Quadrotor Trajectory Generation in Dynamic Complex Environments

Ruocheng Li, Bin Xin
School of Automation, Beijing Institute of Technology, Beijing 100081, P. R. China
E-mail: ruochengli@bit.edu.cn

Abstract: Recent advances in trajectory replanning have enabled quadrotors to navigate autonomously in static complex environments. However, navigation in dynamic environments still remains a significant challenge. In this paper, we present a trajectory planner that generates collision-free trajectories in environments with static and dynamic obstacles. A velocity obstacle (VO) based gradient field, called gradient velocity obstacle (GVO), is proposed to solve dynamic obstacles. The main improvement is that GVO maintains the original feasible set while ensuring computational efficiency. Using the output of GVO as an initial guess, a trajectory parameterized by a uniform b-spline is derived to avoid static obstacles. Multiple sets of comparative experiments show the validness and effectiveness of the proposed method.

Key Words: Motion planning, Velocity obstacle, Aerial robotics

1 Introduction

In recent years, quadrotors have received more and more attention due to its high agility, low cost, and scalability, whose applications can be found in many fields, such as search and rescue, aerial photography, industrial investigation, and scientific research. Among these tasks, autonomous navigation is a fundamental component, in which the vehicle can handle environments with static and dynamic obstacles.

Some existing methods such as [1–4], have already enabled quadrotors to autonomously navigate in static environments, nevertheless it remains a challenge in dynamic environments. Difficulties mainly come from the fact that the quadrotors need to develop a strategy for evading dynamic obstacles. A class of methods, such as [5],[6], uses predicted obstacle trajectories to avoid collisions. The core of these approaches is to ensure that there is no spatial and temporal overlap between the quadrotor trajectory and the predicted trajectory. However, these methods either do not consider the uncertainty brought by the prediction, or gradually inflate the obstacle volume to offset the uncertainty during prediction. The former may lead to unreliable solutions in practice, while the latter could yield the conservativeness of the trajectory and result in poor numerical solutions in complex environments. Another class of methods is implemented based on VO, like [7], [8], where they only need positions and velocities of the obstacles. These types of methods leverage information on current moving obstacles to compute one-step actions, which makes them more accurate but short-sighted. Moreover, such methods shrink the initial feasible set for the computational feasibility, which may suffer from unsolvability in the presence of multiple dynamic obstacles.

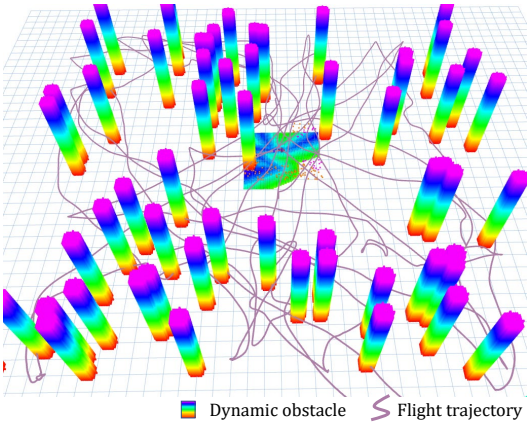

Fig. 1: Snapshot of dynamic obstacle avoidance in simulation.

Motivated by the above facts, this paper proposes a VO-based trajectory planning framework to achieve autonomous navigation in dynamic complex environments. Firstly, we propose a gradient field approach, called **G**radient **V**elocity **O**bstacle (GVO), which transforms velocity obstacle sets into Signed Distance Fields (SDF), within that the current velocity is modified by gradient descent methods to avoid collisions. Compared with existing VO-based frameworks, this method retains the initial feasible domain and is capable of obtaining solutions within milliseconds. Meanwhile, the short-sightedness of VO is addressed by gradient based trajectory planning, which constructs a cost function to generate smooth and collision-free trajectories to avoid static obstacles. Different from methods like [5],[6], the planning framework developed in this paper does not use prediction sequences, but changes the initial velocity of the trajectory in real-time to avoid collisions, which reduces the impact of perception uncertainty.

We fully exploit the method in simulation. Results show that our method performs well and is able to traverse environments with dynamic and static obstacles without prior information. The contributions of this paper are summarized as follows:

(1) We propose the Gradient Velocity Obstacle (GVO) model, a transformative approach to characterizing the set of infeasible velocities. By transforming the geometric

This work was supported in part by the National Outstanding Youth Talents Support Program 61822304, in part by the Basic Science Center Programs of NSFC under Grant 62088101, in part by Beijing Advanced Innovation Center for Intelligent Robots and Systems, in part by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100), in part by Shanghai Municipal Commission of Science and Technology Project (19511132101).

