

# Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths

Boyu Zhou, Fei Gao, Jie Pan, and Shaojie Shen

**Abstract**—Gradient-based trajectory optimization (GTO) has gained wide popularity for quadrotor trajectory replanning. However, it suffers from local minima, which is not only fatal to safety but also unfavorable for smooth navigation. In this paper, we propose a replanning method based on GTO addressing this issue systematically. A path-guided optimization (PGO) approach is devised to tackle infeasible local minima, which improves the replanning success rate significantly. A topological path searching algorithm is developed to capture a collection of distinct useful paths in 3-D environments, each of which then guides an independent trajectory optimization. It activates a more comprehensive exploration of the solution space and output superior replanned trajectories. Benchmark evaluation shows that our method outplays state-of-the-art methods regarding replanning success rate and optimality. Challenging experiments of aggressive autonomous flight are presented to demonstrate the robustness of our method. We will release our implementation as an open-source package<sup>1</sup>.

## I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are gaining popularity for many applications such as autonomous inspection, transportation, and photography, in which the UAV is required to navigate along a global reference trajectory to finish its missions. To ensure safety, trajectory replanning is of vital importance to cope with previously unknown obstacles.

The problem of trajectory replanning has been investigated actively. Recent works [1]–[5] reveal that gradient-based trajectory optimization (GTO), which typically formulates trajectory replanning as a non-linear optimization problem that trades off smoothness, safety, and dynamically feasibility, is particularly effective for this problem. It is widely applied thanks to its convenience to deform an infeasible trajectory segment into a feasible one, with very high efficiency and low memory requirement.

Despite its advantages, GTO-based replanning is cursed by the issue of local minima. The issue arises from the collision cost of the optimization, which should be evaluated according to the structure of the environment. Since there are multiple safe and unsafe regions, this cost is non-convex by nature. This issue could cause fatal crashes since it frequently makes the trajectory get stuck in infeasible local minima and results in replanning failure. Besides, it also leads to the lack of optimality guarantee, as only a small fraction of solution

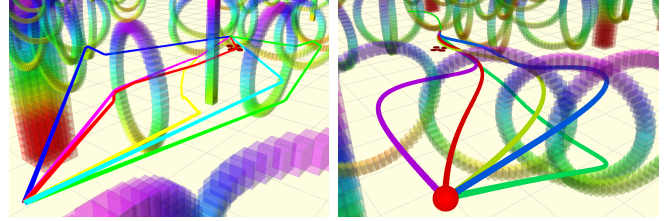


Fig. 1. A set of topologically distinct paths (left). Multiple trajectories generated with the guidance of different paths (right). Video demonstrating the proposed replanning method in aggressive flights is available at <https://www.youtube.com/watch?v=YcEaFTjs-a0>.

space around the initial trajectory is searched. Consequently, the so-called local optimal trajectory is usually unsatisfactory for smooth and safe flights. Researchers have realized this critical issue and employed strategies like random restart [1] and iterative refinement [5] to relieve it. Nonetheless, the problem is not resolved essentially and prohibits GTO to be applied to more challenging scenarios such as aggressive flight.

In this paper, the local minima problem is addressed systematically by a new GTO-based replanning method, which comprised of a **path-guided optimization (PGO)**, an efficient algorithm to discover topologically distinct paths, and the parallel trajectory optimization guided by the paths (as depicted in Fig.1). Firstly, we answer the question of how infeasible local minima of GTO can be reliably prevented. The typical reasons for infeasible local minima are investigated. Based on them, we propose PGO in which a geometric path is utilized to guide the optimization effectively so that the success of replanning is guaranteed. Secondly, we answer how optimality of the replanning can be improved considerably, with only minor computational overhead. We propose an efficient sampling-based topological path searching approach to extract a comprehensive set of paths that capture the structure of the environment. With the guidance of several carefully selected paths, PGO is invoked to explore the solution space more thoroughly. It consistently yields better replanning than previous methods, while the total computation time is comparable.

We conduct extensive benchmark comparisons with state-of-the-art methods and challenging real-world experiments to validate our proposed replanning method. Results show that it is superior to previous ones in terms of significantly higher success rate and stronger optimality guarantee, with only slightly longer computational time induced by the topological path searching. The contributions of this paper are summarized as follows:

This work was supported by HKUST-DJI Joint Innovation Laboratory and HKUST institutional fund. All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. {bzhouai, fgaoaa, jpanai, eeshaojie}@ust.hk

<sup>1</sup>The open-source code will be released at <https://github.com/HKUST-Aerial-Robotics/Fast-Planner> and <https://github.com/HKUST-Aerial-Robotics/TopoTraj>

- 1) A robust optimization approach for real-time trajectory replanning named PGO, to boost the success rate of the replanning.
- 2) An efficient topological path searching algorithm, and its integration with the proposed PGO, to search the solution space more thoroughly and yield better replanning.
- 3) Benchmark comparisons and real-world experiments that validate the performance of our proposed method. The source code of our implementation will be released.

