A Safe Topological Waypoints Searching-Based Conservative Adaptive Motion Planner in Unknown Cluttered Environment

Jiachi Xu[1], Jiefu Tan[1], Chao Xue[2], Yaqianwen Su[2,3], Xionghui He[1], and Yongjun Zhang[2(✉)]

[1] College of Computer, National University of Defense Technology, Changsha 410073, China
jcxu@nudt.edu.cn

[2] Artificial Intelligence Research Center (AIRC), National Innovation Institute of Defense Technology (NIIDT), Beijing 100071, China
yjzhang@nudt.edu.cn

[3] Tianjin Artificial Intelligence Innovation Center, Tianjing 300457, China

Abstract. Autonomous navigation of unmanned aerial vehicles (UAVs) in unknown and complex environments is still a challenge. Because the environment is partially observable to the drone, it is hard to consider trajectory safety and exploration efficiency simultaneously in autonomous navigation. In this paper, we present a motion planning method composed of a geometrically topological waypoints searching method and an adaptive trajectory replanning framework, which improves trajectory safety without sacrificing navigation efficiency. Our waypoint searching approach considers the safety distance and reduces pathfinding's search space by extracting some feasible path points on both sides of the obstacle. And this is based on the ESDF gradient and geometry information of a given obstacle. Besides, the found waypoints keep a safe distance from the obstacles, making the method work well in a scene that contains large obstacles. Based on the waypoint searching method, we proposed an adaptive trajectory replanning framework to improve trajectory safety and navigation efficiency further. The replanning procedure is event-triggered. When the planned trajectory is too close to an obstacle according to our safe condition, the trajectory will be re-planned. The proposed method is tested extensively in various simulation environments. Results show that the trajectory safety of our method is improved by 27.8%, and the computing time for replanning is reduced by 90.8% compared to the state-of-the-art method.

Keywords: Autonomous systems · Unmanned aerial vehicle · Motion planning

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 206–224, 2021.
https://doi.org/10.1007/978-3-030-92638-0_13

1 Introduction

Recently, unmanned aerial vehicles (UAVs) have been widely used in many practical applications such as flying film, search-and-rescue, autonomous inspection. The flexible feature and small size allow them to carry out missions in dangerous or unfavorable areas for humans. In these application scenarios, environmental information is usually unknown and complex. The drones need to frequently re-plan safe and smooth trajectories to pass by unpredicted obstacles, and the motion planning module plays a vitally important role in an autonomous navigation system.

The motion planning module can be decomposed into front-end pathfinding and back-end trajectory optimization [5,13]. The pathfinding part is responsible for searching an obstacle-free path in the low-dimensional discrete space. After that, the path is time parameterized to an executable trajectory in high-dimensional continuous space by the trajectory optimization part. There are many classic methods can generate a high-quality trajectory in seconds [1–4,11,12,15].

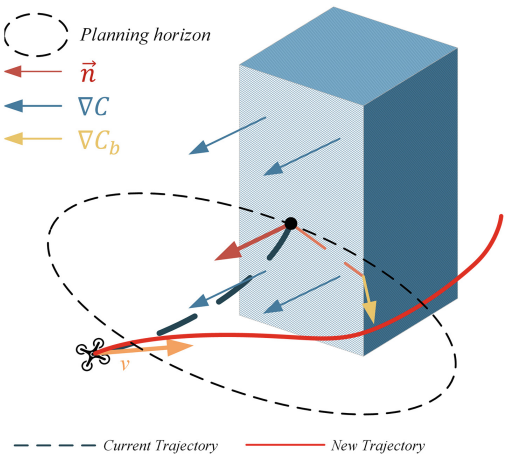


Fig. 1. Illustration of topological waypoints searching. When an impending collision is detected on the current trajectory of the UAV, a new safe trajectory located in the planning horizon is generated.

Plenty of above works are based on optimistic assumptions and use standard pathfinding algorithms, which will bring about the trajectory safety problem. As is well-known, the onboard sensor has a limited field-of-view (FOV) and can only gain information about the nearby environment. In an optimistic motion planning system, the unknown environment is treated as collision-free. This speeds up the trajectory optimization process but may not guarantee the safety of the planned trajectory. On the other hand, the standard pathfinding algorithms,

such as A* and RRT*, search for the shortest optimal path. However, the points on this path are close to obstacles, making the optimized trajectory low safety clearance. Besides, when there are large obstacles in the environment, such methods usually crash and cannot navigate to the target.

In this paper, we proposed a safe motion planning method composed of a geometrically topological waypoints searching method and an adaptive conservative trajectory replanning strategy, which improves trajectory safety and navigation efficiency (displayed in Fig. 1). To guarantee the safety of UAVs, we treat the environment in a conservative way. The unknown environment is regarded as obstacles occupied, and we only allow motions in known-free space. Firstly, instead of using the standard pathfinding algorithm, we propose to combine the Euclidean signed distance field (ESDF) gradient and geometric information of obstacles to search for feasible topological points, which can significantly reduce the search space and improve the exploration efficiency. Secondly, we generate polynomial trajectories from the current state of the UAV to each found topological point parallelly. After that, the one with the least transfer cost is selected as the new global guiding trajectory. Finally, the part of the new trajectory located in known-free space is reparameterized to a smooth, safe, and dynamically feasible B-spline trajectory.

We conduct a lot of comparison experiments in the simulation environment to verify the performance of our proposed motion planning method, and the results show that the trajectory safety and replanning cost of our method is significantly better than the state-of-the-art method. The contributions of this paper are summarized as follows:

- A fast and safe geometrically topological waypoints searching method is proposed to improve the efficiency of searching safe waypoint by combining the geometric properties of obstacles and ESDF map to reduce the search space.
- An adaptive trajectory replanning method based on the conservative estimation is proposed to reduce the computation cost for replanning and further guarantee the trajectory's safety.

