Decentralized Spatial-Temporal Trajectory Planning for Multicopter Swarms

Xin Zhou, Zhepei Wang, Xiangyong Wen, Jiangchao Zhu, Chao Xu and Fei Gao*

***Abstract*—Multicopter swarms with decentralized structure possess the nature of flexibility and robustness, while efficient spatial-temporal trajectory planning still remains a challenge. This report introduces decentralized spatial-temporal trajectory planning, which puts a well-formed trajectory representation named MINCO into multi-agent scenarios. Our method ensures high-quality local planning for each agent subject to any constraint from either the coordination of the swarm or safety requirements in cluttered environments. Then, the local trajectory generation is formulated as an unconstrained optimization problem that is efficiently solved in milliseconds. Moreover, a decentralized asynchronous mechanism is designed to trigger the local planning for each agent. A systematic solution is presented with detailed descriptions of careful engineering considerations. Extensive benchmarks and indoor/outdoor experiments validate its wide applicability and high quality. Our software will be released for the reference of the community.**

I. Introduction

Aerial swarm robotics capable of three-dimensional operations show superior flexibility (adaptability, scalability, and maintainability) and robustness (reliability, survivability, and fault-tolerance) compared to ground vehicles or single-agent systems [4]. In recent years, great amount of attentions have been paid to develop advanced architectures and algorithms that push the boundary of fully autonomous aerial swarms.

The key module in the swarm is its planning algorithm, which dominates the efficiency and feasibility of the formation flight. Investigating the most underlying demand for swarm planning, it is expected to not only deform the shape of trajectories to avoid collisions, but also adjust the time profiles to exploit the sequential solution space and squeeze the feasibility of agents. Spatial-temporal joint trajectory optimization is primary to achieve this, but is challenging even for a single agent. If only spatial deformation is performed [26], as shown in Fig. 3, agents tend to circumnavigate to wait for others while passing through a narrow passage, which hinders latter agents and results in inferior solutions. To this end, we adopt a recently developed trajectory representation named MINCO by Wang et al. [24], which is specially designed for spatial-temporal trajectory optimization for integrator chain systems. Moreover, since trajectory representations are critical but often ambiguous, starting with MINCO, we present a discussion (Sec. II-A) of several commonly used trajectory parameterization methods to developers in robotics community.

*Corresponding author.

All authors are with the State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China and Huzhou Institute, Zhejiang University, Huzhou 313000, China. (email: {iszhouxin, fgaoaa}@zju.edu.cn).

Fig. 1: Various outdoor experiments. Please watch the attached video for more information.

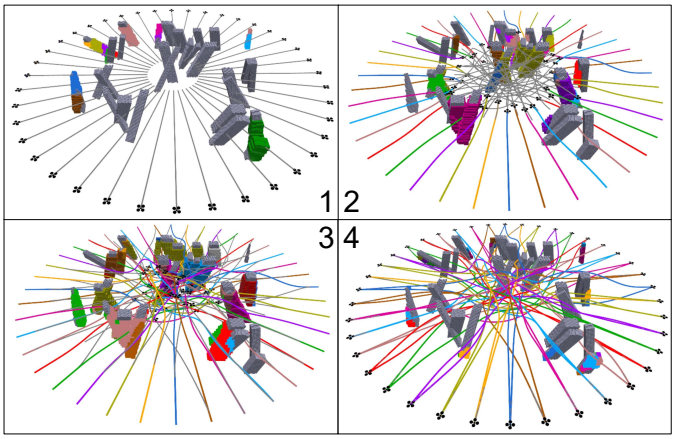


Fig. 2: Large scale position exchange. Colored curves indicate the positions agents have passed; gray curves are planned trajectories, colored obstacles represent the maps agents build. All the agents perform independent sensing, planning, and controlling. The program of all agents runs on a desktop CPU in real-time.

Based on MINCO, we propose a decentralized planner capable of spatial-temporal optimization for aerial swarms. To utilize this novel trajectory representation, several functions are designed to either penalize collision, restrict dynamical infeasibility or regularize the trajectory duration. Using the gradient propagation scheme of MINCO, the gradients of penalty functions on typical polynomial trajectories are efficiently propagated to those of MINCO. The planning problem is then formulated as an unconstrained optimization, which is

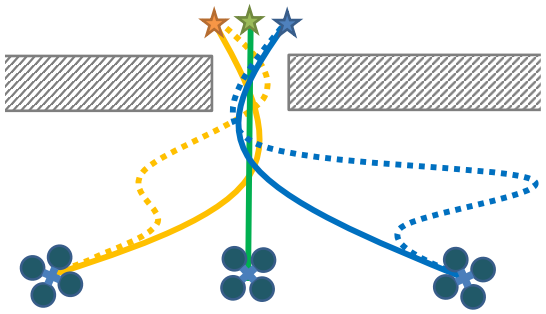


Fig. 3: Trajectory comparison with/without temporal optimization. Dotted curves: without temporal optimization, agents use detours to delay time. Solid curves: with temporal optimization, shorter and smoother trajectories are generated.

solved efficiently with a customized solver in milliseconds. We further deploy our proposed algorithm into a hardware system built with careful engineering considerations.

Besides algorithmic research, another fundamental requirement for the swarm is the architecture, which necessitates moderate inter-dependency, robust communication mechanism, and flexible system scale. This report presents a decentralized and asynchronously triggered planning strategy, which reduces the scale of the problem and distributes the computation burden to every single agent. Key components that bring this system to real-world, such as reciprocal drone detection, world frame alignments, networked trajectory broadcasting, and on-demand time synchronization, are also presented. Finally, the proposed fully autonomous swarm system is verified by extensive experiments in unknown, obstacle-rich, in/outdoor environments as Fig. 1 and 2.

The main points of this report are summarized as follows:

1) A decentralized and asynchronous planning framework, which decouples the whole swarm planning problem and makes the system robust to communication failure.
2) A spatial-temporal trajectory optimization method, extending MINCO representation to multi-agent scenarios.
3) Engineering practices that bring the fully autonomous swarm to real-world. The software will be released after the reception of our previous work [24] for the reference of the community.

