2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Spatiotemporal Co-Design Enabling Prioritized Multi-Agent Motion Planning

Yunshen Huang, Wenbo He, Yiannis Kantaros, Shen Zeng

***Abstract*— This paper introduces an innovative planner for prioritized multi-agent motion planning, employing a sequential integration of spatial and temporal designs. The planner initiates a smooth trajectory in space for each agent, ignoring the presence of other agents. Subsequently, by treating spatial collisions as 2D obstacles from a temporal perspective, the planner dynamically fine-tunes the trajectory-tracking speed of agents to avoid collisions, ensuring optimal time consumption for the last agent to reach the target as well. Additionally, the proposed approach systematically coordinates priority for each agent. The efficacy of the approach is validated through both simulations and comparative experiments with a recent planner.**

I. Introduction

Collision avoidance is a fundamental problem in Multi-Agent Motion Planning (MAMP), requiring each agent to navigate safely among static and dynamic entities while approaching its target. Over the past decades, this problem has found numerous real-world applications such as traffic management, warehouse automation, and autonomous agriculture. This paper specifically addresses situations where certain spaces impose highly restrictive volumes, allowing only one agent to pass at a time—an example being multiple vehicles crossing a narrow bridge. Solving such problems is recognized as notoriously difficult, due to restrictive constraints and scalability issues.

MAMP has been an active area in the robotics field for decades. Early works (see, e.g. [1], [2]) tend to formulate MAMP as a high-dimensional optimization problem by treating all agents collectively, yielding high-quality optimal solutions. However, the computational complexity exponentially grows with the number of agents, restricting the range of applications of these approaches. To bridge the gap between precision and scalability, Optimal Reciprocal Collision Avoidance (ORCA) framework [3] and its variants (see, e.g. [4], [5]) have emerged as widely adopted algorithms for MAMP. Stemmed from the concept of velocity obstacle [6], at each time frame, ORCA solves a sequence of linear programs to find an optimal velocity for each agent, ensuring collision-free during the following certain amount of period. However, these approaches assume that collisions can be resolved by relocating agents, requiring a sufficiently large space—an assumption that does not hold in our specific problem. Addressing this limitation, prioritized planning methods have been explored (see, e.g. [7], [8]), assigning unique priorities to agents and having lower-priority agents treat higher-priority ones as dynamic obstacles. However, especially in highly constrained environments with large swarms of agents, the effectiveness of these methods is sensitive to the predefined priority orderings. This issue is systematically addressed in [9] through the exploration of a priority ordering space. However, its performance hinges on the quality of the initial ordering setup. Additionally, a more recent planner, called Scalable and Safe Multi-agent Motion (S2M2) [10], tackles MAMP by first generating a piecewise path for each agent and subsequently scheduling safe motions for the swarm based on the optimal priority ordering found by [9]. Nevertheless, due to the non-smooth planned path, S2M2 relies on a robust low-level controller for tracking, involving additional design efforts.

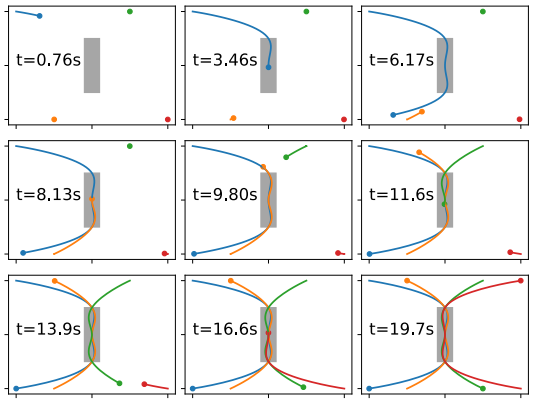


Fig. 1: Four agents, represented by different colors, navigate from their starting point to the targets by traversing a tunnel in order of priority while remaining safe movements.

Addressing these limitations, we present a novel Spatiotemporal Co-Design (Co-STD) planner that can efficiently generate time-optimal scheduling for solving MAMP within a highly-constrained environment, through an integrated spatial and temporal perspective. From a spatial viewpoint, the Co-STD first generates a smooth trajectory expressed in Bézier curves for each agent. As a type of parametric trajectory, a widely-adopted choice in motion planning [11]–[13], the Bézier curve is selected for its simple representation and inherent smoothness [13]–[15]. These properties enable fast plannings and easy track for differentially flat systems [16], [17], a large family including mobile robots [18], quadcopters [19], etc. Given two obtained trajectories, collisions between agents in physical space are projected to obstacles in a 2D temporal space. Within it, the planner then finds an time-optimal and collision-free path that can be leveraged to systematically adjust both agents' initial velocities derived from their trajectories. When following the modified velocities, agents are guaranteed to remain collision-free while traversing their given paths, spatial information of the derived trajectory, and reach the targets in minimum makespans.

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

We additionally integrate the agent priority into the path-finding procedure within the temporal space, and extend the approach to accommodate any number of agents by leveraging the transitivity of such a temporal projection. An example of the resulting safe planning for four quadcopters is demonstrated in Figure 1. Meanwhile, we also find that our idea of the temporal projection coincides with, not inspired by, the method of Path-Velocity Decomposition (PVD) [20] and Robust Multi-Robot Trajectory Tracking Strategy (RMTRACK) [21]. However, PVD is inadequate for addressing prioritized MAMP and faces challenges in scenarios with more than two agents. Similarly, RMTRACK is limited by its nonsmoothness, where the agents are constrained to either being stationary or moving at specific timestamps.

A. Contributions

The contributions of this paper are threefold:

- Decoupled MAMP: We systematically decouple MAMP into a trajectory-generation problem in the environment and a series of 2D path-finding problems in the projected temporal space.
- Smooth trajectory: Unlike other path-based methods (see, e.g. [10], [22]), we additionally construct Bézier curve-based trajectories that offer a high degree of smoothness, facilitating easy tracking by systems, and provide velocity information for later utilization.
- Improved scheduling performance: Through comparisons with S2M2 [10] on identical MAMP scenarios, our proposed Co-STD planner demonstrates shorter solver runtime and reduced makespan for the last agents to reach its target.