Corresponding author: Bin Xin

shape of the conventional Velocity Obstacle (VO) into an Euclidean signed distance field, GVO not only preserves the original feasible set but also boosts computational efficiency. This enhancement facilitates more effective detection of collision-free velocities.

(2) We present a novel static-dynamic decoupled trajectory planning framework. This framework navigates effectively in dynamic environments by using collision-free velocities generated by the GVO model, which serve as initial conditions in trajectory planning. This approach significantly improves navigation success rates by ensuring the generated trajectories satisfy safety, feasibility, and smoothness constraints.

2 Related Work

2.1 Planning for Dynamic Obstacle Avoidance

Planning for dynamic obstacle avoidance, which aims to enable robots to avoid dynamic obstacles, has been extensively studied. Some methods are implemented based on VO [7–14]. The mechanism of the VO-based approach is that if the relative velocity of the robot and the obstacle falls into the VO, design algorithms to find a variation of velocity and change the current velocity to achieve collision avoidance. Most of the existing methods use geometric methods to find variations. Among these, the optimal reciprocal collision avoidance (ORCA) method [14], which models collision avoidance in terms of quadratic programming (QP), simplifies the problem sufficiently to be able to solve it online. Some practical applications based on ORCA can be found in [7, 8, 10]. The drawback of these methods is the reduction of the original feasible domain, which may lead to unfeasible solutions in complex environments.

Another type of method relies on position information, such as [6, 15–17]. In [15, 16], dynamic obstacles are modeled as ellipsoids, and the ellipsoid constraints are linearized as half-space constraints. Based on the linear constraints, quadrotors solve model predictive control (MPC) online to achieve collision avoidance. However, discarding a half-space in order to avoid an ellipsoid is impracticable when facing multiple dynamic obstacles. In [6], motions of dynamic obstacles are predicted based on the Kalman filter. The core to avoiding collision is to keep the desired trajectory of the quadrotor and the predicted motion sequence from overlapping in spatial and temporal terms. In [17], collision avoidance is achieved based on artificial potential fields (APFs), which give fast and feasible solutions but still suffer from short-sightedness.

In this paper, we combine artificial potential fields and VO to construct gradient fields in velocity space, preserving the initial feasible domain. We solve an unconstrained gradient optimization problem and generate collision-free velocities using proposed gradient fields.

2.2 Quadrotor Trajectory Planning

The study of trajectory planning for quadrotors encompasses both hard-constrained and soft-constrained methods as primary categories. Quadrotors, as demonstrated by [18], exhibit differential flatness, enabling the generation of minimum-snap trajectories for control, with these trajectories formulated in terms of QP. Additionally, [19] provides a closed-form solution for these trajectories. Building upon [18], [20] delineates convex safe regions to facilitate the creation of secure trajectories. In pursuit of trajectories that offer greater maneuverability, [1] introduces a method grounded in mixed integers, whereas [21] advances a method based on B-splines for trajectory creation. More recently, [22] has developed an efficient framework for generating trajectories, with related applications highlighted in [23–25].

Soft-constrained methods often engage in non-linear optimization to balance multiple objectives, such as smoothness, dynamical feasibility, and safety. The application of these methods to local replanning has been effectively demonstrated by [2, 3, 26], showcasing their impressive performance. Inspired by [27], [26] introduces a planning framework based on B-splines, ensuring safety through the use of Euclidean Signed Distance Fields (ESDF). [3] enhances the trajectory's initial estimation and its robustness in navigating complex environments. Furthermore, [2] presents a technique for generating environmental gradients from grid maps, with applications detailed in [6, 28].

In this paper, we base our trajectory planning on B-spline and employ ESDF to generate collision-free terms. The initial velocity of each replanned trajectory is given by GVO, which makes the quadrotor keep safe distances with dynamic obstacles, and the resulting trajectory satisfies smoothness, dynamical feasibility, and safety. The entire planner runs at a high frequency to adapt to the dynamic changes of the environment. In this way, the total trajectory is able to be collision-free with static and dynamic obstacles.

3 Real-time collision avoidance

As presented in the classic VO-based method [14], the optimal collision-free velocity is obtained by solving a QP problem with inequality constraints, where the cost function minimizes the two-norm of the difference between the collision-free velocity and the current velocity. Traditionally, such methods prune the original solution space and thus reduce the problem constraints to a linear constraint space. This operation causes the feasible solution space to be drastically reduced and can lead to unsolvable situations in complex scenarios. Besides, the velocity obstacle-based method is difficult to deal with static obstacles, and the algorithm can be challenging to work in environments containing static and dynamic obstacles. In this work, we transform the velocity obstacle into a signed gradient field, which allows us to formulate the problem of solving collision-free velocities as a nonlinear unconstrained gradient descent optimization problem, which makes it easier to obtain feasible solutions.