II. Related Works

A. Trajectory Parameterization for Aerial Robots

Pros and cons of typical trajectory parameterization for aerial robots are summarized in Tab. I. **Polynomial** curves are widely used by traditional works [12, 13, 17], who mostly use polynomial coefficients as decision variables, along with explicit continuity constraints. Many works [13, 17] deform a piecewise polynomial trajectory by adding intermediate waypoints. However, the distribution of waypoints requires a careful configuration to balance feasibility satisfaction and computational burden. For optimizing the time profile of a polynomial, the representative method uses gradient descent with numerical differences [17]. Nevertheless, this method requires a dense matrix inversion to transform waypoints to coefficients, making the optimization quickly intractable when the problem scales-up. **Bézier** and **B-Spline** share the *convex hull* nature, which indicates that the curve is entirely contained in the convex hull of its control points [7, 20]. Therefore, these curves are convenient to add constraints, which can be done by merely restricting the control points inside feasible convex regions. Many works [8, 14, 18] follow this underlying property and show successful applications in aerial swarms. However, this property indeed brings conservativeness, as analyzed by Tordesillas and How [19], preventing a trajectory from being aggressive near its physical limits. Besides, an n order **B-Spline** is naturally $n-1$ order continuous [6]. Thus no continuity constraint is required as long as the B-Spline has a qualified order. For a **Bézier** curve, its time profile can be adjusted with acceptable complexity by its basis [7], while this is rather complicated for a B-Spline due to its time evaluation contains high nonlinearity. **MINCO** [24] is a newly developed trajectory representation specialized for integrator chain systems. The core feature of MINCO is that it efficiently handles a wide variant of constraints while retains spatial and temporal optimality. By discretizing the trajectory to add constraints, MINCO is less conservative but requires further checks after getting the solution. Although MINCO is the most complicated representation and requires a sophisticated implementation, it provides a solid fundamental for spatial-temporal drone swarm planning.

B. Decentralized Multicopter Swarms

For decentralized strategies, **VO** (velocity obstacle) is a lightweight approach to generate collision-free control commands [21, 22]. However, it does not produce trajectories with high-order continuity. Thus it is hard for a multicopter to execute. Arul and Manocha [1] incorporate **VO** into an MPC as velocity constraints, which improves the trajectory quality significantly. Moreover, MPC is widely used in the literature on aerial swarm robotics. Van Parys and Pipeleers [23] achieve formation control for a group of vehicles. Luis and Schoellig [10] use distributed MPC to perform point-to-point swarm transitions. Except that, Chen et al. [3] implements SCP for multiagent trajectory planning in non-convex scenarios by tightening collision constraints incrementally. However, the above methods require synchronization between the replans of different agents, which is hard to be guaranteed in real-world. To further decouple the system, Liu et al. [9], Tordesillas and How [18] propose decentralized and asynchronous planning strategies for drones to avoid static/dynamic obstacles and inter-vehicle collisions. However, the above optimization-based methods require at least tens of milliseconds, even several seconds, to compute a control command or a trajectory. What's more, those algorithms are only validated through simulations without integrating sensing, mapping, and localization. Early decentralized experimental results have been shown by McGuire et al. [11]. However, the naive minimum navigation method in that paper is doomed to produce discrete control commands with no consideration of system constraints. Recently, Zhou et al. [26] builds up a

TABLE I: Pros and cons of trajectory parameterization methods in several frequently concerned aspects. More descriptions of these methods, including implementations, related papers, and detailed comparisons, are presented in Sec. II-A.

Method	Continuity	Safety	Dynamical Feasibility	Temporal Optimization	Implementation Difficulty
Polynomial	Extra Overhead	By Discretization		Coupled	Easy
Bézier Curve		Analytical but Conservative			Medium
B-Spline	By Nature			Intractable	
MINCO		By Discretization		Decoupled	Difficult

fully autonomous multicpoter swarm system using B-Spline parameterized trajectories. Nevertheless, the difficulty of time adjustment of B-Spline curves results in a twisty trajectory shape when multiple agents need to pass through the same area. However, this limitation is broken in the proposed method.

III. Spatial-Temporal Trajectory Optimization For Swarms

A. Overview

The capability of spatial-temporal optimization comes from a recently developed trajectory representation named $\mathfrak{T}_{\text{MINCO}}$ [24]. It's basic definition and features are explained in Sec.III-B. Then Sec.III-C is a general description of how to add various continuous-time constraints to $\mathfrak{T}_{\text{MINCO}}$ trajectories: by constraint quadrature. Finally in Sec.III-D, we formulate an unconstrained nonlinear optimization, which is then solved by gradient-decent methods. To solve it in this way, costs and gradients of various objectives are defined and derived in Sec.III-D as well.

B. MINCO Trajectory Class

Thanks to the differential flatness of multicopters [12], their motion planning is sufficient to be directly performed on a time differentiable curve. In this report, we adopt $\mathfrak{T}_{\text{MINCO}}$ [24] for our trajectory representation, which is a minimum control polynomial trajectory class defined as

$$\mathfrak{T}_{\text{MINCO}} = \Big\{ p(t) : [0,T] \mapsto \mathbb{R}^m \Big| \ \mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T}), \\ \mathbf{q} \in \mathbb{R}^{m(M-1)}, \ \mathbf{T} \in \mathbb{R}^M_{>0} \Big\},$$

where $p(t)$ is an m-dimensional M-piece polynomial trajectory with degree $N = 2s-1$, s the order of the relevant integrator chain, $\mathbf{c} = (\mathbf{c}_1^{\mathrm{T}}, \ldots, \mathbf{c}_M^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{2Ms\times m}$ the polynomial coefficient, $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{M-1})$ the intermediate waypoints and $\mathbf{T} = (T_1, T_2, \ldots, T_M)^{\mathrm{T}}$ the time allocated for all pieces. Specifically, the trajectory is evaluated as

$$p(t) = p_i(t - t_{i-1}), \ \forall t \in [t_{i-1}, t_i]. \tag{1}$$

The i-th piece $p_i(t) : [0, T_i] \mapsto \mathbb{R}^m$ is defined by

$$p_i(t) = \mathbf{c}_i^{\mathrm{T}} \beta(t), \ \forall t \in [0, T_i], \tag{2}$$

where $\beta(t) := [1, t, \cdots, t^N]^{\mathrm{T}}$ is the natural basis, $\mathbf{c}_i \in \mathbb{R}^{2s\times m}$ the coefficient matrix, $T_i = t_i - t_{i-1}$ and $T = \sum_{i=1}^M T_i$. The core of $\mathfrak{T}_{\text{MINCO}}$ is the parameter mapping $\mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T})$ constructed from Theorem 2 in [24]. Technically, the mapping directly constructs a minimum control trajectory for an m-dimensional s-integrator chain for any specified initial and terminal conditions. An instance in $\mathfrak{T}_{\text{MINCO}}$ is intuitively shown in Fig. 4. Since this report focuses on swarm planning, we only intuitively introduce the features of $\mathfrak{T}_{\text{MINCO}}$. Detailed proofs are presented in [24].

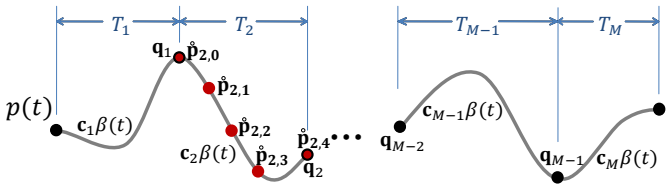


Fig. 4: This figure illustrates the parameters of MINCO trajectory representation and its constraint evaluation. The trajectory is directly parameterized by every waypoint $\mathbf{q}_i$ and time T_i. Red dots represent *constraint points* $\mathring{\mathbf{p}}$ (introduced in Sec. III-C), where any constraint is evaluated and propagated to every $\mathbf{q}_i$ and T_i.