## II. RELATED WORK

### A. Gradient-based Trajectory Optimization

GTO is one of the major trajectory generation approaches, which formulates the problem as a non-linear optimization that minimizes an objective function. Interest in this method was revived recently by [6], which generates discrete-time trajectories by minimizing its smoothness and collision costs using covariant gradient descent. [7] has a similar formulation, but solves the problem by sampling neighboring candidates iteratively. The stochastic sampling strategy partially overcomes the local minima issue but is computationally intensive. [1] extended the method to continuous-time polynomial trajectories to avoid differential errors. It also does random trajectory perturbation and optimization restart for a higher success rate. However, the improvement is insignificant. [2] improved the success rate by providing a high-quality initial path, which is found by an informed sampling-based path searching. However, it only applies to low-speed flight. In [3], the trajectory is parameterized as a uniform B-spline. It showed that the continuity and locality properties of B-spline are particularly useful for trajectory replanning. [4] further exploited the convex hull property of B-spline and improve the optimization efficiency and robustness by a large margin. However, given a poor initial trajectory in complex environments, this method still suffers. As a result, [5] adopts an iterative post-process to improve the practical success rate of [4]. By far, local minima still remains a challenge, since no method copes with it essentially. In this paper, we propose PGO, which incorporates a geometric path in the optimization. As the path effectually guides the optimization to escape from infeasible local minima, the planning success rate is guaranteed. Moreover, multiple distinct paths produced by the topological path searching are integrated with the PGO to seek for plentiful locally optimal solutions, which ensures higher trajectory quality.

### B. Topological Path Planning

There have been works utilizing the idea of topologically distinct paths for planning, in which paths belonging to different homotopy (homology) [8]–[12] or visibility deformation [13] classes are sought. [8] constructs a variant of probabilistic roadmap (PRM) to capture homotopy classes, in which path searching and redundant path filtering are conducted simultaneously. In contrast, [9, 10] firstly creates a PRM or Voronoi diagram, after which a homology equivalence relation based on complex analysis [11] is adopted to filter out redundant paths. These methods only apply to 2-D

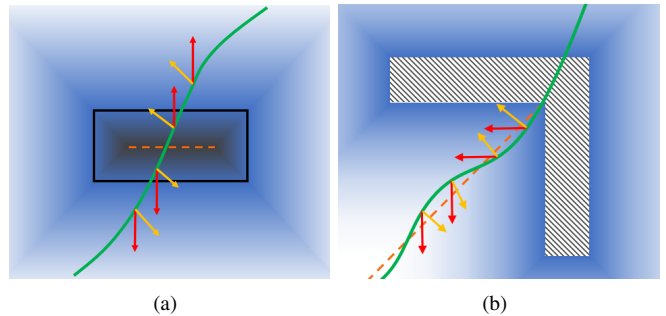


Fig. 2. Typical examples of optimization failure, where adjacent parts of a trajectory are pushed in opposing directions if the trajectory crosses the "valley" (a) or "ridge" (b) of the ESDF (denoted by orange dashed lines). Red arrows denote gradients of ESDF and yellow ones are gradients of the objective function.

scenarios. To seek for 3-D homology classes, [12] exploited the theory of electromagnetism and propose a 3-D homology equivalence relation. However, it requires occupied space to be decomposed into "genus 1" obstacles, which is usually impractical. Besides, capturing only homotopy classes in 3-D space is insufficient to encode the set of useful paths, as indicated in [13], since 3-D homotopic paths may be too hard to deform into each other. To this end, [13] leverages a visibility deformation roadmap to search for a richer set of useful paths. [14, 15] convert maps built from SLAM systems into sparse graphs representing the topological structure of the environments. [13]–[15] focus on global offline planning and are too time-consuming for online usage. Our topological path searching is conceptually closest to [13], but with a reinvented algorithm for real-time performance.

