T-STAR: Time-Optimal Swarm Trajectory Planning for Quadrotor Unmanned Aerial Vehicles

Honghao Pan, Mohsen Zahmatkesh, Fatemeh Rekabi-Bana, Farshad Arvin, *Senior Member, IEEE*, and Junyan Hu, *Senior Member, IEEE*

***Abstract*—This paper introduces a time-optimal swarm trajectory planner for cooperative unmanned aerial vehicle (UAV) systems, designed to generate collision-free trajectories for flocking control in cluttered environments. To achieve this goal, model predictive contour control is utilised to generate time-optimal trajectories for each UAV. By demonstrating the differential flatness dynamic equations, the system state constraints are simplified, the algorithm's complexity is reduced, and the overall stability is improved. Additionally, flocking control is achieved among multiple UAVs by applying virtual repulsive and attractive forces. Furthermore, an event-triggered trajectory deconflict strategy for trajectory replanning is considered to resolve multiple trajectory conflicts. Comparative experiments with baseline methods have confirmed that the proposed planner can generate faster and safer trajectories than conventional methods.**

***Index Terms*—Unmanned aerial vehicles, optimisation, model predictive contour control, distributed trajectory planning, collision avoidance, swarm robotics.**

I. Introduction

Unmanned aerial vehicles (UAVs) are widely recognised for their high agility, performing exceptionally well in various scenarios, such as trajectory planning in unknown environments [1]–[3], and high-speed competitions in known environments [4]. There is also an increased research interest focusing on formation control for multiple UAVs where a team of UAVs is required to form a desired pattern in a given workspace [5]. However, most studies on formation control often compromise the agility of drones [6]–[8], which may cause some limitations in emergency situations like search and rescue [9]. Therefore, this paper aims to develop a multi-UAV trajectory planner that focuses on enhancing agility, enabling aerial clusters to rapidly adapt and manoeuvre through cluttered environments.

The high-agility UAV swarm trajectory planner faces three significant challenges. Firstly, the planner is designed to navigate complex environments by avoiding dense obstacles and dynamically adjusting trajectories to prevent collisions with other UAVs while maintaining optimal flight speeds [10]–[12]. Additionally, generating smooth and high-quality trajectories is crucial for stable flight, allowing quadrotor drones to operate at their performance limits. Furthermore, another critical challenge lies in resolving trajectory conflicts within the swarm, requiring efficient distributed strategies to ensure safe manoeuvring in highly dynamic and constrained environments [13].

The authors are with the Department of Computer Science, Durham University, Durham, UK. (e-mail: {honghao.pan; mohsen.zahmatkesh; fatemeh.rekabi-bana; farshad.arvin; junyan.hu}@durham.ac.uk)

In this paper, these challenges are addressed by designing a Time-optimal Swarm TrAjectory planner (T-STAR). A novel distributed swarm trajectory planning strategy that significantly improves the efficiency of task completion is proposed. This method consists of three main parts. Firstly, the planner uses local perception data for the convex decomposition of a cluttered space and builds a time-optimal model based on the cost function from Model Predictive Contour Control (MPCC) [14]. Unlike established approaches, the proposed strategy does not use the constraints of MPCC to update control variables because the nonlinear dynamics during the high-velocity flight of quadrotor drones significantly increase the computational load. The differential flatness is applied to represent the drone's position, yaw angle, and their finite derivatives as control variables, updating them by the smoothness of flat output, thereby preventing numerical instabilities associated with MPCC-based state updates. Then, by predetermining the relative positions between individuals, a resilient distributed strategy is established in which each UAV exerts repulsive forces on the other UAVs while generating attractive forces to maintain swarm collaboration. Finally, an Event-triggered Trajectory Deconflict strategy (ETD) is designed to avoid trajectory conflicts between multiple UAVs. The contributions of this paper are as follows.

1) A new time-optimal trajectory planning method is proposed for quadrotor drones based on MPCC, utilising the differential flatness to reduce model complexity and enhance numerical stability.
2) A novel decoupled swarm planning strategy is designed, which maintains high swarm speeds while keeping formation errors bounded. This is achieved by incorporating slack variables related to formation errors, enabling a flexible trade-off between swarm speed and formation quality.
3) A new event-triggered mechanism strategy is introduced that allows individual UAVs to locally replan their trajectories, using information from neighbouring UAVs when a trajectory conflict is detected, in order to generate a collision-free swarm trajectory.
4) Extensive benchmark testing, simulations, and real-world experiments are conducted to validate the effectiveness of our method. Compared to conventional trajectory optimisation-based control methods, our approach can generate trajectories that are shorter in duration and faster in speed.

II. Related Work

A. Differentially Flat Control of Quadrotor Drones

Differential flatness was proposed by Fliess et al. [15] and has been widely used in control systems [16]–[18]. This property indicates that all states and inputs of a flat system can be determined by flat outputs and their finite derivatives. Mellinger and Kumar demonstrated the flatness of quadrotor aircraft model, defining position and yaw angle as the flat outputs [19]. Therefore, the planning of states for a quadrotor can be transformed into planning based on flat outputs. For instance, a fault-tolerant trajectory planning algorithm was proposed by combining differential flatness and the uncertainties of the quadrotor model [20]. Differential flatness was applied to model predictive control, resulting in a trajectory planning method based on quadratic programming and feedforward linearisation [21]. However, these methods do not consider the aerodynamic effects associated with the high-speed flight of quadrotors. Faessler et al. proved the flatness of quadrotors affected by linear rotor drag effects, with position and heading remaining as the flat outputs [22]. An aerodynamic model for fixed-pitch quadrotors was developed in [23], with the introduction of a thrust controller to improve tracking accuracy. In [17], the incremental nonlinear dynamic inversion method was proposed to handle aerodynamic drag, utilising optical sensors to measure motor speed for closed-loop control.

However, the above methods focus on the motion planning of a single UAV and do not consider the complexity and coordination requirements of multi-UAV systems. As task complexity increases, multi-UAV systems need to achieve efficient and reliable coordinated motion in dynamic environments, which imposes higher demands on trajectory planning algorithms.

B. Model Predictive Control for Trajectory Planning

Model Predictive Control (MPC) is widely studied for its ability to predict future behaviours in highly uncertain environments and to handle complex constraints [24]–[26]. MPC solves an optimisation problem over a finite time horizon to obtain a locally optimal solution, thereby iteratively updating the optimal state of the control system.

However, MPC requires significant computational resources. This is particularly challenging when establishing the nonlinear dynamics models for quadrotors, making nonlinear MPC computationally intensive. Therefore, many studies have reduced the computational complexity by linearising the nonlinear MPC (NMPC) [27]. With the development of nonlinear optimisation solvers, NMPC has become a popular control method for quadrotors. In [28], a time-optimal trajectory planning method using NMPC is proposed by setting the quadrotor's progression along the trajectory as waypoints. The NMPC model, which used Gaussian processes, was established to compensate for aerodynamics in high-speed flight [29]. However, traditional NMPC heavily relies on reference trajectories, and generating these trajectories is a complex task.

MPCC is a variant of NMPC that expands the search space of MPC by maximising the progress along a reference trajectory and minimising the distance to reference points [30]. Therefore, MPCC reduces the dependence on reference trajectories to enhance the robustness of the tracking system effectively. Liniger et al. proposed an MPCC model for autonomous racing cars, integrating trajectory generation with tracking [31]. The feasibility of the MPCC approach was demonstrated in racing tasks for quadrotors [14]. Additionally, collision-avoidance constraints were incorporated in MPCC to generate safe trajectories [32].

Although these approaches provide reliable solutions for agile flight in quadrotor drones, the high computational resources required by NMPC remain unresolved. Moreover, the issue of numerical stability in NMPC still persists.

C. Distributed Trajectory Planning

Distributed swarm trajectory planning is a key research area in multi-UAV systems, involving obstacle avoidance [33], collaboration [34], and formation [35] of multiple UAVs in complex environments. In [36], a velocity obstacle strategy was introduced for avoiding dynamic obstacles. The core of this strategy is to construct a velocity obstacle space based on the velocity and position of obstacles, with the UAV planning velocity outside this space. Van den Berg et al. applied and extended the velocity obstacle strategy to multi-UAV systems, allowing UAVs to plan collision-free trajectories [37]. However, as the number of obstacles increases, the computational load of the velocity obstacle method significantly increases. Additionally, the velocity obstacle strategy focuses on velocity planning, which may lead to non-smooth trajectories.

To address this issue, optimisation-based multi-UAV trajectory planning has been extensively explored [38]. Zhou et al. embedded the buffered Voronoi cell method into quadratic programming to achieve mutual collision avoidance in multi-robot systems [39]. The predicted collision constraint was added in [40] to distributed MPC to generate collision-free trajectories. A gradient-based distributed trajectory planning framework was proposed to generate safe trajectories in real-time [12]. To ensure trajectory smoothness, Bernstein polynomials are applied to generate multi-UAV trajectories [41]. However, these methods typically do not consider time optimisation, leading to paths that may not be time-efficient. Additionally, they do not fully utilise the agility and high manoeuvrability of quadrotor drones, limiting performance in dynamic and complex environments.

To overcome these shortcomings, this paper aims to restructure the trajectory planner and design a novel cost function to balance the stability and agility of the drone swarm. Additionally, we aim to adopt a distributed optimisation approach to ensure effective collision avoidance among multiple UAVs.

III. Preliminaries

A. Quadrotor Model

The state space of a quadrotor is defined with respect to two reference frames, the world frame W and the body frame B, as shown in Fig. 1. The state variables is $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{\mathrm{W}} & \boldsymbol{v}_{\mathrm{W}} & {}^{\mathrm{W}}\boldsymbol{R}_{\mathrm{B}} & \boldsymbol{\omega}_{\mathrm{BW}} \end{bmatrix} \in \mathbb{R}^{3\times 6}$, where $\boldsymbol{p}_{\mathrm{W}} = \begin{bmatrix} \mathrm{x}_{\mathrm{W}} & \mathrm{y}_{\mathrm{W}} & \mathrm{z}_{\mathrm{W}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^3$ is the position, $\boldsymbol{v}_{\mathrm{W}} \in \mathbb{R}^3$ is the linear

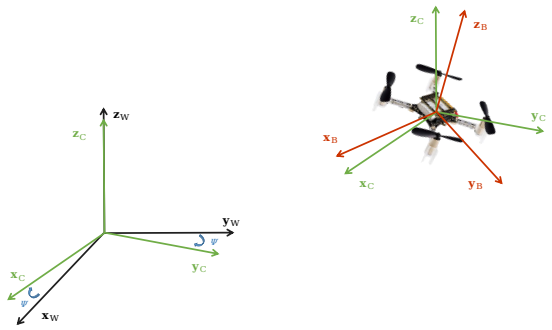

Fig. 1. A schematic of the reference frames. Includes world frame W, intermediate frame C, and body frame B.

velocity vector, ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{B}} = [\mathbf{x}_{\mathrm{B}} \quad \mathbf{y}_{\mathrm{B}} \quad \mathbf{z}_{\mathrm{B}}] \in \mathbb{SO}^3$ is rotation matrix from body frame to world frame, and $\boldsymbol{\omega}_{\mathrm{BW}} \in \mathbb{R}^3$ is the angular velocity vector in body frame. And control variable is $\boldsymbol{u}_c = [f \quad \boldsymbol{\tau}^{\mathrm{T}}] \in \mathbb{R}^4$, where $f \in \mathbb{R}^+$ is thrust and $\boldsymbol{\tau} \in \mathbb{R}^3$ is quadrotor torque. Consider the dynamic model of a quadrotor with linear rotor drag effects [22]. To improve readability, the subscripts and superscripts of each element in the state variables $\boldsymbol{x}$ have been omitted, for example $\mathbf{R} = {}^{\mathrm{W}}\boldsymbol{R}_{\mathrm{B}}$, the dynamic model can be written as

$$\dot{\boldsymbol{p}} = \mathbf{v} \tag{1}$$

$$\dot{\mathbf{v}} = -g\mathbf{e}_{\mathrm{z}} + \frac{1}{m}\mathbf{R}\mathbf{f}_{\mathrm{B}} - \mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}}\mathbf{v} \tag{2}$$

$$\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}} \tag{3}$$

$$\dot{\boldsymbol{\omega}} = \mathcal{I}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathcal{I}\boldsymbol{\omega} - \mathbf{F}_d\mathbf{R}^{\mathrm{T}}\mathbf{v} - \mathbf{T}_u\boldsymbol{\omega}\right), \tag{4}$$

where $\mathbf{f}_{\mathrm{B}} = [0 \ 0 \ f]^{\mathrm{T}} \in \mathbb{R}^3$ is the total rotor thrust, $\boldsymbol{e}_{\mathrm{z}} = [0 \ 0 \ 1]^{\mathrm{T}}$ from the orthonormal world frame $\{\boldsymbol{e}_{\mathrm{x}}, \boldsymbol{e}_{\mathrm{y}}, \boldsymbol{e}_{\mathrm{z}}\}$, $\mathbf{D} = \mathrm{diag}\,(d_{\mathrm{x}}, d_{\mathrm{y}}, d_{\mathrm{z}}) \in \mathbb{R}^{3\times3}$ represents a constant diagonal matrix with rotor drag coefficients, $\hat{\boldsymbol{\omega}}$ is a skew symmetric matrix from $\boldsymbol{\omega}$, $\mathcal{I} \in \mathbb{R}^{3\times3}$ is inertia matrix in body frame, and $\mathbf{F}_d \in \mathbb{R}^{3\times3}$ and $\mathbf{T}_u \in \mathbb{R}^{3\times3}$ are constant matrices.

B. Differential Flatness

Quadrotor dynamic model with rotor drag is proven to have differential flatness properties [22]. Therefore, the following shows that the model of the quadrotor with linear rotor drag can be uniquely determined by the flat output $\boldsymbol{z} = [\boldsymbol{p}^{\mathrm{T}} \quad \psi]^{\mathrm{T}}$.