Feature 1: Compactly represented by $\mathbf{q}$ and $\mathbf{T}$, $\mathfrak{T}_{\text{MINCO}}$ is a class of polynomial trajectories satisfying the following control effort minimization:

$$\min_{p(t)} \int_{t_0}^{t_M} p^{(s)}(t)^{\mathrm{T}} \mathbf{W} p^{(s)}(t) \mathrm{d}t, \tag{3a}$$

$$s.t. \ \ p^{[s-1]}(t_0) = \bar{p}_o, \ p^{[s-1]}(t_M) = \bar{p}_f, \tag{3b}$$

$$p(t_i) = q_i, \ 1 \le i < M, \tag{3c}$$

$$t_{i-1} < t_i, \ 1 \le i \le M, \tag{3d}$$

where $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a diagonal matrix with positive entries, $\bar{p}_o \in \mathbb{R}^{ms}$ and $\bar{p}_f \in \mathbb{R}^{ms}$ the specified initial and terminal conditions, $q_i \in \mathbb{R}^m$ is the given intermediate waypoints that the trajectory is enforced to pass at time t_i.

Feature 2: The evaluation and differentiation of the mapping $\mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T})$ enjoys linear complexity. A more specific correspondence can be expressed as the following function

$$\mathbf{M}(\mathbf{T})\mathbf{c} = \mathbf{b}(\mathbf{q}), \tag{4}$$

where $\mathbf{M}(\mathbf{T}) \in \mathbb{R}^{2Ms\times 2Ms}$ is a banded matrix with non-singularity for any $\mathbf{T} \succ \mathbf{0}$ ensured by Theorem 2 in [24], $\mathbf{b}(\mathbf{q}) \in \mathbb{R}^{2Ms\times m}$. The conversion between these two trajectory representations, i.e., $(\mathbf{c},\mathbf{T})$ and $(\mathbf{q},\mathbf{T})$ is achieved using *Banded PLU Factorization* with $O(M)$ linear time and space complexity. The evaluation is extremely fast, which takes about $1\mu s$ per piece for minimum-jerk trajectory generation on a desktop CPU.

Feature 3: Feature 2 allows any user-defined objective or penalty function $F(\mathbf{c},\mathbf{T})$ with available gradients applicable

to $\mathfrak{T}_{\text{MINCO}}$ parameterized in $(\mathbf{q},\mathbf{T})$. Specifically, the corresponding objective of $\mathfrak{T}_{\text{MINCO}}$ is computed as

$$H(\mathbf{q},\mathbf{T}) = F(\mathcal{M}(\mathbf{q},\mathbf{T}),\mathbf{T}). \tag{5}$$

Then the mapping $\mathbf{c} = \mathcal{M}(\mathbf{q},\mathbf{T})$ gives a linear-complexity way to compute $\partial H/\partial \mathbf{q}$ and $\partial H/\partial \mathbf{T}$ from the corresponding $\partial F/\partial \mathbf{c}$ and $\partial F/\partial \mathbf{T}$. After that, a high-level optimizer is able to optimize the objective efficiently.

C. Time Integral Constraints

The way how various constraints are implemented closely relevant to trajectory parameterization methods. Conventionally, requirements for dynamical feasibility and collision avoidance are formulized as functional-type constraints $\mathcal{G}(p(t),\dots,p^{(s)}(t)) \preceq \mathbf{0}, \forall t \in [0,T]$. However, this formulation cannot be directly handled by constrained optimization, since $\mathcal{G}$ contains infinitely many inequality constraints. Thus, we transform $\mathcal{G}$ into a finite-dimensional one via integral of constraint violation. Practically, the integral is transformed into weighted sum $J_\Sigma(\mathbf{c},\mathbf{T})$ of the sampled penalty function. In most cases that constraints are decoupled, i.e., constraints $\mathcal{G}(p^{[s]}(t))$ with $t_i \le t < t_{i+1}$ are merely determined by $\mathbf{c}_i$ and T_i, the penalty for i-th trajectory piece is computed as

$$J_i(\mathbf{c}_i,T_i,\kappa_i) = \frac{T_i}{\kappa_i}\sum_{j=0}^{\kappa_i} \bar{\omega}_j \chi^{\mathrm{T}} \max\left(\mathcal{G}(\mathbf{c}_i,T_i,\frac{j}{\kappa_i}),\mathbf{0}\right)^3, \tag{6}$$

where κ_i is the sample number on the i-th piece, $\chi \in \mathbb{R}^{n_g}_{\ge 0}$ a vector of penalty weights with appropriately large entries, $(\bar{\omega}_0,\bar{\omega}_1,\dots,\bar{\omega}_{\kappa_i-1},\bar{\omega}_{\kappa_i}) = (1/2,1,\cdots,1,1/2)$ are the quadrature coefficients following the trapezoidal rule [15]. We define the points determined by $\{\mathbf{c}_i,T_i,j/\kappa_i\}$ as *constraint points* $\mathring{\mathbf{p}}_{i,j} = p_i((j/\kappa_i)T_i)$ with $p_i(t)$ the i-th polynomial piece. Then $\mathcal{G}(\mathbf{c}_i,T_i,j/\kappa_i) = \mathcal{G}(\mathring{\mathbf{p}}_{i,j})$. Note that $\mathring{\mathbf{p}}_{i,\kappa_i} = \mathring{\mathbf{p}}_{i+1,0}$. The cubic penalty is used here for its twice continuous differentiability. $J_\Sigma(\mathbf{c},\mathbf{T})$ is then defined as the summation:

$$J_\Sigma(\mathbf{c},\mathbf{T}) = \sum_{i=1}^{M} J_i(\mathbf{c}_i,T_i,\kappa_i). \tag{7}$$

Since the gradients w.r.t. $(\mathbf{c},\mathbf{T})$ are constructed from all $(\mathbf{c}_i,\mathbf{T}_i)$, we only give gradient templates to $\mathbf{c}_i$ and T_i using the chain rule,

$$\frac{\partial J_\Sigma}{\partial \mathbf{c}_i} = \frac{\partial J_\Sigma}{\partial \mathcal{G}}\frac{\partial \mathcal{G}}{\partial \mathbf{c}_i}, \tag{8}$$

$$\frac{\partial J_\Sigma}{\partial T_i} = \frac{J_i}{T_i} + \frac{\partial J_\Sigma}{\partial \mathcal{G}}\frac{\partial \mathcal{G}}{\partial t}\frac{\partial t}{\partial T_i}, \tag{9}$$

$$\frac{\partial J_\Sigma}{\partial \mathcal{G}} = 3\frac{T_i}{\kappa_i}\sum_{j=0}^{\kappa_i} \bar{\omega}_j \max\left(\mathcal{G}(\mathbf{c}_i,T_i,\frac{j}{\kappa_i}),\mathbf{0}\right)^2 \circ \chi. \tag{10}$$

$$\frac{\partial t}{\partial T_i} = \frac{j}{\kappa_i}, \quad t = \frac{j}{\kappa_i}T_i. \tag{11}$$

From above equations, once the gradients of constraints $\mathcal{G}$ relative to polynomial coefficients $\mathbf{c}_i$ and time t are evaluated, the gradients of $J_\Sigma(\mathbf{c},\mathbf{T})$ are efficiently computed.