## III. PATH-GUIDED TRAJECTORY OPTIMIZATION

### A. Optimization Failure Analysis

Previous work [16] showed that failure of GTO is relevant to unfavorable initialization, i.e., initial paths that pass through obstacles in certain ways usually get stuck. Underlying reason for this phenomenon is illustrated in Fig.2. Typical GTO methods incorporate the gradients of a Euclidean signed distance field (ESDF) in a collision cost to push the trajectory out of obstacles. This cost is combined with the smoothness and dynamic feasibility cost to form an objective function, whose gradients iteratively deform the trajectory into a smooth and safe one. Yet there are some "valleys" or "ridges" in the ESDF, around which the gradients differ greatly. Consequently, if a trajectory is in collision and crosses such regions, the gradients of ESDF will change abruptly at some points. This can result in gradients of the objective function pushing different parts of the trajectory in opposing directions and fails the optimization.

Normally, such "valleys" and "ridges", which corresponds to the space that has an identical distance to the surfaces of nearby obstacles, are difficult to avoid, especially in complex environments. Therefore, optimization depending solely on the ESDF fails inevitably at times. To solve the problem, it is essential to introduce extra information that can produce an objective function whose gradients consistently deform the trajectory to the free space.

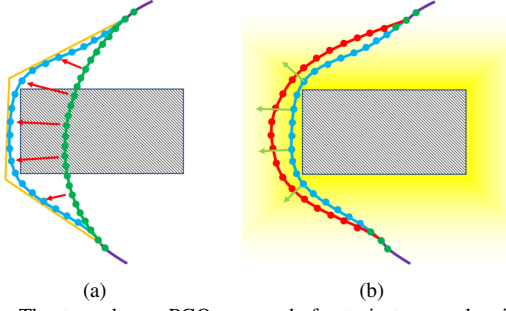


Fig. 3. The two-phases PGO approach for trajectory replanning. (a) A geometric guiding path (orange) attracts the original B-spline trajectory (green) into the free space. (b) The warmup trajectory is further optimized in the ESDF to improve its smoothness and clearance.

### B. Problem Formulation

We propose PGO built upon our previous work [4] that represents trajectories as B-splines for more efficient cost evaluation. For a trajectory segment in collision, we reparameterize it as a  $p_b$  degree uniform B-spline with control points  $\{\mathbf{Q}_0, \mathbf{Q}_1, \dots, \mathbf{Q}_N\}$ .

PGO consists of two different phases. The first phase generates an intermediate **warmup trajectory**. As concluded above, external information should be included to effectually deform the trajectory, since solely applying the ESDF could be futile. We employ a geometric guiding path to attract the initial trajectory to the free space (depicted in Fig. 3(a)) since collision-free paths are readily available from standard methods like A\* and RRT\*. In our work, the paths are provided by the sampling-based topological path searching (Sect. IV). The first-phase objective function is:

$$f_{p1} = \lambda_{1s}f_s + \lambda_{1g}f_g \quad (1)$$

where  $f_s$  is the smoothness cost of the trajectory, while  $f_g$  is the cost to penalize the distance between the guiding path and the B-spline trajectory. As in [4],  $f_s$  is designed as a elastic band cost function<sup>2</sup> that simulates the elastic forces of a sequence of springs:

$$f_s = \sum_{i=p_b-1}^{N-p_b+1} \|\mathbf{Q}_{i+1} - 2\mathbf{Q}_i + \mathbf{Q}_{i-1}\|^2 \quad (2)$$

Because the shape of a B-spline is finely controlled by its control points, we utilize this property to simplify the design of  $f_g$ . Each control point  $\mathbf{Q}_i$  is assigned with an associated point  $\mathbf{G}_i$  on the guiding path, which is uniformly sampled along the guiding path. Then  $f_g$  is defined as the sum of the squared Euclidean distance between these point pairs:

$$f_g = \sum_{i=p_b}^{N-p_b} \|\mathbf{Q}_i - \mathbf{G}_i\|^2 \quad (3)$$

Notably, it is an unconstrained quadratic programming problem, so its optimal solution can be obtained in closed form. It outputs a smooth trajectory in the vicinity of the guiding path. Since the path is already collision-free, usually

<sup>2</sup>Only the subset of control points  $\{\mathbf{Q}_{p_b}, \mathbf{Q}_{p_b+1}, \dots, \mathbf{Q}_{N-p_b}\}$  is optimized due to the boundary state constraints of the trajectory.  $\mathbf{Q}_{p_b-2}$ ,  $\mathbf{Q}_{p_b-1}$ ,  $\mathbf{Q}_{N-p_b+1}$  and  $\mathbf{Q}_{N-p_b+2}$  are needed to evaluate the smoothness.