3.1 GVO Construction Process

Alg. 1 shows the entire construction process. Assuming there are n dynamic obstacles in the field of view, with positions, velocities, and bounding radius denoted as $\mathbf{p}^d \in \mathbb{R}^3$, $\mathbf{v}^d \in \mathbb{R}^3$, and $r^d \in \mathbb{R}$ respectively, then for the i-th obstacle, the Velocity Obstacle (VO) can be defined as follows:

$$\mathrm{VO}_i = \left\{ \mathbf{v} | \exists t \in [t_0, t_f], t\mathbf{v} \in D\left(\mathbf{p}_i^d - \mathbf{p}^m, r_i^d + r^m\right) \right\} \tag{1}$$

where $D(\mathbf{p}, r)$ denotes an open disc of radius r centered at $\mathbf{p}$. Based on this, the boundary equation of VO_i can be defined as follows:

$$\mathrm{A}\mathbf{v} = \mathbf{b} \tag{2}$$

where A is a 4×3 matrix, with the values in the first column being $[k_1, k_2, k_3, k_3]$,in the second column being -1, and in the third column being 0; $\mathbf{v} = \mathbf{v} - \mathbf{v}_i^d$ and $\mathbf{b} = [0, 0, -b_1, -b_2]$. The calculation is as follows:

$$\begin{cases} k_1 = (a + \sqrt{\Delta})/b \\ k_2 = (a - \sqrt{\Delta})/b \\ k_3 = -\mathbf{p}_{ix}^d / \mathbf{p}_{iy}^d \\ b_1 = \min(|c + d|, |-c + d|)/t_f \\ b_2 = \max(|c + d|, |-c + d|)/t_0 \end{cases} \tag{3}$$

where $a = -\mathbf{p}_{ix}^d \mathbf{p}_{iy}^d$, $\Delta = (\mathbf{p}_{ix}^d r)^2 + (\mathbf{p}_{iy}^d r)^2 - r^4$, $b = r^2 - (\mathbf{p}_{ix}^d)^2$ $r = r_i^d + r^m$, $c = r\sqrt{k_3^2 + 1}$, $d = -k_3 \mathbf{p}_{ix}^d + \mathbf{p}_{iy}^d$. $\mathbf{p}^m \in \mathbb{R}^3$, $r^m \in \mathbb{R}$ respectively denote the position and bounding radius of the quadrotor; t_0 and t_f represent the starting and ending moments, during which no collision conflict occurs. It is worth noting that the positions and velocities of obstacles are derived from the quadrotor's local coordinate system. By using the boundary equation, a set $\mathrm{A}\mathbf{v} < \mathbf{b}$ can be obtained. Perform the calculations for each observed dynamic obstacle to obtain the VO (lines 3-4).

Secondly, map the velocity space to grid space, which can be achieved through the occupancy grid-based approach. Create a grid plane in the velocity space, record the occupied part as 1 and the unoccupied part as 0, and get the velocity grid map VO_{grid} (lines 8-12). Finally, apply the Signed Distance Field (SDF) Transform to VO_{grid} to obtain GVO (lines 15). In GVO, the value of each grid stores the distance to the nearest obstacle, i.e., if this grid is outside the obstacle, the distance is positive, and the further away from the obstacle the larger the value is; if the grid is inside the obstacle, the distance is negative and the closer to the center of the obstacle the smaller the value is. The construction process of GVO is illustrated in Fig. 2.

Algorithm 1 GVO construction

Input: $\mathbf{p}^d, \mathbf{v}^d, r^d, n, \mathbf{p}^m, r^m$
Output: VO, $\mathrm{VO}_{\mathrm{grid}}$, GVO
1: **function** SETVO($\mathbf{p}^d$, $\mathbf{v}^d$, r^d, $\mathbf{p}^m$, r^m, n)
2: **for** $i = 1 : n$ **do**
3: tmp = CalcVObound($\mathbf{p}_i^d$, $\mathbf{v}_i^d$, r_i^d, $\mathbf{p}^m$, r^m)
4: VO.$push\_back$(tmp)
5: **end for**
6: **end function**

7: **function** SETVOGRID(VO)
8: **for** $i = 1$: size($\mathrm{VO}_{\mathrm{grid}}$) **do**
9: **if** indexToPos(i) $\in$ VO **then**
10: $\mathrm{VO}_{\mathrm{grid}}(\mathrm{i}) = 1$
11: **end if**
12: **end for**
13: **end function**

14: **function** SETGVO($\mathrm{VO}_{\mathrm{grid}}$)
15: SDFTransform($\mathrm{VO}_{\mathrm{grid}}$, GVO)
16: **return** GVO
17: **end function**