To derive the rotation matrix $\mathbf{R}$, reformulate (2) as follow

$$\begin{aligned}\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} - \frac{1}{m}f\mathbf{z}_{\mathrm{B}} + d_{\mathrm{x}}\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v}\mathbf{x}_{\mathrm{B}} \\ + d_{\mathrm{y}}\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v}\mathbf{y}_{\mathrm{B}} + d_{\mathrm{z}}\mathbf{z}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v}\mathbf{z}_{\mathrm{B}} = 0.\end{aligned} \tag{5}$$

Because $\mathbf{R}$ is a unitary matrix and its elements are orthonormal, left-multiplying (5) by $\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}$ and $\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}$ respectively

$$\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}\left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{x}}\mathbf{v}\right) = 0 \tag{6}$$

$$\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{y}}\mathbf{v}\right) = 0. \tag{7}$$

The quadrotor rotation order is ZYX, and the intermediate frame C is obtained by rotating the world frame by yaw angle ψ, as illustrated in Fig. 1, with the intermediate rotation ${}^{\mathrm{W}}\boldsymbol{R}_{\mathrm{C}} = [\mathbf{x}_{\mathrm{C}} \quad \mathbf{y}_{\mathrm{C}} \quad \mathbf{z}_{\mathrm{C}}]$. Given the yaw angle ψ, we get

$$\mathbf{x}_{\mathrm{C}} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \end{bmatrix}^{\mathrm{T}} \tag{8}$$

$$\mathbf{y}_{\mathrm{C}} = \begin{bmatrix} -\sin(\psi) & \cos(\psi) & 0 \end{bmatrix}^{\mathrm{T}}. \tag{9}$$

The rotation order shows that $\mathbf{y}_{\mathrm{C}}$ is orthogonal to $\mathbf{x}_{\mathrm{B}}$. so $\mathbf{x}_{\mathrm{B}}$ can be determined by (6) and (9)

$$\mathbf{x}_{\mathrm{B}} = \frac{\mathbf{y}_{\mathrm{C}} \times \left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{x}}\mathbf{v}\right)}{\left\|\mathbf{y}_{\mathrm{C}} \times \left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{x}}\mathbf{v}\right)\right\|}. \tag{10}$$

Since $\mathbf{x}_{\mathrm{B}}, \mathbf{y}_{\mathrm{B}}$, and $\mathbf{z}_{\mathrm{B}}$ are mutually orthogonal, consider the equation (7), the $\mathbf{y}_{\mathrm{B}}$ and $\mathbf{z}_{\mathrm{B}}$ can be formulated as follows

$$\mathbf{y}_{\mathrm{B}} = \frac{\mathbf{x}_{\mathrm{B}} \times \left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{y}}\mathbf{v}\right)}{\left\|\mathbf{x}_{\mathrm{B}} \times \left(\dot{\mathbf{v}} + g\mathbf{e}_{\mathrm{z}} + d_{\mathrm{y}}\mathbf{v}\right)\right\|} \tag{11}$$

$$\mathbf{z}_{\mathrm{B}} = \mathbf{x}_{\mathrm{B}} \times \mathbf{y}_{\mathrm{B}}. \tag{12}$$

Thus, $\boldsymbol{R} = [\mathbf{x}_{\mathrm{B}} \quad \mathbf{y}_{\mathrm{B}} \quad \mathbf{z}_{\mathrm{B}}]$ is obtained. And from (2), $\mathbf{f}_{\mathrm{B}} = f\boldsymbol{e}_{\mathrm{z}}$, rotor thrust f is,

$$f = \left\|(\boldsymbol{R}\mathbf{e}_{\mathrm{z}})^{\mathrm{T}}(m\dot{\mathbf{v}} + mg\mathbf{e}_{\mathrm{z}} + m\mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}}\mathbf{v})\right\|$$

To demonstrate the intrinsic connection between body rate $\boldsymbol{\omega}$ and flat output $\boldsymbol{z}$, take the derivative of (2) and reformulate

$$\begin{aligned}\ddot{\mathbf{v}} - \frac{1}{m}\mathbf{R}\hat{\boldsymbol{\omega}}f\mathbf{e}_{\mathrm{z}} - \frac{1}{m}\mathbf{R}\dot{f}\mathbf{e}_{\mathrm{z}} + \mathbf{R}\hat{\boldsymbol{\omega}}\mathbf{D}\mathbf{R}^{\mathrm{T}}\mathbf{v} + \mathbf{R}\mathbf{D}\hat{\boldsymbol{\omega}}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{v} \\ + \mathbf{R}\mathbf{D}\mathbf{R}^{\mathrm{T}}\dot{\mathbf{v}} = 0.\end{aligned} \tag{13}$$

Construct a system of equations via left-multiplying (13) by $\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}$ and $\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}$,

$$\begin{aligned}&\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}\ddot{\mathbf{v}} - \omega_{\mathrm{y}}\frac{1}{m}f + \omega_{\mathrm{y}}\left(d_{\mathrm{z}} - d_{\mathrm{x}}\right)\mathbf{z}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v} + \omega_{\mathrm{z}}\left(d_{\mathrm{x}} - d_{\mathrm{y}}\right)\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v} \\ &+ d_{\mathrm{x}}\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}\dot{\mathbf{v}} = 0 \\ &\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\ddot{\mathbf{v}} + \omega_{\mathrm{x}}\frac{1}{m}f + \omega_{\mathrm{x}}\left(d_{\mathrm{y}} - d_{\mathrm{z}}\right)\mathbf{z}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v} + \omega_{\mathrm{z}}\left(d_{\mathrm{x}} - d_{\mathrm{y}}\right)\mathbf{x}_{\mathrm{B}}^{\mathrm{T}}\mathbf{v} \\ &+ d_{\mathrm{y}}\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\dot{\mathbf{v}} = 0\end{aligned} \tag{14}$$

Given that ω_{z} is the projection of $\dot{\mathbf{x}}_{\mathrm{B}}$ onto $\mathbf{y}_{\mathrm{B}}$,

$$\omega_{\mathrm{z}} = \boldsymbol{\omega}^{\mathrm{T}}\mathbf{z}_{\mathrm{B}} = \boldsymbol{\omega}^{\mathrm{T}}(\mathbf{x}_{\mathrm{B}} \times \mathbf{y}_{\mathrm{B}}) = \mathbf{y}_{\mathrm{B}}^{\mathrm{T}}(\boldsymbol{\omega} \times \mathbf{x}_{\mathrm{B}}) = \mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\dot{\mathbf{x}}_{\mathrm{B}}. \tag{15}$$

Since $\mathbf{x}_{\mathrm{B}}$ is orthogonal to $\mathbf{y}_{\mathrm{C}}$ and $\mathbf{z}_{\mathrm{B}}$, $\mathbf{x}_{\mathrm{B}} = \frac{\mathbf{y}_{\mathrm{C}} \times \mathbf{z}_{\mathrm{B}}}{\|\mathbf{y}_{\mathrm{C}} \times \mathbf{z}_{\mathrm{B}}\|}$, then we get the derivative of vector $\mathbf{x}_{\mathrm{B}}$,

$$\dot{\mathbf{x}}_{\mathrm{B}} = \frac{\dot{\tilde{\mathbf{x}}}_{\mathrm{B}}}{\|\tilde{\mathbf{x}}_{\mathrm{B}}\|} - \tilde{\mathbf{x}}_{\mathrm{B}}\frac{\tilde{\mathbf{x}}_{\mathrm{B}}^{\mathrm{T}}\dot{\tilde{\mathbf{x}}}_{\mathrm{B}}}{\|\tilde{\mathbf{x}}_{\mathrm{B}}\|^3}, \ \text{with } \tilde{\mathbf{x}}_{\mathrm{B}} = \mathbf{y}_{\mathrm{C}} \times \mathbf{z}_{\mathrm{B}}. \tag{16}$$

Based on the $\tilde{\mathbf{x}}_{\mathrm{B}}$ perpendicular to $\mathbf{y}_{\mathrm{B}}$, $\mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\tilde{\mathbf{x}}_{\mathrm{B}} = 0$, rewrite (15),

$$\begin{aligned}\omega_{\mathrm{z}} &= \mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\frac{\dot{\tilde{\mathbf{x}}}_{\mathrm{B}}}{\|\tilde{\mathbf{x}}_{\mathrm{B}}\|} = \mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\frac{\dot{\mathbf{y}}_{\mathrm{C}} \times \mathbf{z}_{\mathrm{B}} + \mathbf{y}_{\mathrm{C}} \times \dot{\mathbf{z}}_{\mathrm{B}}}{\|\tilde{\mathbf{x}}_{\mathrm{B}}\|} \\ &= \mathbf{y}_{\mathrm{B}}^{\mathrm{T}}\frac{(-\dot{\psi}\mathbf{x}_{\mathrm{C}}) \times \mathbf{z}_{\mathrm{B}} + \mathbf{y}_{\mathrm{C}} \times (\omega_{\mathrm{y}}\mathbf{x}_{\mathrm{B}} - \omega_{\mathrm{x}}\mathbf{y}_{\mathrm{B}})}{\|\tilde{\mathbf{x}}_{\mathrm{B}}\|} \\ &= \frac{\dot{\psi}\mathbf{x}_{\mathrm{C}}^{\mathrm{T}}\mathbf{x}_{\mathrm{B}} + \omega_{\mathrm{y}}\mathbf{y}_{\mathrm{C}}^{\mathrm{T}}\mathbf{z}_{\mathrm{B}}}{\|\mathbf{y}_{\mathrm{C}} \times \mathbf{z}_{\mathrm{B}}\|}.\end{aligned} \tag{17}$$

The body rate $\boldsymbol{\omega}$ can be obtained by solving (14) and (17). To calculate the torque $\boldsymbol{\tau}$ using the derivative of $\boldsymbol{\omega}$,

$$\boldsymbol{\tau} = \mathcal{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathcal{I}\boldsymbol{\omega} + \mathbf{F}_d \mathbf{R}^{\mathrm{T}} \mathbf{v} + \mathbf{T}_u \boldsymbol{\omega}. \tag{18}$$

This section proves that the state space $\boldsymbol{x}$ and control input can be represented as an algebraic combination of the flat output $\boldsymbol{z}$ and its finite number of derivatives.

C. Multi-stage Polynomial Trajectory

Multi-stage polynomial trajectories are widely utilised in quadrotor trajectory planning to ensure the smoothness of trajectories [42]. Considering the K-stage polynomial trajectories,

$$\boldsymbol{z}(t) = \begin{cases} \sum_{k_p=0}^{K_p} \beta_{k_p}^{1} (t - t_1)^{k_p} & t \in [t_0, t_1] \\ \vdots \\ \sum_{k_p=0}^{K_p} \beta_{k_p}^{k+1} (t - t_k)^{k_p} & t \in [t_k, t_{k+1}] \\ \vdots \\ \sum_{k_p=0}^{K_p} \beta_{k_p}^{K} (t - t_{K-1})^{k_p} & t \in [t_{K-1}, t_K], \end{cases} \tag{19}$$

where k_p is degree, $\boldsymbol{\beta}_{k_p}^{k}$ is k_pth coefficient, t_k and t_{k+1} represents the time boundaries of kth trajectory respectively.

D. Problem Statement

Based on the above conclusions, this paper concentrates on the time-optimal trajectory planning problem for distributed UAV swarm in flat systems with high obstacle density unknown environments. The optimal control problem can be formulated within the T-STAR framework as follows,

$$\min_{\boldsymbol{z}(t)} \sum_{k=1}^{K} \left\{ \mathbf{J}_{opt}(\boldsymbol{z}(t_k)) + \lambda_s e_{s,k}^2 \right\} \tag{20a}$$

$$\text{s.t. } \mathcal{F}(\boldsymbol{z}(t_k), \ldots, \boldsymbol{z}^{(n)}(t_k)) \preceq \mathbf{0} \quad \forall k = 1, \ldots, K \tag{20b}$$

$$\|\mathbf{L}(\boldsymbol{z}(t_k))\|_F^2 \leq e_{s,k} \quad \forall k = 1, \ldots, K \tag{20c}$$

$$\boldsymbol{z}(t_k) \in \mathcal{H}_k \quad \forall k = 1, \ldots, K \tag{20d}$$

$$\boldsymbol{z}^{[n-1]}(t_0) = \bar{\boldsymbol{z}}_{init}, \quad \boldsymbol{z}^{[n-1]}(t_K) = \bar{\boldsymbol{z}}_{final}, \tag{20e}$$

where $\mathbf{J}_{opt}$ is cost function for optimising speed. Augmented flat output $\boldsymbol{z}^{[n]}(t_k) = (\boldsymbol{z}(t_k)^{\mathrm{T}}, \dot{\boldsymbol{z}}(t_k)^{\mathrm{T}}, \ldots, \boldsymbol{z}^{(n)}(t_k)^{\mathrm{T}})^{\mathrm{T}}$, n is a constant, and $\boldsymbol{z}^{(n)}(t_k)$ represents the n-th order derivative of the flat output $\boldsymbol{z}(t_k)$. Laplacian matrix $\mathbf{L}(\boldsymbol{z}(t_k))$ of each drone, defined based on flat output $\boldsymbol{z}(t_k)$, formalises the distributed UAV swarm. Slack variable $e_{s,k}$ indicates the error between the UAV swarm and the desired formation. λ_s is a weight. $\mathcal{F}$ represents the constraints on the smoothness and continuity of the trajectory. $\bar{z}_{init}$ and $\bar{z}_{final}$ is boundary conditions. The solution space $\mathcal{H}_k = \mathcal{H}_{p_k} \times \mathcal{H}_\psi$ consists of the feasible region $\mathcal{H}_\psi = [0, 2\pi]$ and the the collision-free space $\mathcal{H}_{p_k}$, which can be described by an m-faced convex polyhedron,

$$\mathcal{H}_{p_k} = \left\{ \boldsymbol{p}(t_k) \mid \bar{\mathbf{A}}_k^{\mathrm{T}} \boldsymbol{p}(t_k) - \bar{\mathbf{B}}_k < 0 \right\} \quad k = 1, \ldots, K,$$

where $\bar{\mathbf{A}}_k \in \mathbb{R}^{m \times 3}$ and $\bar{\mathbf{B}}_k \in \mathbb{R}^m$ are coefficients matrices that vary with t_k.