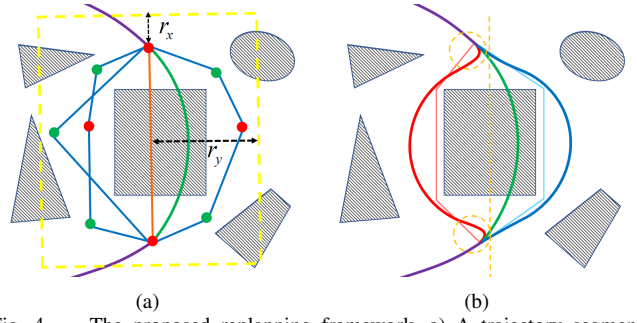


Fig. 4. The proposed replanning framework. a) A trajectory segment within the checking horizon (green) collides with an obstacle and triggers the topological roadmap generation (Sect.IV-B) within a cube. b) Paths are extracted, shortened and pruned to guide the PGO in parallel. The red path is shorter than the blue one; however, it leads to a more jerky trajectory which takes sharp turns near the start and end position.

the warmup trajectory is also so. Even though it is not completely collision-free, its major part will be attracted to the free space. At this stage, the gradients of ESDF along the trajectory vary smoothly, and the gradients of the objective function (green arrows in Fig.3(b)) push the trajectory to the free space in consistent directions. Hence, standard GTO methods can be utilized to improve the trajectory.

In the second phase, we adopt our previous B-spline optimization method [4] to further refine the warmup trajectory into a smooth, safe, and dynamically feasible one, whose objective function is:

$$f_{p2} = \lambda_{2s}f_s + \lambda_{2c}f_c + \lambda_{2d}(f_v + f_a) \quad (4)$$

$f_c$  is the collision cost evaluated on the ESDF, which grows rapidly if the trajectory gets close to obstacles.  $f_v$  and  $f_a$  penalize infeasible velocity and acceleration. The formulations of  $f_c$ ,  $f_v$ , and  $f_a$  are simplified based on the convex hull property of B-spline, thanks to which it suffices to constrain the control points of the B-spline to ensure safety and dynamic feasibility, without evaluating expensive line integrals. For brevity, we refer the readers to [4] for detailed formulations.

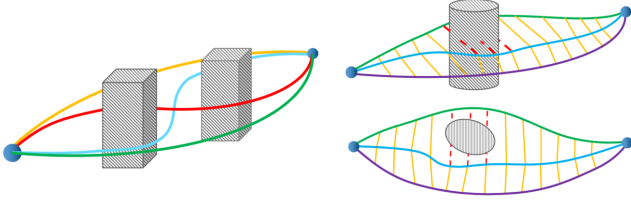
Although PGO has one more step of optimization compared with previous methods, it can generate better trajectories within shorter time. The first-phase takes only negligible time, but generate a warmup trajectory that is easier to be further refined, which improve the overall efficiency.

## IV. TOPOLOGICAL PATH SEARCHING

Given a geometric guiding path, our PGO method can replan a locally optimal trajectory. However, this trajectory is not necessarily satisfactory, even with the guidance of the shortest path, as seen in Fig. 4(b). Actually, it is difficult to determine the best guiding path, since the paths do not contain high order information (velocity, acceleration, etc.), and can not completely reflect the true motion. Searching a kinodynamic path [17, 18] may suffice, but it takes excessive time to obtain a promising path with boundary state constraints at the start and end of the replanned trajectory.

For a better solution, a variety of paths is required. We propose a sampling-based topological path searching to find a collection of distinct paths to guide the PGO. Although





(a) The four trajectories are equivalent under the definition of homotopy, trajectory is distinct to the green one, but represent substantially different motions of a quadrotor. (b) An illustration of UVD. The blue trajectory is distinct to the green one, but is equivalent to the purple one.

Fig. 5. Topology equivalence relation.

methods [8, 13]–[15] are for this problem, none of them runs in real-time in complex 3-D environments. We redesign the algorithm carefully to solve this challenging problem in real time.

#### A. Topology Equivalence Relation

Although the concept of homotopy is widely used, it captures insufficient useful trajectories in 3-D environments, as shown in Fig. 5(a). [13] proposes a more useful relation in 3-D space named visibility deformation (VD), but it is computationally expensive for equivalence checking. Based on VD, we define *uniform visibility deformation* (UVD), which also captures abundant useful trajectories, and is more efficient for equivalence checking.