3.2 Dynamic Obstacle Avoidance

As depicted in Fig. 2, every time a dynamic obstacle appears in the FOV, its corresponding GVO will be generated. According to the typical VO method, we need to find a velocity that is not in VO to avoid collision. That is,

$$\mathbf{v}^{opt} = \operatorname{argmin} \left(\|\mathbf{v} - \mathbf{v}^m\| \notin \mathrm{VO} \right). \tag{4}$$

In GVO, we solve the following optimization problem:

$$\mathbf{v}^{opt} = \operatorname{argmin} \ \|\mathbf{v} - \mathbf{v}^g\|, \tag{5}$$

where $\mathbf{v}^{opt}$ represents the optimal collision avoidance velocity and $\mathbf{v}^g$ represents the guide velocity. The calculation of $\mathbf{v}^g$ is shown in Alg. 2. First, uniformly sample in the velocity space to generate m random points. Then, obtain the value of all points in GVO. If it exceeds the given threshold v_{safe}, store it in the candidate point list. Finally, identify the point closest to the current velocity as the guidance velocity $\mathbf{v}^g$. As we aim to find a velocity that is optimal in dynamic obstacle avoidance, the core of the cost function (5) is to maintain a safe distance from dynamic obstacles while minimizing changes to the current velocity, where the safe distance is determined by the v_{safe}. Meanwhile, since the gradient of (5) can be calculated directly, the result can be obtained by a closed-form solution.

Algorithm 2 Guide velocity calculation

Input: GVO,v_{safe}
Output: $\mathbf{v}^g$
1: **function** CALCGUIDEVELOCITY()
2: RandPoints = GenerateRandomPoints(m)
3: **for** $i = 1$: m **do**
4: **if** GVO(i) $> v_{safe}$ **then**
5: GPoints.$push\_back$(RandPoints(i))
6: **end if**
7: **end for**
8: $\mathbf{v}^g = \operatorname{argmin} \ \|\mathbf{v} - \mathbf{v}^m\|, \mathbf{v} \in$ GPoints
9: **end function**

4 Autonomous Navigation in Complex Environments

As mentioned in Sect. 3.2, dynamic obstacles can be avoided by changing the current velocity. However, in the actual scene, there are static obstacles in the environment at the same time. Therefore, it is necessary to give a reference trajectory that enables the quadrotor to navigate autonomously in an environment with static obstacles, which needs to satisfy smoothness, dynamic feasibility, and safety [3].

4.1 Uniform B-splines

In this paper, we utilize a B-spline curve as the foundational element for crafting the reference trajectory. This choice is characterized by its degree p, a set of $N + 1$ control points $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_{N-1}, \mathbf{C}_N\}$, and a knot vector $\{u_0, u_1, u_2, ..., u_M\}$. Here, each control point $\mathbf{C}_i$ is a vector in $\mathbb{R}^3$, each u_m is a real number, and the relationship $M = N + p + 1$ holds. For our method, we employ a uniform B-spline, which means the difference $\Delta u = u_{i+1} - u_i$ remains constant throughout. Echoing the approach of [21],