B. Outline

The problem is stated in Section II, and preliminaries are given in Section III. Section IV delves into the trajectory generation, followed by scheduling the tracking velocity in Section V. In addition, numerical and experimental results are demonstrated in Section VI. Finally, the paper is concluded in Section VII.

II. Problem Statement

Given a swarm of α agents $\mathcal{A} = \{\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(\alpha)}\}$, each modeled as a differentially flat system, navigating in an environment containing narrow tunnels where only one agent can pass at a time. Agents are expected to start from their individual starting points $P^{(i)}_{\text{start}}$, navigate through a sequence of tunnels in a predefined order, and finally reach their targets $P_{\text{target}^{(i)}}$. Each agent $\mathcal{A}^{(i)}$ has a unique priority, with lower values indicating higher priority. Given the requirement of prioritized traversal of tunnels, the objective is to schedule the safe motions for $\mathcal{A}$ with minimum makespan, the time needed for the last agent to reach its target.

As the exploration of the environment is beyond the interest of this paper, we assume the environmental information is given. Therefore, a sequence of waypoints can be specified regarding $P^{(i)}_{\text{start}}$, $P^{(i)}_{\text{target}}$, tunnels, and some other points of interest $w^{(i)}$. For convenience, we assume a tunnel as a cylinder taking the form of a tuple $(c_1 \in \mathbb{R}^d, c_2 \in \mathbb{R}^d, r \in \mathbb{R}_+)$, where c_1, c_2 are the centers of the end surfaces, and r denotes the radius. This expression effectively encapsulates a tunnel as two waypoints. Given the starting point $P^{(i)}_{\text{start}}$, target $P^{(i)}_{\text{target}}$, original waypoints $w^{(i)}$, and a number of h tunnels, one can collect them and form into a waypoint set

$$\{\boldsymbol{w}\} \triangleq \{P_{\text{start}}, c_1^{(1)}, c_2^{(1)}, \cdots, w^{(i)}, \cdots, c_1^{(h)}, c_2^{(h)}, P_{\text{target}}\}. \tag{1}$$

Note that the ordering of $(c_1^{(i)}, c_2^{(i)})$ determines the direction of tunnel traversing. An illustrative example of constructing $\boldsymbol{w}$ is presented in Figure 2.

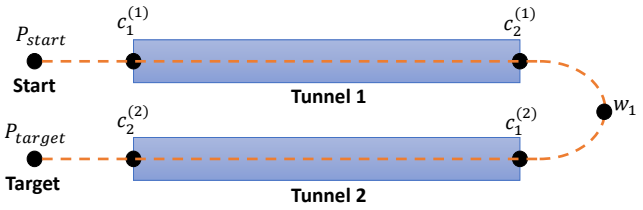


Fig. 2: By viewing a tunnel as two waypoints, a waypoints set is formed as $\{\boldsymbol{w}\} \triangleq \{P_{\text{start}}, c_1^{(1)}, c_2^{(1)}, w^{(1)}, c_1^{(2)}, c_2^{(2)}, P_{\text{target}}\}$.

III. Preliminaries

Considering the simple construction and extrinsic smoothness, we formulate the trajectory in Bézier curves.

Definition 1 (Bézier curve) *A Bézier curve of degree n denoted $\boldsymbol{B} : [0,1] \mapsto \mathbb{R}^d, d > 1$ is defined as a convex interpolation in $n+1$ control points $\mathcal{P} = \{\bar{\boldsymbol{p}}_i \in \mathbb{R}^d,\ i \in [0, ..., n]\}$, giving the expression*

$$\boldsymbol{B}(\lambda; \mathcal{P}) = \sum_{i=0}^{n} \binom{n}{i} (1-\lambda)^{(n-i)} \lambda^i \bar{\boldsymbol{p}}_i \triangleq \sum_{i=0}^{n} \alpha_i^n(\lambda) \bar{\boldsymbol{p}}_i, \tag{2}$$

where $\binom{n}{i}$ defines the binomial coefficient, and $\alpha_i^n(\lambda)$ denotes the so-called Bernstein polynomial.

Note that we can conveniently describe the set of control points $\mathcal{P}$ as an augmented vector $\boldsymbol{p} = \text{vec}(\mathcal{P}) \triangleq (\bar{\boldsymbol{p}}_0^\top, ..., \bar{\boldsymbol{p}}_n^\top)^\top \in \mathbb{R}^{d(n+1)}$. Recognizing that agent's motion is defined in time, we further define time-scaled Bézier curve.

Definition 2 (Time-scaled Bézier curve) *Given a Bézier curve $\boldsymbol{B}(\lambda; \mathcal{P})$, its time-scaled counterpart is defined by a coordinate transform $\lambda = tT^{-1}$ with $t \in [0, T]$ for $T > 0$.*

The following properties can be analytically concluded.

Property 1 *The k^{th} derivative of a Bézier curve yields another Bézier curve with an order of $n-k$, i.e. $(\mathrm{d}^k/\mathrm{d}t^k)\boldsymbol{B}(tT^{-1}; \mathcal{P}) = \boldsymbol{B}(tT^{-1}; \mathcal{P}^{(k)'})$. The control points $\mathcal{P}$ and $\mathcal{P}^{(k)'}$ are related by a linear map $\boldsymbol{p}^{(k)'} = \boldsymbol{M}_n^k \boldsymbol{p}$,*

$$\boldsymbol{M}_n^k(T) = \boldsymbol{M}_{n-k+1}^1(T) \cdots \boldsymbol{M}_{n-1}^1(T) \boldsymbol{M}_n^1(T). \tag{3}$$

$$\boldsymbol{M}_n^1(T) = \frac{n}{T} \begin{bmatrix} \boldsymbol{I}_d & -\boldsymbol{I}_d & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_d & -\boldsymbol{I}_d & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{I}_d \end{bmatrix} \in \mathbb{R}^{nd \times (n+1)d},$$

Proof: The calculation of the first derivative is fairly easy by directly writing out $(\mathrm{d}/\mathrm{d}\lambda)\alpha_n^i(\lambda)$. An iterative manner shall be followed to conduct higher-order derivatives up to k, where $k < n$. ∎