---

#### Algorithm 1: Topological Roadmap

---

```

1 Initialize()
2 AddGuard( $\mathcal{G}, s$ ), AddGuard( $\mathcal{G}, g$ )
3 while  $t \leq t_{max} \wedge N_{sample} \leq N_{max}$  do
4    $p_s \leftarrow \text{Sample}()$ 
5    $g_{vis} \leftarrow \text{VisibleGuards}(\mathcal{G}, p_s)$ 
6   if  $g_{vis}.size() == 0$  then
7     AddGuard( $\mathcal{G}, p_s$ )
8   if  $g_{vis}.size() == 2$  then
9      $path_1 \leftarrow \text{Path}(g_{vis}[0], p_s, g_{vis}[1])$ 
10     $distinct \leftarrow \text{True}$ 
11     $\mathcal{N}_s \leftarrow \text{SharedNeighbors}(\mathcal{G}, g_{vis}[0], g_{vis}[1])$ 
12    for each  $n_s \in \mathcal{N}_s$  do
13       $path_2 \leftarrow \text{Path}(g_{vis}[0], n_s, g_{vis}[1])$ 
14      if Equivalent( $path_1, path_2$ ) then
15         $distinct \leftarrow \text{False}$ 
16        if Len( $path_1$ ) < Len( $path_2$ ) then
17          Replace( $\mathcal{G}, p_s, n_s$ )
18        break
19    if  $distinct$  then
20      AddConnector( $\mathcal{G}, p_s, g_{vis}[0], g_{vis}[1]$ )

```

---

**Definition 1.** Two trajectories  $\tau_1(s), \tau_2(s)$  parameterized by  $s \in [0, 1]$  and satisfying  $\tau_1(0) = \tau_2(0), \tau_1(1) = \tau_2(1)$ , belong to the same uniform visibility deformation class, if for all  $s$ , line  $\tau_1(s)\tau_2(s)$  is collision-free.

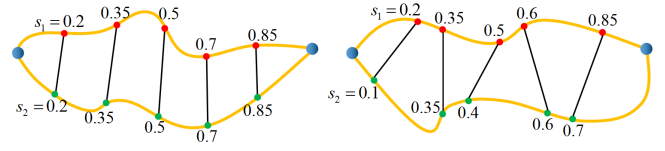


Fig. 6. Comparison between UVD (left) and VD (right). Each red point on one path is transformed to a point (green) on the other path. Any two associated points correspond to the same parameter  $s$  in UVD, but not in VD.

Fig. 5(b) gives an example of three trajectories belonging to two UVD classes. The relation between VD and UVD is depicted in Fig. 6. Both of them define a continuous map between two paths  $\tau_1(s)$  and  $\tau_2(s)$ , in which a point on  $\tau_1(s)$  is transformed to a point on  $\tau_2(s)$  through a straight-line. The major difference is that for UVD, point  $\tau_1(s_1)$  is mapped to  $\tau_2(s_2)$  where  $s_1 = s_2$ , while for VD  $s_1$  does not necessarily equals  $s_2$ . In concept, UVD is less general and characterizes subsets of VD classes. Practically, it captures slightly more classes of distinct paths than VD, but is far less expensive<sup>3</sup> for equivalence checking.

To test UVD relation, one can uniformly discretize  $s \in [0, 1]$  to  $s_i = i/K, i = 0, 1, \dots, K$  and check collision for lines  $\tau_1(s_i)\tau_2(s_i)$ . For the piece-wise straight line paths (as in Alg. 1, **Equivalent()**), we simply parameterize it uniformly, so that for any  $s$  except  $\tau(s)$  is the join points of two straight lines,  $\left\| \frac{d\tau(s)}{ds} \right\| = \text{const.}$

#### B. Topological Roadmap

Alg. 1 is used to construct a UVD roadmap  $\mathcal{G}$  capturing an abundant set of paths from different UVD classes. Unlike standard PRM, which results in many redundant loops, our method generates a more compact roadmap where each UVD class contains just one or a few paths (displayed in Fig. 7).

We introduce two different kinds of graph nodes, namely *guard* and *connector*, similar to the Visibility-PRM [19]. The guards are responsible for exploring different part of the free space, and any two guards  $g_1$  and  $g_2$  are not *visible* to each other (line  $\overline{g_1g_2}$  is in collision). Before the main loop, two guards are created at the start point  $s$  and end point  $g$ . Every time a sampled point is invisible to all other guards, a new guard is created at this point (Line 6-7). To form paths of the roadmap, connectors are used to connect different guards (Line 7-19). When a sampled point is visible to exactly two guards, a new connector is created, either to connect the guards to form a topologically distinct connection (Line 19-20), or to replace an existing connector to make a shorter path (Line 16-17). Limits of time ( $t_{max}$ ) or sampling number ( $N_{max}$ ) are set to terminate the loop.

With the UVD roadmap, a depth-first search augmented by a visited node list is applied to search for the paths between  $s$  and  $g$ , similar to [10].

