Contents lists available at YXpublications

International Journal of Applied Mathematics in Control Engineering

Journal homepage: http://www.ijamce.com

Safety Mechanism-based Local Trajectory Optimization in Real-time for Quadrotor Fast Flight

Jie Tang, Xingguo Song*, Qiu Hou, Lulu Gong, Yue Zan

Mechanical Engineering School, Southwest Jiaotong University, Chengdu 610031, China

ARTICLE INFO

Article history:
Received 20 January 2023
Accepted 25 February 2023
Available online 2 March 2023

Keywords:
Safety mechanism
Trajectory optimization
Gradient descent
Quadrotor navigation

ABSTRACT

Trajectory planning has been widely applied for autonomous quadrotors navigation in unstructured environments. However, when faced with unexplored environments, trajectory replanning is difficult to ensure the safety of quadrotors fast flight since the limitations of kinodynamic feasibility. In this paper, a safety mechanism-based local trajectory optimization (SMLTO) is proposed to replan a safe and kinodynamic feasible trajectory in real-time, by adjusting the position of local control points for occlusion effect. Expansion radius of the obstacle is calculated by extracting the center of point clouds of obstacle, which is utilized to predict the distances between control points and obstructive surface. When the distance fails to reach the set safety threshold, the gradient descent method will be used to calculate the adjustment direction of control points. The distance from adjusted control points to the obstacle surface is not less than safety threshold. Each control point can be adjusted to conservative position with safety distance through the proposed SMLTO method, even if quadrotor will face some unforeseeable obstacles. The proposed method is validated extensively through benchmark comparisons in multifold challenging simulation environments.

1. Introduction

Quadrotor is playing an increasingly important role in contemporary society, and is increasingly used in a wide variety of fields, including terrain survey and cargo delivery. The quadrotor has the ability to recognize surrounding obstacles and generate reliable collision-free trajectory is a fundamental guarantee to accomplish the autonomous tasks. Although a lot of work has been performed on trajectory generation and autonomous navigation of quadrotor in unknown environments, however, many problems still exist.

First of all, the quadrotor will not fly exactly according to the pre-planned trajectory due to external factors, which will lead to a collision between the quadrotor and the obstacle when the quadrotor passes the obstacle at a certain moment. On the other hand, the fast flight speed of the quadrotor makes it difficult to re-path planning timely to avoid obstacles when the quadrotor suddenly encounters obstacle during flight (e.g. obstacles obscured by walls and dynamic obstacles, etc.), which eventually leads to the failure of quadrotor trajectory planning and navigation or even crashing.

In this paper, we propose a trajectory control point adjustment algorithm to obtain a safer flight trajectory. The algorithm calculates the distance between the initial trajectory control point and the corresponding obstacle, and adjusts the initial trajectory control point according to the distance size, so that the planned trajectory can keep relatively farther distance away from the obstacle.

A novel objective function is introduced in this algorithm to limit the minimum distance between the trajectory control point and the obstacle, and the gradient descent algorithm is adopted to adjust the trajectory control point to the most suitable position, i.e., the distance between the trajectory control point and the obstacle needs to satisfy the given safety threshold as much as possible. The trajectory adjusted by the algorithm eliminates potential risks and the quadrotor is able to maintain a relatively safe distance from the obstacle when it encounters obstacle. The relatively large distance gives the quadrotor more time to re-trajectory planning when encountering obscured obstacles and dynamic obstacles, which makes the quadrotor fast flight process gets better safety assurance.

Compared to the already existing work, our proposed method is

* Corresponding author.
E-mail addresses: songxg@swjtu.edu.cn (Xingguo Song*)
doi:123456

able to generate safer and more stable trajectory in various complex environments. We have tested the robustness and security of our proposed algorithm in many simulation environments with satisfactory results.