D. Optimization Problem Construction

According to the above definitions of trajectory parameterization and constraints imposition, we present the complete optimization for trajectory planning in this section. The basic requirements on a desired trajectory are smoothness, dynamical feasibility, as well as safety among obstacles and other agents. Extra objectives such as minimization of control effort and execution time are also desired. In this report, we adopt $\mathfrak{T}_{\text{MINCO}}$ to address above all concerns. Since $\mathfrak{T}_{\text{MINCO}}$ is naturally guaranteed smooth by Theorem 2 in [24], no extra effort needs to be paid on trajectory continuity.

In this work, we formulate the trajectory generation as an unconstrained optimization problem:

$$\min_{\mathbf{q},\mathbf{T}} \ \sum_x \lambda_x J_x, \tag{12}$$

where J_x are J_Σ instances, λ_x is the weight for either an objective or penalty, subscripts $x = \{e,t,d,o,w,u\}$ represent control effort (e), execution time (t), dynamical feasibility (d), obstacle avoidance (o), swarm reciprocal avoidance (w), and uniform distribution of constraint points (u). Commonly, a sufficiently large weight suffices for penalties. The problem is solved via unconstrained nonlinear optimization.

1) Control Effort J_e: Control effort follows the definition in Eq. 3b. Without loss of generality, consider a single dimension of the i-th piece $p_i(t) : [0,T_i] \mapsto \mathbb{R}$ of a polynomial spline, its derivatives with respect to $\mathbf{c}_i$ and T_i are

$$\frac{\partial J_e}{\partial \mathbf{c}_i} = 2\left(\int_0^{T_i} \beta^{(s)}(t)\beta^{(s)}(t)^{\mathrm{T}} dt\right)\mathbf{c}_i, \tag{13}$$

$$\frac{\partial J_e}{\partial T_i} = \mathbf{c}_i^{\mathrm{T}}\beta^{(s)}(T_i)\beta^{(s)}(T_i)^{\mathrm{T}}\mathbf{c}_i. \tag{14}$$

2) Execution Time J_t: A shorter execution time is desirable, so we also minimize the weighted total execution time $J_t = \sum_{i=1}^{M} T_i$. Obviously, its gradient $\partial J_t/\partial \mathbf{c} = \mathbf{0}$, $\partial J_t/\partial \mathbf{T} = \mathbf{1}$.

3) Dynamical Feasibility Penalty J_d: For the trajectory generation of the multicopter, dynamical feasibility is always guaranteed by limiting the trajectory's derivatives. In our work, we limit the amplitude of velocity, acceleration, and jerk. Note that the dynamical feasibility penalty is acquired using time integral in Sec. III-C. Following the Eq. 6, constraints of velocity, acceleration, and jerk are denoted as

$$\mathcal{G}_v = \dot{p}(t)^2 - v_m^2, \quad \mathcal{G}_a = \ddot{p}(t)^2 - a_m^2, \quad \mathcal{G}_j = \dddot{p}(t)^2 - j_m^2, \tag{15}$$

where v_m, a_m, j_m are maximum allowed velocity, acceleration and jerk. The corresponding gradients are

$$\frac{\partial \mathcal{G}_x}{\partial \mathbf{c}_i} = 2\beta^{(n)}(t)p^{(n)}(t)^{\mathrm{T}}, \quad \frac{\partial \mathcal{G}_x}{\partial t} = 2\beta^{(n+1)}(t)^{\mathrm{T}}\mathbf{c}_i p^{(n)}(t), \tag{16}$$

where $x = \{v,a,j\}$, $n = \{1,2,3\}$, and $t = jT_i/\kappa_i + t_{i-1}, t \in [t_{i-1},t_i]$, respectively. Substituting Eq. 15 into 6 and 16 into Eq.8, 9 gets the penalty and the gradient of $\mathbf{c}_i$ and $\mathbf{T}_i$;

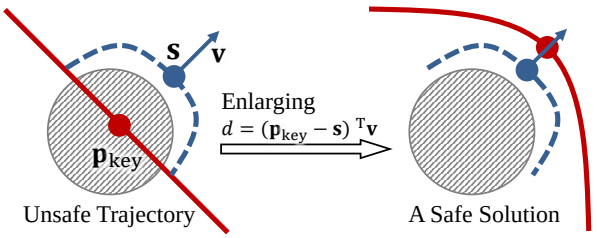


Fig. 5: Collision avoidance object formulation. The red curve is the optimized trajectory; the blue curve is a collision-free path starts and ends on the red curve. Then a fixed point $\mathbf{s}$ with a vector $\mathbf{v}$ is generated to determine the collision penalty.

4) Obstacle Avoidance Penalty J_c: In this work, we adopt the front-end of collision evaluation from Zhou et al. [27]. Also a brief description is given here. In [27], the information for an unsafe trajectory to escape collision is extracted by comparing the unsafe initial trajectory with a collision-free guiding path. As depicted in Fig. 5, a fixed safe point $\mathbf{s}$ with a fixed safe vector $\mathbf{v}$ is recorded by a corresponding key point $\mathbf{p}_{\text{key}}$ on the trajectory. Then the distance of $\mathbf{p}_{\text{key}}$ to the obstacle which the $\{\mathbf{s}, \mathbf{v}\}$ pair generated is defined as

$$d(\mathbf{p}_{\text{key}}, \{\mathbf{s}, \mathbf{v}\}) = (\mathbf{p}_{\text{key}} - \mathbf{s})^{\mathrm{T}}\mathbf{v}. \tag{17}$$

Using the distance information, the trajectory deforms iteratively during optimization. If $\mathbf{p}_{\text{key}}$ discovers a new obstacle, a new $\{\mathbf{s}, \mathbf{v}\}$ pair is stacked to $\mathbf{p}_{\text{key}}$'s records. Therefore each $\mathbf{p}_{\text{key}}$ may record several $\{\mathbf{s}, \mathbf{v}\}$ pairs.

In this work, constraint points $\mathring{\mathbf{p}}_{i,j}$ with $1 \le i \le M, 0 < j \le \kappa_i$ are selected as the key points $\mathbf{p}_{\text{key}}$. To enforce safety, we penalize distance to obstacles less than a safe clearance $\mathcal{C}_o$. Therefore, obstacle avoidance constraint is defined as

$$\mathcal{G}_o(p(t)) = (\cdots, \mathcal{G}_{o_k}(p(t)), \cdots)^{\mathrm{T}} \in \mathbb{R}^{N_{\mathbf{sv}}}, \tag{18}$$

$$\mathcal{G}_{o_k}(p(t)) = \mathcal{C}_o - d(p(t), \{\mathbf{s}, \mathbf{v}\}_k), \tag{19}$$

where $N_{\mathbf{sv}}$ is the number of $\{\mathbf{s}, \mathbf{v}\}$ pairs recorded by a single $\mathring{\mathbf{p}}_{i,j}$ and $t = jT_i/\kappa_i$ the relative time on this piece. For the case that $\mathcal{C}_o \ge d(p(t), \{\mathbf{q}, \mathbf{v}\}_k), 0)$, the gradient is computed:

$$\frac{\partial \mathcal{G}_{o_k}}{\partial c_i} = -\beta(t)\mathbf{v}^{\mathrm{T}}, \quad \frac{\partial \mathcal{G}_{o_k}}{\partial t} = -\mathbf{v}^{\mathrm{T}}\dot{p}(t). \tag{20}$$

Otherwise, $\partial \mathcal{G}_{o_k}/\partial c_i = \mathbf{0}_{2s\times m}, \partial \mathcal{G}_{o_k}/\partial t = 0$.