Property 2 *The squared $\mathcal{L}_2$-norm of $\sum_{k=0}^{n} \boldsymbol{B}(tT^{-1}; \mathcal{P}^{(k)'})$ is strictly convex in $\mathcal{P}$ over its domain $t \in [0, T]$, written as*

$$\begin{aligned}&\Big\| \sum_{k=0}^{N} w_k (\mathrm{d}^k/\mathrm{d}t^k) \boldsymbol{B}(tT^{-1}; \mathcal{P}) \Big\|^2_{\mathcal{L}_2([0,T])} \\ &= \int_0^T \Big\| \sum_{k=0}^{N} w_k (\mathrm{d}^k/\mathrm{d}t^k) \boldsymbol{B}(tT^{-1}; \mathcal{P}) \Big\|_2^2 \mathrm{d}t \\ &= \boldsymbol{p}^\top \Big(T \sum_{a=0}^{N} \sum_{b=0}^{N} w_a w_b (\boldsymbol{M}_n^a(T))^\top (\boldsymbol{C}_{mn} \otimes \boldsymbol{I}_d) \boldsymbol{M}_n^b(T) \Big) \boldsymbol{p}\end{aligned} \tag{4}$$

where w_k denotes the weight, $m = N - a$, $m = N - b$, $\boldsymbol{C}_{mn} \in \mathbb{R}^{(m+1)\times(n+1)}$, with

$$[\boldsymbol{C}_{mn}]_{i+1,j+1} = \frac{mn(j+i)!(m+n-i-j)!}{i!j!(m-i)!(n-j)!(m+n+1)!}.$$

Proof: This follows from considering all of the dot products resulting from (4) and the closed form of the integrals $\int_0^1 \alpha_m^i(\lambda)\alpha_n^j(\lambda)\mathrm{d}\lambda$, collected in $\boldsymbol{C}_{mn}$. In doing so, we have found a quadratic form (4) which is well defined for all $T > 0$ and zero when $\boldsymbol{p} \equiv \boldsymbol{0}$. As such the hessian of (4) is positive semi-definite, and the function is convex in $\boldsymbol{p}$ for fixed $T > 0$. ∎

IV. Spatial Trajectory Design for Single agent

Given an agent $\mathcal{A}^{(i)}$ along with its starting point, target, and a collection of tunnels within the environment, we first convert them into a waypoint set $\{\boldsymbol{w}\}$ shown in (1). Subsequently, we generate a minimum-snap trajectory represented in Bézier polynomials that connects every element from $\{\boldsymbol{w}\}$ in sequence. To ensure the connection, one could leverage the method of time gridding (see e.g. [23]), forcing specific points of the trajectory to be affixed to waypoints. However, challenges arise when dealing with more complex geometries of waypoint sets and the need for time optimality. To this end, given $m+1$ waypoints, we consider m segments $\{\mathcal{B}_i : [0, T_i] \mapsto \mathbb{R}^d\}_{i=1}^m$, where $\mathcal{B}_i(t) = \boldsymbol{B}(tT_i^{-1}; \mathcal{P}_i)$ for some $T_i > 0, d > 0$, and construct a Bézier trajectory $\mathcal{B} : [0, T] \mapsto \mathbb{R}^d$, as

$$\mathcal{B}(t) = \mathcal{B}_i(t - \tau_i) \quad \forall t \in [\tau_i, \tau_{i+1}], \quad \tau_i = \sum_{j=1}^{i-1} T_j, \tag{5}$$

where $T = \sum_{i=1}^m T_i$ represents the total time duration. To simplify notation, we let $\boldsymbol{p}_i = \mathrm{vec}(\mathcal{P}_i) \in \mathbb{R}^{d(n+1)}$ be the control points associated with the i-th curve, and define all of the control points associated with $\mathcal{B}$ in (5) as $\boldsymbol{p} = (\boldsymbol{p}_1^\top, \ldots, \boldsymbol{p}_m^\top)^\top \in \mathbb{R}^{dm(n+1)}$ and collect time intervals in $\boldsymbol{t} = (T_1, \ldots, T_m) \in \mathbb{R}_+^m$.

A. Minimum-Snap Trajectory

Having done so, the trajectory planning can be formulated as an optimization problem concerning $\boldsymbol{p}$.

1) Objective Function: We attempt to minimize

$$\sum_{k=0}^{n} c_k \|(\mathrm{d}^k/\mathrm{d}t^k)\mathcal{B}(t)\|^2_{\mathcal{L}_2([0,T])} = \sum_{j=1}^{m} \boldsymbol{p}_j^\top \boldsymbol{Q}_S(T_j) \boldsymbol{p}_j, \tag{6}$$

where c_k specifies the weight for the k^{th} derivative. In the minimum-snap context, we set weights to zero except for $c_4 > 0$. According to **Property 2**, such an objective is notably a strictly convex function in $\boldsymbol{p}$ with a fixed time duration at each segment.

2) Waypoint Constraints: To ensure all waypoints are sequentially connected, we consider the endpoints for each segment to satisfy the following.

$$\begin{aligned}\mathcal{B}_j(0) &= \{\boldsymbol{w}\}_j, \quad j \in \{1, \cdots, m\}, \\ \mathcal{B}_m(T) &= \{\boldsymbol{w}\}_{m+1} = P_{\text{target}},\end{aligned}$$

which can be equivalently expressed as equality constraints

$$\begin{aligned}\begin{bmatrix}\boldsymbol{I}_d & \boldsymbol{0}\end{bmatrix} \boldsymbol{p}_j &= \{\boldsymbol{w}\}_j, \quad i \in \{1, \cdots, m\}, \\ \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}_d\end{bmatrix} \boldsymbol{p}_m &= \{\boldsymbol{w}\}_{m+1} = P_{\text{target}}.\end{aligned} \tag{7}$$

3) Continuity Constraints: To ensure the dynamic feasibility of the planned trajectory, we must guarantee the smooth conjunction of adjacent segments. To achieve this, we express the condition

$$(\mathrm{d}^k/\mathrm{d}\lambda^k)\mathcal{B}_j(T_j) = (\mathrm{d}^k/\mathrm{d}\lambda^k)\mathcal{B}_{j+1}(0), \quad j = 1, \cdots, m-1,$$

as an equality constraint

$$\begin{bmatrix}\boldsymbol{M}_n^k(T_j) & -\boldsymbol{M}_n^k(T_{j+1})\end{bmatrix} \begin{bmatrix}\boldsymbol{p}_j \\ \boldsymbol{p}_{j+1}\end{bmatrix} = \boldsymbol{0}. \tag{8}$$

where the user specified k guarantees the resulting trajectory to be k-times continuously differentiable i.e. $\mathcal{B} \in \mathcal{C}^K([0, T], \mathbb{R}^d)$. From a physical viewpoint, this requirement implies continuity in planned velocity, acceleration, and jerk, with $k = 3$.