#### C. Path Shortening & Pruning

As shown in Fig. 7(e), some paths obtained from Alg. 1 may be detoured. Such paths are unfavorable, since in the

<sup>3</sup>To test VD relation, one should compute a visibility diagram and do path searching within it [13], which has higher complexity than testing UVD.

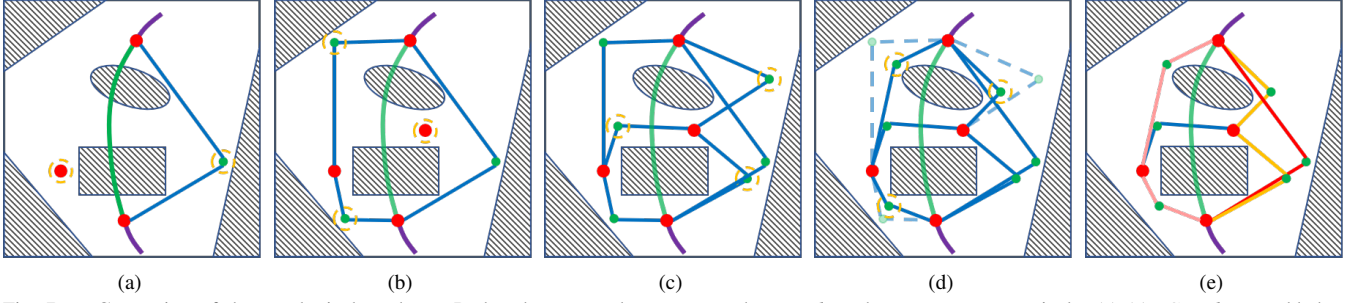


Fig. 7. Generation of the topological roadmap. Red and green nodes represent the *guards* and *connectors* respectively. (a)-(c): *Guards* are added to occupy different regions of space, and *connectors* are added to form new connections between the *guards*. (d): new *connectors* replace the old ones, making the connections shorter. (e): some of the paths found by the depth-first search. Both the red and orange paths belong to the same UVD class, while the pink path is the only member of its UVD class.

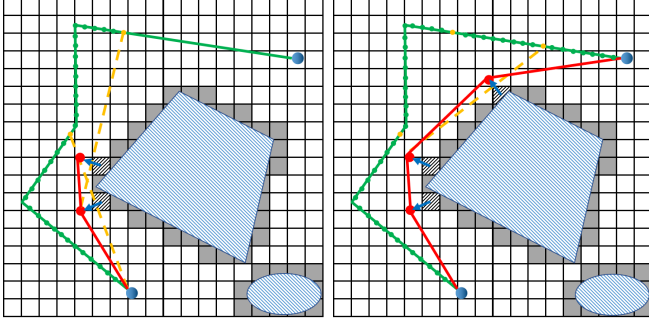


Fig. 8. A detoured and long path (green line) is shortened. For each discretized point on the original path (green point), its visibility to the last waypoint of the shortened path (red points) is checked. The center points of the blocking voxel (dashed) are pushed and appended as new waypoints.

first phase of PGO it can deform a trajectory excessively and make it unsmooth. Hence, Alg. 2 find a topologically equivalent shortcut path  $P_s$  for each  $P_r$  found by the depth-first search (illustrated in Fig. 8). The algorithm uniformly **Discretizes**  $P_r$  to a set of points  $P_d$ . In each iteration, if a point  $p_d$  in  $P_d$  is invisible from the last point in  $P_s$  (Line 3, 4), the center of the first occupied voxel blocking the view of  $P_s$ .**back()** is found (Line 5). This point is then pushed away from obstacles in the direction orthogonal to  $l_d$  and coplanar to the ESDF gradient at  $p_b$  (Line 6), after which it is appended to  $P_s$  (Line 7). The process continues until the last point is reached.

Although in Alg. 1, redundant connection between two guards are avoided, there may exist a small number of redundant paths between  $s$  and  $g$  (Fig. 7(e)). To completely exclude repeated ones, we check the equivalence between any two paths and only preserve topologically distinct ones.

---

**Algorithm 2:** Finding a shortcut path  $P_s$  for  $P_r$ .

---

```

1  $P_d \leftarrow \text{Discretize}(P_r)$ ,  $P_s \leftarrow \{P_d.\text{front}()\}$ 
2 for each  $p_d \in P_d$  do
3    $l_d \leftarrow \text{Line}(P_s.\text{back}(), p_d)$ 
4   if  $\neg \text{LineVisib}(l_d)$  then
5      $p_b \leftarrow \text{BlockPoint}(l_d)$ 
6      $p_o \leftarrow \text{PushAwayObs}(p_b, l_d)$ 
7      $P_s.\text{push\_back}(p_o)$ 
8  $P_s.\text{push\_back}(P_d.\text{back}())$ 
```