2 Related Works

2.1 Standard Pathfinding Algorithm

Pathfinding is the front-end part of the motion planning module, aiming at finding an obstacle-free geometric path in low-dimensional discrete space. The standard pathfinding algorithms can be divided into sampling-based and searching-based. For sampling-based pathfinding algorithms, one of the most representative methods is Rapidly-Exploring Random Tree (RRT) [8]. RRT* [7] is an asymptotically optimal sampling-based method. The solution found by RRT* will converge to the global optimality as the samples increase. However, this method requires prior knowledge of the global map. For searching-based algorithms which usually convert the pathfinding to graph searching problem, the A* [6] is the most widely used method. Many works use standard A* or the

derivative of A* algorithms to search for a safe path to bypass the new-emerged obstacles in case of collision. [1] initializes an obstacle-free flight corridor by A* algorithm and then generated a feasible trajectory within the flight corridor through hard-based optimization method. [20] adopts the hybrid A* algorithm to find a rough time-parameterized path and optimizes it to a refined trajectory in the back-end.

However, the limited sensor FOVs make the environment locally observable, and the path obtained by front-end methods may not be globally optimal. To ensure the safety of the trajectory, frequent search is required, which increases the computational overhead.

2.2 Trajectory Replanning

Local trajectory planning is a mechanism to generate a new safe trajectory when the current trajectory collides with an unpredicted obstacle. Existing methods can be categorized into optimistic and conservative methods. Plenty of methods are based on the optimistic assumption. PGO [19], and TGK-Planner [18] initialize a global trajectory that does not consider collisions as a guideline, and then get the position where the global trajectory goes in and out of obstacles. After that, a new local trajectory that guides the drone to bypass the obstacle is generated. Interest in using B-spline to represent the trajectory was revived by [17], which utilizes the local modification scheme of B-spline curve to achieve real-time local trajectory re-planning.

Another category is conservative local trajectory planning methods, which consider the unknown space as obstacle occupied. [9,10] discretizes the control space and only allows the motion primitives in known-free space. FASTER [16] proposes a local planner to generate a committed trajectory in both known-free and unknown space and a safe back-up trajectory in known-free space. Nevertheless, the former does slow control space discretization and may cause infeasibility, while the latter decomposes the free space to a set of overleaping polyhedral, which is computationally expensive.

Optimistic assumption-based methods simplify the replanning process and improve the chance of reaching target. However, it may not ensure the safety of trajectory, especially in a scene with large obstacles. In contrast, conservative planning methods often require tremendous computation to ensure the trajectory's safety, leading to a decrease in navigation efficiency.

3 Geometrically Topological Waypoints Searching

The standard pathfinding methods search for the shortest optimal path in a large space, which is inefficient. Since the environment is only locally observable, the shortest path is not necessarily optimal. Besides, points on the shortest path are usually too close to the obstacle, which makes the generated trajectory a low clearance and safety, especially in scenarios with large obstacles. We proposed a geometrically topological waypoints searching method to extract some feasible

path points on both sides of the obstacle, which considers the safety distance and reduces searching space.

3.1 Topological Points Searching

Algorithm 1. Topological points searching

function GETTOPOPOINT(cur_{traj})
  $P_{in} \leftarrow$ FINDCOLLISION(cur_{traj})
  Let($\nabla c, \vec{n}$) be the ESDF gradient
    and normal vector in P_{in}
  if $\nabla c \parallel \vec{n}$ **then**
    $P_{topo} \leftarrow$ FINDCORNERPTS($P_{in}, \nabla c$)
  else
    $P_{topo} \leftarrow grad \times d_{thr}$
  if no topoPoint were found before **then**
    $P_{topo} \leftarrow$ PUSHAWAYFROMOBS($P_{in}, \nabla c, d_{thr}$)
  return P_{topo}
function FINDCOLLISION(cur_{traj})
  for P_c inside the planning horizon **do**
    Let $safe$ be whether P_c in the obstacle
    if $last_{safe}$ and !$safe$ **then**
      $P_{in} \leftarrow P_c$
      return P_{in}
function FINDCORNERPTS($P_{in}, \nabla c$)
  Let $\vec{d}$ be the vectors perpendicular to ∇c
    and pointing to the sides of obstacle
  topoPoint $\leftarrow$ RAYTRACING($P_{in}, \vec{d}$)
function PUSHAWAYFROMOBS($P_{in}, \nabla c, d_{thr}$)
  Let $\vec{d}$ be the vector perpendicular to ∇c
    and at an acute angel to d_{thr}
  $P_b \leftarrow$ RAYTRACING($P_{in}, \vec{d}$)
  $grad \leftarrow$ the ESDF gradient in P_b
  $P_{topo} \leftarrow grad \times d_{thr}$
function RAYTRACING(P_{in}, dir)
  scan from P_{in} in direction $\vec{d}$
  if ray touches the planning horizon boundary **then**
    return the intersection P_b
  else
    $P_s \leftarrow$ first point satisfying safe distance

Pseudocode implementing of the method is presented in Algorithm 1. We periodically check whether the local trajectory in the planning horizon ahead of

the UAV collides with obstacles. If a collision is detected, the topological points searching process is triggered (line 1). Firstly, the intersection P_{in} between the trajectory and the obstacle surface is obtained (line 2). Then the algorithm calculates the Euclidean signed distance field (ESDF) gradient and the normal vector of the obstacle surface at P_{in} through the local ESDF map (line 3 to 4).