5) Swarm Reciprocal Avoidance Penalty J_w: In this work, a non-cooperative swarm framework is adopted, which means that one agent receives other agents' trajectories as constraints and generates trajectories only for itself. Considering the u-th agent in a multicpoter swarm containing U agents, swarm collision avoidance is guaranteed when the trajectory $p_u(t)$ of agent u maintains a distance greater than a safe clearance $\mathcal{C}_w$ to all the trajectories at the same global timestamps of other agents as depicted in Fig. 6. However, one caution must be taken that we have always been using a relative time $t = jT_i/\kappa_i$ in our optimization, while the other agents' trajectories take an absolute timestamp $\tau = T_1 + \cdots + T_{i-1} + jT_i/\kappa_i$. This indeed makes the penalty evaluated on the i-th piece depend on all its preceding piece times. Thus we denote by $G_w^{i,j}$ the

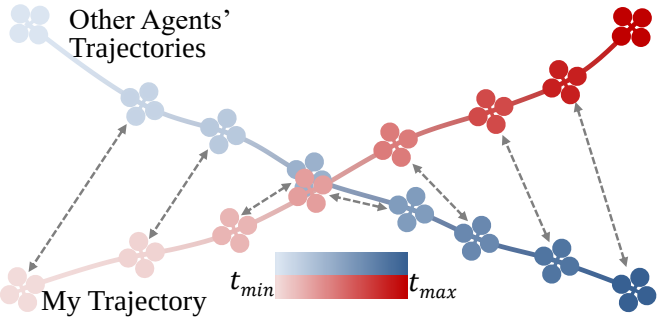


Fig. 6: Collision avoidance formulation between two agents. Gray dotted arrows indicate states at the identical global time. The goal of reciprocal avoidance is to maintain a safe distance at any time.

constraint evaluated at the j-th constraint point on the i-th piece of $p_u(t)$. Therefore the reciprocal avoidance constraint is defined as

$$\mathcal{G}_w^{i,j}(p_u(t), \tau) = (\cdots, \mathcal{G}_{w_k}(p_u(t), \tau), \cdots)^{\mathrm{T}} \in \mathbb{R}^U, \tag{21}$$

$$\mathcal{G}_{w_k}^{i,j}(p_u(t), \tau) = \begin{cases} \mathcal{C}_w^2 - d^2(p_u(t), p_k(\tau)) & k \ne u, \\ 0 & k = u, \end{cases} \tag{22}$$

$$d(p_u(t), p_k(\tau)) = \left\| \mathbf{E}^{1/2}(p_u(t) - p_k(\tau)) \right\|, \tag{23}$$

where $p_k(\tau)$ is the trajectory of the k-th agent that the u-th agent has to avoid at the same but relative stamp t. The matrix $\mathbf{E} := \mathrm{diag}(1, 1, 1/\mathrm{c})$ with $c > 1$ transforms Euclidean distance into *ellipsoidal distance* with the minor axes at the z-axis to relieve downwash risk from rotors. Squared distance is used to avoid *square root* operations.

When $\mathcal{C}_w^2 \ge d^2(p_u(t), p_k(\tau))$, the gradient to $\mathbf{c}_i$ is

$$\frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial \mathbf{c}_i} = \begin{cases} -2\beta(t)(p_u(t) - p_k(\tau))^{\mathrm{T}}\mathbf{E} & k \ne u, \\ \mathbf{0} & k = u. \end{cases} \tag{24}$$

Then substituting the sum of $\mathcal{G}_{w_k}^{i,j}$ into Eq.8 gets the gradient $\partial J_w/\partial \mathbf{c}_i$. However, the gradient to $\mathbf{T}$ is more complicated, since the gradient template equation 9 does not hold here. Considering previous time profile, the proper gradient to the preceding time T_l for any $1 \le l \le i$ should be computed as

$$\frac{\partial J_w}{\partial T_l} = \sum_{k=1}^{U} \frac{\partial J_{w_k}}{\partial T_l} = \sum_{k=1}^{U}\sum_{i=1}^{M}\sum_{j=0}^{\kappa_i} \frac{\partial J_{w_k}^{i,j}}{\partial T_l}, \tag{25}$$

$$\frac{\partial J_{w_k}^{i,j}}{\partial T_l} = \frac{J_{w_k}^{i,j}}{T_l} + \frac{\partial J_{w_k}^{i,j}}{\partial \mathcal{G}_{w_k}^{i,j}} \frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial T_l}, \tag{26}$$

$$\frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial T_l} = \frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial t}\frac{\partial t}{\partial T_l} + \frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial \tau}\frac{\partial \tau}{\partial T_l}, \tag{27}$$

$$\frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial t} = \begin{cases} 2\left(p_k(t) - p_u(\tau)\right)^{\mathrm{T}}\mathbf{E}\dot{p}_u(t) & k \ne u, \\ 0 & k = u, \end{cases} \tag{28}$$

$$\frac{\partial \mathcal{G}_{w_k}^{i,j}}{\partial \tau} = \begin{cases} 2\left(p_u(t) - p_k(\tau)\right)^{\mathrm{T}}\mathbf{E}\dot{p}_k(\tau) & k \ne u, \\ 0 & k = u, \end{cases} \tag{29}$$

$$\frac{\partial t}{\partial T_l} = \begin{cases} \frac{j}{\kappa_i} & l = i, \\ 0 & l < i, \end{cases} \quad \frac{\partial \tau}{\partial T_l} = \begin{cases} \frac{j}{\kappa_i} & l = i, \\ 1 & l < i. \end{cases} \tag{30}$$

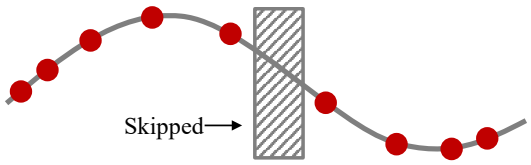


Fig. 7: Non-uniformly distributed constraint points (red dots) fail to detect a thin obstacle.

Note that $J_{wk}^{i,j}$ denotes the swarm penalty of j-th constraint point at i-th polynomial piece to agent k's current trajectory. The trajectory $p_k(\tau)$ for any other agent is already known, thus its derivatives are all available in computation.