2. Related works

2.1 Hierarchical Motion Planning

The implement of flight corridors to limit the range of generated trajectory has been extensively researched in the field of trajectory planning. Continuous large overlapping grids are taken in unknown space as free space and feasible trajectory is found in a series of connected three-dimensional cubical grids[1]. The set of convex overlapping polyhedra is defined as a flight corridor, and then polynomial trajectory is generated in the flight corridor using a quadratic programming approach[2]. Some methods construct flight corridors directly on point cloud data as free space and employ quadratically constrained quadratic programming methods[3] to generate collision-free trajectories, as well as trajectories being represented as piecewise Bézier curves[4] to satisfy the dynamic feasibility of the trajectory. A k-order function of time is used to express the trajectory, which then is optimized by solving a quadratic programming problem to generate the optimal trajectory[5]. The existing polynomial trajectory generation method is extended to guarantee trajectory safety by adding intermediate path points[6], however, this method increases the computational effort and lacks the guarantee of a globally optimal trajectory. Topological path algorithm is proposed in [7] to explore the three-dimensional space more completely and finds the relatively optimal initial path. The algorithm mentioned above relies heavily on the time allocation, and an improper time allocation may significantly degrade the quality of the trajectory. In order to achieve a better time allocation, a ESDF-induced velocity field is utilized to search for trajectory directly[8].

2.2 Kinodynamic Motion Planning

Finding a dynamically feasible initial trajectory and adjusting local control points is also quite effective in the trajectory planning process.

A b-spline curve is utilized to represent the initial trajectory to resolve the dynamic infeasibility caused by non-static motion, and an elastic optimization method is presented to optimize the trajectory discretization[9]. The shortest trajectory search algorithm is converted to a dynamic trajectory search algorithm that a more reasonable initial trajectory is generated, and an elastic optimization algorithm is adopted to address trajectory dispersion[10]. A b-spline-based trajectory search algorithm (BNUK) is proposed to generate dynamically feasible initial trajectory, and then the trajectory is refined by utilizing the proposed control point optimization method to improve the smoothness of the trajectory[11]. An initial path in 3D space based on a non-uniform b-spline curve representation can be generated by using the kinodynamic path searching method[12], and an iterative time-adjustment algorithm is adopted for time allocation of paths to improve the feasibility of dynamic trajectory.

Methods based on searching and sampling are often employed to search for dynamically feasible trajectories. A search-based planning method is proposed to compute dynamically feasible trajectories for a quadrotor flying in an obstacle-cluttered environment by exploiting the explicit solution of a linear quadratic minimum time problem[13]. A set of short-time motion primitives is utilized to explore the unknown to find smooth as well as least time-consuming trajectory[14]. The algorithms PRM* and RRT* proposed in [15] guarantee asymptotic optimality and less computational complexity than the traditional sampling-based path search algorithms PRM and RRT. The kinodynamic RRT* algorithm proposed in [16] has been extended with work on RRT*, which guarantees the asymptotic optimality of the algorithm for any controllable linear system with minimal computational overhead.

Some methods add collisionality and smoothness of the trajectory to the cost function and optimize the trajectory by adopting gradient optimization method.

The CHOMP algorithm is proposed to optimize the generated trajectory using the covariance gradient technique[17], but which tends to get stuck with local minima. The cost function is optimized during the iterative process to avoid local minima[18], but the computational effort is relatively larger. A sampling method is adopted to search for a collision-free initial safe trajectory[19], and then refines the trajectory with the gradient information of the smoothness of the trajectory to obtain a dynamically feasible trajectory. An ESDF-free gradient-based planning algorithm is proposed, which reduces computation time in planning the collision-free trajectory, and lengthens the time allocation for dynamical feasibility[20].

The hierarchical motion planning approach cannot meet the dynamics of quadrotor flight in some cases and places great importance on the time allocation of the second stage. The trajectory searched by using kinodynamic motion can meet the dynamic feasibility, however the computation is relatively large, and the gradient-based optimization method for local trajectory is easy to get stuck with local minima.