B. Optimal-Time Allocation

In scenarios where specifying a sequence of arrival times at each waypoint is crucial, the aforementioned optimization can be directly solved given a time interval for each segment. However, when arrival time is non-critical, we allow these segment time intervals to vary, achieving a time-optimal allocation. Specifically, we additionally consider time intervals $\boldsymbol{t} = (T_1, \cdots, T_m)$ into the optimization (6), giving the form

$$J(\boldsymbol{p}, \boldsymbol{t}) = \sum_{j=1}^{m} \boldsymbol{p}_j^\top \boldsymbol{Q}_S(T_j) \boldsymbol{p}_j + k_t \sum_{j=1}^{m} T_j, \tag{9}$$

where k_t defines the time consumption weight. Introducing $\boldsymbol{t}$ not only achieves minimum time consumption but also allows the trajectory to automatically adjust the time spent in each segment based on the geometry relationship of waypoints. For instance, the resulting motion near sharp corners turns to be slower. See [12] for more discussions and demonstrations of a similar treatment on time allocation. To ensure valid time intervals, a minimal duration constraint is enforced for each segment:

$$\boldsymbol{t} \geq \underline{\boldsymbol{t}} > \boldsymbol{0}, \quad \underline{\boldsymbol{t}} \in \mathbb{R}_+^m. \tag{10}$$

In summary, given the cost in (9), and the constraints in (7), (8), and (10), generating a minimum-snap trajectory with optimal-time allocation for a single agent involves solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & J(\boldsymbol{p},\ \boldsymbol{t}) \\ \text{subject to} \quad & \boldsymbol{C}(\boldsymbol{t})\boldsymbol{p} = \boldsymbol{0}, \\ & \boldsymbol{t} - \underline{\boldsymbol{t}} \geq \boldsymbol{0}. \end{aligned} \tag{11}$$

It is observed that the objective function (9) is bi-convex in $\boldsymbol{p}$ and $\boldsymbol{t}$ that can be resolved using a block coordinate descent approach. Given an initial point $(\boldsymbol{p}^{(0)}, \boldsymbol{t}^{(0)})$, we solve (11) with fixed $\boldsymbol{t} = \boldsymbol{t}^{(0)}$ as a quadratic program. We then fix $\boldsymbol{p} = \boldsymbol{p}^{(1)}$ and minimize (11) using a Newton method, leveraging the fact that we can obtain the gradient and Hessian of the cost functions analytically easily.

Remark: If the planned trajectory does not completely reside inside the tunnel, we simply treat the geometric centers of those tunnels as new waypoints, and then re-solve (11). Compared with sticking the trajectory with some sampled points inside the tunnel, in practice, our strategy takes better advantage of the time allocation without adding significant computation complexity.

V. Temporal Scheduling for Multiple agents

This section introduces an innovative methodology for collision-free MAMP. The proposed approach, viewing trajectories as functions of time, systematically adjusts the speed of agents as they track their trajectories. Collision avoidance is guaranteed, even when the trajectories overlap in space. Furthermore, the methodology preserves agents' priority, allowing those with higher priority to navigate through tunnels first.

A. View Collision from Temporal Perspective

To start, we consider a simple swarm comprising two agents $A^{(1)}$ and $A^{(2)}$, each with designated starting point, pass-through tunnels, target, and unique priority. For each agent i, where $i \in \{1, 2\}$ the followings are conducted:

1) Generate optimal trajectory $\mathcal{B}^{(i)}$ by solving (11).
2) Sample $\mathcal{B}^{(i)}$ with a fixed step dt, resulting in a discretized trajectory $\mathcal{S}^{(i)}$ with the indexing rule

$$\mathcal{S}_r^{(i)} = \mathcal{B}^{(i)}(r \cdot dt), \quad r = \{0, \cdots, N^{(i)} = T^{(i)}/dt\}.$$

We then introduce the following concepts to aid in visualizing collisions from the temporal perspective.

1) Temporal Collision Map (TCM): Formed upon $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$, the Temporal Collision Map (TCM) takes the form of a matrix, denoted as $\boldsymbol{C}$, with $N^{(2)}$ rows and $N^{(1)}$ columns. Each element, for example, $\boldsymbol{C}_{r,c}$, represents a moment when $\mathcal{A}^{(1)}$ is at $\mathcal{S}_c^{(1)}$, and $\mathcal{A}^{(2)}$ is at $\mathcal{S}_r^{(2)}$. This value serves as an indicator of whether the two agents collide at that specific moment along their trajectories. Formally, we define:

$$\boldsymbol{C}_{r,c} = \begin{cases} 1 & \text{if } \|\mathcal{S}_c^{(1)} - \mathcal{S}_r^{(2)}\|_2 \leq c_{\text{th}} \\ 0 & \text{else} \end{cases}, \tag{12}$$
$$\forall r \in \{1, \cdots, N^{(2)}\},\ \forall c \in \{1, \cdots, N^{(1)}\},$$

where c_{th} denotes the minimum collision threshold.