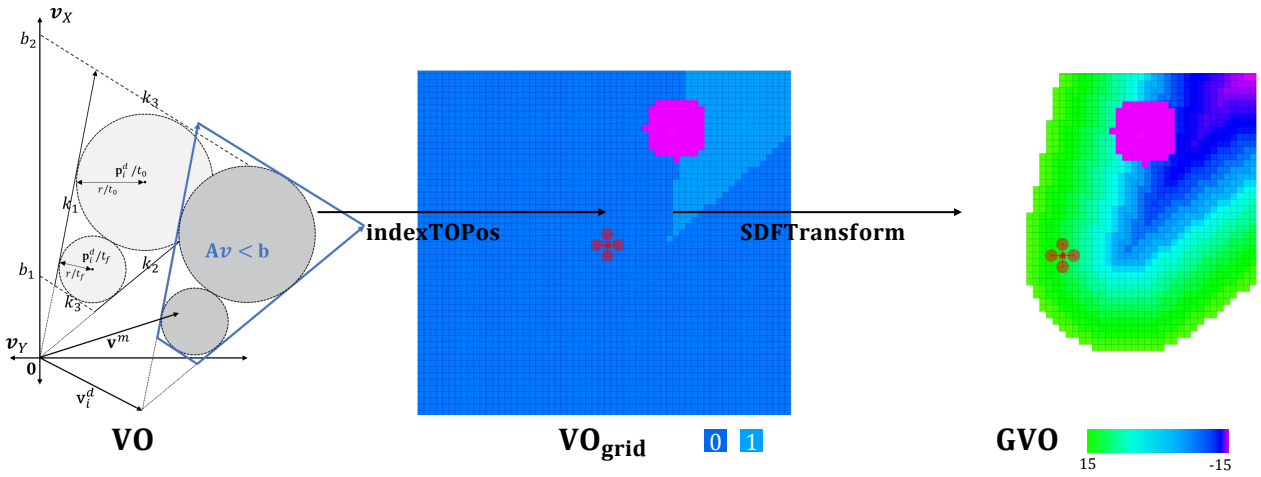

Fig. 2: An explanation of how the GVO is constructed. Firstly, construct the VO by considering the positions and velocities of dynamic obstacles and represent it as a set of inequalities $\mathrm{A}\mathbf{v} < \mathbf{b}$. Secondly, build a grid plane in the velocity space and map the VO set onto this plane to obtain the $\mathrm{VO}_{\mathrm{grid}}$. Finally, transform the $\mathrm{VO}_{\mathrm{grid}}$ into GVO using the SDF.

a linear relationship is established between the knot vector u and the time vector $\{t_0, t_1, ..., t_{M-p}\}$ through the equation $t = (u - u_p)/\beta$, where t ranges from 0 to t_{M-p}, with $t_{M-p} = (M-p)/\beta$. Based on these definitions, the position is articulated as

$$\mathbf{p}(u(t)) = \sum_{i=0}^{N} \mathbf{C}_i \, N_i^p(u(t)), \; N_i^p \in \mathbb{R}, \tag{6}$$

where N_i^p represents the differential basis function.

The property of a B-spline curve extends such that its derivative remains a B-spline. For velocity, we define control points as $\{\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_{N-1}\}$, for acceleration as $\{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_{N-2}\}$, and for jerk as $\{\mathbf{J}_1, \mathbf{J}_2, ..., \mathbf{J}_{N-3}\}$. These control points are derived using Equ. 7, taking into account that $\Delta t = \Delta u/\beta$:

$$\mathbf{V}_i = \frac{\mathbf{C}_{i+1} - \mathbf{C}_i}{\Delta t}, \mathbf{A}_i = \frac{\mathbf{V}_{i+1} - \mathbf{V}_i}{\Delta t}, \mathbf{J}_i = \frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\Delta t}. \tag{7}$$

4.2 B-spline trajectory optimization

In our work, a 3rd-degree B-spline is selected for generating the reference trajectory. The entirety of the trajectory is determined through the optimization of control points, given that the basis functions remain invariant during this optimization phase. Throughout the optimization process, our focus is on optimizing the set of control points $\{\mathbf{C}_p, \mathbf{C}_{p+1}, ..., \mathbf{C}_{N-p}\}$, deliberately excluding the three control points at both the start and the end. This exclusion is due to their role in defining the boundary conditions, which are not subject to alteration during the optimization. The optimization problem is thus established as follows:

$$min \; J_{total} \; = \; \lambda_s J_s + \lambda_c J_c + \lambda_f J_f, \tag{8}$$

where J_s represents the smoothness cost, J_c the collision cost, and J_f the feasibility cost, with λ_s, λ_c, and λ_f serving as the respective weights for these cost terms. The formulation of each term is precisely defined and resolved through the application of the gradient descent method.

4.2.1 Smooth cost

Building on our preceding research [4], we facilitate trajectory smoothing by minimizing the jerk term. Given that B-splines possess the convex hull property, optimizing the control points alone suffices to smooth the entire trajectory. For the sake of simplicity, we disregard Δt, leading to the approach where

$$J_s = \sum_{i=0}^{N-3} \|\mathbf{J}_i\|_2^2 = \sum_{i=0}^{N-3} \|\mathbf{C}_{i+3} - 3\mathbf{C}_{i+2} + 3\mathbf{C}_{i+1} - \mathbf{C}_i\|_2^2. \tag{9}$$

4.2.2 Collision cost

We utilize Euclidean Signed Distance Fields (ESDF) to ensure a collision-free trajectory. The formulation is presented as follows:

$$J_c = \sum_{i=p}^{N-p} \mathcal{F}(\mathbf{C}_i, d_{thr}), \tag{10}$$

where

$$\mathcal{F}(\mathbf{C}_i, d_{thr}) = \begin{cases} (d(\mathbf{C}_i) - d_{thr})^2 & , \; if \; d(\mathbf{C}_i) < d_{thr} \\ 0 & , if \; d(\mathbf{C}_i) > d_{thr} \end{cases}, \tag{11}$$

$d(\mathbf{C}_i)$ denotes the distance from $\mathbf{C}_i$ to the nearest obstacle, with d_{thr} defining the minimum clearance distance from obstacles. The function $\mathcal{F}$ aims to displace the control points away from proximate obstacles towards a safe region. Leveraging the convex hull property of B-splines, this approach ensures the safety of the entire trajectory.