3. Problem Description

When flying fast in an unexplored or partially unexplored environment, the quadrotor cannot see the obscured space in advance since the camera has no vision of the back of the obstacle. So there is a significant safety risk, if quadrotor follows the initial reference trajectory quickly by treating all unknown space as free. To avoid the newly found obstacles, quadrotor needs plan local trajectories in real-time. EGO-Planner[20] is an efficient and robust local trajectory planning algorithm which adjusts the collision trajectory to free space and reduces computation time significantly. However, EGO-Planner cannot handle dynamic obstacles and leaves less safety margin for unknown views, and the quadrotor failed to detect new obstacles in time under the current sensor field of view. When quadrotor enters an unknown space and detects a new obstacle, the original planned trajectory collides with obstacles (the blue trajectory as in Fig.1). And under EGO-Planner, the replanned trajectory is dynamically infeasible due to too large a change in yaw angle rate, making it more risky for the quadrotor to follow the replanned trajectory (the red curve replanned in Fig.1(a)), resulting in a greater risk of collision with sudden obstacle. The brown trajectory in Fig.1(b) is intentionally kept away from obstacles, which allows the quadrotor to reach the yellow control point on the brown trajectory with a greater safety margin for the unknown environment, and results in a low rate of change in yaw angle for the replanned trajectory, which ensures dynamic feasibility(the green curve replanned in Fig.1(b)).

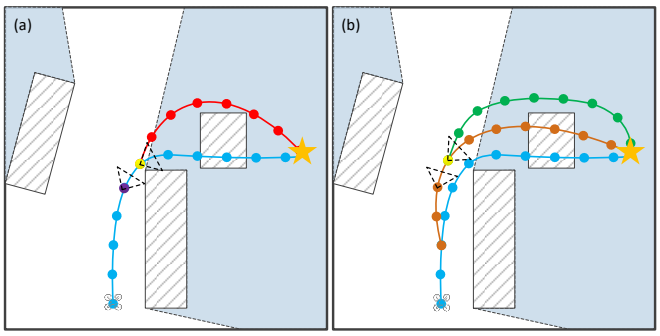


Fig. 1. (a) Quadrotor generates real-time trajectories based on EGO-Planner algorithm. (b) Quadrotor generates real-time trajectories based on our algorithm. The blue curve is the initial trajectory, the shaded part is the blind area of sensor field of view.

4. Safety Mechanism-Based Local Trajectory Optimization

4.1 SMLTO Overview

The Safety mechanism-based local trajectory optimization algorithm is proposed to improve the local trajectory generated by EGO-Planner[20]. When a quadrotor flies fast in an unknown environment, a large potential risk exists if there is no relative safety margin between the planned trajectory and the obstacles. The idea of proposed algorithm is to adjust the quadrotor trajectory control point to a certain threshold distance away from the surface of the obstacle to guarantee the safety of the fast flight process, as shown in Algorithm 1.

Algorithm 1 Control point adjustment

Input: Enter the initial control point $pt = \{pt_1, pt_2, ...pt_n\}$

Output: Control points after adjustment

```
for pt_i = pt_1 → pt_n do
    for obs_i = obs_1 → obs_n do
        d_i ⇐ ||pt_i − obs_i||^2
        if d_i < d_imin then
            d_imin = d_i
        end if
    end for
    d_min.push_back(d_imin)
end for
for i = 0, ..., len(d_min) do
    if d_i >= threshold then
        adjust_point.push_back((x_i, y_i, z_i))
        return
    end if
    x_i = x_i − η * ∂cost/∂x_i
    y_i = y_i − η * ∂cost/∂y_i
    z_i = z_i − η * ∂cost/∂z_i
    adjust_point.push_back((x_i, y_i, z_i))
end for
```

The algorithm begins with the reception of a set of control points $pt = \{pt_1, pt_2, \ldots, pt_n\}$.

And then the distance d_i between each control point pt_i of the trajectory and the surrounding obstacles obs_i is calculated and the distance d_{imin} between each control point and its nearest obstacle is figured out. The distances between a set of control points and their corresponding nearest obstacles are stored into the container d_{min}. And the d_i in d_{min} is traversed to judge the size relationship between d_i and the distance threshold set in advance. If the closest distance between the control point and the surface of the obstacle is greater than or equal to the given threshold, the relevant trajectory control point is directly pushed into the adjusted container *adjust_point*. Conversely, the gradient $\partial cost / \partial \mu$ of the objective function cost with respect to the coordinates of the control points is calculated respectively. And formula (1) is adopted to adjust and optimize the coordinates of control points.

$$\mu = \mu - \eta \cdot \partial cost / \partial \mu \tag{1}$$

where $\mu \in \{x, y, z\}$. The algorithm does not terminate until the unsafe trajectory control point is adjusted to the appropriate position.