2) Trajectory Tracer (TT): Treating TCM as a 2D environment, we can find a path starting from $\boldsymbol{C}_{0,0}$ and ending at $\boldsymbol{C}_{N^{(1)},N^{(2)}}$ in TCM. This path, given the name of the Trajectory Tracer (TT), contains the spatiotemporal information regarding two agents' locations along their trajectories at each moment in time. Moreover, TT can be defined as vector $\boldsymbol{f}$ containing the corresponding indices of $\boldsymbol{C}$. From this perspective, TT reveals how these agents track their trajectories over time. Given that nonzero-valued elements in TCM indicate collisions, we introduce the following:

Definition 3 *A TT is safe, if and only if* $\boldsymbol{C}_{f_i} = 0,\ \forall f_i \in \boldsymbol{f}$.

Proposition 1 *If two agents follow the trajectories in a way specified by a safe TT, they are guaranteed to not collide.*

3) Clocks: It is worth noting that if two agents follow a given TT, their trajectory-tracking speeds are adjusted simultaneously. To better elaborate such temporal adaption, we introduce two distinct clocks: wall and agent clocks.

- **Wall clock** t_{w} elapses uniformly in time, indicating the time of the world frame.
- **Agent clock** $t_{\text{a}}^{(i)}$ represents the agent's eternal-time progression of agent i, defined as a monotonically increasing mapping $t_{\text{w}} \mapsto t_{\text{a}}^{(i)}(t_{\text{w}})$,

Therefore, the variation in the trajectory-tracking speed can be interpreted as a result of the agent's eternal clock progressing differently from the wall clock. For instance, given an agent clock elapses faster than the wall clock, i.e. $t_{\text{a}}(t_{\text{w}}) > t_{\text{w}}$, an observer in the world frame will perceive the agent tracking the trajectory at a higher speed, as it takes less time to reach the target. Notably, allowing t_{a} to decrease implies that the agent can track the trajectory in reversed time, which is beyond the scope of this paper.

To quantify the effect of a given TT being followed by two agents, we introduce the following

Definition 4 *Denoting* $t_a^{(i)'} \triangleq \frac{d}{dt_w} t_a^{(i)}(t_w)|_t$ *and the slop of TT at time* t *is given as* $\Delta \boldsymbol{f}|_t = t_a^{(2)'}/t_a^{(1)'}$.

Proof: With a slightly loosing representation, we have

$$\Delta \boldsymbol{f} = \frac{\Delta t_{\text{a}}^{(2)}}{\Delta t_{\text{a}}^{(1)}} = \frac{\Delta t_{\text{a}}^{(2)}/\Delta T_{\text{w}}}{\Delta t_{\text{a}}^{(1)}/\Delta T_{\text{w}}} = \frac{t_{\text{a}}^{(2)'}}{t_{\text{a}}^{(1)'}}.$$

■

The value of $\Delta \boldsymbol{f}$ can be obtained by taking the finite difference of TT's elements. Essentially, a regulation is enforced to the tracking speed ratio between two agents by following a given TT, leading to the following proposition.

Proposition 2 *If both agents track their trajectories at the original speed, then* $t_a^{(1)}(t_w) = t_a^{(2)}(t_w) = t_w \Rightarrow \Delta \boldsymbol{f} = 1$. *Otherwise, if* $\Delta \boldsymbol{f} < 1$, $\mathcal{A}^{(1)}$ *tracks faster than* $\mathcal{A}^{(2)}$ *does, and vice versa for* $\Delta \boldsymbol{f} > 1$.

An example of TCM and four TTs are illustrated in Figure 3. If two agents follow the red TT, with the slop as $\Delta \boldsymbol{f} = 1$, they will track their trajectories at the original (more rigorously, same) speed. However, because the red TT

traverses through the collision area, thus it is unsafe, and two agents will collide during the movement.

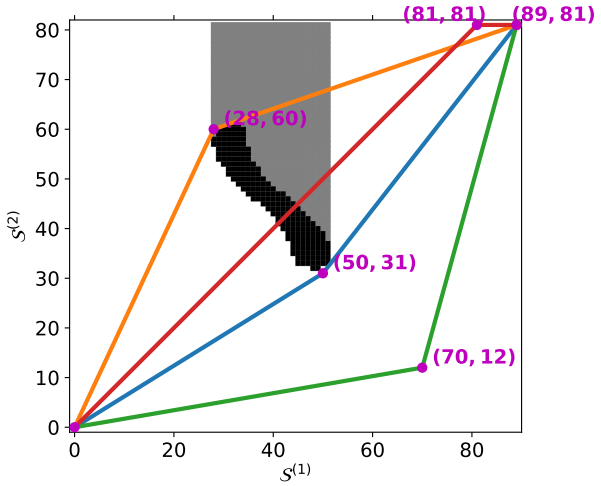


Fig. 3: TCM is constructed by piecewisely examining $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ in the way of (12). The elements indicating no collision are shown in white pixels, otherwise in black, forming an obstacle as a whole. To preserve the priority, an artificial obstacle is created as shown in the gray area. The vertices of the polyline-segment TTs are shown in purple along with their position values.

B. Avoid Collision from Temporal Perspective

Having viewed the spatial collision from a temporal perspective, it is clear that finding a safe TT is key to preventing collisions. Definition 3 implies that searching a safe TT in TCM can be reformulated to a 2D path-finding problem by viewing the collision areas as obstacles. In this section, we introduce a method on finding a TT that leads to an optimal makespan with the fulfillment of the priority ordering.

1) Time Optimality: As the slope of TT plays a crucial role in manipulating the trajectory tracking speeds of agents, we choose to form TT as a polyline segment in TCM. The motivation lies in the expression of the associated acceleration along the motion

$$\begin{aligned}\frac{d^2}{dt_{\mathrm{w}}^2}\mathcal{B}(t_{\mathrm{a}}) &= \frac{d}{dt_{\mathrm{w}}}\left(\frac{d}{dt_{\mathrm{a}}}\mathcal{B}(t_{\mathrm{a}})\frac{d}{dt_{\mathrm{w}}}t_{\mathrm{a}}\right)\\ &= \frac{d^2}{dt_{\mathrm{a}}^2}\mathcal{B}(t_{\mathrm{a}})\left(\frac{d}{dt_{\mathrm{w}}}t_{\mathrm{a}}\right)^2 + \frac{d}{dt_{\mathrm{a}}}\mathcal{B}(t_{\mathrm{a}})\frac{d^2}{dt_{\mathrm{w}}^2}t_{\mathrm{a}}.\end{aligned}$$

By representing TT as piecewise linear, we specifically set $t_{\mathrm{a}}^{(k)'} = 0,\ k \geq 2$ for any k^{th} derivative of the motion, which significantly simplifies the motion analysis.