Once we get the gradient and normal vector, the search direction of the topological points is determined according to the relationship between the direction of the two vectors. The geometric relationship between the ESDF gradient and the normal vector at the collision point is as follows:

$$\xi = \|\nabla c \times \vec{n}\| \tag{1}$$

If the ESDF gradient and the normal vector are collinear, then $\xi = 0$, otherwise, $\xi \neq 0$. After calculating ξ, the search direction vector $\vec{d}$ can be calculated as:

$$\vec{d} = \begin{cases} \nabla c & \xi \neq 0 \\ \vec{v} + \frac{\langle -\nabla c, \vec{v} \rangle}{\|\nabla c\|} \cdot \nabla c & \xi = 0 \end{cases} \tag{2}$$

At this time, the search direction vector $\vec{d}$ is related to the UAV's motion property. We can simply rotate $\vec{d}$ by 180° to obtain the topological search direction $\vec{d'}$ with symmetric characteristics.

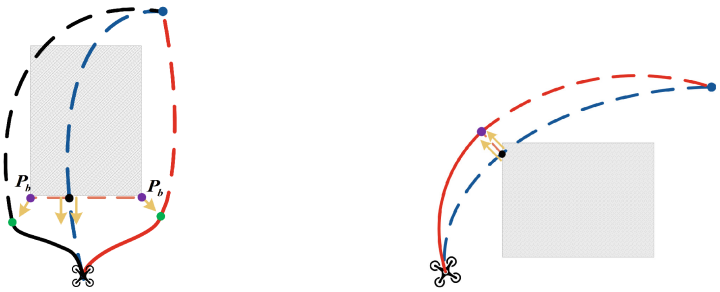


Fig. 2. The proposed waypoint searching method. (a) The ESDF gradient at the collision point (black point) is collinear with the normal vector. (b) Push the collision point away from the obstacle along the ESDF gradient at the collision point.

If the gradient is collinear with the normal vector, the search direction $\vec{d}$ is set perpendicular to both of them and points to the sides of the obstacle (line 6, illustrated as Fig. 2(a)). We do ray tracing from the collision point P_{in} along the search direction $\vec{d}$. The tracing stops when finding the corner point of obstacles or the projection exceeds the planning horizon, and the boundary point P_b is

obtained. Then the topological waypoint can be calculated by Eq. 3, and the calculated point will be added to the topological waypoint set (line 24 to 28).

$$P_{topo} = P_b + \nabla c \cdot d_{thr} \tag{3}$$

where d_{thr} is the preset safety distance threshold, which make the generated trajectory maintain a certain buffer distance between obstacles. In actual mission scenarios, the UAV cannot accurately move along the trajectory. If the trajectory is too close to the obstacle, it may cause the UAV to hit the obstacle due to deviation. Another situation is that the trajectory collides with an obstacle's corner, as shown in Fig. 2(b). At this time, the ESDF gradient and the normal vector at the collision point are no longer collinear. And the search direction $\vec{d}$ is set to be collinear with the gradient direction to guide the drone away from obstacles (line 8).

If no topological points are found in the previous step, it means the discovered obstacle is too large, and the ray tracing exceeds the planning horizon of the UAV. By this time, we need to reset the search direction. An UAV's motion has high-order dynamic properties (velocity, acceleration, etc.), which cannot be mutated. Therefore, we tend to search in the direction that can reflect the motion properties of the UAV. The motion orientation is a critical property, which can be calculated by the coordinates of the trajectory points near the current position. After selecting a new search direction, the consistency of the search direction should be ensured. This is because the environment for the UAV is only partially observable. If the UAV is not specified to search along one direction all the time, it may fall into the "local optimal trap", spinning around in the same place. The search direction is perpendicular to the gradient at the collision point and is acute to the motion orientation. Similarly, we do ray tracing from the collision point along the search direction until getting P_b at the planning horizon boundary.

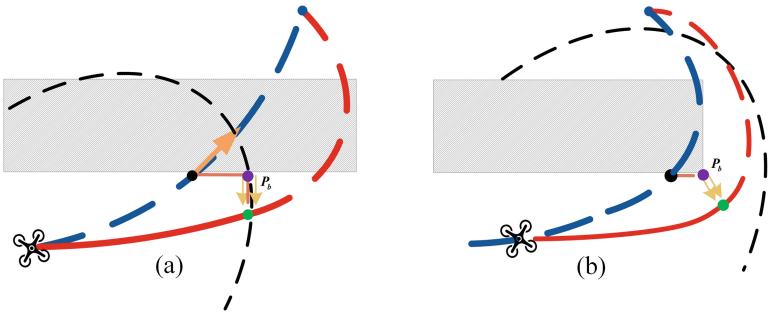


Fig. 3. A scenario with the large obstacle. (a) The obstacle exceeds the planning horizon, and the waypoint satisfying safe distance is obtained at the boundary. (b) Bypass the large obstacle by searching for a safe waypoint progressively.

This point is then pushed away from obstacles toward the ESDF gradient descent direction, after which the next waypoint is obtained. Repeat the above searching process until the drone bypasses the obstacle. (depicted in Fig. 3).

3.2 Optimal State Transition Waypoint

When there are multiple topological points, we need to choose one of them as the next key waypoint. Given the drone's high-order dynamic properties, the best choice may not be the point that guides a shorter path to the end. As shown in Fig. 4, the trajectory generated by the shorter path takes a sharp turn, which is dynamic infeasibility and leads to a high control cost. Actually, it is challenging to rely on geometrically topological points alone to make decisions due to the lack of high-order dynamic information and the UAV's true motion at the current moment. We propose generating multiple jerk-controlled polynomial trajectories from the UAV's current state to the points obtained by the geometrically topologic points finding method. After that, we choose the one with the minimum transition cost as the waypoint according to the control cost of the polynomial trajectories.