4.2 Obstacle Center Extraction Method and Obstacle Expansion Radius

It is crucial to obtain the obstacle center coordinates for our proposed algorithm, and we adopt the strategy of fitting the irregular obstacle information into a regular cube to obtain the obstacle center coordinates and the expansion radius. The depth camera on the quadrotor can obtain the obstacle information directly in the real environment. We utilize PCL standard library to cluster the point cloud information obtained by the depth camera, and traverse each clustered point cloud cluster to find the maximum X,Y,Z value and the minimum X,Y,Z value in the point cloud coordinates respectively, and we adopt the difference between the minimum coordinate value and the maximum coordinate value as the length, width and height of the obstacle to be fitted to form a cubic obstacle, and then obtain the center coordinates of the obstacle and the expansion radius *obs_r*. However, if the length, width and height of the fitted obstacle are large, as in Fig.2(a), using the geometric center of the fitted rectangle to calculate the distance to the trajectory control point (red line segment in Fig.2(a)), the gap with the actual distance (green line segment in Fig.2(a)) will be large, resulting in inaccurate judgment with respect to the safety threshold. Therefore, we propose a method where we will split larger obstacles into smaller ones to obtain more geometric centers, and the calculated distance between the trajectory control point and the obstacles is very close to the real distance, which makes the judgment relationship between the calculated distance and the safety threshold more accurate, such as the green and red line segments in Fig.2(b). The blue dashed circle in Fig.2(c) represents the obstacle expansion area.

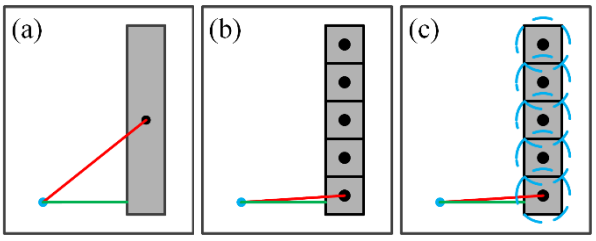


Fig. 2. (a) The error between the Euclidean distance calculated by the huge obstacle and the actual distance is large. (b)The huge obstacle is divided into small obstacles. (c) The obstacles are expanded into a sphere.

4.3 Calculating the Distances Between Trajectory Control Points and Obstacles

First, the distance between the initial control point and the obstacle surface should be calculated to determine whether the corresponding control point needs to be adjusted. For a safer fast flight, we adopt the strategy of drawing a sphere that can wrap the obstacle as the actual obstacle range(blue dashed circles as in Fig. 2(c)) to ensure that the distance from the control point to the surface of the obstacle is greater than the given safety threshold regardless of the relative position of the control point and the obstacle. The distance between the control point and the center of the obstacle is defined as *d_cal* and is calculated as in (2).

$$d\_cal = \sqrt{(x_{obs} - x_{cpt})^2 + (y_{obs} - y_{cpt})^2 + (z_{obs} - z_{cpt})^2} \quad (2)$$

where x_{obs}, y_{obs}, z_{obs} and x_{cpt}, y_{cpt}, z_{cpt} represent the 3D coordinates of the obstacle and the trajectory control point, respectively. And the actual distance d_{fct} between the control point and the obstacle surface is calculated according to Equation (3), which is used to determine the relationship with the safety threshold.

$$d_{fct} = d\_cal - obs\_r \quad (3)$$

where *obs_r* indicates the radius of the obstacle sphere and control points need to be adjusted only in the case of its corresponding d_{fct} is less than the safety threshold.

On the other hand, during the process of distance adjustment between the control point and the obstacle, it is necessary to obtain the actual distance between the control point being adjusted and the obstacle. When the Euclidean distance adjusted between the control point and the obstacle is equal to the safe threshold, the distance from the control point on the local trajectory in orange to the obstacle surface does not reach the safety threshold(d_1 and d_2 in Fig.3(a)). In order to avoid such situations, an algorithm is proposed to calculate the actual distance and the main thought of the algorithm is shown in Fig.3(b).

The algorithm defines a triangle with two trajectory control points and an obstacle point, fixes the obstacle point as well as the previous trajectory control point, adjusts the latter trajectory control point so that the distance between the latter trajectory control point and the surface of the obstacle satisfies the given condition.