Taking time optimality into account, both agents tend to operate at full speed whenever possible. Consequently, the time taken for two agents to traverse a specific segment of the TT is directly proportional to the Chebyshev distance between the endpoints of that segment. For example, considering the green TT from Figure 3, the time taken to traverse the first segment is $\max(70, 12) = 70$, given that $\mathcal{A}^{(1)}$ is operating at full speed $\overline{v} = 1$ at certain points along the trajectory. Similarly, the time taken to traverse the second segment is found as $\max(19, 69) = 69$, while $\mathcal{A}^{(2)}$ is operating at full speed. As a result, following the green TT takes $70+69 = 139$ units of the makespan for two agents to reach their targets. However, only 100 units of the makespan are needed by following the blue TT. Consequently, the time-optimal TT can be defined as a safe TT with the minimum length which is measured by the sum of Chebyshev distances of each segment's endpoints.

Given an optimal TT, the ratio of the tracking speed is determined by $\Delta \boldsymbol{f} = t_{\mathrm{a}}^{(2)'}/t_{\mathrm{a}}^{(1)'}$ from Definition 4. Together with the expression of the bounded agent's velocity

$$\frac{d}{dt_{\mathrm{w}}}\mathcal{B}(t_{\mathrm{a}})^{(i)} = \mathcal{B}'^{(i)}(t_{\mathrm{a}})t_{\mathrm{a}}^{'(i)} \leq \overline{v} \quad i \in \{1, 2\},$$

where $\mathcal{B}'^{(1)}(t_{\mathrm{a}})$ and $\mathcal{B}'^{(2)}(t_{\mathrm{a}})$ can be readily obtained from (3), we can systematically obtain respective agent tracking speed by letting one agent track at full speed.

2) Priority Preserving: Examining Figure 3, we note that both the orange and blue TTs are optimal and safe, yet their distinct routes around the obstacle indicate how priorities are assigned to each agent. For instance, the blue TT, circumventing the obstacle from the lower space, results in $\Delta f < 1$ in the first segment. During this interval, as per Proposition 2, $\mathcal{A}^{(1)}$ tracks the trajectory faster than $\mathcal{A}^{(2)}$, i.e., $t_{\mathrm{a}}^{(1)}(t_{\mathrm{w}}) > t_{\mathrm{a}}^{(2)}(t_{\mathrm{w}}),\ \forall t_{\mathrm{w}} > 0$, leading $\mathcal{A}^{(1)}$ to complete the trajectory tracking earlier and thus traverse the tunnel ahead of $\mathcal{A}^{(2)}$. This implies a higher priority assigned to $\mathcal{A}^{(1)}$. Conversely, following the orange TT assigns higher priority to $\mathcal{A}^{(2)}$, given that the associated $\Delta f > 1$ during the first segment.

Generally speaking, as a result, the agent's higher priority can be preserved by planning TT leaning towards its associated axis. For instance, if $\mathcal{A}^{(1)}$ is assigned with higher priority, we simply treat all areas above and including the obstacles as artificial obstacles, i.e.

$$\{\boldsymbol{C}_{r,c} = 1 \mid r \geq r_{\mathrm{ob}}, c = c_{\mathrm{ob}}\}, \tag{13}$$

where $(c_{\mathrm{ob}}, r_{\mathrm{ob}})$ represents the position of the original obstacles. Having done so, as the upper space is blocked, the safe TT can only exist in the lower one, thus preserving the higher priority of $\mathcal{A}^{(1)}$.

It should be noted that, to achieve the minimum length, the vertices of the optimal TT have to be on the edge of the artificial obstacles. Therefore, the optimal TT can be found by equivalently identifying these vertices. Specifically, after collecting the boundary points of the obstacles and both $\boldsymbol{C}_{0,0}$, $\boldsymbol{C}_{N^{(1)},N^{(2)}}$, we employ the Graham scan [24] to efficiently compute the smallest convex hull of this points set. The points forming the obtained convex hull are the vertices constructing the optimal TT.

C. Multiple Agents

In this part, we introduce the transitivity of TT which can be exploited to address MAMP with more than two agents. Without loss of generality, we assume the priority for each agent equals its index, i.e. $\mathcal{A}^{(1)}$ receives the highest

priority, while $\mathcal{A}^{(\alpha)}$ owns the lowest. After discretizing the optimal trajectory obtained by solving optimization (11) for each agent, we then determine the tracking speed ratio between $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ by finding the optimal safe TT$^{(1)}$ from TCM$^{(1)}$ (12), as shown in the steps above. Proceeding to $\mathcal{A}^{(3)}$, to visualize potential collisions among three agents in the temporal space, we construct a Generalized TCM$^{(2)}$ (GTCM$^{(2)}$) as

$$\boldsymbol{C}_{r,c} = \begin{cases} 1 & \text{if } \|\mathcal{S}_c^{(1)} - \mathcal{S}_r^{(3)}\|_2 \leq c_{\text{th}} \text{ or } \|\mathcal{S}_c^{(2)} - \mathcal{S}_r^{(3)}\|_2 \leq c_{\text{th}} \\ 0 & \text{else} \end{cases},$$
$$\forall r \in \{1, \cdots, N^{(3)}\},\ \forall c \in \{1, \cdots, \max(N^{(1)}, N^{(2)})\}, \tag{14}$$

where $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ represent the temporally twisted trajectories based on TT$^{(1)}$. Consequently, $t_{\text{a}}^{(3)}$ can be derived based on the optimal TT$^{(2)}$ planned in GTCM$^{(2)}$ (14).

Following this procedure in a priority-descending order, we can derive each agent's clock based on all previous agents, thus demonstrating the transitivity of TT. Notably, as all previous agents receive higher priority than the newly added one, similar to the consideration in (13), artificial obstacles need to be placed to block the upper space for all GTCMs. Finally, by setting the fastest trajectory tracking speed equal to the agent's velocity upper bound, we determine the exact tracking speed for the entire swarm, ensuring safe and efficient navigation through the environment.

