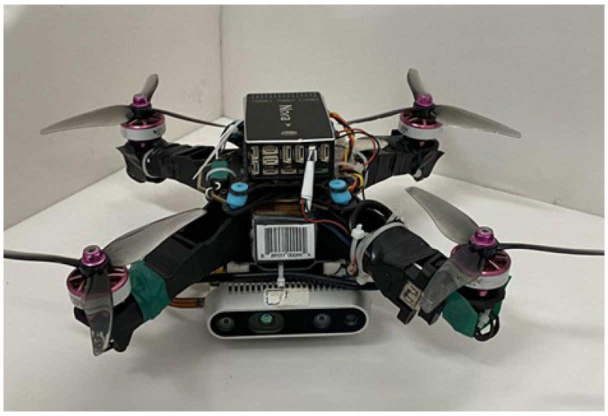


Figure 11

Software architecture of the UAV system

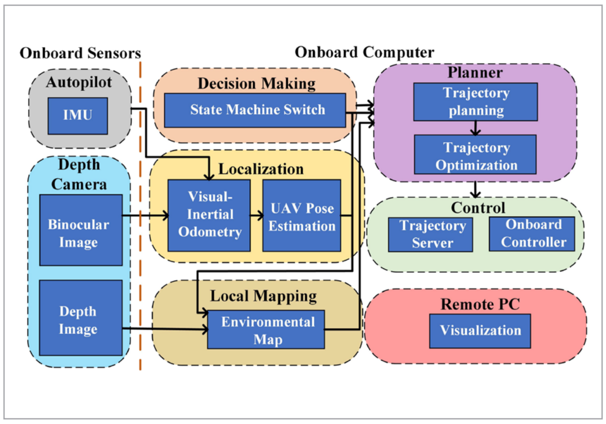


Figure 12

Map and trajectory by Ego [39] (left) and the proposed method (middle) in the same environment (red trajectories: planned, blue trajectories: executed, right graph: position and velocity by the two methods)

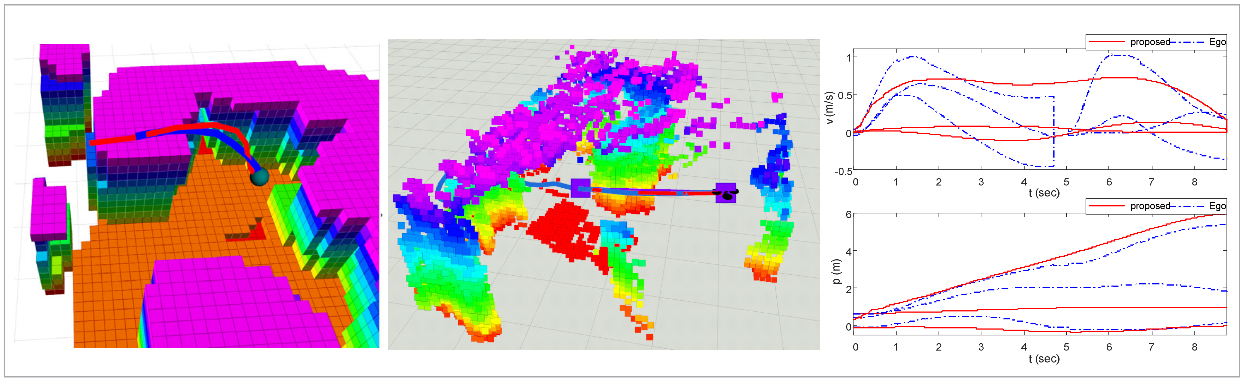


Figure 13

Map and trajectory in the UAV passing a square hole (left) and passing dense obstacles by foam boxes and iron rods (right) by the proposed method (red trajectories: planned, blue trajectories: executed)