To eliminate additional optimisation variables $e_{s,k}$, the optimisation model (20a) is reformulated as follows,

$$\min_{\boldsymbol{z}(t)} \sum_{k=1}^{K} \left\{ \mathbf{J}_{opt}(\boldsymbol{z}(t_k)) + \lambda_s \mathbf{J}_s(\mathbf{L}_k) \right\} \tag{21a}$$

$$\text{s.t. } \mathcal{F}(\boldsymbol{z}(t_k), \ldots, \boldsymbol{z}^{(n)}(t_k)) \preceq \mathbf{0} \quad \forall k = 1, \ldots, K \tag{21b}$$

$$\boldsymbol{z}(t_k) \in \mathcal{H}_k \quad \forall k = 1, \ldots, K \tag{21c}$$

$$\boldsymbol{z}^{[n-1]}(t_0) = \bar{\boldsymbol{z}}_{init}, \quad \boldsymbol{z}^{[n-1]}(t_K) = \bar{\boldsymbol{z}}_{final}, \tag{21d}$$

where $\mathbf{J}_s(\mathbf{L}_k) = \|\mathbf{L}(\boldsymbol{z}(t_k))\|_F^2$ is cost function for flocking control.

Therefore, a trajectory planner designed for this paper aims to solve the following three problems: 1. Finding collision-free space in unknown environments with abundant obstacles and dynamic UAVs; 2. Generating time-optimal trajectories for a single UAV planned on the flat output; and 3. Considering flocking control algorithms for cooperative UAVs.

Remark 1: In this paper, the concept of time optimality builds upon the advanced work on MPCC [14]. We effectively extend this concept by introducing swarm flocking systems, which aim to increase the flight speed of the UAV swarm to reduce mission completion time, thereby achieving swarm time optimality. However, simultaneously optimising both the swarm's speed and formation error by enforcing constraint (20c) as a hard constraint and fixing e_s to zero can render the optimisation model (20a) easily infeasible. Therefore, unlike most works that focus on formation control of UAV swarms, the slack variable e_s essentially relaxes the formation error and adjusts e_s through the cost function to mediate the trade-off between speed and formation error, with speed being guaranteed. The weight parameter λ_s represents the importance of the formation error and is user-defined. A smaller λ_s can increase the swarm's speed; conversely, a larger λ_s reduces e_s by sacrificing speed.

Remark 2: The terminal time t_K in the optimal control framework does not necessarily coincide with the actual task execution time. This distinction arises because the UAV could reach the terminal state before t_K. Consequently, the time at which a UAV arrives at the target point, referred to as the task completion time, can be shorter than the given terminal time.

Remark 3: Coefficients matrices $\bar{\mathbf{A}}_k$ and $\bar{\mathbf{B}}_k$ are updated based on the environment perceived by the UAV. Furthermore, before each optimisation of the control problem (21a), the values of $\bar{\mathbf{A}}_k$ and $\bar{\mathbf{B}}_k$ are obtained using the method described in Subsection IV-D.

E. System Structure

In this paper, we propose a method called T-STAR, which consists of five distinct functional modules, as shown in Fig. 2. First, quadrotor UAVs exchange adjacent trajectory information through a distributed communication network and establish a safe flight corridor (SFC) based on Support Vector Machine (SVM) $H_{svm-sfc}$. Then, the Global Planner uses the unfeasible shortest path obtained by Jump Point Search and the collision-free space $H_{svm-sfc}$ to generate the reference

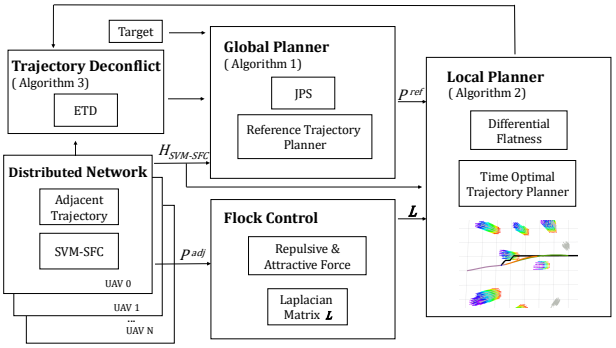

Fig. 2. The structure diagram demonstrates five modules of the global planner, distributed communication network, flocking control, local planner, and trajectory deconflict.

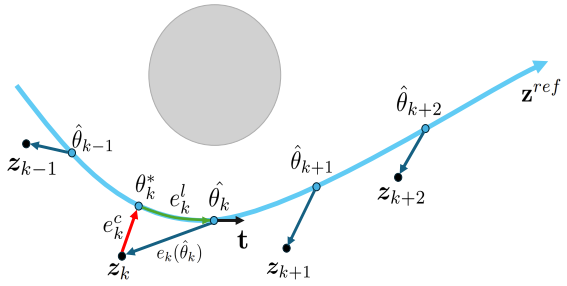

Fig. 3. Model Predictive Contouring Control. Minimising contour error e_k^c and lag error e_k^l for reference trajectory tracking.

trajectory. Next, the UAVs establish a Laplacian matrix for flocking control based on the adjacent trajectories. After calculating the reference trajectory $\mathbf{p}^{ref}$, collision-free space $H_{svm-sfc}$, and matrix $\mathbf{L}(\boldsymbol{z}(t))$, the Local Planner generates the time-optimal trajectory. If there are trajectory conflicts, replanning is conducted in the ETD to get the conflict-free trajectories.

IV. Low-Level Optimal Trajectory Planning

A. Trajectory Optimal Control

A general form of the continuous-time cost functional is,

$$\mathbf{J}(\boldsymbol{z}^{(n)}(t)) = \int_{t_0}^{t_K} \left[(\boldsymbol{z}^{(n)}(t))^2 + \lambda \right] dt, \tag{22}$$

where λ is a weight factor for time optimal. Time interval $\Delta_T := t_K - t_0$ is a constant defined by the user. From this, $\int_{t_0}^{t_K} \lambda \, dt = \lambda \Delta_T$ also is a constant, the discrete cost function $\mathbf{J}$ simplifies to

$$\mathbf{J}(\boldsymbol{z}^{(n)}(t)) = \sum_{k=1}^{K} (\boldsymbol{z}^{(n)}(t_k))^2. \tag{23}$$

This cost function is applied in Subsection IV-E to generate the reference trajectory. According to Pontryagin's Maximum Principle, under the constraints (21b) and (21d), the optimal control (23) is bang-bang control. Therefore, a closed-form solution for the optimal trajectory can be obtained without obstacles. And, in environments with a high density of obstacles, the trajectory is obtained using numerical methods.

However, the trajectory generated by (23) is not time-optimal. In the next subsection, a novel cost function is designed for the time-optimal trajectory.

B. Unconstrained Time-Optimal Control

In contrast to optimal control (23), T-STAR is designed to generate faster trajectories to exploit the high-speed capability of the quadrotor. Based on differential flatness, the flat output $\boldsymbol{z}$ is planned instead of the full state vector. Our method considers tracking the reference trajectory $\mathbf{z}^{ref}$ while maximising the speed. As illustrated in Fig. 3. The pose of the quadrotor is $\boldsymbol{z}_k = \boldsymbol{z}(t_k)$ and $\theta_k^* = \begin{bmatrix} \mathbf{p}_x^{ref}(\theta_k^*) & \mathbf{p}_y^{ref}(\theta_k^*) & \mathbf{p}_z^{ref}(\theta_k^*) & \mathbf{p}_\psi^{ref}(\theta_k^*) \end{bmatrix}^T$ represents the vertical projection point of $\boldsymbol{z}_k$ on reference path $\mathbf{z}^{ref}(\theta)$, defining *contour error* e_k^c is

$$e_k^c = \min_{\theta} \left\| \boldsymbol{z}_k - \mathbf{z}^{ref}(\theta) \right\| , \tag{24}$$

where $\mathbf{z}^{ref}(\theta)$ is an arc length $\theta = [\theta_x \quad \theta_y \quad \theta_z \quad \theta_\psi]^T$ parameterised trajectory. Arc-length parameterisation allows our method to maximise the speed of the projection point θ along the reference path, thereby increasing the flight speed of the quadrotor. Moreover, as the reference trajectory does not explicitly depend on the time parameter t, T-STAR can track dynamically infeasible trajectories. The cost function is,

$$\mathbf{J}(\boldsymbol{z}, \theta) = \sum_{k=1}^{K} \| e_k^c(\boldsymbol{z}_k, \theta_k) \|_{q_c}^2 - \eta v_{\theta,k}, \tag{25}$$

where $q_c \in \mathbb{R}^+$ is the contour weight and $\eta \in \mathbb{R}^+$ is the speed weight. $v_{\theta,k} = \frac{\Delta \theta_k}{\Delta t}$ is the velocity of the quadrotor projected onto the reference trajectory. Notably, the arc length θ and the velocity $v_{\theta,k}$ are virtual control state and corresponding virtual control input, guiding the planner toward faster trajectories.

However, coupling optimisation model (24) within (25) can significantly decrease computational efficiency. To ensure the real-time performance of the control system, an approximation point $\hat{\theta}_k$ of the optimal point θ_k^* is considered. As shown in Fig 3, the *lag error* $e_k^l = \left\| \theta_k^* - \hat{\theta}_k \right\|$ represents the link of θ_k^* and $\hat{\theta}_k$. For computational simplicity, $\hat{e}_k^l$ and $\hat{e}_k^c$ are used to approximate e_k^l and e_k^c. To get $\hat{e}_k^l$ and $\hat{e}_k^c$, firstly defining tangent line $\mathbf{t}(\hat{\theta}_k)$ of $\mathbf{z}^{ref}(\hat{\theta}_k)$ at point $\hat{\theta}_k$,

$$\begin{aligned} \mathbf{t}(\hat{\theta}_k) &= \frac{d\mathbf{z}^{ref}(\hat{\theta}_k)}{d\hat{\theta}_k} = [t_x \quad t_y \quad t_z \quad t_\psi] \\ &= \left[\dot{\mathbf{p}}_x^{ref}(\hat{\theta}_k) \quad \dot{\mathbf{p}}_y^{ref}(\hat{\theta}_k) \quad \dot{\mathbf{p}}_z^{ref}(\hat{\theta}_k) \quad \dot{\mathbf{p}}_\psi^{ref}(\hat{\theta}_k) \right], \end{aligned} \tag{26}$$

then, $\hat{e}_k^l$ can be calculated as the projection of $e_k(\hat{\theta}_k)$ onto normalised tangent line $\hat{\mathbf{t}}(\hat{\theta}_k) = \frac{\mathbf{t}(\hat{\theta}_k)}{\|\mathbf{t}(\hat{\theta}_k)\|}$, and $\hat{e}_k^c$ is the orthogonal vector to $\hat{e}_k^l$,

$$\begin{aligned} e_k(\hat{\theta}_k) &= \boldsymbol{z}_k - \mathbf{z}^{ref}(\hat{\theta}_k), \\ \hat{e}_k^l(\hat{\theta}_k) &= (e_k(\hat{\theta}_k) \cdot \hat{\mathbf{t}}(\hat{\theta}_k)) \hat{\mathbf{t}}(\hat{\theta}_k), \\ \hat{e}_k^c(\hat{\theta}_k) &= e_k(\hat{\theta}_k) - \hat{e}_k^l(\hat{\theta}_k), \end{aligned} \tag{27}$$

thus cost function (25) can be rewritten as

$$\begin{aligned}\mathbf{J}(e_k,\hat{\theta}_k) &= \sum_{k=1}^{K}\left\|\hat{e}_k^c(\hat{\theta}_k)\right\|_{q_c}^2 + \left\|\hat{e}_k^l(\hat{\theta}_k)\right\|_{q_l}^2 - \eta v_{\hat{\theta},k} \qquad (28)\\ &= \sum_{k=1}^{K} e_k(\hat{\theta}_k)^{\mathrm{T}}\mathbf{T_c}(\hat{\theta}_k)^{\mathrm{T}}\mathbf{Q}_c\mathbf{T_c}(\hat{\theta}_k)e_k(\hat{\theta}_k) + \\ &\quad e_k(\hat{\theta}_k)^{\mathrm{T}}\hat{\mathbf{t}}(\hat{\theta}_k)^{\mathrm{T}}\mathbf{Q}_l\hat{\mathbf{t}}(\hat{\theta}_k)e_k(\hat{\theta}_k) - \eta v_{\hat{\theta},k},\end{aligned}$$

$$\begin{aligned}\text{with}\quad \mathbf{T_c}(\hat{\theta}_k) &= \mathbf{I} - \hat{\mathbf{t}}(\hat{\theta}_k)^{\mathrm{T}}\hat{\mathbf{t}}(\hat{\theta}_k)\\ &= \begin{bmatrix} 1-t_x^2 & -t_xt_y & -t_xt_z & -t_xt_\psi \\ -t_yt_x & 1-t_y^2 & -t_yt_z & -t_yt_\psi \\ -t_zt_x & -t_zt_y & 1-t_z^2 & -t_zt_\psi \\ -t_\psi t_x & -t_\psi t_y & -t_\psi t_z & 1-t_\psi^2 \end{bmatrix}, \qquad (29)\end{aligned}$$

where $\mathbf{Q}_c = q_c \cdot \mathbf{I}_4, q_c \in \mathbb{R}^+$ is the contour weight. $\mathbf{Q}_l = q_l \cdot \mathbf{I}_4, q_l \in \mathbb{R}^+$ is the lag weight.

Remark 4: Incorporating the yaw angle error into the contour error is a natural choice with differential flatness. Standard MPCC is inherently nonlinear [14] and requires significant computational resources for its solution [43]. Different from standard MPCC, the elements of the flat output $\boldsymbol{z} = [\boldsymbol{p}^{\mathrm{T}} \quad \psi]^{\mathrm{T}}$ are decoupled. Consequently, planning the $\boldsymbol{z}$ reduces the complexity of constraints associated with the coupled full state of the quadrotor dynamic model, thereby decreasing the computational time.