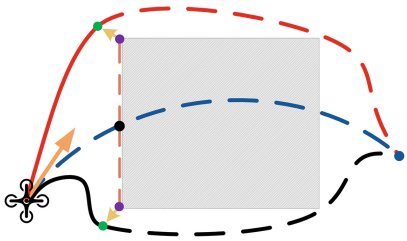


Fig. 4. Dynamic infeasibility led by the shortest path. The blue dash solid line is the current trajectory. The black line is the trajectory generated by a shorter path. The arrow is the velocity direction at the current moment, and the red line is the trajectory that conforms to the dynamic feasibility. (Color figure online)

For each topological point, we consider to generate a m-segment, n-th order polynomial trajectory through it to the target point, and the i-th segment is parameterized as:

$$p_i(t) = c_n t^n + c_{n-1} t^{n-1} + \cdots + c_1 t + c_0 \tag{4}$$

where $\{c_0, c_1, \cdots, c_n\}$ is the coefficient of the i-th segment.

To find the optimal state transition point, we need to calculate the control cost of each trajectory:

$$J = \sum_{k \in \{x,y,z\}} \sum_{i \in \{0,1,\cdots,m\}} \int_0^T [p_{k,i}^{(3)}(t)]^2 dt \tag{5}$$

where T is the total transition time of m trajectory segments. Then the one with the least control overhead J is chosen as the new global guiding trajectory and input to the back-end trajectory optimization. In the following section, we introduce the process of generating an executable B-spline local trajectory, which will make the drone flight safer and smoother.

4 Conservative Trajectory Replanning

In an optimistic trajectory replanning method, it is usually assumed that the generated local trajectory can bypass obstacles and rejoin the global trajectory. However, in an environment with dense or large obstacles, such a hypothesis may lead to unsafe trajectories and frequent replanning in a high probability. We propose an adaptive conservative trajectory replanning framework for ensuring trajectory safety and reducing replanning frequency. Moreover, we parameterize the trajectory by B-spline to improve the efficiency of trajectory optimization. The proposed method adopts a hierarchical structure of global trajectory guiding motion and local trajectory avoiding unpredicted obstacles. The monomial polynomial represents the global trajectory without considering whether it passes through obstacles. In contrast, the local trajectory is represented by a B-spline and is only located in the drone's currently known-free space.

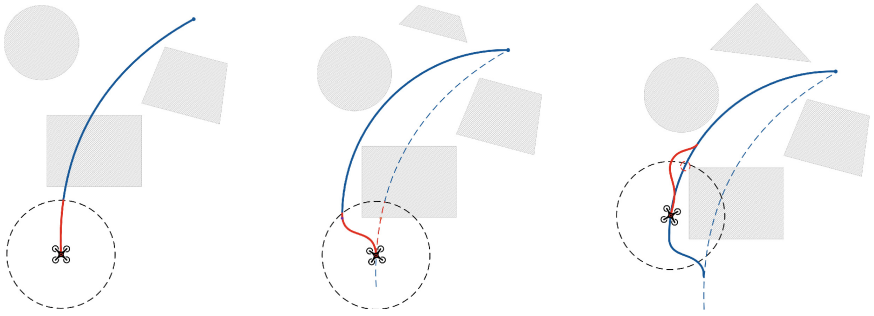


Fig. 5. The trajectory replanning method. (a) A segment of the current global trajectory (blue solid line) within the planning horizon (black dotted circle) is reparameterized as a B-spline local trajectory (red solid line). (b) The replanning process is triggered when detecting a collision on the current trajectory. A new global guiding trajectory is generated by the geometrically topological waypoint searching. Meanwhile, a local trajectory is also generated to avoid the obstacle. (c) When the local trajectory is detected with low clearance, it is refined to a safer trajectory by the B-spline's local modification scheme. (Color figure online)

4.1 Adaptive Trajectory Replanning

The replanning module is responsible for adjusting the flight trajectory in time to avoid unpredicted obstacles and guarantee the drone's safety. In our method, the global guiding trajectory is represented as the monomial polynomial, which does not take the obstacles in the environment into account. Moreover, the UAV has a limited sensing range, making it necessary to replan the trajectory to avoid unpredicted obstacles frequently. We assume that the replanned local trajectory cannot return to the original global trajectory, which is reasonable for conservative considerations. Therefore, when detecting a collision between the current trajectory and an obstacle in the environment, a new globalguiding trajectory is generated through the geometrical topological waypoint searching. Afterward, the segment of the new global trajectory located in the planning horizon is optimized to a local trajectory parameterized by the B-spline curve (depicted in Fig. 5(b)). In addition, the current local trajectory may be too close to the newly discovered obstacle. In this case, we need to use the ESDF gradient information to push this part of the trajectory segment away from the obstacle. The obtained trajectory is called refined trajectory (displayed in Fig. 5(c)).

Algorithm 2. Conservative trajectory replanning

```
function TRAJREPLAN(cur_traj)
    if a collision is detected on cur_traj then
        P_topo ← GETTOPOPOINT(cur_traj)
        update the global guiding trajectory gl_traj
        local_new ← REPARAMLOCALTRAJ(gl_traj)
    else if cur_traj is close to the obstacle then
        refine the cur_traj by the local modification
        scheme of B-spline
    else
        local_new ← REPARAMLOCALTRAJ(gl_traj)
```

4.2 B-Spline Trajectory Representation

The monomial polynomial is elementary in form and easy to calculate, but it is pure numerical formula and does not contain any spatial geometric information. Therefore, it is necessary to iteratively detect the extreme value in optimization to ensure that the trajectory meets the safety and dynamic constraints, which will bring about numerical instability [17] and increase the calculation time. On the contrary, the B-spline curve has appropriate geometric properties, which can be well combined with the ESDF gradient information to reduce the complexity of optimization.