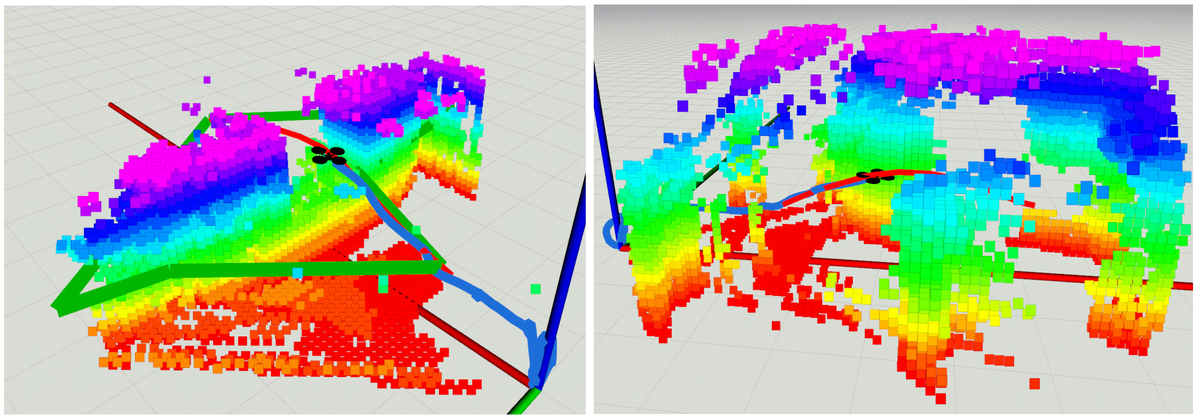
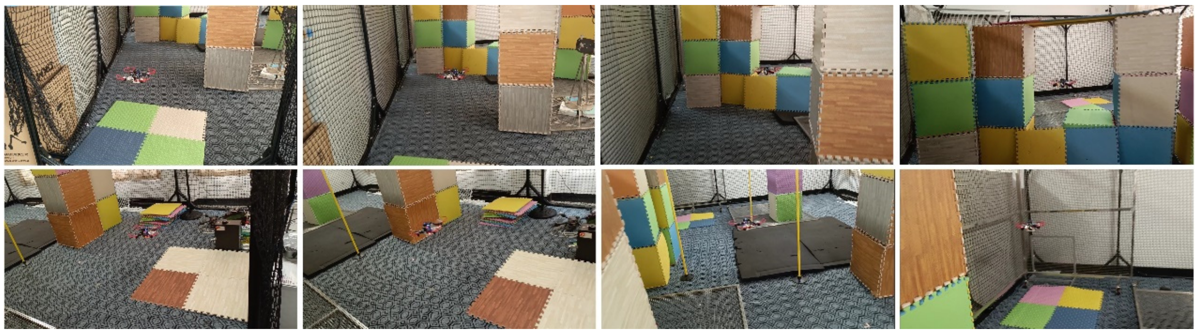


Figure 14

Photos for the UAV passing a square hole (top) and passing dense obstacles by foam boxes and iron rods (bottom) by the proposed method



tories by Ego and the middle graph is by the proposed method. It can be seen that the UAV flies directly to the target endpoint by the proposed method, while flies to high place then to the target endpoint by Ego. Position and velocity by the two methods are provided in the right graph of Figure 12, where the proposed method makes the UAV reach the endpoint in about 90s, and the speed is with approximately trapezoidal variation for the forward direction.

Moreover, two experiments are conducted to let the UAV pass a square hole and pass dense obstacles by foam boxes and iron rods, respectively. Figure 13 shows map and trajectory in the two processes, where green lines in the left plot are the topology guided graph formed by the environment, and flight photos are provided in Figure 14. In the passing hole experiment, front-end trajectory planning algorithm in the proposed method is tested, and it is seen that this method helps the UAV plan the optimal trajectory and makes the UAV fly out from the hole center. In the passing dense obstacle experiment, size and location of obstacles are randomly configured, with foam boxes simulating enormous obstacles and iron rods simulating small size obstacles. In this challenging scenario for real-time planning, based on the topology guided graph results, the proposed planner samples and generates the initial trajectory and then optimizes the trajectory in relative smoothness.