C. Properties of the Cost Function

The cost function (28) can be extended to m-dimensional space. Consider the m-dimensional Euclidean space $\mathbb{R}^m$ formed by $\boldsymbol{z}_k$, which is equipped with an inner product structure. Therefore, equation (27) holds in this space. Then, calculate the optimal solution $\boldsymbol{z}_k^*$ of (28) without constraints, yielding

$$\begin{aligned}&\frac{de_k(\hat{\theta}_k)^{\mathrm{T}}\mathbf{W}_c(\hat{\theta}_k)e_k(\hat{\theta}_k) + e_k(\hat{\theta}_k)^{\mathrm{T}}\mathbf{W}_l(\hat{\theta}_k)e_k(\hat{\theta}_k) - \eta v_{\hat{\theta},k}}{d\boldsymbol{z}_k}\\ &= 2(\mathbf{W}_c(\hat{\theta}_k) + \mathbf{W}_l(\hat{\theta}_k))(\boldsymbol{z}_k - \mathbf{z}^{ref}(\hat{\theta}_k)) = 0, \qquad (30)\end{aligned}$$

where symmetric matrix $\mathbf{W}_c(\hat{\theta}_k) = \mathbf{T_c}(\hat{\theta}_k)^{\mathrm{T}}\mathbf{Q}_c\mathbf{T_c}(\hat{\theta}_k)$ and $\mathbf{W}_l(\hat{\theta}_k)) = \hat{\mathbf{t}}(\hat{\theta}_k)^{\mathrm{T}}\mathbf{Q}_l\hat{\mathbf{t}}(\hat{\theta}_k)$.

The cost function (28) is optimal if and only if $\boldsymbol{z}_k^* = \mathbf{z}^{ref}(\hat{\theta}_k)$. Therefore, the extension of $\boldsymbol{z}_k$ with mutually independent components will not affect the optimality of the result.

Next, the optimality of (28) and the form of the optimal solution $\boldsymbol{z}_k^*$ is discussed. Based on the differential flatness of a quadrotor, establish a continuous time-invariant system,

$$\dot{\mathbf{z}_k} = \mathbf{A}\mathbf{z}_k + \mathbf{B}u_k, \qquad (31)$$

where state variable $\mathbf{z}_k = \boldsymbol{z}_k^{[n-1]}$, control vector $u_k = \boldsymbol{z}_k^{(n)}$, state matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, input matrix $\mathbf{B} \in \mathbb{R}^n$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{(n-1)} \\ 0 & \mathbf{0} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \qquad (32)$$

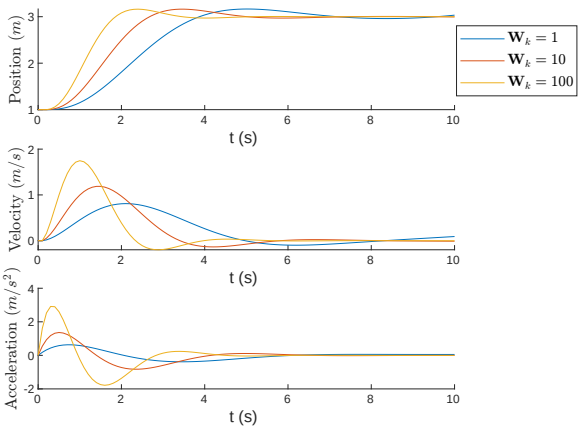

Fig. 4. Optimal control validation with terminal constraints. Given the same initial state $\mathbf{z}_k = [\mathbf{p}, \boldsymbol{v}, \boldsymbol{a}] = [1, 0, 0]$, as $\mathbf{W}_k$ increases, the velocity and acceleration increase significantly, allowing the state variables to reach the target values $[3, 0, 0]$ in a shorter time.

In the k-th time interval $[t_k, t_{k+1}]$, equation (28) is rewritten as follow,

$$\mathbf{J} = (\mathbf{z}_k(t_{k+1}) - \tilde{\mathbf{z}}^{ref}(\hat{\theta_k}))^T\mathbf{W}_k(\mathbf{z}_k(t_{k+1}) - \tilde{\mathbf{z}}^{ref}(\hat{\theta_k})) + \mathbf{J}_1, \qquad (33)$$

$$\text{with}\quad \mathbf{J}_1 = \frac{1}{2}\int_{t_k}^{t_{k+1}} u_k(t)^T h_k(t) u_k(t) dt, \quad \mathbf{z}_k(t_k) = \mathbf{z}_k^{init}$$

where $\tilde{\mathbf{z}}^{ref}(\hat{\theta_k}) = [\mathbf{z}^{ref}(\hat{\theta_k}), \mathbf{0}]^{\mathrm{T}} \in \mathbb{R}^n$ is augmented reference vector, $\mathbf{W}_k = \mathrm{diag}(\mathbf{W}_c(\hat{\theta_k}) + \mathbf{W}_l(\hat{\theta_k}), \mathbf{0}) \in \mathbb{R}^{n\times n}$ is terminal weighting matrix, $\mathbf{J}_1$ is a penalty term for control inputs, used to limit the infinite increase of the control input, positive control weight $h_k(t)$.

Considering the costate variable $\boldsymbol{\rho}(t) = [\rho_1(t), \cdots, \rho_s(t)]^{\mathrm{T}}$, the Hamiltonian function [44] is

$$\mathcal{H} = \frac{1}{2}u_k(t)^T h_k(t) u_k(t) + \boldsymbol{\rho}^{\mathrm{T}}(t)\mathbf{A}\mathbf{z}_k(t) + \boldsymbol{\rho}^{\mathrm{T}}(t)\mathbf{B}u_k(t). \qquad (34)$$

Based on Pontryagin's maximum theorem, the optimal control input satisfies

$$\frac{\partial \mathcal{H}}{\partial u_k} = h_k(t)u_k(t) + \mathbf{B}^{\mathrm{T}}\boldsymbol{\rho}(t) = 0, \qquad (35)$$

therefore, the optimal control input $u_k^*(t)$ is

$$u_k^*(t) = -\frac{\mathbf{B}^{\mathrm{T}}\boldsymbol{\rho}(t)}{h_k(t)} = -h_k^{-1}(t)\rho_s(t). \qquad (36)$$

The costate variable $\boldsymbol{\rho}(t)$ satisfies

$$\dot{\boldsymbol{\rho}(t)} = -\frac{\partial \mathcal{H}}{\partial \mathbf{z}_k} = -\mathbf{A}^{\mathrm{T}}\boldsymbol{\rho}(t), \qquad (37)$$

expand (37), get

$$\dot{\rho_i(t)} = \begin{cases} 0, & \text{if } i = 1 \\ -\rho_{i-1}(t), & \text{if } 1 < i \le n \end{cases} \qquad (38)$$

thus, $\boldsymbol{\rho}(t)$ consists of polynomials. It follows that $u_k^*(t)$ and $\boldsymbol{z}_k$ are also polynomials of degree $n-1$ and $2n-1$ respectively.

Finally, the effect of the final state on the optimal control is analysed. Considering the boundary condition,

$$\begin{aligned}&\frac{\partial \mathbf{J}}{\partial \mathbf{z}_k(t_{k+1})} - \boldsymbol{\rho}(t_{k+1}) \\ &= 2\mathbf{W}_k(\mathbf{z}_k(t_{k+1}) - \tilde{\mathbf{z}}^{ref}(\hat{\theta_k})) - \boldsymbol{\rho}(t_{k+1}) \\ &= 0.\end{aligned} \tag{39}$$

From (38) and (39), obtain

$$\rho_1(t) = 2\mathbf{W}_k(\mathbf{z}_k(t_{k+1}) - \tilde{\mathbf{z}}^{ref}(\hat{\theta_k})). \tag{40}$$

It is obvious that $\boldsymbol{\rho} \propto \mathbf{W}_k$ through (40), thus $u_k^* \propto \mathbf{W}_k$ through (36). An example in Fig. 4 shows that u_k^* increases with $\mathbf{W}_k$, causing the high velocity v, which allows the state to reach the reference value in a shorter time. Hence, increasing terminal weight $\mathbf{W}_k$ can generate trajectories with higher velocity and tracking accuracy.

D. SVM-Based Safe Flight Corridor

Compared to trajectory planning for a single quadrotor, which primarily concentrates on obstacle avoidance, the control of quadrotor swarms also requires collision avoidance among quadrotors. T-STAR constructs the safe flight space, referred to as the SVM-based safe flight corridor, as shown in Fig. 5.

It is assumed that the environmental information has been collected by sensors in advance. The optimal path is found from a given start point $\boldsymbol{p}_{start}$ to a goal point $\boldsymbol{p}_{end}$ by using a graph search algorithm. *Jump Point Search* (JPS) is considered as the path planner for this work because JPS guarantees path optimality and reduces computational overhead by pruning unnecessary nodes [45].

Specifically, JPS enhances the efficiency of the pathfinding process by selectively skipping certain nodes in the search grid that do not need to be individually checked. This 'jumping' mechanism allows JPS to move rapidly through open areas of the grid, directly advancing to important points. As a result, JPS significantly reduces the number of nodes processed compared to traditional A* search while still ensuring that the shortest path is found.

Then, based on the path generated by the JPS, the space is decomposed into multiple sequentially linked convex polyhedra. This can be achieved by dilating an ellipsoid which contains the optimal paths until it intersects an obstacle with the point $\boldsymbol{p}_e$, calculating the tangent plane of the ellipsoid over $\boldsymbol{p}_e$ to get the half-space $H_{obstacle} = \left\{\boldsymbol{p} \mid \mathbf{a}_o^{\mathrm{T}}\boldsymbol{p} - \mathbf{b}_o < 0\right\}$ containing the paths, deleting the obstacles outside the half-space, and repeating the process until there are no obstacles. More details can be found in [46].

According to subsection V-C, a quadrotor can receive trajectory $\mathbf{p}^{adj}$ of adjacent quadrotors. To avoid collisions between quadrotors, hyperplanes $H_{traj} = \mathbf{W}_t^{\mathrm{T}}\boldsymbol{p} - b_t$, where $\mathbf{W}_t$ is the weight matrix defining the orientation of the normal to the hyperplane and b_t is the offset of hyperplane, are found to separate the reference trajectory $\mathbf{p}^{ref}$ and adjacent

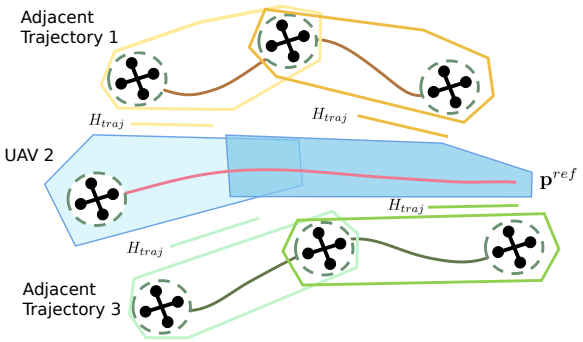

Fig. 5. A schematic of SVM-based Safe Flight Corridors. After receiving the adjacent trajectories, UAV2 calculates the hyperplanes H_{traj} using the reference trajectory and SVM algorithm to get the final safe space $H_{svm-sfc}$. (blue polyhedra)

trajectory $\mathbf{p}^{adj}$. Consider the boundaries of space occupation of a quadrotor $\mathcal{B}$,

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{\mathbf{p}}^{ref} \\ \hat{\mathbf{p}}^{adj} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^{ref} \oplus \mathcal{B} \\ \mathbf{p}^{adj} \oplus \mathcal{B} \end{bmatrix} \tag{41}$$

where the $\oplus$ is the Minkowski sum. Using the hard margin support vector machine (SVM), the following Quadratic Programming problem is constructed

$$\begin{aligned}&\min_{\mathbf{W}_t, b_t} \ \frac{1}{2}\left\|\mathbf{W}_t\right\|^2 \\ &\text{s.t. } 1 - \gamma_i\left(\mathbf{W}_t^{\mathrm{T}}\hat{\mathbf{p}}_i - b_t\right) \le 0 \quad \forall i \in \{1, 2, \ldots, N_p\},\end{aligned} \tag{42}$$

where $\gamma_i \in [+1 \ \ -1]$ is classification matrix, N_p is the size of $\hat{\mathbf{p}}$. Utilising the Lagrangian duality, (42) is equivalent to:

$$\begin{aligned}&\max_{\alpha}\min_{\mathbf{W}_t, b_t} \mathcal{L}\left(\mathbf{W}_t, b_t, \alpha\right) \\ &= \max_{\alpha}\min_{\mathbf{W}_t, b_t} \frac{1}{2}\left\|\mathbf{W}_t\right\|^2 - \sum_{i=1}^{N}\alpha_i\left[\gamma_i\left(\mathbf{W}_t^{\mathrm{T}}\hat{\mathbf{p}}_i - b_t\right) - 1\right],\end{aligned} \tag{43}$$

where α is Lagrange multiplier, $\mathcal{L}\left(\mathbf{W}_t, b_t, \alpha\right)$ is Lagrangian function. The optimal values $\mathbf{W}_t^*$ and b_t^* can be obtained by setting the partial derivatives to zero $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_t} = 0$ and $\frac{\partial \mathcal{L}}{\partial b_t} = 0$. By reformulating, we arrive at the following quadratic programming problem concerning the variable α:

$$\begin{aligned}\min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\gamma_i\gamma_j\hat{\mathbf{p}}_i^{\mathrm{T}}\hat{\mathbf{p}}_j - \sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i=1}^{N}\alpha_i\gamma_i = 0 \\ & \alpha_i \ge 0 \quad \forall i \in \{1, 2, \ldots, N_p\}.\end{aligned} \tag{44}$$

By solving equation (44) to obtain the optimal values α^*, we can subsequently calculate $\mathbf{W}_t^*$ and b_t^*. With these parameters, the hyperplane H_{traj} is fully defined, allowing us to construct the safe flight corridor $H_{svm-sfc} = \left\{\boldsymbol{p} \mid \bar{\mathbf{A}}^{\mathrm{T}}\boldsymbol{p} - \bar{\mathbf{B}} < 0\right\}$, with $\bar{\mathbf{A}} = [\mathbf{a}_o \quad \mathbf{W}_t]^{\mathrm{T}}$ and $\bar{\mathbf{B}} = [\mathbf{b}_o \quad b_t]^{\mathrm{T}}$.