Remark: Given a GTCM, the existence of an optimal safe TT planned within its lower space ensures a solution for agents to safely track their trajectories in minimum makespan. Leveraging the transitivity introduced above, this optimal solution extends to the entire swarm. The worst-case scenario, also trivial, involves every agent remaining stationary until all higher-priority agents reach their targets. Only one simple assumption is made here: no agent remains stationary at a point that also lies on another agent's planned path throughout the entire traversal.

We summarize the safe scheduling from the temporal perspective for multiple agents in Algorithm 1.

Algorithm 1 Temporal Scheduling for Multiple Agents

```
Input: α agents with individual sequence of waypoints
Output: α trajectories can be safely followed by agents
for each agent do
    Obtain optimal trajectory by solving (9)
end for
Sort agents in a priority descending order
for each sorted agent do
    Construct generalized TCM
    Search for time-optimal TT by using Graham scan
end for
Determine trajectory tracking speed for all agents
```

VI. Experiments Results

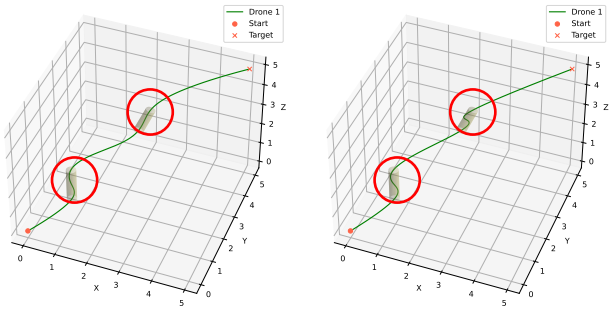
To showcase the effectiveness of the proposed method, we implement the proposed Co-STD to address MAMP in different scenarios. The parameters shared among experiments are listed in Table I. All simulations were run on a desktop equipped with a 3.60 GHz Intel Core i7-9700K CPU and 32GB RAM.

Notation	Explanation	Value
m	Order of the Bèzier curves	7
k_t	Time consumption weight	70
$\underline{\boldsymbol{t}}_i$	Element of time threshold	0.5 s
r	Radius of all tunnels i	0.2 m
c_{th}	Collision threshold	0.2 m
dt	Sampling time of trajectory	0.1 s
$\overline{v}$	Upper bounded velocity	1 m/s

TABLE I: Parameter setups

A. Single Agent

We begin by utilizing the planner to generate a minimum-snap and time-optimal trajectory for a single agent. In this scenario, the agent must navigate from the starting point $(0, 0, 0)$ to the target $(5, 5, 5)$ while sequentially traversing two tunnels positioned between $(1, 1, 1)$, $(1, 1, 1.5)$ and $(2.5, 2.5, 3.5)$, $(2.5, 3.5, 3.5)$, respectively. The trajectory, based on Bézier curves without time allocation, is generated in 0.09 s by solving a quadratic problem. With the addition of time optimality considerations, the planning of the trajectory using the coordinate descent method takes 0.39 s. A comparison of the resulting trajectories is illustrated in Figure 4.

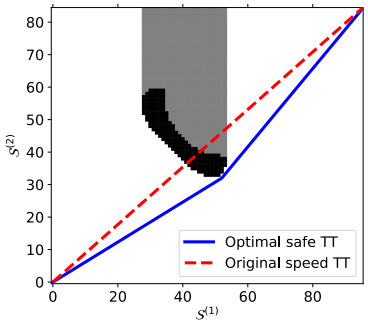


(a) Considering optimal time allocation, $T_{\text{total}} = 5.18$ s. (b) Without considering optimal time allocation, $T_{\text{total}} = 6.00$ s.

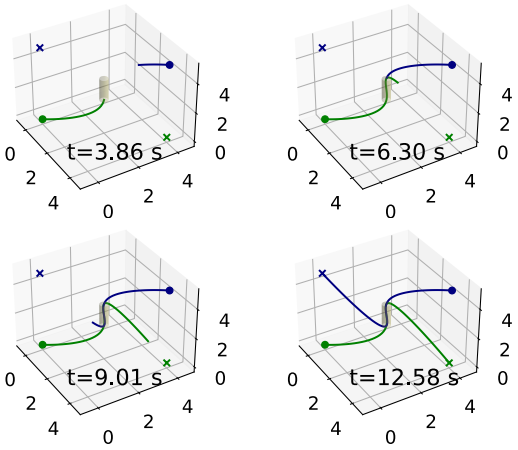
Fig. 4: Two optimal trajectories are planned. In addition to reducing time consumption, time-optimal allocation yields a more naturally curved trajectory indicated in the red circles.

B. Two Agents

We then investigate a scenario where two agents must pass through a tunnel from opposite directions before reaching their targets, with $\mathcal{A}^{(1)}$ being assigned higher priority. For both agents, the starting points are set to $(0, 0, 0)$ and $(0, 5, 0)$, while the targets are set to $(0, 5, 5)$ and $(0, 0, 5)$, respectively. The tunnel is positioned between the points $(2, 2, 2)$ and $(2, 2, 3.5)$. Initially, the planner generates a time-optimal trajectory for each agent, which is then adjusted to ensure safe motions by modifying their trajectory tracking speeds. The results are demonstrated in Figure 5. Additionally, we explore another case with identical setups, except that the agents traverse the tunnel from the same direction, while $\mathcal{A}^{(2)}$ receives higher priority. The results for this scenario are shown in Figure 6.