6) Uniform Distribution Penalty J_v: The purpose of this penalty is to make the constraint points $\mathring{\mathbf{p}}_{i,j}$ equally spaced for all $1 \leq i \leq M$ and $0 < j \leq \kappa_i$, which is a simpler substitution to the space-uniform variant of $\mathfrak{T}_{\text{MINCO}}$ mentioned in [24]. It is necessary since if all $\mathfrak{T}_{\text{MINCO}}$ parameters $\mathbf{q}$ and $\mathbf{T}$ are optimized freely, some pieces of the trajectory tend to vanish to reduce the total cost. This phenomenon is harmful for trifold reasons. Firstly, $T_i = 0$ for the i-th trajectory piece is an undefined singular point for $\mathfrak{T}_{\text{MINCO}}$, which does not consider consecutive aligned waypoints. Secondly, since the spatial collision avoidance is constrained by a finite number of constraint points, non-uniform distribution increases the possibility of skipping some tiny or thin obstacles, as shown in Fig. 7. Thirdly, since we compute the distance to obstacles following Eq. 17, the obstacles are modeled as planes. The accuracy of this modeling decreases when constraint points move along the plane.

To enforce the uniform distribution of constraint points, we penalize the variance of the squared distances between each pair of adjacent constraint points. For simplicity, we denote $N_c = \sum_{i=0}^{M} \kappa_i$ as the total number of distances. The variance is computed as

$$J_u = \mathrm{E}\left[\mathbf{D}^2\right] - \mathrm{E}\left[\mathbf{D}\right]^2 = \frac{1}{N_c}\left\|\mathbf{D}\right\|_2^2 - \frac{1}{N_c^2}\left\|\mathbf{D}\right\|_1^2, \tag{31}$$

where $\mathbf{D} \in \mathbb{R}^{N_c}$ is defined by

$$\mathbf{D} = \left(\left\|\mathring{\mathbf{p}}_{1,1} - \mathring{\mathbf{p}}_{1,0}\right\|_2^2, \cdots, \left\|\mathring{\mathbf{p}}_{M,\kappa_M} - \mathring{\mathbf{p}}_{M,\kappa_M-1}\right\|_2^2\right). \tag{32}$$

$\mathbf{D}$ is the squared distance vector for all $N_c + 1$ constraint points. Denote by $\mathbf{D}_k$ the k-th entry in $\mathbf{D}$ based on the conversion between two kinds of indices $k = j + \sum_{l=1}^{i-1} \kappa_l$. The gradient is computed as

$$\frac{\partial J_u}{\partial \mathbf{c}_i} = \sum_{j=1}^{\kappa_i} \beta\left(\frac{jT_i}{\kappa_i}\right)\left(\frac{\partial J_u}{\partial \mathring{\mathbf{p}}_{i,j}}\right)^{\mathrm{T}}, \tag{33}$$

$$\frac{\partial J_u}{\partial T_i} = \frac{1}{\kappa_i}\sum_{j=1}^{\kappa_i} j\left(\frac{\partial J_u}{\partial \mathring{\mathbf{p}}_{i,j}}\right)^{\mathrm{T}} \mathbf{c}_i^{\mathrm{T}}\dot{\beta}\left(\frac{jT_i}{\kappa_i}\right), \tag{34}$$

$$\begin{aligned}\frac{\partial J_u}{\partial \mathring{\mathbf{p}}_{i,j}} = &\frac{4}{N_c}\left(\mathbf{D}_{k-1} - \frac{\left\|\mathbf{D}\right\|_1}{N_c}\right)\left(\mathring{\mathbf{p}}_{i,j} - \mathring{\mathbf{p}}_{i,j-1}\right), \\ &- \frac{4}{N_c}\left(\mathbf{D}_k - \frac{\left\|\mathbf{D}\right\|_1}{N_c}\right)\left(\mathring{\mathbf{p}}_{i,j+1} - \mathring{\mathbf{p}}_{i,j}\right).\end{aligned} \tag{35}$$

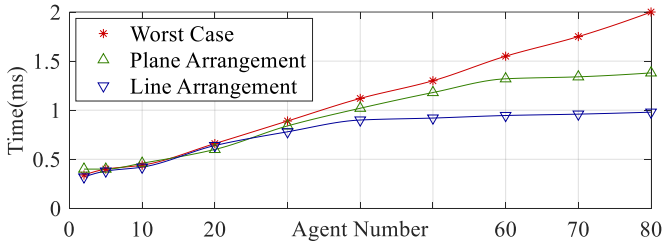


Fig. 8: Scalability evaluation in obstacle-free scenarios. Agents in the ***Worst Case*** that considering all the others achieve linear complexity relative to the agent amount. If ignoring agents' trajectories outside the planning horizon, the complexity is further reduced.

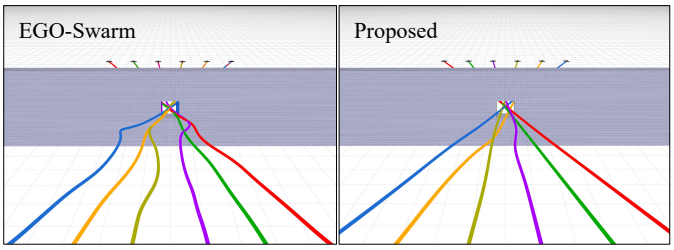


Fig. 9: Passing through a narrow gate with EGO-Swarm and the proposed planner. Due to spatial-temporal planning capability, the proposed method achieves smoother trajectories

7) Temporal Constraint Elimination: An open-domain constraint is $\mathbf{T} \succ \mathbf{0}$. Unlike previous constraints from Sec.III-D3 to III-D6 which are transformed into penalty functions, this constraint is directly eliminated by a diffeomorphism map as is done in [24]:

$$T_i = e^{\tau_i}, \tag{36}$$

where τ_i is the unconstrained virtual time. Note that the meaning of notion τ here is different from Sec.III-D5. Gradients w.r.t. T_i is propagated to τ_i following $\partial J/\partial \tau_i = (\partial J/\partial T_i)e^{\tau_i}$. More generally, any twice continuously differentiable bijection $f : \mathbb{R} \mapsto \mathbb{R}_{>0}$ suffices for this map.

IV. Benchmark

A. Large Scale Simulation

We conduct tests on large-scale planning scenarios, where 40 agents perform position exchanging on a circle with a $12.5m$ radius, as depicted in Fig.2.

B. Scalability

Since a non-cooperative swarm framework is adopted, the complexity of the proposed method relative to agent number U is $O(U)$. What's more, the complexity can be further reduced based on a consensus that the capacity of a given space is finite. Therefore, the received trajectory is neglected if it is outside the planning horizon. Here we demonstrate the scalability on replanning time of each agent in three scenarios. In the ***Worst Case***, the trajectory ignoring mechanism is disabled. In the ***Line Arrangement*** case, agents start in a straight line and fly to targets uniformly placed on another line $50m$ away. In the ***Plane Arrangement*** case, agents are

TABLE II: Comparisons in an $8 \times 8m$ empty space containing eight agents with radius of $0.25m$. Maximum velocity and acceleration are set to $1.7m/s$ and $6m/s^2$. *Safety Ratio* is the minimum agent interval divided by two times of radius. **int**(a^2) and **int**(j^2) are time integral of squared acceleration and jerk, indicating smoothness and control effort. The units of time and distance are seconds and meters.