We use B-spline to parameterize the local trajectory. Given $n + 1$ control points $\{\mathrm{P}_0,\mathrm{P}_1,\cdots,\mathrm{P}_n\}$ and a knot vector $U = \{\mathrm{u}_0, \mathrm{u}_1, \cdots, u_m\}$, the B-spline of degree p can be uniquely defined as:

$$C(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i \tag{6}$$

where $N_{i,p}(u)$ are the basis functions of B-spline which can be computed by De Boor-Cox recursive formula and t is the normalized parameter which can be computed according to $t = (u - u_i)/\Delta u$, for u$\in [u_i, u_{i+1}]$. In proposed method, the degree p is set to 3, and n,m,p must satisfy $m = n + p + 1$.

4.3 Problem Formulation

In our proposed method, to ensure the smoothness, safety and dynamic feasibility of the final executable trajectory, the local trajectory planning problem is formalized as an optimization problem of the cost function:

$$f_{total} = \lambda_s f_s + \lambda_c f_c + \lambda_d (f_v + f_a) \tag{7}$$

where f_s is the smoothness cost that is designed as an elastic band cost function; f_c is the collision cost that penalizes the point on trajectory which are less than the given threshold distance d to the nearest obstacles; f_v *and* f_a are dynamic feasibility cost that penalize infeasible high-order motion properties (velocity and acceleration).

The purpose of the smoothness cost function is to make the generated trajectory smoother so that the control overhead for UAV moving along the trajectory is as slight as possible. For the consideration of being able to adjust the time allocation in optimization, rather than the integral over squared snap or jerk, we adopt the elastic band cost function as [14]:

$$f_s = \sum_{i=p+1}^{n-p+1} \left\| (P_{i+1} - P_i) + (P_{i-1} - P_i) \right\|^2 \tag{8}$$

To ensure that the generated trajectory is far from the obstacle and meets the dynamic constraints, we use the piecewise quadratic functions to punish collisions and feasibility costs:

$$F_c(d) = \begin{cases} (d - d_{thr})^2 & d \le d_{thr} \\ 0 & d > d_{thr} \end{cases} \tag{9}$$

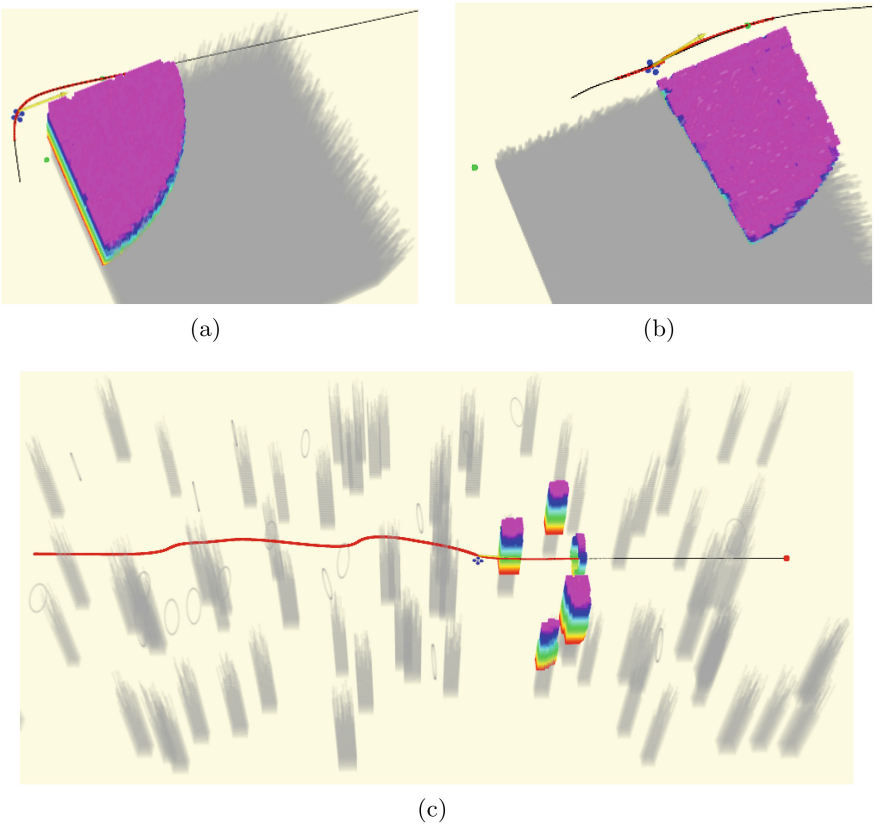
$$F_d(v) = \begin{cases} (v^2 - v_{max}^2)^2 & v^2 \ge v_{max}^2 \\ 0 & v^2 < v_{max}^2 \end{cases} \tag{10}$$

5 Experiments

In this section, we present the experimental result of the proposed method. First, we evaluate the searching part with the baseline method that uses a standard front-end pathfinding algorithm. Second, we evaluate the proposed adaptive conservative trajectory planning approach with the state-of-the-art method.