---

## V. REAL-TIME TOPOLOGICAL TRAJECTORY REPLANNING

Algorithms in Sect. IV output a fruitful set of paths that can guide trajectory optimization. We integrate them appropriately with the PGO for real-time replanning, as illustrated in Fig. 4. During the flight, a segment of the global trajectory within a specific horizon is checked periodically for safety. Once collisions are detected, topological roadmap construction is triggered within a cube, which is determined by the start and end position of the segment, and  $(r_x, r_y, r_z)$  specifying the size of the cube. Paths extracted from the roadmap are shortened and pruned (Sect. IV-C), after which each of the paths invokes an independent PGO.

Noticeably, the number of alternative UVD classes grows exponentially with the number of obstacles. So in complex environments it may be intractable to optimize for all paths. For this reason, we only select the first  $K_{max}$  shortest paths. Paths more than  $r_{max}$  times longer than the shortest one are also excluded. Such strategies bound the complexity and does not lead to the missing of potential optimality, since very long paths are unlikely to produce the optimal trajectory. Practically we find  $K_{max} = 5$  is sufficient.

## VI. RESULTS

### A. Benchmark Comparisons

We first compare our local replanning method with two state-of-the-art methods Ewok [3] and TRR [5]. Both methods parameterize local trajectory as uniform B-spline and use GTO to do replanning efficiently. TRR further exploits the convex hull property of B-spline to simplify the cost function. We compare all methods in random maps with 3 different densities of obstacles, with 500 random replanning tasks for each density, as shown in Fig. 9. For fair comparisons, we initialize all methods with the same reference trajectories computed by [20]. Besides, we limit the optimization time of Ewok and TRR to 15 *ms* according to their settings in the original work. For ours, the time for topological roadmap construction<sup>4</sup> and trajectory optimization is limited to 3 *ms*

<sup>4</sup>The computation time of topological path searching can not be determined exactly, because time for path shortening and pruning varies slightly in different environments. So we determine the roadmap sampling time empirically according to the desired time budget.

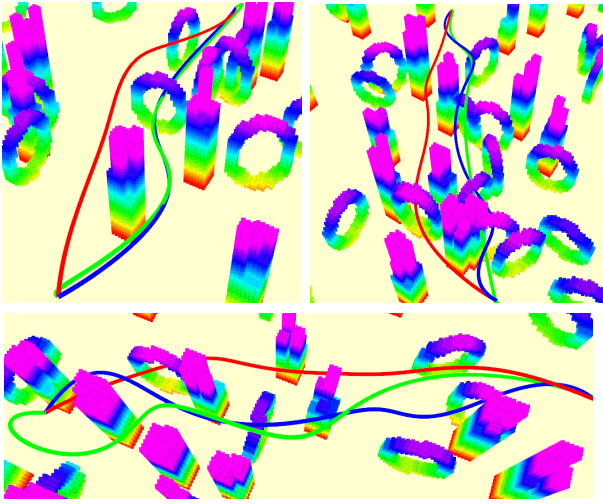


Fig. 9. Benchmark comparisons of the proposed method (red) with Ewok [3] (green) and TRR [5] (blue) in environments of different complexities.

TABLE I  
COMPARISONS OF REPLANNING METHODS.

Density	Method	Comp. time ( $ms$ )	Success rate (%)	Smoothness ( $m^2/s^5$ )
Low	Ewok [3]	15.00	88.0	9.0151
	TRR [5]	15.00	90.4	6.5102
	Proposed	5.75 + 10.00	<b>100.0</b>	<b>5.4357</b>
Medium	Ewok [3]	15.00	81.4	9.5042
	TRR [5]	15.00	85.6	8.3942
	Proposed	6.83 + 10.00	<b>100.0</b>	<b>6.7833</b>
High	Ewok [3]	15.00	78.8	9.4845
	TRR [5]	15.00	82.6	9.1762
	Proposed	7.05 + 10.00	<b>100.0</b>	<b>7.7038</b>

and 10  $ms$ <sup>5</sup>. We check for collisions and smoothness of replanned trajectories.

As is shown in Tab. I, our method outperforms both benchmark methods in terms of success rate and smoothness. Our method successfully finds feasible replanning in all environments, while the success rates of others decrease with increasing complexity of environments. Also, our method generates the smoothest trajectories. Our overall computation is only slightly longer, in which 5-7  $ms$  is spent on topological path searching and 10 $ms$  on parallel optimization. It is notable that although less time is spent on optimization, the generated trajectories are better. The reasons are that the proposed two-phases optimization (Sect. III-B) is easier to converge, and that the parallel optimization (Sect. V) explores the solution space more thoroughly.