4.2.3 Feasibility cost

The feasibility cost is designed to ensure the trajectory adheres to dynamic feasibility criteria, specifically not exceeding the limits of maximum velocity and acceleration. The

construction of the feasibility cost is outlined as follows:

$$J_f = \sum_j \left(\sum_{i=0}^{N-1} \mathcal{G}\left({\mathbf{V}_i^j}^2 - v_m^2 \right) + \sum_{i=0}^{N-2} \mathcal{G}\left({\mathbf{A}_i^j}^2 - a_m^2 \right) \right), \tag{12}$$

where $\mathcal{G}(x) = max\{x, 0\}^2$, v_m and a_m represent the maximum velocity and acceleration limits respectively, and $j \in \{x, y, z\}$ denotes each spatial dimension.

We employ A-star for front-end path searching and use the result as an initial guess for trajectory planning. Since the initial path is close to the obstacle, it will be pushed away during the trajectory planning phase. Due to the limited sensing range, the quadrotor cannot perceive the full environment. Assume the unknown environment is empty, when new obstacles appear, the planner will generate a new trajectory based on the updated local map to meet the safety requirements, in which the initial state of the new trajectory is the current position $\mathbf{p}_{curr}$, velocity $\mathbf{v}_{curr}$, and acceleration $\mathbf{a}_{curr}$. If there exist dynamic obstacles, the initial state of the re-planned trajectory is given as follows:

$$\begin{bmatrix} 1/6 & 2/3 & 1/6 \\ -\beta/2 & 0 & \beta/2 \\ \beta^2 & -2\beta^2 & \beta^2 \end{bmatrix} \begin{bmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{curr} \\ \mathbf{v}^{opt} \\ \mathbf{a}_{curr} \end{bmatrix}, \tag{13}$$

where $\mathbf{v}^{opt}$ is given by GVO, which ensures that the reference trajectory can avoid the observed dynamic obstacles. The entire process is shown in Fig. 3.

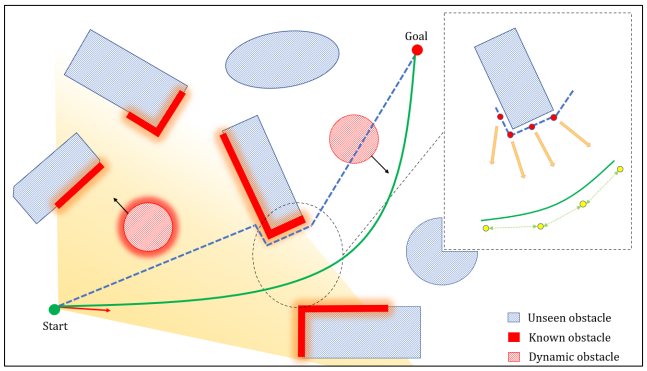

Fig. 3: An illustration of autonomous navigation. The blue dotted line is the initial path and the green curve is the B-spline after optimization. The red points are the initial path points and the yellow points are the control points of the B-spline. The initial path is close to the obstacle while the B-spline is pushed away by the gradient-based optimization, which is safer for the quadrotor. The red arrow indicates the initial velocity, which could avoid the observed dynamic obstacles.

5 Results

5.1 Benchmark Comparisons in Dynamic Environment

As shown in Fig.1, the first scenario consists of multiple dynamic cylindrical obstacles with a motion velocity set about 1 m/s. The parameters are set as $\text{GVO}_{60\times60}$, $m = 200$, and $v_{safe} = 8.0$. Additionally, to simulate real-world scenarios, we define the perception range of the quadrotor as $r = 5m$. We compare the proposed method with several VO-based methods, as shown in Tab. 1. It should be noted that some of the compared methods were originally designed for collision avoidance in multi-robot interactions, where the strategy involved distributing the velocity change for collision avoidance equally between two agents. In our comparison process, the velocity change is solely borne by the quadrotor directly.

As shown in Tab. 1, we compare GVO with other approaches in terms of computation time (**Comp.t**), feasible set size (**Fea.set**), and success rate (**Su.rate**). Approaches [13] and [9] utilize geometric methods to find feasible solutions, which inherently restrict the feasible solutions to be around the VO. These methods naturally produce solutions that are near the surface of obstacles. In the scenario, the VO-based methods have repeatedly failed, and the chances of successful traversal are extremely low. ORCA [14] employs Quadratic Programming (QP) to evade dynamic obstacles by linearizing the feasible set of VO. The velocity feasible set is represented as a collection of linear inequalities obtained by intersecting multiple half-planes, thereby reducing the complexity of the problem. However, the linearization of the feasible set increases the probability of encountering infeasible solutions. Similarly, the ORCA-based approach [7] also encounters similar issues.