5.1 Implementation Details

The motion planning method proposed in this letter is implemented in C++11 with a non-linear optimization solver NLopt. We built a $40 \times 20 \times 5$ m random forest environment with 100 deployed pillar and circle obstacles for simulation experimental (as is shown in Fig. 6). All computations are done in a 3.0 GHz Intel i7-9700 processor. In the topological points finding part, we set the safety distance threshold to obstacles $d_{thr} = 1.0$ m. For the trajectory optimization part, we set the $\lambda_s = 10$, $\lambda_c = 8, \lambda_d = 0.01$. Finally, to simulate the autonomous flight experiment of the UAV under an unknown environment, we set the sensing radius to be 6m, and the UAV maintains only a $6 \times 6 \times 3$ m local map.



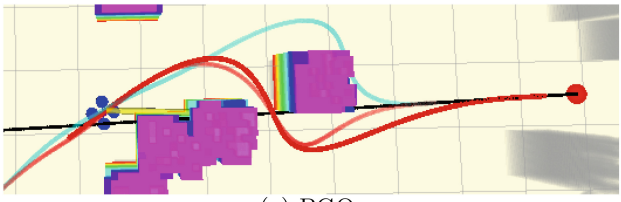
(a) (b)

(c)

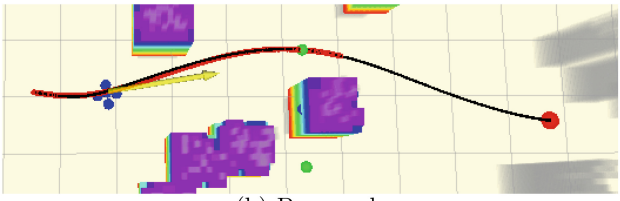
Fig. 6. Autonomous flight experiments in Rviz simulation environments. The colorful voxels represent the mapped obstacles in the UAV's FOV, while the grey voxels represent the unknown obstacles.

5.2 Topological Waypoints Searching

We compare the proposed geometrically topological waypoints searching method with PGO [19], which use the standard sampling-based pathfinding algorithms. Since the proposed approach returns a set of feasible topological points rather than a path, we mainly compare the searching time of each method. We conduct 10 random maps with different obstacle size and density and run both methods in each map 10 times from the same start point to endpoint.



(a) PGO



(b) Proposed

Fig. 7. Benchmark comparison of the proposed method with PGO in waypoint searching. Our method find two topological waypoints in 1.12 ms, while the PGO find three in 6.92 ms.

PGO [19] proposes to find multiple safe and feasible topological paths to improve the quality of the final trajectory. It uses a sampling-based PRM-like algorithm and integrates time-consuming path shortening and pruning. For better visualization, we optimize the path to the displayed trajectory. As displayed in Fig. 7 and Table 1, our method finds a fewer topological path, but the searching time has been reduced by 80% compared to the PRM-like pathfinding method in PGO. This is expected for three reasons. First, our approach considers that the UAV lacks a view of the upper and lower fuselage areas, so it avoids exploring the path in the z-axis direction. Second, our method is based on conservative estimates of the environment and only searches for topological path points in the known-free space. Third, we reduce the search space of pathfinding by taking the geometric and gradient information of the obstacle into account and extracting several feasible waypoints on both sides of the obstacles.

Table 1. Comparisons of waypoint searching

Obstacle size	Method	Average time (ms)	Max time (ms)
<1 m	Proposed	**1.2**	**1.7**
	PGO	7.1	8.7
1–4 m	Proposed	**1.2**	**1.2**
	PGO	7.2	7.9
>4 m	Proposed	**1.3**	**1.8**
	PGO	–	–

5.3 Conservative Trajectory Replanning

Moreover, our method considers the safety distance when searching for waypoints. The found waypoints keep a safe distance from the obstacle, which gives the drone sufficient reaction time and braking distance when encountering unpredicted obstacles. This is meaningful for drones navigating autonomously in environments containing large obstacles. As is shown in Table 1, our method is still stable when there are obstacles with a diameter of more than 4 m in the environment. However, the baseline method rarely successes, even if we increase the sampling range to 10×10 m and set the maximum number of iterations to 2000.

In the following, we compare the performance of our conservative trajectory planning framework with the baseline method. We set the maximum velocity and acceleration constraints are 3 m/s and 4 m/s^2, respectively. The UAV is initialized with a global trajectory generated by [15] and does not consider collision. For a fair comparison, we use the open-source implementation of PGO [19] and the default parameter settings. We check for the re-planning frequency and time, as well as the safety and quality of trajectories generated by different methods.

Re-planning Frequency and Time: Our proposed trajectory replanning method is adaptive and event-triggered. When an unsafe event in the current trajectory is detected, such as collisions or being too close to an obstacle according to our safe condition, the trajectory will be re-planned. We compare the replanning frequency and average planning time in random maps with different obstacle densities. As shown in Table 2, the replanning frequency of the proposed strategy is significantly less compared to the baseline method, no matter in high-density or low-density environments, and it has a considerable average planning time. Our method is based on the conservative estimation of the environment and only generates a new local trajectory in known-free space, which can naturally ensure trajectory safety. In contrast, the approach based on optimistic assumptions requires frequent replanning to ensure safety.

Safety: For autonomous navigation in the unknown and complex environment, we hope that the drone can always maintain a safe distance from the surrounding

Table 2. Comparisons of the trajectory planning.