Online/ Offline	Method	Solver Time	Traj. Time	Traj. Length	Safety Ratio	Safe?	int(a^2)	int(j^2)
Offline	SCP, h_scp=0.3s	1.46	7.65	**9.64**	0.267	No	2.63	7.01
	SCP, h_scp=0.17s	7.72	**7.40**	9.70	0.480	No	3.14	17.5
	RBP,no batches	**0.320**	15.4	11.3	1.02	Yes	**1.30**	0.608
	RBP,batch_size=4	0.413	15.4	11.5	1.06	Yes	1.32	**0.604**
Online	Mader	0.0239	**7.50**	12.2	1.37	Yes	12.0	125.1
	EGO, horizon=7.5m	0.000554	8.12	10.1	1.34	Yes	7.39	72.0
	EGO, horizon=10m	0.000844	8.17	10.3	1.62	Yes	8.15	94.1
	Proposed, $\kappa_i = 5$	**0.000465**	7.57	**9.70**	1.17	Yes	5.33	30.4
	Proposed, $\kappa_i = 8$	0.000557	7.58	9.71	1.22	Yes	**5.15**	**28.2**

TABLE III: Comparisons in a $20\times20\times5m$ space with 100 obstacles and 8 agents. This table shares the same parameters, units, and notations as Tab .II.

Method	Solver Time	Traj. Time	Traj. Length	Safety Ratio	Dist. to Obs.	Safe?	int(a^2)	int(j^2)
MADER	0.0539	19.4	30.25	1.52	0.466	Yes	39.6	960.9
EGO	0.000882	19.6	29.8	1.57	0.623	Yes	20.3	198.4
Proposed	**0.000755**	**16.9**	**29.1**	1.22	0.600	Yes	**13.5**	**75.3**

initialized on a plane with targets uniformly placed on another plane. The order of targets in all tests is randomly selected. Results are shown in Fig. 8

C. Comparisons

We compare the proposed planner against **SCP** [2], **RBP** [14], **MADER** [18], and **EGO** (EGO-Swarm) [26]. All presented data is the average of 8 agents in 10 runs, except for **Safety Ratio** and **Distance to Obstacles** (in abbreviation **Dis. to Obs.**) are the minimal value in all records.

*1) Comparison of Reciprocal Collision Avoidance **without** Static Obstacles:* Results are given in Tab. II. A bold term indicates a better value in the corresponding category (Online/Offline), while an underlined term indicates the best value among all planners. Indicated by Tab. II, **SCP** achieves a better flight time at the sacrifice of trajectory safety. **RBP** achieves better dynamical performance **int**(a^2) and **int**(j^2) with the cost of a relatively long flight time. Both of these methods fail to balance various aspects of trajectory quality. Compared with offline methods, three online planners show more balances while requiring less computation by orders of magnitude. Among them, **MADER** shows more conservativeness due to the simplex representation of the trajectory, although the simplex enjoys almost the minimum volume.

*2) Comparison of Reciprocal Collision Avoidance **with** Static Obstacles:* Results are presented in Tab. III for three online planners. Note that **MADER** uses a pre-built map while **EGO-Swarm** and the proposed method perform online sensing and mapping. From the data, **MADER** demands aggressive maneuvers for the vehicle while the other two are more superior in their trajectory smoothness. Another experiment to intuitively visualize the benefits of temporal optimization is illustrated in Fig. 9.

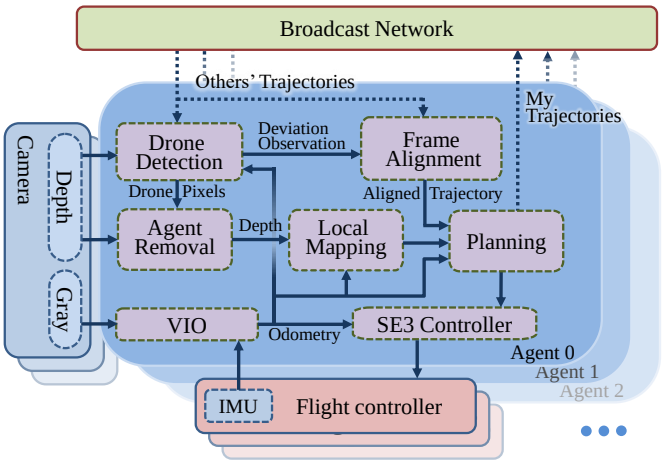


Fig. 10: System Architecture.

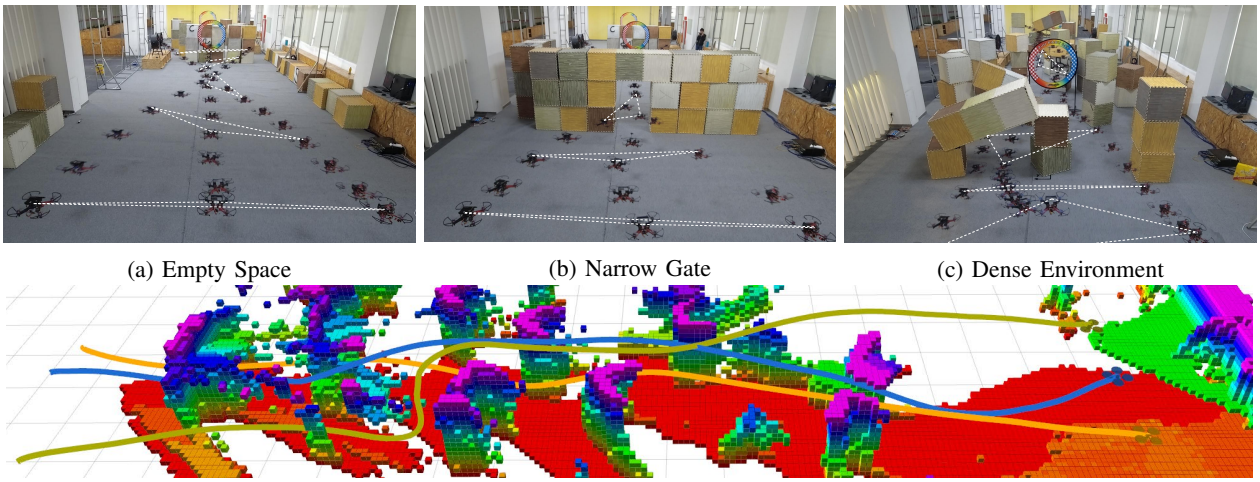
3) A Brief Conclusion: From the comparison, the proposed method achieves top-level performance with the shortest computation time. This achievement is attributed to low complexity of $\mathfrak{T}_{\text{MINCO}}$-based problem formulation and decoupling of spatial-temporal parameters. All the programmers run on an Intel Core i7-10700KF CPU at 5.1GHz.