Table 1: Comparisons between several VO-based methods

Method	Comp.t(ms)	Fea.set	Su.rate(%)
RVO [13]	0.10	geometric set	59
HRVO [9]	0.16	geometric set	65
ORCA [14]	**0.097**	half-space	55
DCAD [7]	76.6	half-space	50
Proposed (Ours)	0.18	**original**	**89**

Compared to the aforementioned methods, GVO demonstrates a clear advantage in terms of both success rate and the existence of solutions. The feasible solutions provided by GVO maintain a safe distance from the obstacle sets, which significantly enhances the success rate. The average computation time stands at a mere 0.18 milliseconds, ensuring real-time performance even in complex environments. Remarkably, as the number of dynamic obstacles increases, the solution time remains unaffected. Moreover, GVO directly operates on the original VO set and leverages the SDF transformation, thereby preserving the original feasible set. Fig. 4 illustrates a successful traversal using GVO.

5.2 Results in Complex Environments

In the second scenario, we set up an area of approximately $110m \times 20m \times 5m$ and placed both static and dynamic obstacles simultaneously, as shown in Fig. 5. The proposed algorithm parameters are set as follows: $p = 3$, $\beta = 3$, $\lambda_s = 5$, $\lambda_c = 10$, $\lambda_f = 1$, $d_{thr} = 0.5$, $v_m = 1.5m/s$, $a_m = 2.5m/s^2$. The proposed algorithm utilizes the open-source solver **NLOPT**. We compare the proposed algorithm with several state-of-the-art local trajectory replanning methods in terms of replanning time (**Re.t**), success number (**Su.num**), flight time (**Fli.t**) and dynamic obstacle avoidance methods (**Dyn.obs**), and the results are shown in Tab. 2. Method [3] utilizes an incremental gradient field approach for local trajectory replanning, with a frequency of up to 20Hz. In scenarios where dynamic obstacles

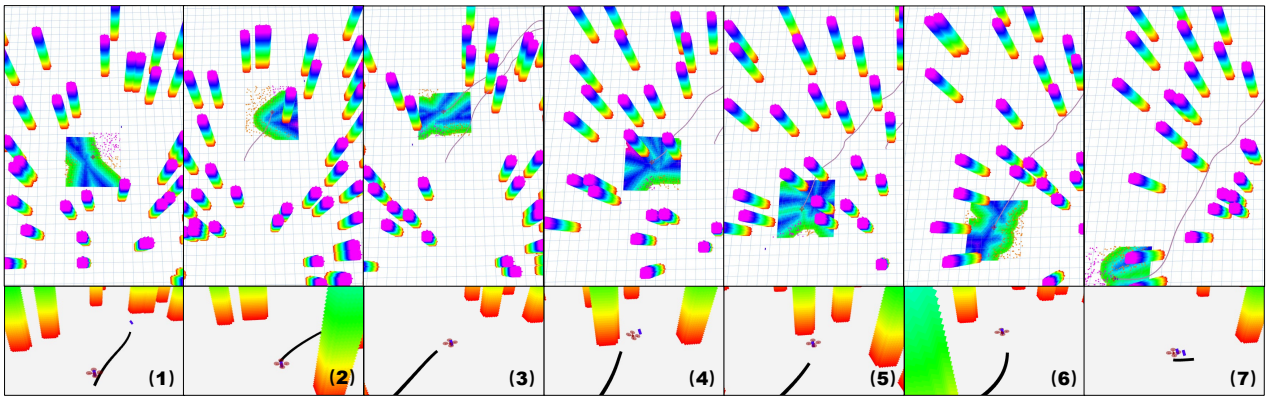

Fig. 4: Dynamic obstacle avoidance based on GVO.

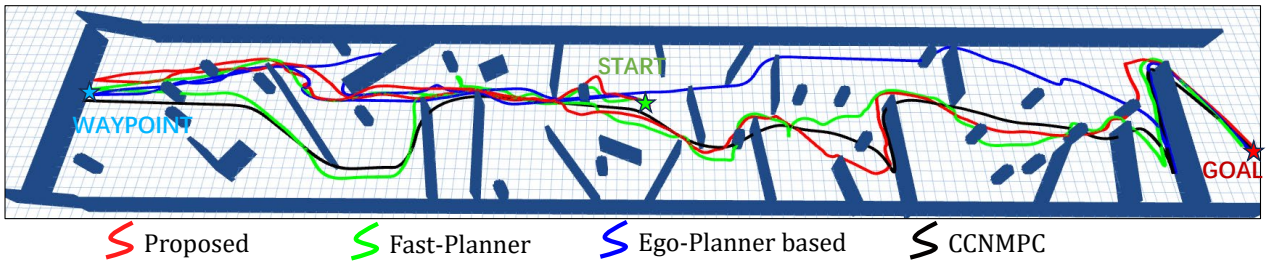

Fig. 5: Trajectories generated by several methods in dynamic complex environments.