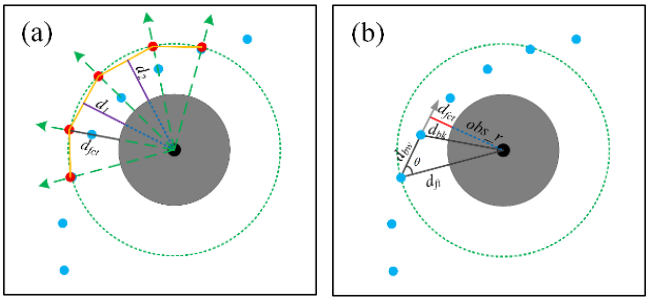


Fig. 3. (a) Define the Euclidean distance between the control point and the surface of the obstacle as the actual distance. (b) Define the distance between the obstacle and the straight line determined by the two control points as the actual distance.

Based on the relationship between the lengths of the three sides, the interior angle (the angle between d_{bw} and d_{ft}) in the triangle of Fig.3(b) is calculated using the cosine theorem as in (4).

$$\theta = \arccos\left(\frac{d_{bw}^{\ 2} + d_{ft}^{\ 2} - d_{bk}^{\ 2}}{2 \times d_{bw} \times d_{ft}}\right) \quad (4)$$

where θ represents the quadrotor flight direction, and the angle will gradually increase during the real-time adjustment process, which allows the quadrotor to adjust the flight direction in real-time during the fast flight and obtain more information about the unknown environment in advance.

The algorithm guarantees that the distance between each point on the actual trajectory and the surface of obstacle is greater than the safety threshold. We employ the strategy that the direction of the connecting line of two adjacent trajectory control points is the velocity direction, as the grey arrow in Fig.3(b). In the vertical line of velocity direction crossing the obstacle center, the distance from the surface of the obstacle to the connecting line is defined as the actual distance d_{fct} (the red line segment in Fig.3(b)), the calculation formula is shown as (5). Adopting this strategy can effectively guarantee that the minimum distance between the straight line formed by the two control points and the surface of the obstacle is larger than the given safety threshold.

$$d_{fct} = d_{ft} \cdot \sin\theta - obs\_r \quad (5)$$

4.4 Objective Function Design

The objective function we designed as follows:

$$cost(d_{fct}) = \begin{cases} \frac{1}{2}(d_{fct} - d_{thres})^2 & d_{fct} < d_{thres} \\ 0 & d_{fct} \geq d_{thres} \end{cases} \quad (6)$$

where d_{fct} represents the actual distance between the trajectory control point and the obstacle, and d_{thres} represents the given safety threshold.

A relatively large gradient is obtained when the actual distance differs significantly from the safety threshold, a larger value of gradient enables the inappropriate control points to iterate quickly close to the optimal control point position. The gradient is decreasing during the adjustment process, and the smaller gradient can prevent the control point from oscillating near the optimal control point, which makes the trajectory control point better for iterating and gaining the optimal solution. This function does not have a local optimum and enables the control points to converge to the global optimum solution quickly and satisfy real-time performance.

5. Experimental Details

5.1 Simulation experiment setup

The trajectory control point adjustment algorithm proposed in this paper is implemented based on C++11 standard, and in each group of comparison experiments, the parameters of the adjusted algorithm are kept consistent with all parameters of the original algorithm, including map resolution, obstacle expansion coefficient, quadrotor flight speed and acceleration. Various simulation conditions are also kept the same, including initial takeoff position, given target point position, camera internal parameters and quadrotor controller, etc. We adopt the physical simulation platform (Gazebo) as the simulation environment, and the robot operating system (ROS) is utilized for communication between the components during the simulation. The simulation quadrotor uses the official open source firmware provided by PX4. All simulations and algorithms are run on a laptop with an octa-core 2.3GHz i7-10875H processor and RTX2060 graphics card. In the next section, we will set up different

simulation environments for verifying the security of our proposed algorithms.

5.2 Simulation experiments

In scene one, a 1×1×5m rectangular obstacle is placed behind the corner of the wall. Our algorithm allows the quadrotor to reach the red dot in Fig.4(a) at a greater distance from the obstacle in the corner. When the quadrotor encounters the obstacle behind the wall corner, the longer safety distance enables the quadrotor to obtain more reaction time which guarantees the success rate of trajectory replanning in high-speed flight situations. In the original algorithm, when the quadrotor reaches the red dot in Fig.4(b), the distance to the obstacle behind the corner is close and due to the inertia of the high-speed flying quadrotor, the quadrotor collides with the obstacle, resulting in planning failure. Compared with Fig.4(d), the trajectory planned by our algorithm (Fig.4(c)) will leave more safety margin for the unknown area.