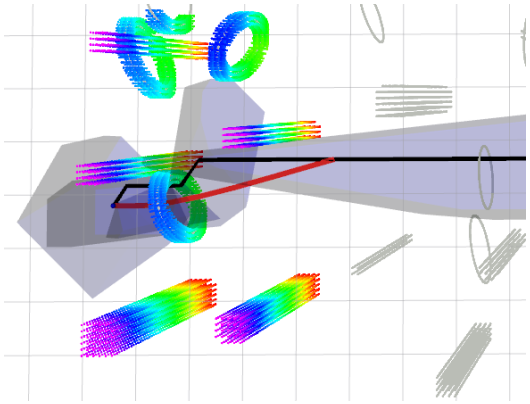

Fig. 6. Visualisation of Reference Trajectory. Given the initial and target points, the shortest path (black) is found by JPS algorithm. The safe corridor (light blue polyhedra) is used as a constraint to calculate the optimal solution of (46a), resulting in the reference path (red).

E. Reference Trajectory Planner

Unlike standard Model Predictive Control, the reference trajectories in our approach are not required to strictly follow the dynamic constraints of the quadrotor. These trajectories only need to be differentiable and contained within the safe flight corridor $H_{svm-sfc}$. This allows for significant flexibility in creating reference trajectories. We utilise the trajectory optimisation model (46a) to generate smooth and safe reference trajectories. It's worth noting that although this method is adopted in our framework, any trajectory generated by alternative planning methods that satisfy the above conditions is acceptable.

Consider the reference trajectory $\mathbf{p}_i^{ref}(t)$ as piecewise K_p-degree polynomial [47]:

$$\mathbf{p}_i^{ref}(t) = \sum_{k_p=0}^{K_p} \boldsymbol{\beta}_{k_p}^i (t - t_{i-1})^{k_p}, \quad t \in [t_{i-1}, t_i], \tag{45}$$

where t_{i-1} and t_i denote the initial and final time of ith piece respectively,

$$\min_{\mathbf{p}^{ref}} \sum_{i=1}^{N_p} \left\| \frac{d^n \mathbf{p}_i^{ref}(t_i)}{dt^n} \right\|^2 \tag{46a}$$

$$\text{s.t.} \ \frac{d^s \mathbf{p}_i^{ref}(t_i)}{d^s t} = \frac{d^s \mathbf{p}_{i+1}^{ref}(t_i)}{d^s t} \quad \forall s \in \{1, \dots, n\} \tag{46b}$$

$$\bar{\mathbf{A}}_i^{\mathrm{T}} \mathbf{p}_i^{ref}(t_i) - \bar{\mathbf{B}}_i < 0, \tag{46c}$$

$$\mathbf{p}_1^{ref}(t_0) = \mathbf{p}_{init}^{ref}, \quad \mathbf{p}_{N_p}^{ref}(t_{N_p}) = \mathbf{p}_{final}^{ref} \tag{46d}$$

where s represents sth derivative, constraint (46b) guarantees the smoothness of the trajectory, constraint (46c) ensures the trajectory stays within the safe flight corridor $H_{svm-sfc}$, boundary conditions (46d). Then, arc-length θ parameterise $\mathbf{p}^{ref}(t)$ as $\mathbf{p}^{ref}(\theta)$. It should be noted that arc-length parameterisation of the reference trajectory is a nontrivial problem. Thus, to improve computational efficiency, we consider the cubic spline with the given parameter $K_p = 3$ for the reference trajectory (45). This configuration enables the efficient generation of approximately arc-length parameterised reference trajectories by utilising the methodology described in [48]. Fig. 6 shows an example of a reference trajectory obtained by running Algorithm 1. However, Algorithm 1 does not explicitly consider the scenario of multi-UAV trajectory conflicts. In such cases, the equation (46a) may yield no feasible solution. Therefore, the ETD mechanism described in subsection V-C is proposed to handle these conflicts.

Algorithm 1 Reference Trajectory Planner.

Notation: Global map $\mathbf{M}_{global}$, Local map $\mathbf{M}_{local}$, Reference trajectory $\mathbf{p}^{ref}(t)$, Adjacent trajectory $\mathbf{p}^{adj}$, Initial start point $\boldsymbol{p}_{start}$, Initial goal point $\boldsymbol{p}_{end}$, Trajectory by JPS $\boldsymbol{p}^{JPS}$, Quadrotor model $\mathcal{B}$

Initialise: $z^* \leftarrow \varnothing$, $\mathbf{M}_{global}$, $\mathbf{p}^{adj} \leftarrow$ Adjacent Quadrotors
$\mathbf{M}_{local} \leftarrow \mathbf{M}_{global}$
$\boldsymbol{p}^{JPS} \leftarrow$ **JPS**($\boldsymbol{p}_{start}$, $\boldsymbol{p}_{end}$, $\mathbf{M}_{local}$)
$H_{obstacle} \leftarrow$ **ConvexDecompse**($\boldsymbol{p}^{JPS}$)
$H_{svm-sfc} \leftarrow$ **SVM-SFC**($H_{obstacle}$, $\mathbf{p}^{adj}$, $\mathcal{B}$)
$\mathbf{p}^{ref}(t) \leftarrow (46a) \leftarrow (\boldsymbol{p}^{JPS}, H_{svm-sfc})$
$\mathbf{p}^{ref}(\theta) \leftarrow$ **ArcLengthParameter**($\mathbf{p}^{ref}(t)$)
Return $\mathbf{p}^{ref}(\theta)$

V. High-Level Swarm Coordination

A. Flocking Control

Flocking control focuses on modelling the interaction of quadrotors. Inspired by the method of artificial potential fields (APF), quadrotors are subjected to repulsive forces U_r and attractive forces U_a. Uniquely, both U_r and U_a are generated from adjacent quadrotors instead of obstacles or target points. This leads to the formation of flocks in completely distributed systems by allowing the quadrotor to track reference distances $\mathbf{d}^{ref}$ of adjacent quadrotors. Thus, the formation cost w_{ij} between quadrotor i and adjacent quadrotor j

$$w_{ij} = U_a\left(d_{ij}, d_{ij}^{ref}\right) + U_r\left(d_{ij}, d_{ij}^{ref}\right), \tag{47}$$

where $d_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$ and d_{ij}^{ref} are the real distance and reference distance between quadrotor i and quadrotor j.

Consider an *undirected graph* $G = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ is the set of $M = N+1$ vertices and $\mathcal{E} \in \mathbb{R}^{M\times M}$ is the set of edges, N represents the number of adjacent quadrotors. Then, quadrotor i calculates local weighted adjacency matrix $\hat{\mathbf{A}}_i \in \mathbb{R}^{M\times M}$ and degree matrix $\hat{\mathbf{D}}_i \in \mathbb{R}^{M\times M}$

$$\hat{\mathbf{A}}_i = \begin{bmatrix} 0 & w_{i\boldsymbol{l}_1} & w_{i\boldsymbol{l}_2} & \dots & w_{i\boldsymbol{l}_N} \\ w_{\boldsymbol{l}_1 i} & 0 & 0 & & 0 \\ w_{\boldsymbol{l}_2 i} & 0 & 0 & & 0 \\ \vdots & & & \ddots & \\ w_{\boldsymbol{l}_N i} & 0 & 0 & & 0 \end{bmatrix},$$

$$\hat{\mathbf{D}}_i = \begin{bmatrix} \sum_{j=1}^{N} w_{i\boldsymbol{l}_j} & 0 & 0 & \dots & 0 \\ 0 & w_{\boldsymbol{l}_1 i} & 0 & & 0 \\ 0 & 0 & w_{\boldsymbol{l}_2 i} & & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & & w_{\boldsymbol{l}_N i} \end{bmatrix},$$

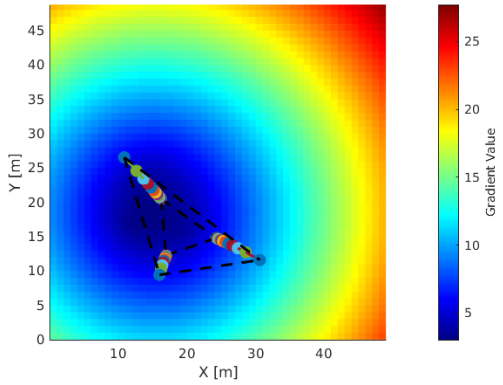

Fig. 7. An example of flocking control. The colour intensity indicates the gradient value of the virtual force. The quadrotor swarm will converge from high gradients (red) to low gradients (blue).

where $\boldsymbol{l}_j$ represents the jth quadrotor in the set of adjacent quadrotors $\boldsymbol{l}$. Thus, the Laplacian matrix is

$$\begin{aligned}\mathbf{L}_i &= \hat{\mathbf{D}}_i - \hat{\mathbf{A}}_i \\ &= \begin{bmatrix} \sum_{j=1}^{N} w_{i\boldsymbol{l}_j} & -w_{i\boldsymbol{l}_1} & -w_{i\boldsymbol{l}_2} & \dots & -w_{i\boldsymbol{l}_N} \\ -w_{\boldsymbol{l}_1 i} & w_{\boldsymbol{l}_1 i} & 0 & & 0 \\ -w_{\boldsymbol{l}_2 i} & 0 & w_{\boldsymbol{l}_2 i} & & 0 \\ \vdots & & & \ddots & \\ -w_{\boldsymbol{l}_N i} & 0 & 0 & & w_{\boldsymbol{l}_N i} \end{bmatrix}.\end{aligned}$$

Obviously, the Laplacian matrix $\mathbf{L}$ indicates the cost associated with the quadrotor formation. Therefore, the formation cost function is written as

$$\begin{aligned}&\min_{\mathbf{L}} \ \mathbf{J}_s(\mathbf{L}) \\ &\text{with } \mathbf{J}_s(\mathbf{L}) = \|\mathbf{L}\|_F^2 = \mathrm{tr}\{(\mathbf{L})^{\mathrm{T}}\mathbf{L}\} \\ &\qquad = \left(\sum_{j=1}^{N} w_{i\boldsymbol{l}_j}\right)^2 + 3\sum_{j=1}^{N}(w_{i\boldsymbol{l}_j})^2,\end{aligned} \tag{48}$$

the optimality condition for (48) is

$$\begin{aligned}\frac{\partial \mathbf{J}_s(\mathbf{L})}{\partial d_{i\boldsymbol{l}_j}} &= \frac{\partial \mathbf{J}_s(\mathbf{L})}{\partial w_{i\boldsymbol{l}_j}} \frac{\partial w_{i\boldsymbol{l}_j}}{\partial d_{i\boldsymbol{l}_j}} \\ &= \left(2\left(\sum_{j=1}^{N} w_{i\boldsymbol{l}_j}\right) + 6w_{i\boldsymbol{l}_j}\right) \frac{\partial w_{i\boldsymbol{l}_j}}{\partial d_{i\boldsymbol{l}_j}} \\ &= 0,\end{aligned} \tag{49}$$

given that virtual forces U_a and U_r are strictly positive ($U_a > 0, U_r > 0$), it means $w > 0$. To ensure that the UAV swarm converges to the reference formation, we expect the optimal value $d^*_{i\boldsymbol{l}_j} = d^{ref}_{i\boldsymbol{l}_j}$, (49) can be reformulated as

$$\begin{aligned}\frac{\partial w_{i\boldsymbol{l}_j}}{\partial d_{i\boldsymbol{l}_j}} &= \frac{\partial U_a\left(d_{i\boldsymbol{l}_j}, d^{ref}_{i\boldsymbol{l}_j}\right)}{\partial d_{i\boldsymbol{l}_j}} + \frac{\partial U_r\left(d_{i\boldsymbol{l}_j}, d^{ref}_{i\boldsymbol{l}_j}\right)}{\partial d_{i\boldsymbol{l}_j}} \\ &= \left(d_{i\boldsymbol{l}_j} - d^{ref}_{i\boldsymbol{l}_j}\right) f\left(d_{i\boldsymbol{l}_j}\right) \\ &= 0,\end{aligned} \tag{50}$$

where $f\left(d_{i\boldsymbol{l}_j}\right)$ is a conventional function of $d_{i\boldsymbol{l}_j}$. Any formulations of U_a and U_r satisfying the above condition can be applied to this work. Thus, we utilise the following general form

$$U_a(d_{i\boldsymbol{l}_j}, d^{ref}_{i\boldsymbol{l}_j}) = \begin{cases} 0 & \text{if} \quad 0 < d_{i\boldsymbol{l}_j} < d^{ref}_{i\boldsymbol{l}_j} \\ \frac{1}{2}\kappa_a\left(d_{i\boldsymbol{l}_j} - d^{ref}_{i\boldsymbol{l}_j}\right)^2 & \text{if} \quad d_{i\boldsymbol{l}_j} \geq d^{ref}_{i\boldsymbol{l}_j} \end{cases}$$

$$U_r(d_{i\boldsymbol{l}_j}, d^{ref}_{i\boldsymbol{l}_j}) = \begin{cases} \frac{1}{2}\kappa_r\left(\frac{1}{d_{i\boldsymbol{l}_j}} - \frac{1}{d^{ref}_{i\boldsymbol{l}_j}}\right)^2 & \text{if} \quad 0 < d_{i\boldsymbol{l}_j} < d^{ref}_{i\boldsymbol{l}_j} \\ 0 & \text{if} \quad d_{i\boldsymbol{l}_j} \geq d^{ref}_{i\boldsymbol{l}_j} \end{cases}$$

where κ_a, κ_r are coefficients.

Fig. 7 shows an example of the above form of the virtual force, demonstrating that the UAV swarm can converge to the reference value.

B. T-STAR with Constrained Control Effort

To ensure the trajectory satisfies the quadrotor dynamics, consider the optimal trajectory as the clamped piecewise cubic polynomial [3], expressed in the form:

$$\boldsymbol{z}_k = \mathbf{a}_k + \mathbf{b}_k t + \mathbf{c}_k t^2 + \mathbf{d}_k t^3, \tag{51}$$

where $\mathbf{a}_k$, $\mathbf{b}_k$, $\mathbf{c}_k$ and $\mathbf{d}_k$ are the polynomial coefficient.