7. Conclusion

An autonomous UAV navigation strategy is proposed in the paper based on kinodynamic planning to solve feasible trajectories in complex environments under finite field of view. A global trajectory is generated first by using environment topology information, and a KRRT* expansion algorithm is designed. Then, a KRRT* expansion strategy is designed to find local collision-free trajectories. In trajectory optimization, bending radius and collision risk function are defined as well as a cost function related to yaw angle optimization by considering sensor field of view and potential risk. Finally, effectiveness of the proposed method is verified through comparative simulation and experiment by considering planning and perceiving ranges.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under grant 61973234, and in part by the Tianjin Natural Science Foundation under grant 20JCYBJC00180.

References

1. Beul, M., Behnke, S. Fast Time-Optimal Avoidance of Moving Obstacles for High-Speed MAV Flight. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2019), Macau, China, November 4-8, 2019, 7240-7247.https://doi.org/10.1109/IROS40897.2019.8968103
2. Campos-Macías, L., Gómez-Gutiérrez, D., Aldana-López, R., de la Guardia, R., Parra-Vilchis, J.I. A Hybrid Method for Online Trajectory Planning of Mobile Robots in Cluttered Environments. IEEE Robotics and Automation Letters, 2017, 4, 935-942. https://doi.org/10.1109/LRA.2017.2655145
3. Choi, J., Curry, R.E., Elkaim, G.H. Curvature-Continuous Trajectory Generation with Corridor Constraint for Autonomous Ground Vehicles. In: Proceedings of the 49th IEEE Conference on Decision and Control, (CDC 2010), Atlanta, GA, USA, December 15-17, 2010, 7166-7171. https://doi.org/10.1109/CDC.2010.5718154
4. Delingette, H., Hebert, M., Ikeuchi. K. Trajectory Generation with Curvature Constraint Based on Energy Minimization. In: Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems ,91, (IROS 1991), Osaka, Japan, 1991, 206-211. https://doi.org/10.1109/IROS.1991.174451
5. Dhullipalla, M.H., Hamrah, R., Sanyal, A.K. Trajectory Generation on SE(3) with Applications to a Class of Underactuated Vehicles. In: Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control , (CDC 2017), Melbourne, VIC, Australia, October 12-15, 2017, 2557-2562. https://doi.org/10.1109/CDC.2017.8264029
6. Escamilla, H., Mora-Camino, F. Generation of Curvature Continuous Trajectories for Transport Aircraft Using Bezier Curves. In: Proceedings of the 19th International Conference on New Trends in Civil Aviation, (NTCA 2017), Prague, Czech Republic, December 7-8, 2017, 7. https://doi.org/10.1201/9781351238649
7. Gammell, J. D., Srinivasa, S. S., Barfoot, T. D. Batch Informed Trees (BIT): Sampling-Based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs. In: Proceedings of the 2015 IEEE International Conference on Robotics and Automation, (ICRA 2015), Seattle, WA, USA, May 26-30, 2015, 3067-3074. https://doi.org/10.1109/ICRA.2015.7139620
8. Gammell, J.D., Barfoot, T.D., Srinivasa, S. S. Informed Sampling for Asymptotically Optimal Path Planning. IEEE Transactions on Robotics, 2018, 34, 966-984. https://doi.org/10.1109/TRO.2018.2830331
9. Gao, F., Lin, Y., Shen, S. Gradient-Based Online Safe Trajectory Generation for Quadrotor Flight in Complex Environments. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2017), Vancouver, BC, Canada, September 24-28, 2017, 3681-3688. https://doi.org/10.1109/IROS.2017.8206214
10. Gao, F., Wang, L., Zhou, B., Zhou, X., Pan, J., Shen, S. Teach-Repeat-Replan: A Complete and Robust System for Aggressive Flight in Complex Environments. IEEE Transactions on Robotics, 2020, 36, 1526-1545. https://doi.org/10.1109/TRO.2020.2993215
11. Guarino, C., Bianco, L., Gerelli, O. Generation of Paths with Minimum Curvature Derivative with η3-Splines. IEEE Transactions on Automation Science and Engineering, 2010, 7, 249-256. https://doi.org/10.1109/TASE.2009.2023206
12. Hauser, K. Lazy Collision Checking in Asymptotically-Optimal Motion Planning. In: Proceedings of the 2015