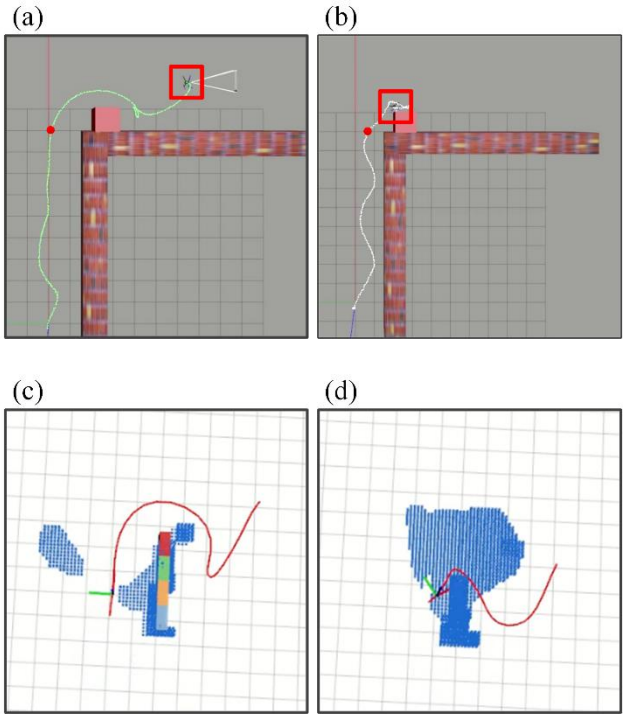


Fig. 4. The actual flight trajectory of the quadrotor in scene 1. (a) Flight trajectory under our proposed algorithm. (b) Flight trajectory under the EGO-Planner. (c) The local trajectory replanned by our algorithm. (d) The local trajectory replanned by the EGO-Planner.

In scene two, we simulated the situation of a real corridor, and the distance between corridors is set to 3 meters. When flying in the simulation corridor under our proposed algorithm, if the distance from the quadrotor to obstacles on both sides cannot meet the given safety threshold, our algorithm will plan the trajectory in the middle of the corridor, such as the yellow elliptical curve circle in Fig.5(c). Flying in the middle of the two obstacles, so that the quadrotor can ensure more reliable flight during the flight in a narrow environment, and there will not be a situation where the planning trajectory cannot be accurately executed due to the influence of external factors leading to collision with obstacles, such as the quadrotor in Fig.5(b), the quadrotor is close to the corner of the wall and collides with the red obstacle. On the other hand, flying between two obstacles, the quadrotor is able to find the obstacles behind the corners altogether easily, ensuring a safer flight, as in Fig.5(a). The trajectory planned by the original algorithm lacks spatial optimality, as Fig.5(d).

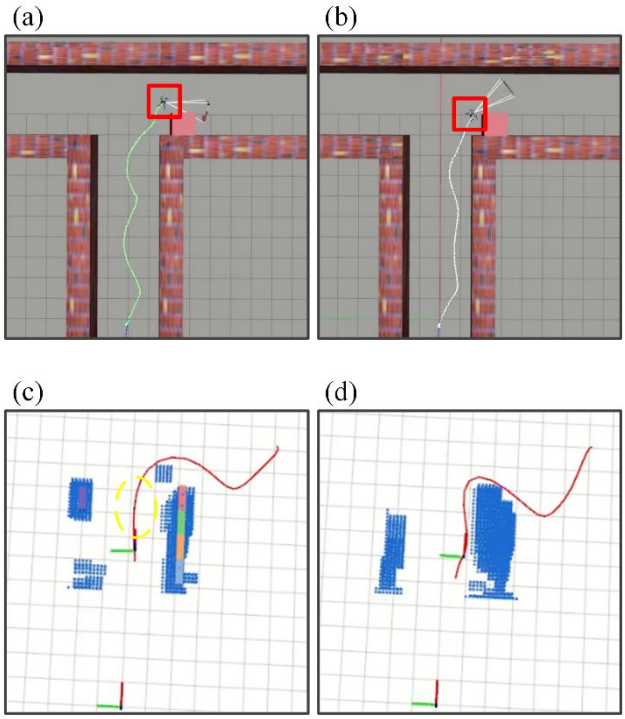


Fig. 5. The actual flight trajectory of the quadrotor in scene 2. (a),(c) Flight trajectory under our proposed algorithm. (b),(d) Flight trajectory under the EGO-Planner. (e) The local trajectory replanned by our algorithm. (f) The local trajectory replanned by the EGO-Planner.