### B. Aggressive Autonomous Flights

We conducted aggressive autonomous flight experiments to validate the robustness of our replanning method. The self-developed drone is localized by a robust visual-inertial state estimator [?]. A local mapping framework [22] fuses the depth images from a stereo camera into a volumetric occupancy map and maintains an ESDF for online replanning. We use a geometric controller [23] for trajectory tracking. All modules run on an Intel Core i7-8550U CPU.

<sup>5</sup>Run-time efficiency is critical for online replanning, therefore we give short computation time to test the performance.

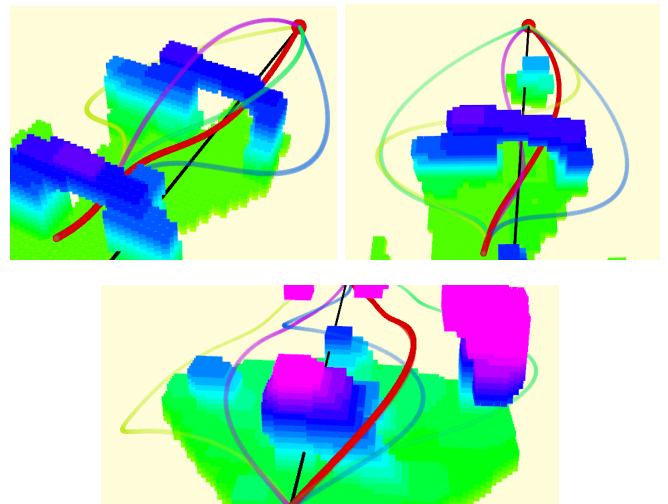


Fig. 10. Autonomous flight experiments in indoor (top) and outdoor (bottom) environments. A set of topologically distinct candidate trajectories are generated to avoid obstacles while keeping the drone close to the global reference trajectory (black). The best one (red) is selected and executed.



Fig. 11. Indoor and outdoor autonomous aggressive flights.

The experiments are conducted in complex indoor and outdoor scenes. In each experiment, a straight-line global reference trajectory is first generated using the approach [20]. During the flight, local trajectories within a horizon of 9  $m$  are replanned to avoid previously unknown obstacles while keeping the drone close to the global trajectory. Aggressive autonomous flights with very limited sensing range in both scenes are quite challenging, since safe trajectories should be generated frequently within extremely short time to cope with sudden and unexpected obstacles. The local trajectories, local maps and velocity profiles of one indoor and outdoor flights are shown in Fig. 10. We refer the readers to the video attachment for more details.

## VII. CONCLUSIONS

In this paper, we propose a robust trajectory replanning method for autonomous quadrotor navigation. It overcomes local minima with the path-guided optimization, topological path searching and parallel trajectory optimization. Extensive benchmark comparisons and aggressive autonomous flight experiments are conducted to validate the robustness of our method.

Currently, the performance of topological path searching is satisfactory, but its completeness is not analyzed in detail. Also, we are not completely certain about the theoretical optimality of the replanning method. In the future, we will investigate these problems. We also plan to extend our method to cope with not only static but also dynamic obstacles to enable safe navigation in more complex scenes.

## REFERENCES

- [1] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*, Daejeon, Korea, Oct. 2016, pp. 5332–5339.
- [2] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*, Sept 2017, pp. 3681–3688.
- [3] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "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)*. IEEE, 2017, pp. 215–222.
- [4] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.
- [5] F. Gao, L. Wang, B. Zhou, L. Han, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *arXiv preprint arXiv:1907.00520*, 2019.
- [6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
- [7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.
- [8] E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of homotopy classes with probabilistic road map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2317–2322.
- [9] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.
- [10] —, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.
- [11] S. Bhattacharya, "Search-based path planning with homotopy class constraints," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
- [12] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.
- [13] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.
- [14] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3d topological graphs for micro-aerial vehicle planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.
- [15] F. Blochliger, M. Fehr, M. Dymczyk, T. Schneider, and R. Siegwart, "Topomap: Topological mapping and navigation based on visual slam maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.
- [16] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [17] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7595–7602.
- [18] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS)*, Sept 2017, pp. 2872–2879.
- [19] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.
- [20] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Intl. Sym. of Robot. Research (ISRR)*, Dec. 2013, pp. 649–666.
- [21] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.
- [22] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," *arXiv preprint arXiv:1903.02144*, 2019.
- [23] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Proc. of the IEEE Control and Decision Conf. (CDC)*, Atlanta, GA, Dec. 2010, pp. 5420–5425.