have slow motions, fast updates of the local gradient field enable effective avoidance of dynamic obstacles. However, when the velocity of dynamic obstacles is comparable to that of the quadrotor itself, the failure probability increases significantly. Method [6] achieves obstacle avoidance by predicting the trajectories of dynamic obstacles. The online trajectory replanning module ensures that there is no spatial-temporal overlap between the quadrotor's and the obstacle's trajectories. This method directly computes gradients on a grid map, resulting in high computational efficiency. However, the trajectory prediction approach assumes that the motion trajectories of obstacles are smooth and do not undergo sudden changes, which is often difficult to guarantee. In the given scenario, the dynamic obstacles frequently exhibit back-and-forth movement between two points, which causes the prediction module to frequently fail. Method [15] employs a spatial linearization approach to avoid both static and dynamic obstacles. However, constructing a linear solution space forcefully discards feasible sets. In the given scenario, the feasible set constructed by Method [15] is frequently empty, leading to solution failures.

In comparison to the methods mentioned above, our proposed approach utilizes GVO to avoid dynamic obstacles while employing an incremental gradient field to ensure collision-free trajectories with static obstacles in the environment. As shown in Equ. 13, changing the initial velocity of each local trajectory segment only modifies the initial state of the trajectory, without the need for trade-offs in the backend optimization. This strategy allows the quadrotor to prioritize avoidance of observed dynamic obstacles, resulting in significantly improved success rates. Fig. 5 illustrates the trajectories generated by several methods in dynamic complex environments. For additional insights, we encourage readers to access our GitHub: https://github.com/SmartGroupSystems/Dynamic-Planner.

Table 2: Comparison of trajectory generation methods

Method	Re.t(ms)	Su.num	Fli.t(s)	Dyn.obs
Fast-Planner [3]	8.30	1	90.75	None
Ego-Planner based [6]	**6.70**	6	94.4	Pre
CCNMPC [15]	11.25	2	97.12	Linear
Proposed (Ours)	8.95	**8**	**88.56**	GVO

6 Conclusions

In this paper, we propose a method for autonomous navigation in dynamic complex scenarios. We propose a gradient-based velocity obstacle set called GVO, which preserves the original solution space and obtains collision-free velocities more efficiently through optimization techniques. Furthermore, we design a local trajectory planner based on an incremental gradient field and improve motion safety in dynamic complex scenarios by optimizing the initial velocities of the trajectories. The planning framework is thoroughly evaluated through benchmark comparisons. The simulation results demonstrate that the proposed method is capable of effectively handling dynamic complex scenarios and ensuring safe flight. Future work will focus on enhancing the robustness of the algorithm in dealing with unstructured dynamic obstacles and expanding its applicability to swarm robotic systems.

References

[1] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[2] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[3] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[4] R. Li, J. Lyu, A. Wang, R. Yu, D. Wu, and B. Xin, "Flagdroneracing: An autonomous drone racing system," *Unmanned Systems*, 2023.

[5] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2021.

[6] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1966–1973.

[7] S. H. Arul and D. Manocha, "Dcad: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1191–1198, 2020.

[8] ——, "Swarmcco: Probabilistic reactive collision avoidance for quadrotor swarms under uncertainty," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2437–2444, 2021.

[9] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[10] Y. Xu, S. Lai, J. Li, D. Luo, and Y. You, "Concurrent optimal trajectory planning for indoor quadrotor formation switching," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 2, pp. 503–520, 2019.

[11] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[12] S. Roelofsen, D. Gillet, and A. Martinoli, "Collision avoidance with limited field of view sensing: A velocity obstacle approach," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1922–1927.

[13] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[14] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Springer, 2011, pp. 3–19.

[15] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[16] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2682–2688.

[17] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

[18] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[19] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[20] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[21] S. Lai, M. Lan, and B. Chen, "Optimal constrained trajectory generation for quadrotors through smoothing splines," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4743–4750.

[22] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, 2022.

[23] Y. Ren, F. Zhu, W. Liu, Z. Wang, Y. Lin, F. Gao, and F. Zhang, "Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors," *arXiv preprint arXiv:2202.12177*, 2022.

[24] Z. Wang, C. Xu, and F. Gao, "Robust trajectory planning for spatial-temporal multi-drone coordination in large scenes," *arXiv preprint arXiv:2109.08403*, 2021.

[25] Q. Wang, B. He, Z. Xun, C. Xu, and F. Gao, "Gpa-teleoperation: Gaze enhanced perception-aware safe assistive aerial teleoperation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5631–5638, 2022.

[26] 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.

[27] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[28] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4101–4107.