V. Real-World Experiments

A. System Architecture

System architecture is depicted in Fig. 10. A broadcast network that shares trajectories and performs on-demand time synchronization is the only connection among all agents. Therefore it is less coupled than [26].

In Fig. 10, three of the modules may be confusing, here is a simple explanation. The module "Drone Detection" is for detecting other agents witnessed. The module "Frame Alignment" compensates single agent localization drift using the



(a) Empty Space (b) Narrow Gate (c) Dense Environment

(d) Trajectory and map created during the flight in dense environment.

Fig. 11: Composite images of indoor experiments, in which three drones fly to three reverse placed targets. Dashed lines indicate the relative positions in the same photographs. As shown in the images, agents maintain a safe distance from each other while avoiding obstacles. Note that the trajectories are smooth, and there is no detour.

position deviation acquired from "Drone Detection" module. The module "Agent Removal" removes pixels of other agents from depth images, since these pixels can interfere the depth-based obstacle mapping. A detailed explanation of these three modules is in our previous work [26].

B. Indoor

We present several indoor experiments at a speed limit of $3.0m/s$ for empty space and $2.0m/s$ for narrow gate and obstacle-rich scenarios, as depicted in Fig.11. The left one shows three quadrotors perform a cross flight in empty space where reciprocal collision avoidance is necessary. In the middle one, quadrotors manage to pass through a narrow gate one after another. In the right figure, we set up a cluttered environment composed of vertical and horizontal obstacles, where the narrowest gate is less than $1m$. In such a cluttered environment, three quadrotors manage to navigate across this environment sequentially and smoothly.

C. Outdoor

Outdoor experiments are presented to validate that the proposed system is capable of field operations. Snapshots of this experiment with the map built during the flight are shown in Fig. 1. Please watch the video at https://www.youtube.com/watch?v=w5GDMpjAoVQ for more information. The hardware settings are the same as [7].

VI. Conclusion and Future Work

In this work, we propose a decentralized spatial-temporal trajectory planning framework for multicopter swarms. Our framework is powered by $\mathfrak{T}_{\text{MINCO}}$ trajectory class capable of spatial-temporal deformation, and the time integral penalty functional based on constraint transcription. All these components enjoy efficiency originating from low-complexity. Extensive benchmarks are performed to show the speedup by orders of magnitude and the top-level solution quality. Real-word experiments also demonstrate the wide applicability of our framework.

It's worth noting that the formulation of reciprocal collision avoidance in Sec. III-D5 enables the planner to avoid moving obstacles as well. Since we use non-cooperative swarms, other agents and moving obstacles behave identically from the planner's view. However, we exclude this module in the current work due to the immature front-end of moving object detection and prediction, which is taken as the future work to enable fully autonomous aerial swarms in dynamical environments.

References

[1] Senthil Hariharan Arul and Dinesh Manocha. DCAD: Decentralized Collision Avoidance With Dynamics Constraints for Agile Quadrotor Swarms. *IEEE Robotics and Automation Letters*, 5(2):1191–1198, 2020.

[2] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. Generation of Collision-Free Trajectories for a Quadrocopter Fleet: A Sequential Convex Programming Approach. In *2012 IEEE/RSJ international conference on Intelligent Robots and Systems (IROS)*, pages 1917–1922. IEEE, 2012.

[3] Yufan Chen, Mark Cutler, and Jonathan P How. Decoupled Multiagent Path Planning via Incremental Sequential Convex Programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961. IEEE, 2015.

[4] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A Survey on Aerial Swarm Robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[5] Flaviu Cristian. Probabilistic Clock Synchronization. *Distributed Computing*, 3(3):146–158, 1989.

[6] Carl De Boor. *A practical guide to splines*, volume 27. Springer-verlag New York, 1978.

[7] Fei Gao, Luqi Wang, Boyu Zhou, Xin Zhou, Jie Pan, and Shaojie Shen. Teach-repeat-replan: A Complete and Robust System for Aggressive Flight in Complex Environments. *IEEE Transactions on Robotics*, 36(5): 1526–1545, 2020.

[8] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory Planning for Quadrotor Swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[9] Zuxin Liu, Baiming Chen, Hongyi Zhou, Guru Koushik, Martial Hebert, and Ding Zhao. MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments. *arXiv preprint arXiv:2007.15724*, 2020.

[10] Carlos E Luis and Angela P Schoellig. Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control. *IEEE Robotics and Automation Letters*, 4(2):375–382, 2019.

[11] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and Guido CHE de Croon. Minimal Navigation Solution for a Swarm of Tiny Flying Robots to Rxplore an Unknown Environment. *Science Robotics*, 4 (35), 2019.

[12] Daniel Mellinger and Vijay Kumar. Minimum Snap Trajectory Generation and Control for Quadrotors. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525. IEEE, May 2011.

[13] Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran. Continuous-Time Trajectory Optimization for Online UAV Replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339. IEEE, 2016.

[14] Jungwon Park, Junha Kim, Inkyu Jang, and H Jin Kim. Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 434–440. IEEE, 2020.

[15] William H Press, H William, Saul A Teukolsky, A Saul, William T Vetterling, and Brian P Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[17] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. In *Robotics Research*, pages 649–666. Springer, 2016.

[18] Jesus Tordesillas and Jonathan P How. MADER: Trajectory Planner in Multi-Agent and Dynamic Environments. *arXiv preprint arXiv:2010.11061*, 2020.

[19] Jesus Tordesillas and Jonathan P How. MINVO basis: Finding Simplexes with Minimum Volume Enclosing Polynomial Curves. *arXiv preprint arXiv:2010.10726*, 2020.

[20] Jesus Tordesillas, Brett T Lopez, and Jonathan P How. Faster: Fast and Safe Trajectory Planner for Flights in Unknown Environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1934–1940. IEEE, 2019.

[21] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-Body Collision Avoidance. In *Robotics Research*, pages 3–19. Springer Berlin Heidelberg, 2011.

[22] Jur Van Den Berg, Jamie Snape, Stephen J Guy, and Dinesh Manocha. Reciprocal Collision Avoidance With Acceleration-Velocity Obstacles. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3475–3482. IEEE, 2011.

[23] Ruben Van Parys and Goele Pipeleers. Distributed Model Predictive Formation Control with Inter-Vehicle Vollision Avoidance. In *2017 11th Asian Control Conference (ASCC)*, pages 2399–2404. IEEE, 2017.

[24] Zhepei Wang, Xin Zhou, Chao Xu, and Fei Gao. Geometrically Constrained Trajectory Optimization for Multicopters. *arXiv preprint arXiv:2103.00190*, 2021.

[25] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8776–8782. IEEE, 2020.

[26] Xin Zhou, Jiangchao Zhu, Hongyu Zhou, Chao Xu, and Fei Gao. EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments. *arXiv preprint arXiv:2011.04183*, 2020.

[27] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2021.