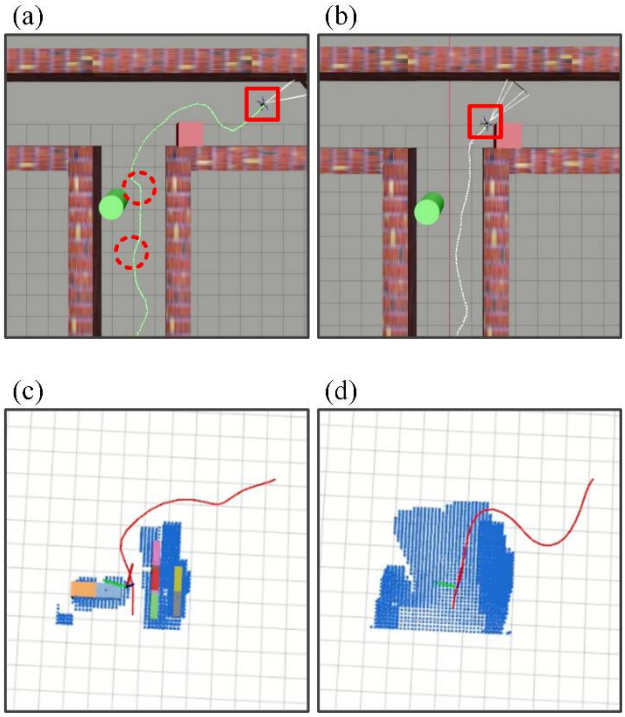


Fig. 6. The actual flight trajectory of the quadrotor in scene 3. (a) Flight trajectory under our proposed algorithm. (b) Flight trajectory under the EGO-Planner. (c) The local trajectory replanned by our algorithm. (d) The local trajectory replanned by the EGO-Planner.

The scene three is shown in Fig.6, the quadrotor will initially fly between two walls, and when new obstacles appear in front of the flight and the flight corridor becomes narrower (i.e., the horizontal

distance between the green cylinder and the right wall in Fig.6(a)), the quadrotor will automatically fly at the safest position according to the distance from the obstacles on both sides, as the red dashed circle in Fig.6(a). After passing the narrow corridor, due to the risk of unknown obstacles behind the corner of the wall, the quadrotor will take the initiative to move away from the corner so that it can get a larger field of view as well as a longer braking distance during the turn, and can discover the obstacles behind the corner in advance to ensure the safe flight. On the contrary, if the distance from the corner is too small when turning, which will lead to the quadrotor collide with the obstacles in the unknown field of view, as in Fig.6(b). The actual planned trajectory of the two algorithms are shown in Fig.6(c)(d), and the trajectory planned by our algorithm (Fig.6(c)) is safety conscious for the unknown environment compared to the original algorithm (Fig.6(d)).

6. Conclusion

In this paper, we propose a safety mechanism-based local trajectory optimization (SMLTO) method for quadrotor autonomous navigation. The method plans a conservative and safe trajectory with fast inference speed. We adopt EGO-Planner to find an initial trajectory and calculate the expansion radius of the obstacle by extracting the center of point clouds with vision sensor. The distances between trajectory points and obstructive surface will be estimated. To avoid these distances are less than the desired safety threshold, local trajectory is further adjusted by a gradient-based optimization. Finally, all trajectory points of quadrotor will be conservative with safety distance, even if quadrotor immediately enters an unknown space. The proposed method is validated through benchmark comparisons in various complex environments and the simulation.

References

[1] Chen J, Liu T, Shen S. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments[C]//2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016: 1476-1483.