Obstacle density and size	Method	Replan number	Average replan time (ms)
High and Small	Proposed	**5.6**	**11.2**
	PGO	28.5	13.4
Medium	Proposed	**3.5**	**9**
	PGO	69.1	9.3
Low and Large	Proposed	**5.0**	**5.1**
	PGO	–	–

obstacles. To evaluate the safety performance of the generated trajectory, we propose a new evaluation indicator: the length of the trajectory segment on which the distance to obstacles is less than the given safety threshold. The safe distance threshold is set as the braking distance that the drone decelerates from the maximum speed to stop state using the maximum acceleration and can be calculated by:

$$d_{safe} = \frac{V_{max}^2}{2A_{max}} \tag{11}$$

If the distance between the UAV and obstacles in the environment is greater than the safety threshold d_{safe}, no matter how fast the UAV is when it observes an imminent danger of collision (less than the maximum speed of the UAV), it can successfully stop in front of the obstacle without hitting the obstacle. Conversely, if the distance between the UAV and the obstacle is less than the safety threshold d_{safe}, the UAV may hit the obstacle due to the long braking distance. The more segments on a trajectory whose distance to the obstacle in the environment is greater than the safety threshold, the smaller the probability of collision when the UAV flying along such trajectory, that is, the safer the trajectory. Therefore, our proposed trajectory safety evaluation indicator is reasonable and feasible.

According to the parameter setting, the safe distance in our simulation experiment is 1.15 m. As displayed in Fig. 8, our method always have a safer trajectory compared with the baseline. In the scene with large obstacles, only 18.86% of the trajectory segments in our method are less than the safe distance, while the unsafe trajectory segments in baseline method account for 34.16% (depicted in Fig. 8(c)). In other words, the safety of our trajectory has improved by 44.8%. Furthermore, when the diameter of the obstacle in the environment is greater than 10 m, our method can navigate to the target point while maintaining an absolute safe distance between the UAV and the obstacle, while the baseline method cannot complete the autonomous navigation task (as is shown in Fig. 8(d)). With the increase of obstacle density, the proportion of unsafe trajectory segments also increases. However, our method is still better than the baseline method, and the safety is improved by 21.1% and 17.5% respectively (displayed in Fig. 8(b) and 8(a)).

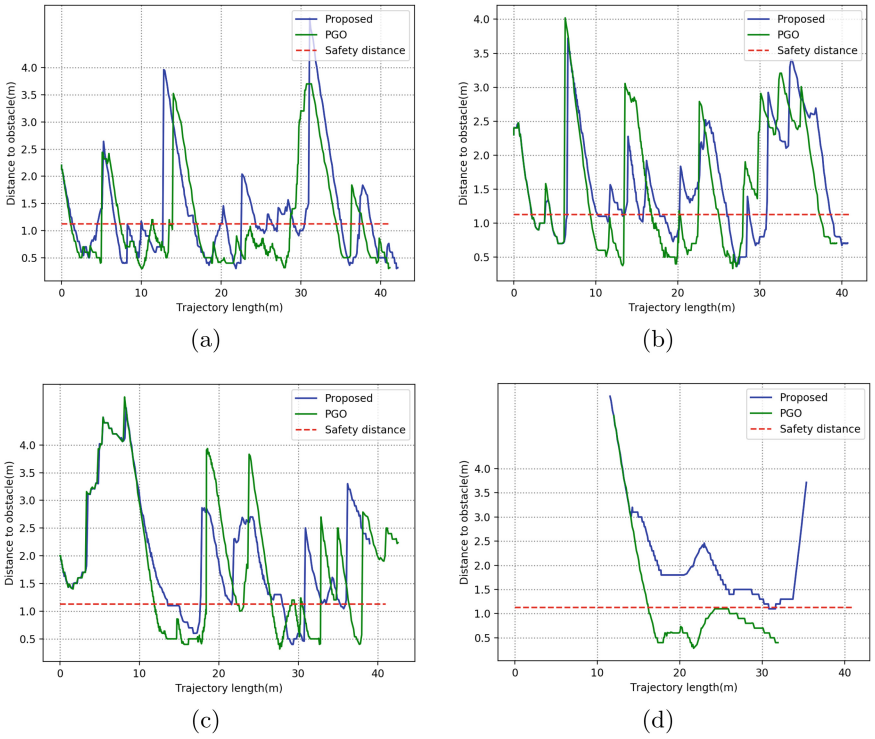


Fig. 8. Comparisons of the trajectory safety in different maps.

Trajectory Quality: The UAV is a motion system with high-order properties (velocity, acceleration, etc.). Therefore, the generated trajectory should satisfy the kinodynamic constraints. We evaluate the trajectory quality by comparing the flight time, flight distance, and the average flight speed. As is shown in Table 3, the proposed method does not perform as well as the baseline method in flight distance and the variance of velocity. This can be explained that the drone in our method tends to take long detours for guaranteeing the safety due to the conservative consideration. However, our method outperforms the baseline method in aspects of flight time, and average flight speed, especially in an environment containing obstacles whose diameter is more than $4m$. For the partially observable environment, the baseline method, which adopts a standard pathfinding algorithm and optimistic assumption, may fall into the local optimal trap when encountering a large obstacle. In contrast, the drone can maintain high speed when flying along the trajectory generated by our method, even in the environment containing large obstacles (displayed in Fig. 9).

Table 3. Comparisons of the trajectory planning.