Considering higher-order continuity constraints on piecewise trajectories at the boundaries to ensure smoothness, the coefficients of (51) give

$$\begin{aligned}&\mathbf{a}_k = \boldsymbol{z}_k \quad \mathbf{b}_k = \frac{\boldsymbol{z}_{k+1} - \boldsymbol{z}_k}{\xi} - \frac{\xi}{6}(4\ddot{\boldsymbol{z}}_k - \ddot{\boldsymbol{z}}_{k+1}) \\ &\mathbf{c}_k = \frac{\ddot{\boldsymbol{z}}_k}{2} \quad \mathbf{d}_k = \frac{\ddot{\boldsymbol{z}}_{k+1} - \ddot{\boldsymbol{z}}_k}{6\xi}\end{aligned} \tag{52}$$

Under the Not-A-Knot condition, i.e., given the initial and terminal states, rewriting (51) and (52) yields

$$\xi^2 \mathbf{M}\ddot{\boldsymbol{z}} - 6\mathbf{b}_c = 0 \tag{53}$$

where ξ is sampling time, define $\ddot{\boldsymbol{z}} \coloneqq [\ddot{\boldsymbol{z}}_0^T, \dots, \ddot{\boldsymbol{z}}_K^T]^T$, $\mathbf{M} \in \mathbb{R}^{(K+1)\times(K+1)}$ and $\mathbf{b} \in \mathbb{R}^{K+1}$ are

$$\mathbf{M} = \begin{bmatrix} 2 & 1 & 0 & 0 & \dots & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 \\ 0 & 1 & 4 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & \dots & 0 & 1 & 2 \end{bmatrix}, \tag{54}$$

$$\mathbf{b}_c = \begin{bmatrix} \boldsymbol{z}_1 - \boldsymbol{z}_0 - \xi\dot{\boldsymbol{z}}_0 \\ \boldsymbol{z}_2 + \boldsymbol{z}_0 - 2\boldsymbol{z}_1 \\ \boldsymbol{z}_3 + \boldsymbol{z}_1 - 2\boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_K + \boldsymbol{z}_{K-2} - 2\boldsymbol{z}_{K-1} \\ \xi\dot{\boldsymbol{z}}_K - \boldsymbol{z}_K + \boldsymbol{z}_{K-1} \end{bmatrix}, \tag{55}$$

Algorithm 2 Constrained Time-Optimal Control.

Notation: Reference trajectory $\mathbf{z}^{ref}(\theta)$, Adjacent trajectory $\mathbf{p}^{adj}$, Adjacent Quadrotors $\boldsymbol{l}$, Approximation point $\hat{\theta_k}$, Normalised Tangent line $\hat{\mathbf{t}}(\hat{\theta_k})$, Contour error $\hat{e}_k^c$, Lag error $\hat{e}_k^l$, Optimal trajectory z^*

Initialise: $z^* \leftarrow \varnothing$, $\mathbf{p}^{adj} \leftarrow$ Adjacent Quadrotors
if *ETD*(z^*) is false **then**
  $\mathbf{z}^{ref}(\theta) \leftarrow$ **Algorithm 1**
  for each time step k **do**
    $\hat{\mathbf{t}}(\hat{\theta_k}) \leftarrow (26) \leftarrow (\mathbf{z}^{ref}(\hat{\theta_k}))$
    $\hat{e}_k^c, \hat{e}_k^l \leftarrow (27) \leftarrow \hat{\mathbf{t}}(\hat{\theta_k})$
    for each adjacent quadrotors j **do**
      $d_{i\boldsymbol{l}_j,k} \leftarrow \left\|\boldsymbol{p}_{i,k} - \boldsymbol{p}_{\boldsymbol{l}_j,k}\right\|$
      $J(\mathbf{L}, k) \leftarrow U_a(d_{i\boldsymbol{l}_j,k}), U_r(d_{i\boldsymbol{l}_j,k})$
    end for
    $\mathbf{M}, \mathbf{b}_c \leftarrow (54), (55)$
    $\boldsymbol{z}_k^* \leftarrow \hat{e}_k^c, \hat{e}_k^l, J(\mathbf{L}, k), H_{svm-sfc}, \mathbf{M}, \mathbf{b}_c$
  end for
  $z^* \leftarrow \boldsymbol{z}_k^*$
end if
Return z^*

According to (28), (42), (48), and (53), the constrained time-optimal control can be defined as,

$$\min_{\boldsymbol{u}_k} \sum_{k=1}^{K} \left\|\hat{e}_k^c(\hat{\theta}_k)\right\|_{q_c}^2 + \left\|\hat{e}_k^l(\hat{\theta}_k)\right\|_{q_l}^2 - \eta v_{\hat{\theta},k} + \left\|\Delta v_{\hat{\theta},k}\right\|_{R_u}^2 + \lambda_s \left(\sum_{j=1}^{N} w_{i\boldsymbol{l}_j,k}\right)^2 + 3\lambda_s \sum_{j=1}^{N} (w_{i\boldsymbol{l}_j,k})^2 \tag{56}$$

$$\begin{aligned}
\text{s.t. } & \xi^2 \mathbf{M}\ddot{\boldsymbol{z}} - 6\mathbf{b}_c = 0 \\
& v_{\hat{\theta}_{k+1}} = v_{\hat{\theta}_k} + \xi \Delta v_{\hat{\theta},k} && \forall k = 0, \ldots, K-1 \\
& \hat{\theta}_{k+1} = \hat{\theta}_k + \xi v_{\hat{\theta},k} + 0.5\xi^2 \Delta v_{\hat{\theta},k} && \forall k = 0, \ldots, K-1 \\
& \bar{\mathbf{A}}_k^{\mathrm{T}} \boldsymbol{z}_k - \bar{\mathbf{B}}_k < 0 && \forall k = 1, \ldots, K \\
& \boldsymbol{z}_{k,min}^{[2]} \leq \boldsymbol{z}_k^{[2]} \leq \boldsymbol{z}_{k,max}^{[2]} && \forall k = 1, \ldots, K \\
& 0 \leq v_{\hat{\theta}_k} \leq v_{\hat{\theta},max} && \forall k = 1, \ldots, K \\
& \boldsymbol{z}_0^{[2]} = \bar{\boldsymbol{z}}_{init}, \ \boldsymbol{z}_K^{[2]} = \bar{\boldsymbol{z}}_{final} \\
& \hat{\theta}_0 = \hat{\theta}_{init}, \ v_{\hat{\theta},0} = v_{\hat{\theta},init}, \ \Delta v_{\hat{\theta},0} = \Delta v_{\hat{\theta},init},
\end{aligned}$$

where control input $\boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{z}_k^{[2]\mathrm{T}} & v_{\hat{\theta},k} \end{bmatrix}^{\mathrm{T}}$, and virtual control input increment $\Delta v_{\hat{\theta},k}$ which prevent non-smooth virtual control input $v_{\hat{\theta},k}$ and weight R_u. Algorithm 2 generates feasible, collision-free time-optimal trajectories for any quadrotor i.

C. Deconfliction

For collaborative multi-quadrotor systems, this section explores an asynchronous information interaction mechanism. This method allows the quadrotor UAV to receive trajectory information from adjacent UAVs in the communication range d_c at non-predetermined frequencies. To ensure that all quadrotors can generate feasible trajectories, we propose an Event-triggered Trajectory Deconflict strategy, as shown in Fig. 8, which is divided into the following three key steps:

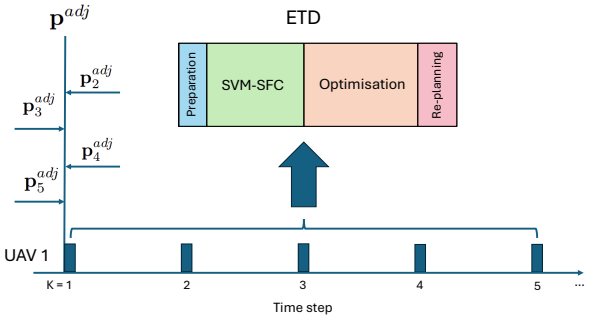

Fig. 8. Illustration of ETD. During the preparation stage (blue), the UAV receives adjacent trajectories and calculates the optimal trajectory (orange) using the SVM-based flight safe corridor (green). If the calculation fails, representing a trajectory conflict, the system will re-plan, establishing a new safe area to obtain a conflict-free trajectory.

Algorithm 3 Event-triggered Trajectory Deconflict Algorithm.

Notation: Reference trajectory $\mathbf{p}^{ref}(t)$, Adjacent trajectory $\mathbf{p}^{adj}$, Local start point $\boldsymbol{p}_{start}$, Goal point $\boldsymbol{p}_{end}$, Local map $\mathbf{M}_{local}$

Initialise: $\mathbf{p}^{adj} \leftarrow$ Adjacent Quadrotors, $z^* \leftarrow \varnothing$
while *ETD*(z^*)
  $\boldsymbol{p}^{JPS} \leftarrow$ **JPS(**$\boldsymbol{p}_{start}$**,** $\boldsymbol{p}_{end}, \mathbf{M}_{local}$**)**
  $H_{obstacle} \leftarrow$ **ConvexDecompse(**$\boldsymbol{p}^{JPS}$**)**
  $H_{svm-sfc} \leftarrow$ **SVM-SFC(**$H_{obstacle}$**,** $\mathbf{p}^{adj}$**,** $\mathcal{B}$**)**
  $\mathbf{z}^{ref}(\theta) \leftarrow$ **Algorithm 1**
  $z^* \leftarrow$ **Algorithm 2**
  if z^* is false
    $\mathbf{M}_{local} \leftarrow (\mathbf{M}_{local}, \mathbf{p}^{adj} \oplus \mathcal{B})$
    ETD(z^*) $\leftarrow$ true
  else
    ETD(z^*) $\leftarrow$ false
  end if
end
Return *ETD*(z^*)

- Trajectory planning preparation: in the distributed communication network, the quadrotor UAV continuously receives adjacent trajectories $\mathbf{P}^{adj}$ within communication range d_c during periods when it is not planning trajectory.
- Trajectory feasibility check: concerning (44) and (56), the feasibility of the SVM-SFC and the constrained optimisation model is checked. If the optimal solution is obtained, meaning the trajectory is conflict-free, the quadrotor will follow this trajectory. Otherwise, go to the next step.
- Trajectory replanning: the airspace ($\mathbf{P}^{adj} \oplus \mathcal{B}$) associated with the received trajectories extended by the quadrotor model is labelled as obstacle areas, which are updated on the local map of the quadrotor. Then, the quadrotor generates a new S-SFC and corresponding optimal trajectory and repeats the feasibility check in step 2 until a conflict-free trajectory is found.

By using Algorithm 3, each quadrotor can dynamically plan and adjust its trajectory to ensure reliable cooperative control.

VI. Simulation And Real-world Experiments

In this section, we presented several simulations and real-world experiments to evaluate the performance of our method with both single and multiple quadrotors against the following three approaches: AMSwarm [49], DMPC [40] and DMPCC [50]. The CPU used for the simulation is i9-14900KF. The parameter configurations used for our method and the baseline approaches were set up with sampling time $\xi = 0.1s$, prediction Horizon $K = 10$, velocity $\boldsymbol{v}_{min} = -9.0m \cdot s^{-1}, \boldsymbol{v}_{max} = 9.0m \cdot s^{-1}$, and acceleration $\boldsymbol{a}_{min} = -21.0m \cdot s^{-2}, \boldsymbol{a}_{max} = 21.0m \cdot s^{-2}$. In addition to the parameters listed above, AMSwarm and DMPC were run using the default settings from their respective open-source libraries. For our method and DMPCC, we configured the same important parameters, as follows: contour weight $q_c = 0.1$, lag weight $q_l = 1000$, speed weight $\eta = 1.0$, formation weight $\lambda_s = 1.0$, control input increment weight $R_u = 0.05$, and polynomial degree $K_p = 3$. Furthermore, we set the flocking control parameters to $\kappa_a = 1.0, \kappa_r = 1.0$ for T-STAR and $a = 1.0, b = 1.0$ for DMPCC, respectively.

A. Performance with a Single UAV

To demonstrate the advantages of the T-STAR algorithm compared to AMSwarm, DMPC, and DMPCC, we tested our proposed algorithm in a 36 m $\times$ 20 m $\times$ 3 m environment with 0.1 m resolution, containing [0, 50, 100, 200] randomly placed static obstacles of various sizes. Fig. 9 illustrates the results of T-STAR running in this simulation, showing the trajectory planning of a single UAV in an unknown environment. In a single UAV scenario, DMPCC provided basic validation for T-STAR.

We compared our proposed method with AMSwarm and DMPC in terms of flight time, velocity, and flight distance. As shown in Table I, through 10 different simulations, we found that our method reduced flight time by 55.09% and 49.12% compared to AMSwarm and DMPC respectively. In terms of velocity, the performance of AMSwarm and DMPC was only 55.71% and 33.27% of our method, respectively. For flight distance, our method decreased the distance by 25.48% and 3% with respect to AMSwarm and DMPC respectively. Fig. 10 shows that under the same maximum velocity and acceleration conditions, our method is able to generate faster and shorter trajectories. This advantage stems from our local planner's ability to achieve time-optimal trajectories within the feasible domain by finding faster velocity. In contrast, AMSwarm and DMPC use the distance between the quadrotor's current position and the target point as the cost function, resulting in more conservative trajectory planning.

B. Performance with a Multi-UAV System

To show the agility of our method in multi-UAV systems, we conducted a series of simulation experiments in 36 m $\times$ 20 m $\times$ 3 m environments with 100 static obstacles and different sizes of UAV swarms, comparing our results with AMSwarm, DMPC, and DMPCC. Additionally, we introduced a new metric, safety ratio, to further validate the reliability of our method. The safety ratio measures the proximity between any two quadrotors during flight, with a higher value indicating safer flight.

The results, as shown in Table II, indicate that compared to AMSwarm and DMPC, our method reduced the average flight

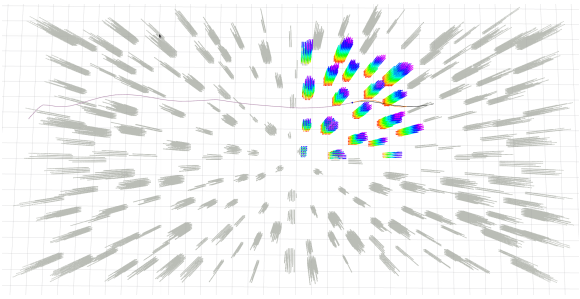

Fig. 9. Single UAV simulation in an environment with 200 static obstacles. Coloured areas represent obstacles detected by the quadrotors, while grey areas indicate unknown regions. The trajectory planner generates a reference trajectory (black), a time-optimal trajectory (orange), and the flown trajectory (purple).