[2] Liu S, Watterson M, Mohta K, et al. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments[J]. IEEE Robotics and Automation Letters, 2017, 2(3): 1688-1695.

[3] Gao F, Shen S. Online quadrotor trajectory generation and autonomous navigation on point clouds[C]//2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2016: 139-146.

[4] Gao F, Wu W, Gao W, et al. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments[J]. Journal of Field Robotics, 2019, 36(4): 710-733.

[5] Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors[C]//2011 IEEE international conference on robotics and automation. IEEE, 2011: 2520-2525.

[6] Richter C, Bry A, Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments[M]//Robotics research. Springer, Cham, 2016: 649-666.

[7] Zhou B, Gao F, Pan J, et al. Robust real-time uav replanning using guided gradient-based optimization and topological paths[C]//2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020: 1208-1214.

[8] Gao F, Wu W, Lin Y, et al. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 344-351.

[9] Ding W, Gao W, Wang K, et al. An efficient b-spline-based kinodynamic replanning framework for quadrotors[J]. IEEE Transactions on Robotics, 2019, 35(6): 1287-1306.

[10] Ding W, Gao W, Wang K, et al. Trajectory replanning for quadrotors using kinodynamic search and elastic optimization[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 7595-7602.

[11] Tang L, Wang H, Liu Z, et al. A real-time quadrotor trajectory planning framework based on B‐spline and nonuniform kinodynamic search[J]. Journal of Field Robotics, 2021, 38(3): 452-475.

[12] Zhou B, Gao F, Wang L, et al. Robust and efficient quadrotor trajectory generation for fast autonomous flight[J]. IEEE Robotics and Automation Letters, 2019, 4(4): 3529-3536.

[13] Liu S, Atanasov N, Mohta K, et al. Search-based motion planning for quadrotors using linear quadratic minimum time control[C]//2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 2872-2879.

[14] Mueller M W, Hehn M, D'Andrea R. A computationally efficient motion primitive for quadrocopter trajectory generation[J]. IEEE transactions on robotics, 2015, 31(6): 1294-1310.

[15] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning[J]. The international journal of robotics research, 2011, 30(7): 846-894.

[16] Webb D J, Van Den Berg J. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics[C]//2013 IEEE international conference on robotics and automation. IEEE, 2013: 5054-5061.

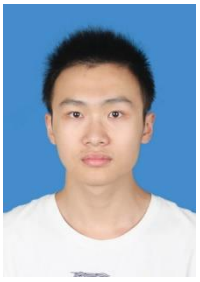
[17] Ratliff N, Zucker M, Bagnell J A, et al. CHOMP: Gradient optimization techniques for efficient motion planning[C]//2009 IEEE International Conference on Robotics and Automation. IEEE, 2009: 489-494.

[18] Kalakrishnan M, Chitta S, Theodorou E, et al. STOMP: Stochastic trajectory optimization for motion planning[C]//2011 IEEE international conference on robotics and automation. IEEE, 2011: 4569-4574.

[19] Gao F, Lin Y, Shen S. Gradient-based online safe trajectory generation for quadrotor flight in complex environments[C]//2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 3681-3688.

[20] Zhou X, Wang Z, Ye H, et al. Ego-planner: An esdf-free gradient-based local planner for quadrotors[J]. IEEE Robotics and Automation Letters, 2020, 6(2): 478-485.

Xingguo Song, Ph.D., graduated from Harbin Institute of Technology, School of Mechanical and Electrical Engineering, majoring in Mechanical Design and Theory, is a visiting scholar at Rice University and a postdoctoral fellow at Johns Hopkins University, USA. His main research interests are intelligent robotics, UAV path planning, bionic robotics, and computer vision.



Jie Tang is currently a graduate student in the School of Mechanical Engineering at Southwest Jiaotong University and his research interests are in UAV control and UAV path planning.

Qiu Hou is currently studying in the School of Mechanical Engineering at Southwest Jiaotong University, and his main research interests are autonomous exploration and map building of UAV in unknown environments.

Lulu Gong is currently studying at the School of Mechanical Engineering at Southwest Jiaotong University, and his research interest is autonomous UAV landing technology.

Yue Zan is a master student and currently studying in the School of Mechanical Engineering, Southwest Jiaotong University, with research interests in the design and multimodal control of variable structure quadrotor UAV.