Obstacle density and size	Method	Filght time(s)	Flight distance(m)	Average speed(m/s)	Var. velocity
High and Small	Proposed	**21.467**	41.935	**1.938**	0.797
	PGO	21.833	**35.334**	1.613	**0.654**
Medium	Proposed	**18.74**	39.843	**2.124**	0.859
	PGO	22.302	**34.113**	1.512	**0.736**
Low and Large	Proposed	**13.454**	**25.347**	1.86	**0.866**
	PGO	–	–	–	–

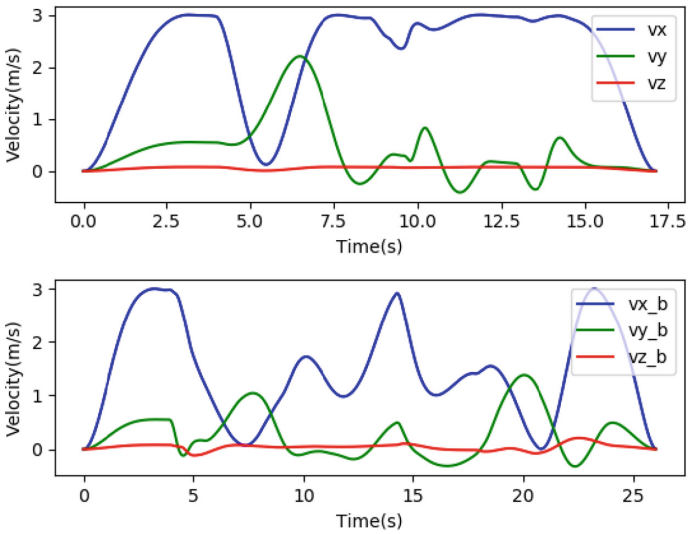


Fig. 9. Velocity curve of simulating flight in environment with large obstacles. The top half is the velocity curve of our method, and the bottom half is the baseline method.

5.4 Comparisons of Planning Efficiency

An efficient trajectory planning method should reduce the computational cost of replanning while ensuring the safety of the trajectory. We propose to use the calculation time of the drone for replanning to evaluate the overall efficiency of the motion planning system, and it can be computed by:

$$E(t) = E(n)E(t_r) \tag{12}$$

where t is the total computing time for replanning, n is the replanning frequency, and t_r is the replanning time.

Replanning time includes the time for waypoint searching, and the quality of found waypoints can influence the replanning frequency. In our proposed method, the front-end geometrically topological waypoint searching method considers the

safe distance and finds a safe enough waypoint rather than the shortest path, which is close to the obstacle. This allows the drone to observes more and plan a long safe trajectory. Thereby, the replanning frequency is decreased. For fairness, we only compare the planning efficiency of two methods in scenarios where the baseline method can work. As shown in Table 2, the expected planning time of our method in different environment can be computed by Eq. 12 as 47.11 ms, while 512.27 ms in the baseline method.

6 Conclusions

In this paper, we present a safe and efficient motion planning method based on conservative estimation for autonomous navigation. The geometrically topological waypoints searching and adaptive trajectory re-planning are devised to improving trajectory safety without sacrificing navigation efficiency. Extensive benchmark comparisons in the simulation environment are conducted to validate the performance of our method. The results show that the trajectory safety of our method is improved by 27.8%, and the replanning frequency is significantly reduced, while the average replanning time is not increased. Currently, this method does not consider active environmental perception and is only suitable for quadrotors. In the future, we will introduce the active planning of yaw angle. Besides, we will extend this method to adapt to other types of UAVs, such as fixed-wing.

Acknowledgment. This work was supported by the National Key Research and Development Program of China under Grant No.2017YFB1001901.

References

1. Chen, J., Liu, T., Shen, S.: Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1476–1483. IEEE (2016)
2. Ding, W., Gao, W., Wang, K., Shen, S.: An efficient B-spline-based Kinodynamic replanning framework for quadrotors. IEEE Trans. Rob. **35**(6), 1287–1306 (2019)
3. Gao, F., Lin, Y., Shen, S.: Gradient-based online safe trajectory generation for quadrotor flight in complex environments. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3681–3688. IEEE (2017)
4. Gao, F., Shen, S.: Online quadrotor trajectory generation and autonomous navigation on point clouds. In: 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 139–146. IEEE (2016)
5. Gao, F., Wu, W., Lin, Y., Shen, S.: Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 344–351. IEEE (2018)
6. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
7. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)

8. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: progress and prospects. Algorithmic Comput. Robot. New Dir. **5**, 293–308 (2001)
9. Liu, S., Atanasov, N., Mohta, K., Kumar, V.: Search-based motion planning for quadrotors using linear quadratic minimum time control. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2872–2879. IEEE (2017)
10. Liu, S., et al.: Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. IEEE Robot. Autom. Lett. **2**(3), 1688–1695 (2017)
11. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE International Conference on Robotics and Automation, pp. 2520–2525. IEEE (2011)
12. Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., Galceran, E.: Continuous-time trajectory optimization for online UAV replanning. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5332–5339. IEEE (2016)
13. Quan, L., Han, L., Zhou, B., Shen, S., Gao, F.: Survey of UAV motion planning. IET Cyber-Systems Robot. **2**(1), 14–21 (2020)
14. Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. In: Proceedings IEEE International Conference on Robotics and Automation, pp. 802–807. IEEE (1993)
15. Richter, C., Bry, A., Roy, N.: Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Inaba, M., Corke, P. (eds.) Robotics Research. STAR, vol. 114, pp. 649–666. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28872-7_37
16. Tordesillas, J., Lopez, B.T., How, J.P.: Faster: fast and safe trajectory planner for flights in unknown environments. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1934–1940. IEEE (2019)
17. Usenko, V., von Stumberg, L., Pangercic, A., Cremers, D.: Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 215–222. IEEE (2017)
18. Ye, H., Zhou, X., Wang, Z., Xu, C., Chu, J., Gao, F.: TGK-planner: an efficient topology guided kinodynamic planner for autonomous quadrotors. IEEE Robot. Autom. Lett. **6**(2), 494–501 (2020)
19. Zhou, B., Gao, F., Pan, J., Shen, S.: Robust real-time UAV replanning using guided gradient-based optimization and topological paths. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 1208–1214. IEEE (2020)
20. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robot. Autom. Lett. **4**(4), 3529–3536 (2019)