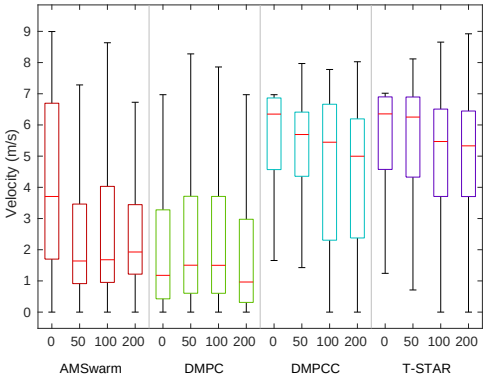

Fig. 10. A boxplot of single UAV velocity distribution with varying obstacle densities. Comparing our method with the baseline methods. Each box's central mark shows the median, the bottom and top edges represent the first and third quartile, and the solid lines extend to the non-outlier minimum and maximum value.

TABLE I
Performance Comparison in Single Quadrotor

Obs	Method	Avg. Time. (s)	Avg. Vel. (m/s)	Distance (m)
0	AMSwarm	9.52	4.05	41.69
	DMPC	12.40	2.17	**33.52**
	DMPCC	6.45	5.40	33.71
	T-STAR	**6.10**	**5.45**	33.69
50	AMSwarm	14.25	2.99	45.17
	DMPC	12.90	2.44	**33.59**
	DMPCC	6.74	4.98	34.46
	T-STAR	**6.19**	**5.38**	33.77
100	AMSwarm	15.41	2.73	44.67
	DMPC	13.60	1.63	**33.29**
	DMPCC	7.13	4.89	34.82
	T-STAR	**6.92**	**4.90**	34.29
200	AMSwarm	19.68	2.71	55.17
	DMPC	16.88	1.99	**33.82**
	DMPCC	7.90	4.42	35.23
	T-STAR	**6.91**	**4.93**	34.51

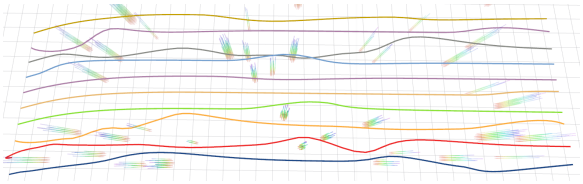

Fig. 11. Trajectories visualisation of ten UAVs in RViz with different colour identification

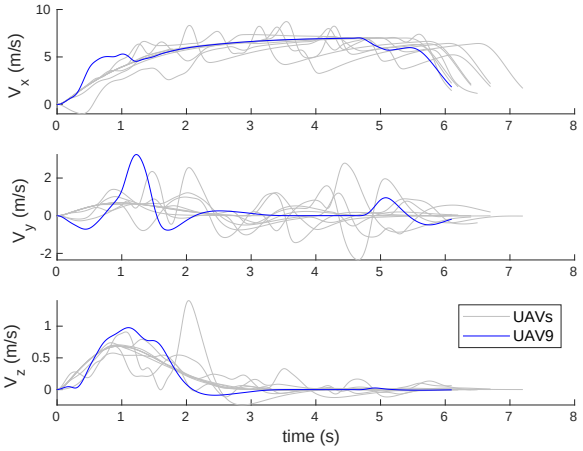

Fig. 12. Velocity of ten UAVs with one of them highlighted for clarity.

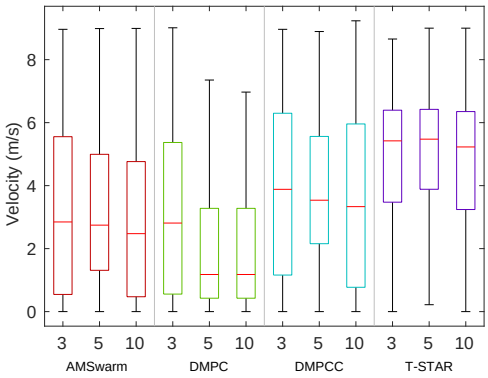

Fig. 13. Boxplot of multi-UAV velocity distribution with different numbers of UAVs, comparing our method with AMSwarm, DMPC and DMPCC.

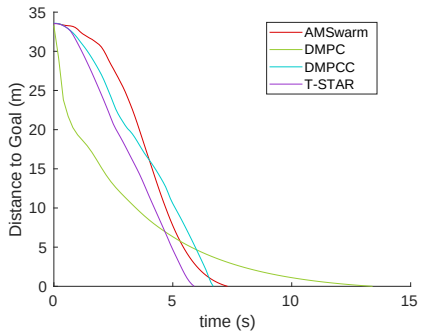

Fig. 14. Responses of distance to target point over time with 3 UAVs. Our method reaches the target point in 5.9 s, faster than AMSwarm (7.3 s), DMPC (13.4 s), and DMPCC (6.7 s).

time by 41.69% and 50.61%, and increased the average flight velocity by 52.15% and 47.81%, respectively. This is consistent with the results of the single quadrotor, demonstrating that our method retains its agility in multi-quadrotor trajectory planning. Furthermore, T-STAR achieved a 23.54% reduction in average flight time and a 31.81% increase in average speed over DMPCC by relaxing formation constraints, which allows for higher-speed trajectory generation. Fig. 13 shows box plots of velocity for our method versus baseline methods, indicating that our method can produce velocity distributions closer to the maximum value $\boldsymbol{v}_{max}$. Fig. 14 demonstrates that our method is able to reach the target point in a shorter time than AMSwarm and DMPC. This effect is also qualitatively visible in Fig. 11 which presents the trajectories when the UAV swarm reached the target point, and Fig. 12 which demonstrates the corresponding velocity curve. For visualisation, we randomly selected and highlighted the velocity curve of UAV 9. It can be observed that the velocity generated by our method is smooth. Additionally, since the UAV primarily flies along the x-axis, V_x consistently keeps at a high value. In safety ratio, our method is 4.91 and 1.42 times higher than AMSwarm and DMPC, respectively. This is because our method considers the trajectories of adjacent quadrotors when constructing obstacle-free spaces and includes significant virtual repulsive forces among UAVs in flocking control, which not only maintains formation but also enhances flight safety. In contrast, although AMSwarm and DMPC designed collision-free constraints, they did not fully address the safety risks posed by excessively close quadrotors.

TABLE II
PERFORMANCE COMPARISON IN MULTI-QUADROTORS

Num of UAVs	Method	Avg. Time. (s)	Avg. Vel. (m/s)	Distance (m)	Safety Ratio (m)
3	AMSwarm	7.70	3.29	36.07	0.35
	DMPC	14.20	3.20	**33.29**	1.00
	DMPCC	8.49	3.79	35.62	1.98
	T-STAR	**7.02**	**4.82**	34.75	**2.33**
5	AMSwarm	23.73	3.31	55.35	0.31
	DMPC	15.40	2.17	**33.77**	0.75
	DMPCC	9.23	3.68	35.44	2.11
	T-STAR	**6.89**	**4.92**	34.82	**2.98**
10	AMSwarm	13.37	2.92	35.79	0.99
	DMPC	16.80	2.17	**33.58**	1.50
	DMPCC	10.02	3.50	35.34	**1.62**
	T-STAR	**7.22**	**4.71**	34.88	1.43

C. Effect with Dynamic and Annular Obstacles

In a 36 m × 20 m × 3 m environment, we placed 20 cylindrical dynamic obstacles with 0.2 m radius. A centre point of motion was randomly chosen for each obstacle, and the obstacle's centre moved along a circular path with 0.2 m radius around that centre point at a linear speed of 1 m/s. Additionally, we placed 30 annular static obstacles, with radii randomly distributed between 0.5 m and 0.7 m. This simulation is used to test the performance of our method in a complex environment.

In Table III, we compared the performance metrics of T-STAR, AMSwarm, DMPC, and DMPCC in terms of average execution time, velocity, flight distance, and success rate. The results showed that the average execution time of T-STAR was reduced by 13.86%, 7.75%, and 27.48% compared to AMSwarm, DMPC, and DMPCC, respectively. Although the average speed of AMSwarm was 21.15% faster than T-STAR,

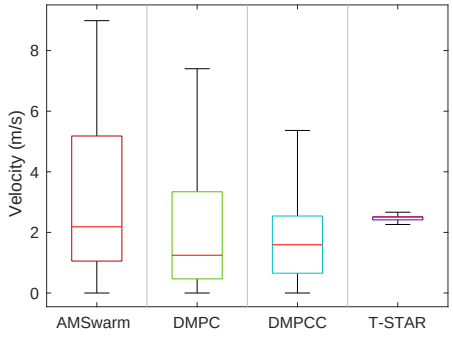

Fig. 15. A boxplot of velocity distribution with dynamic and annular obstacles.

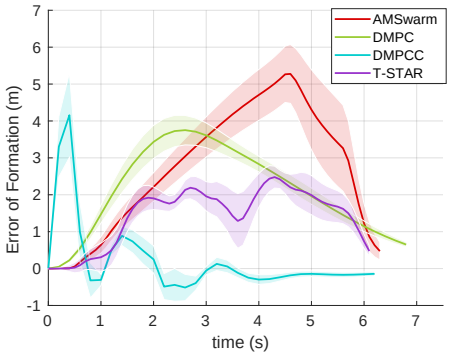

Fig. 16. Error of Formation. The red, green, blue, and purple zones display the 10 times simulation results of AMSwarm, DMPC, DMPCC, and T-STAR, respectively. The solid lines represent the average values.

its total flight distance was 34.99% longer, and AMSwarm has a lower success rate. The reason behind this is that our method predicts the trajectory of dynamic obstacles for the next 10 time steps based on their velocity direction when detected, establishing obstacle regions and then calculating a safe flight corridor using the method in Subsection IV-D. This approach compresses the feasible space but significantly increases the success rate. As shown in Fig. 15, the velocity fluctuations of our method are very small, indicating that the trajectory, although relatively conservative, has a higher success rate. In contrast, AMSwarm adds obstacle avoidance constraints only when the quadrotor is close enough to the obstacles, making it difficult to efficiently avoid dynamic obstacles, resulting in longer flight distances or failure.

Our method has a higher success rate than DMPC because DMPC considers collision constraints as soft constraints, which cannot guarantee collision-free optimisation results. In contrast, our method uses the flight safety corridor as a hard constraint, ensuring a collision-free trajectory.

TABLE III
PERFORMANCE COMPARISON IN COMPLICATED ENVIRONMENT

Method	Avg. Time. (s)	Avg. Vel. (m/s)	Distance (m)	Success Rate (%)
AMSwarm	16.02	**3.12**	52.44	50
DMPC	14.96	2.24	34.12	60
DMPCC	19.03	2.13	42.59	100
T-STAR	**13.80**	2.46	**34.09**	**100**

D. Flocking Analysis

In an environment of 36 m × 20 m × 3 m, we randomly placed 50 static cylindrical obstacles, with radii ranging from 0.3 m to 0.5 m and heights from 0.1 m to 3 m. This experiment is used to present the improvements of our method over the benchmark methods in flocking control. We define a new metric, "Error of Formation", which quantifies the error of the quadrotor swarm from the given formation in the form

$$Error = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \|\boldsymbol{p}_i - \boldsymbol{p}_j\| - D_{ij}^{ref},$$

where the position of ith quadrotor $\boldsymbol{p}_i$, nominal distance D_{ij}^{ref} between ith and jth quadrotor. In Table IV, Root Mean Square Error (RMSE) is used to describe the distribution of the error.

TABLE IV
PERFORMANCE COMPARISON IN FLOCKING CONTROL

Method	Avg. Error (m)	Max. Error (m)	RMSE (m)
AMSwarm	2.67	8.66	3.16
DMPC	2.10	4.82	2.40
DMPCC	**0.19**	4.16	**1.01**
T-STAR	1.43	**3.86**	1.62

The experimental results are shown in Fig. 16 and Table IV. Our method reduced the average error, maximum error, and RMSE by 46.44%, 55.43%, and 47.11%, respectively, compared to AMSwarm. Additionally, in comparison to DMPC, our method reduces the average error by 31.90%, the maximum error by 19.92%, and the RMSE by 26.73%. Fig. 16 shows that our method maintained a lower error for most of the flight time. This is because our method considers the relative distances between quadrotors in flocking control, proving that it can effectively form a tighter quadrotor swarm while keeping high speed. Moreover, the average error and RMSE of T-STAR are higher than that of DMPCC. This is because DMPCC primarily focuses on achieving formation consensus at the expense of flight speed, which has been demonstrated in previous experimental results. This is detrimental to the agility of UAVs in time-limited scenarios.