IEEE International Conference on Robotics and Automation, (ICRA 2015), Seattle, WA, USA, May 26-30, 2015, 2951-2957. https://doi.org/10.1109/ICRA.2015.7139603

13. Jaillet L., Cortes, J., Simeon, T. Transition-Based RRT for Path Planning in Continuous Cost Spaces. In: Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, September 22-26, 2008, 2145-2150. https://doi.org/10.1109/IROS.2008.4650993

14. Jaillet, L., Hoffman, J., van den Berg, J., Abbeel, P., Porta, J. M., Goldberg, K. EG-RRT: Environment-Guided Random Trees for Kinodynamic Motion Planning with Uncertainty and Obstacles. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, September 25-30, 2011, 2646-2652. https://doi.org/10.1109/IROS.2011.6094802

15. Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., Teller, S. Anytime Motion Planning Using the RRT*. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, (ICRA 2011), Shanghai, China, May 9-13, 2011, 1478-1483. https://doi.org/10.1109/ICRA.2011.5980479

16. Ko, I., Kim, B., Park, F. Randomized Path Planning on Vector Fields. International Journal of Robotics Research, 2014, 33, 1664-1682. https://doi.org/10.1177/0278364914545812

17. Kunz, T., Thomaz, A., Christensen, H. Hierarchical Rejection Sampling for Informed Kinodynamic Planning in High-Dimensional Spaces. In: Proceedings of the 2016 IEEE International Conference on Robotics and Automation, (ICRA 2016), Stockholm, Sweden, May 16-21, 2016, 89-96. https://doi.org/10.1109/ICRA.2016.7487120

18. Lai, T., Ramos, F., Francis, G. Balancing Global Exploration and Local-Connectivity Exploitation with Rapidly-Exploring Random Disjointed-Trees. In: Proceedings of the 2019 International Conference on Robotics and Automation (ICRA 2019), Montreal, QC, Canada, 20-24 May 2019, 5537-5543. https://doi.org/10.1109/ICRA.2019.8793618

19. Lewis, A.S., Overton, M.L. Nonsmooth Optimization via Quasi-Newton Methods. Mathematical Programming, 2013, 141, 135-163. https://doi.org/10.1007/s10107-012-0514-2

20. Liu S., Atanasov N., Mohta K., Kumar V. Search-Based Motion Planning for Quadrotors Using Linear Quadratic Minimum Time Control. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2017), Vancouver, BC, Canada, September 24-28, 2017, 2872-2879. https://doi.org/10.1109/IROS.2017.8206119

21. Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C. J., Kumar, V. Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments. IEEE Robotics and Automation Letters, 2017, 2, 1688-1695. https://doi.org/10.1109/LRA.2017.2663526

22. Lopez, B. T., How J. P. Aggressive Collision Avoidance with Limited Field-of-View Sensing. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2017), Vancouver, BC, Canada, September 24-28, 2017, 1358-1365. https://doi.org/10.1109/IROS.2017.8202314

23. Lopez, B. T., How, J. P. Aggressive 3-D Collision Avoidance for High-Speed Navigation. In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation, (ICRA 2017), Singapore, May 29 - June 3, 2017, 5759-5765. https://doi.org/10.1109/ICRA.2017.7989677

24. Mellinger, D., Kumar, V. Minimum Snap Trajectory Generation and Control for Quadrotors. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, (ICRA 2011), Shanghai, China, May 9-10, 2011, 2520-2525. https://doi.org/10.1109/ICRA.2011.5980409

25. Neto, A. A., Macharet, D. G., Campos, M. F. M. Feasible RRT-Based Path Planning Using Seventh Order Bézier Curves. In: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, China, October 18-22, 2010, 1445-1450. https://doi.org/10.1109/IROS.2010.5649145

26. Nieuwenhuisen, M., Behnke, S. Local Multiresolution Trajectory Optimization for Micro Aerial Vehicles Employing Continuous Curvature Transitions. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2016), Daejeon, Korea, October 9-14, 2016, 3219-3224. https://doi.org/10.1109/IROS.2016.7759497

27. Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., Galceran, E. Continuous-Time Trajectory Optimization for Online UAV Replanning. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2016), Daejeon, Korea, October 9-14, 2016, 5332-5339. https://doi.org/10.1109/IROS.2016.7759784

28. Preiss, J., Hausman, K., Sukhatme, G., Weiss, S. Trajectory Optimization for Self-Calibration and Navigation. In: Proceedings of the Robotics: Science and Systems,

(RSS 2017), Cambridge, MA, USA, July 12-16, 2017. https://doi.org/10.15607/RSS.2017.XIII.054

29. Salzman, O., Halperin, D. Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning. IEEE Transactions on Robotics, 2016, 32, 473-483. https://doi.org/10.1109/TRO.2016.2539377

30. Tordesillas, J., Lopez, B. T., Everett, M., How, J. P. FASTER: Fast and Safe Trajectory Planner for Navigation in Unknown Environments. IEEE Transactions on Robotics, 2022, 38, 922-938. https://doi.org/10.1109/TRO.2021.3100142

31. Wang, K., Gao, F., Shen S. Real-Time Scalable Dense Surfel Mapping. In: Proceedings of the 2019 International Conference on Robotics and Automation, (ICRA 2019), Montreal, QC, Canada, May 20-24, 2019, 6919-6925. https://doi.org/10.1109/ICRA.2019.8794101

32. Watterson, M., Liu, S., Sun, K., Smith, T., Kumar, V. Trajectory Optimization on Manifolds with Applications to SO(3) and R3×S2. In: Proceedings of the Robotics: Science and Systems, (RSS 2018), Pittsburgh, Pennsylvania, USA, June 26-30, 2018. https://doi.org/10.15607/RSS.2018.XIV.023

33. Watterson, M., Smith, T., Kumar, V. Smooth Trajectory Generation on SE(3) for a Free Flying Space Robot. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2016), Daejeon, Korea, October 9-14, 2016, 5459-5466. https://doi.org/10.1109/IROS.2016.7759803

34. Webb, D.J., van den Berg, J. Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics. In: Proceedings of the 2013 IEEE International Conference on Robotics and Automation, (ICRA 2013), Karlsruhe, Germany, May 6-10, 2013, 5054-5061. https://doi.org/10.1109/ICRA.2013.6631299

35. Yang, L., Song, D., Xiao, J., Han, J., Yang, L., Cao, Y. Generation of Dynamically Feasible and Collision-Free Trajectory by Applying Six-Order Bezier Curve and Local Optimal Reshaping. In: Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2015), Hamburg, Germany, September 28 - October 02, 2015, 643-648. https://doi.org/10.1109/IROS.2015.7353440

36. Ye, H., Zhou, X., Wang, Z., Xu, C., Chu, J., Gao, F. TGK-Planner: An Efficient Topology Guided Kinodynamic Planner for Autonomous Quadrotors. IEEE Robotics and Automation Letters, 2021, 6, 494-501. https://doi.org/10.1109/LRA.2020.3047798

37. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S. Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight. IEEE Robotics and Automation Letters, 2019, 4, 3529-3536. https://doi.org/10.1109/LRA.2019.2927938

38. Zhou, B., Pan, J., Gao, F., Shen, S. RAPTOR: Robust and Perception-Aware Trajectory Replanning for Quadrotor Fast Flight. IEEE Transactions on Robotics, 2021, 37, 1992-2009. https://doi.org/10.1109/TRO.2021.3071527

39. Zhou, X., Wang, Z., Ye, H., Xu, C., Gao, F. Ego-Planner: An ESDF Free Gradient-Based Local Planner for Quadrotors. IEEE Robotics and Automation Letters, 2021, 6, 478-485. https://doi.org/10.1109/LRA.2020.3047728

40. Zhou, X., Zhu, J., Zhou, H., Xu, C., Gao, F. EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments. In: Proceedings of the 2021 IEEE International Conference on Robotics and Automation, (ICRA 2021), Xi‘an, China, May 30 - June 5, 2021, 4101-4107. https://doi.org/10.1109/ICRA48506.2021.9561902