E. Experimental Validation

To assess the practicality and effectiveness of the proposed approach under real-world conditions, an experimental flight test was carried out with three Crazyflie 2.1 drones. The test environment was carefully designed to replicate the conditions of the simulation as closely as possible. For positioning purposes, the Lighthouse V2 system functioned as the transmitter, while the Lighthouse deck acted as both the receiver and the estimator of the drones' positions. Considering the performance of the Crazyflie, we set the velocity $\boldsymbol{v}_{min} = -2.0m \cdot s^{-1}, \boldsymbol{v}_{max} = 2.0m \cdot s^{-1}$, and acceleration $\boldsymbol{a}_{min} = -3.0m \cdot s^{-2}, \boldsymbol{a}_{max} = 3.0m \cdot s^{-2}$. All initial scenario parameters are detailed in TABLE V. The UAVs are expected to form an equilateral triangle with 0.5 m sides. We configured the initial positions in a line with a spacing of 0.4 m or 0.8 m between each UAV. Furthermore, landing positions were

TABLE V
REAL-WORLD SCENARIO PARAMETERS

Parameters	Value	
Positions ($\boldsymbol{p}$)	Initial (m)	Landing (m)
UAV1 UAV2 UAV3	$[0.1, 0.1, 0.0]$ $[0.1, 0.5, 0.0]$ $[0.1, 0.9, 0.0]$	$[2.3, 1.1, 0.0]$ $[2.3, 1.9, 0.0]$ $[2.3, 1.5, 0.0]$
Positions ($\boldsymbol{p}$)	Centre (m)	Size ($W * D * H$ m)
Obstacle1 Obstacle2	$[0.7, 0.6, 0.5]$ $[1.9, 1.4, 0.5]$	$[0.4, 1.2, 1.0]$
Nominal Distance (D^{ref})	$0.5m$	

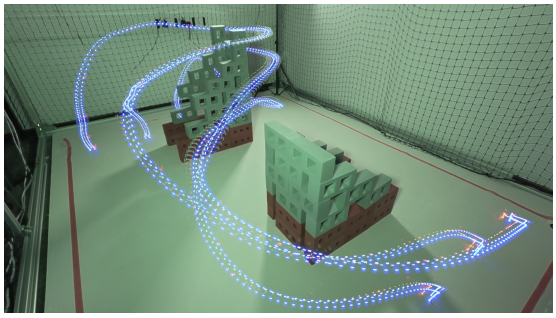

Fig. 17. Real-flight trajectories of a UAV swarm using T-STAR.

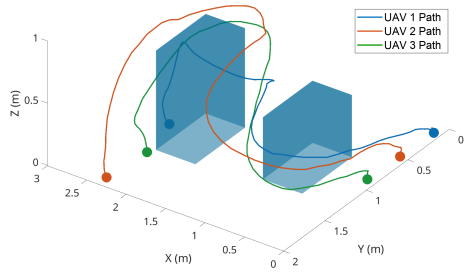

Fig. 18. Real-flight trajectories of the micro-UAV swarm generated by T-STAR.

assigned in an order (UAV1, UAV2, UAV3) that differs from the initial configuration (UAV1, UAV3, UAV2) to increase the complexity of the experiment and thereby validate the effectiveness and robustness of our method.

As illustrated in Fig. 17 and 18, all three Crazyflie drones took off from their designated initial points and avoided obstacles while following the optimal trajectory, thereby demonstrating the effectiveness of the proposed algorithm. The video for the experimental results can be found at https://youtu.be/uNTLgc3Iv4o?si=Eh3LtdlP1ttKX1qh

In addition, Fig. 19 illustrates the velocities of the drones during the experiment. The data reveals that the drones maintained a consistently smooth and high-speed flight, while effectively avoiding obstacles. Furthermore, Fig. 20 presents the actual formation errors of the three Crazyflie drones. The formation errors fluctuate normally due to the limited safe space, with an average value of 0.14 m, which demonstrates the robustness of our method under challenging conditions.

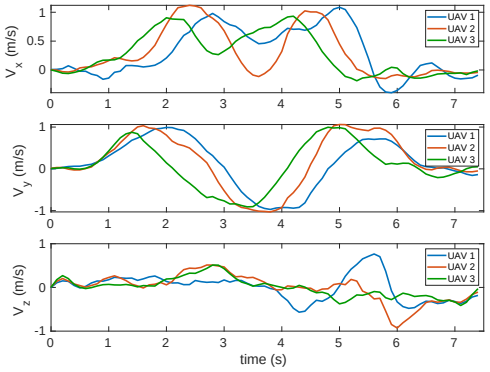

Fig. 19. Velocity time history of a real-flight swarm.

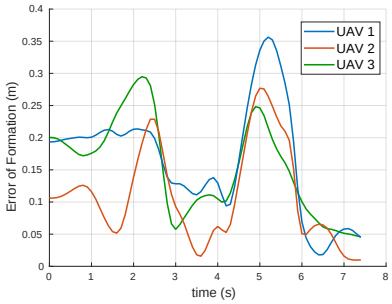

Fig. 20. Actual Error of Formation. Due to the limited safe space between the two obstacles, the UAVs deviated from formation to avoid collisions, causing the formation error to fluctuate at approximately 2.2 s and 5.1 s.

VII. CONCLUSION

This work proposed a distributed time-optimal swarm trajectory planner for environments with unknown and high-density obstacles. Leveraging differential flatness, we restructured the state update equations of model predictive contour control into a polynomial format, thereby reducing the complexity of the standard MPCC. Additionally, we applied virtual attractive and repulsive forces among multiple UAVs to achieve flocking control. Furthermore, to improve the speed of the UAV swarms, we relax the formation error by incorporating slack variables. Compared with AMSwarm and DMPC, our planner generates higher speeds, shorter task completion times, and safer trajectories in both single-UAV and multi-UAV experiments. Using DMPCC as a benchmark confirms the basic performance of our approach in single-UAV experiments and demonstrates its ability to generate high-speed trajectories in swarm systems. Although this requires a trade-off in formation quality, real-world experiments show that the formation error remains within acceptable bounds. Consequently, the real-world implementations validate the effectiveness and robustness of our method. In future work, we plan to investigate methods to further improve the robustness of the flocking control, such as considering UAV swarm faults and recovery. Additionally, communication delays among UAVs are a critical issue that can affect trajectory updates and overall performance. Therefore, we are interested in exploring how to adjust trajectory deconflict strategies to accommodate certain time delays due to unreliable communication networks in extreme environments.

REFERENCES

[1] B. Lopez, J. Munoz, F. Quevedo, C. A. Monje, S. Garrido, and L. Moreno, "4D Trajectory Planning Based on Fast Marching Square for UAV Teams," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 6, pp. 5703–5717, 6 2024.

[2] M. R. Rezaee, N. A. W. A. Hamid, M. Hussin, and Z. A. Zukarnain, "Comprehensive Review of Drones Collision Avoidance Schemes: Challenges and Open Issues," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[3] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1934–1940, 11 2019.

[4] A. Romero, R. Penicka, and D. Scaramuzza, "Time-Optimal Online Replanning for Agile Quadrotor Flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 7 2022.

[5] K. Wu, J. Hu, Z. Li, Z. Ding, and F. Arvin, "Distributed collision-free bearing coordination of multi-uav systems with actuator faults and time delays," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 11 768–11 781, 2024.

[6] N. Bashir, S. Boudjit, and G. Dauphin, "A Connectivity Aware Path Planning for a Fleet of UAVs in an Urban Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 10, pp. 10 537–10 552, 10 2023.

[7] Y. Liu, J. M. Montenbruck, D. Zelazo, M. Odelga, S. Rajappa, H. H. Bulthoff, F. Allgower, and A. Zell, "A Distributed Control Approach to Formation Balancing and Maneuvering of Multiple Multirotor UAVs," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 870–882, 8 2018.

[8] L. Quan, L. Yin, T. Zhang, M. Wang, R. Wang, S. Zhong, X. Zhou, Y. Cao, C. Xu, and F. Gao, "Robust and Efficient Trajectory Planning for Formation Flight in Dense Environments," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4785–4804, 12 2023.

[9] L. Bartolomei, L. Teixeira, and M. Chli, "Fast Multi-UAV Decentralized Exploration of Forests," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5576–5583, 9 2023.

[10] W. Honig, J. A. Preiss, T. K. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 8 2018.

[11] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online Distributed Trajectory Planning for Quadrotor Swarm with Feasibility Guarantee Using Linear Safe Corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 4 2022.

[12] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 4101–4107, 2021.

[13] B. Convens, K. Merckaert, M. M. Nicotra, and B. Vanderborght, "Safe, fast, and efficient distributed receding horizon constrained control of aerial robot swarms," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4173–4180, 2022.

[14] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model Predictive Contouring Control for Time-Optimal Quadrotor Flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 12 2022.

[15] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[16] 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, 4 2020.

[17] E. Tal and S. Karaman, "Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 5 2021.

[18] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-Racing: An Open-Source Strong Baseline for SE(3) Planning in Autonomous Drone Racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 10 2021.

[19] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011.

[20] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2832–2847, 2012.

[21] M. Greeff and A. P. Schoellig, "Flatness-Based Model Predictive Control for Quadrotor Trajectory Tracking," *IEEE International Conference on Intelligent Robots and Systems*, pp. 6740–6745, 12 2018.

[22] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 4 2018.

[23] M. Bangura and R. Mahony, "Thrust Control for Multirotor Aerial Vehicles," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 390–405, 4 2017.

[24] B. Lindqvist, S. S. Mansouri, A. A. Agha-Mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for Collision Avoidance and Control of UAVs with Dynamic Obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 10 2020.

[25] J. Ulrich, M. Stefanec, F. Rekabi-Bana, L. A. Fedotoff, T. Rouček, B. Y. Gündeğer, M. Saadat, J. Blaha, J. Janota, D. N. Hofstadler *et al.*, "Autonomous tracking of honey bee behaviors over long-term periods with cooperating robots," *Science Robotics*, vol. 9, no. 95, p. eadn6848, 2024.

[26] Y. Song and D. Scaramuzza, "Policy Search for Model Predictive Control With Application to Agile Drone Flight," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2114–2130, 8 2022.

[27] F. Nan, S. Sun, P. Foehn, and D. Scaramuzza, "Nonlinear MPC for Quadrotor Fault-Tolerant Control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5047–5054, 4 2022.

[28] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. 1221, 7 2021.

[29] G. Torrente, E. Kaufmann, P. Fohn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 4 2021.

[30] D. Lam, C. Manzie, and M. Good, "Application of Model Predictive Contouring Control to an X-Y Table," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 325–10 330, 1 2011.

[31] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 9 2015.

[32] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe Nonlinear Trajectory Generation for Parallel Autonomy with a Dynamic Vehicle Model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 9 2018.

[33] B. Lindqvist, P. Sopasakis, and G. Nikolakopoulos, "A Scalable Distributed Collision Avoidance Scheme for Multi-agent UAV systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 9212–9218, 2021.

[34] B. Zhou, H. Xu, and S. Shen, "RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1816–1835, 6 2023.

[35] C. J. Stamouli, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Multi-Agent Formation Control Based on Distributed Estimation with Prescribed Performance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2929–2934, 4 2020.

[36] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[37] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3475–3482, 2011.

[38] F. Rekabi-Bana, J. Hu, T. Krajnik, and F. Arvin, "Unified Robust Path Planning and Optimal Trajectory Generation for Efficient 3D Area Coverage of Quadrotor UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 2492–2507, 3 2024.

[39] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, Online Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 4 2017.

[40] C. E. Luis and A. P. Schoellig, "Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 1 2019.

[41] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 434–440, 5 2020.

[42] 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, 2 2022.
[43] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 12 2022.
[44] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.
[45] D. Harabor and A. Grastien, "Online Graph Pruning for Pathfinding On Grid Maps," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 1114–1119, 8 2011.
[46] 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, 7 2017.
[47] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically Constrained Trajectory Optimization for Multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 10 2022.
[48] H. Wang, J. Kearney, and K. Atkinson, "Arc-length parameterized spline curves for real-time simulation," in *Proc. 5th International Conference on Curves and Surfaces*, vol. 387396, 2002.
[49] V. K. Adajania, S. Zhou, A. K. Singh, and A. P. Schoellig, "AMSwarm: An Alternating Minimization Approach for Safe Motion Planning of Quadrotor Swarms in Cluttered Environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2023-May, pp. 1421–1427, 2023.
[50] M. Zhao and H. Li, "Distributed formation control of quadrotors using model predictive contouring control," in *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, 2023, pp. 1–6.

Honghao Pan received his BSc degree in Electrical Engineering and Automation from Shanghai Normal University in 2020, and the MSc degree in Aerial Robotics from University of Bristol in 2022. He is currently working toward the PhD degree in Computer Science from Durham University. His research interests include swarm intelligence, optimal control, distributed optimisation, and motion planning.

Mohsen Zahmatkesh is a Research Associate in Mechatronics, specialising in Control Systems. With expertise spanning Swarm Robotics, Intelligent Control, Reinforcement Learning, and Visual Servo Control, his academic foundation includes a BSc from Civil Aviation Technology College and an MSc from the prestigious Sharif University of Technology.

He is affiliated with the Swarm Robotics Lab, operating as a division of the Department of Computer Science at Durham University. His research contributions are featured in high-impact international scientific journals, reflecting his engagement in cutting-edge advancements in control and robotics.

Fatemeh Rekabi-Bana received her BSc (2010) and MSc (2012) in Aerospace Engineering from Amirkabir University of Technology (Tehran Polytechnic). She received her PhD (2020) in Mechanical Engineering Dynamics, Control and Vibration from the University of Tehran. She joined the EU-H2020 project (RoboRoyale) as a postdoc research associate at the University of Manchester and now she is working as a postdoctoral research associate at Durham University.

Farshad Arvin received the BSc degree in Computer Engineering, the MSc degree in Computer Systems Engineering, and the PhD degree in Computer Science, in 2004, 2010, and 2015, respectively. Farshad is a Professor of Robotics in the Department of Computer Science at Durham University in the UK. Prior to that, he was a Senior Lecturer in Robotics at The University of Manchester, UK. He visited several leading institutes including Artificial Life Laboratory at the University of Graz, Institute of Microelectronics, Tsinghua University, Beijing, and Italian Institute of Technology (iit) in Genoa as a Senior Visiting Research Scholar. His research interests include swarm robotics, multi-agent systems, and biohybrid robotics. He is the Founding Director of the Swarm & Computation Intelligence Laboratory formed in 2018, `www.SwaCIL.com`.

Junyan Hu received BSc degree in Automation from Hefei University of Technology in 2015 and PhD degree in Electrical and Electronic Engineering from the University of Manchester in 2020.

Junyan is an Assistant Professor with the Department of Computer Science at Durham University and a Fellow of Durham Energy Institute. Prior to that, he worked as a Lecturer at University College London and a Postdoctoral Research Associate at the University of Manchester. His research interests include swarm intelligence, multi-agent systems, cooperative planning and control, with applications to autonomous vehicles and robotics. He is a member of IEEE Technical Committee on Networks and Communication Systems and IEEE Technical Committee on Multi-Robot Systems. He served as an Associate Editor for IEEE Robotics and Automation Letters, and a Conference Editorial Board member for ICRA and CASE.