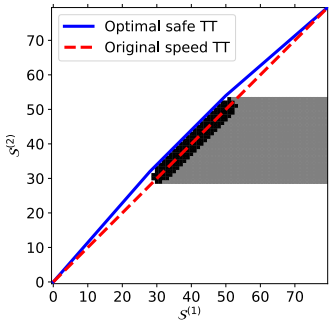


(a) Constructed TCM and TTs

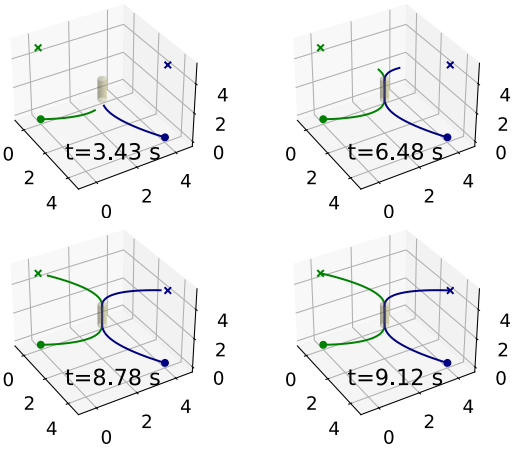


(b) Snapshots of the motions

Fig. 5: (**a**) TCM is constructed where two agents traverse the tunnel from opposite directions. The red-dashed TT indicates that two agents will collide if they operate at the original velocity. As $\mathcal{A}^{(1)}$ has a higher priority, an artificial obstacle (gray area) is created, enforcing the optimal safe TT (blue segments) obtained from the lower space of TCM. (**b**) By following the blue TT, $\mathcal{A}^{(1)}$ (green) passes through the tunnel ahead of $\mathcal{A}^{(2)}$ (blue), and they reach the targets in $12.58\,\mathrm{s}$.



(a) Constructed TCM and TTs



(b) Snapshots of the motions

Fig. 6: (**a**) TCM is constructed where two agents traverse a tunnel from the same direction. An artificial obstacle is created to preserve the higher priority of $\mathcal{A}^{(2)}$. (**b**) By following the blue TT, $\mathcal{A}^{(2)}$ (blue) passes through the tunnel ahead of $\mathcal{A}^{(1)}$ (green), and they reach the targets in $9.12\,\mathrm{s}$.

C. Multiple Agents

We further emphasize the efficiency of Co-STD by comparing it with a recent S2M2 planner [10] using one of its demonstrated testbeds, namely the Wall map introduced by [25]. In this environment, a swarm of agents are initially positioned beside the wall, and they are required to pass through three windows on the wall to reach their target located on the opposite side. Windows are basically tunnels within this paper's scope, rendering the Wall map an equitable benchmark for assessing both methods. Particularly, we are interested in two metrics

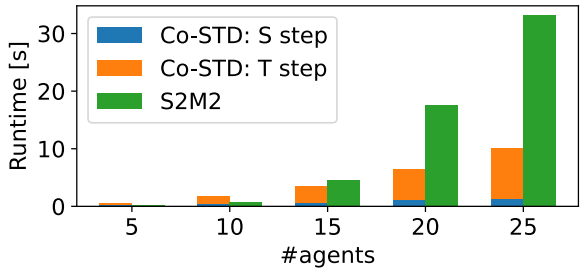
- Runtime: the time needed to fully solve MAMP, indicating the algorithm's time complexity
- Makespan: the time taken for the last agent to reach its target, indicating the time optimality of the result

Except for the modification of the thickness of the wall to be $1\,\mathrm{m}$, the other parameter configurations remain the same for S2M2. Through a series of experiments involving varying numbers of agents, the outcomes, as illustrated in Figure 7, underscore the superior efficiency of Co-STD, in terms of both algorithmic time complexity and time optimality consideration. Here, we highlight a few comparison results. Leveraging the transitivity of TT, Co-STD is capable of scheduling the entire swarm through only one iteration, whereas S2M2 must check collisions between each agent and the remaining ones. Consequently, Co-STD demonstrates better scalability with a larger number of agents. Moreover, the obtained minimum-length TT allows for a shorter makespan than the solution yielded by S2M2.

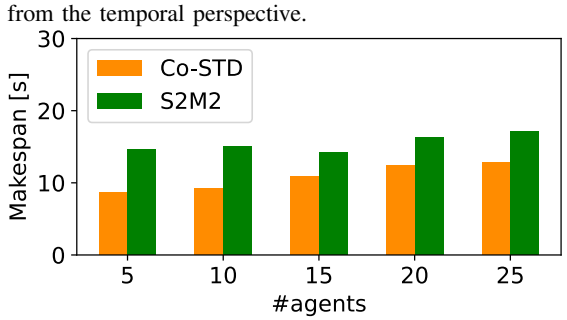
We eventually leverage the scenario with 25 agents to showcase the safe scheduling provided by Co-STD. This is demonstrated by observing that the minimum distance among them remains above the collision threshold during motion, as depicted in Figure 8.

VII. Conclusion

In this paper, we have introduced an innovative algorithm on planning time-optimal and collision-free trajectories for multi-agent systems from a joint spatial and temporal perspective. To solve this challenging prioritized multi-agent



(a) Runtimes for Wall. Co-STD's runtime is divided for into two components: **S step:** trajectory generation for all agents in physical space; **T step:** time-optimal and safe scheduling from the temporal perspective.



(b) Makespan for wall

Fig. 7: Compared performance with S2M2.

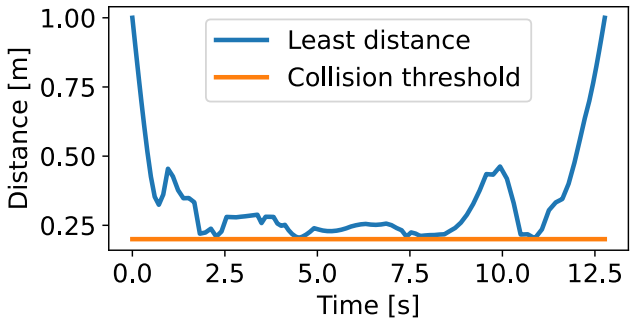


Fig. 8: Least distance among 25 agents during the motion, where no collision is found.

motion planning problem, our approach initially generates a smooth trajectory for each agent. Consequently, by projecting spatial collisions into temporal obstacles, it reformulates the scheduling task into a sequence of 2D path-finding subproblems which can be efficiently addressed using Graham scan. With the obtained paths, the trajectory-tracking speeds of all agents are adjusted in descending order of priority, ensuring collision avoidance and optimal makespan. We demonstrate the superiority of our planner through comparison with a recent approach, highlighting its advantages in terms of the algorithm's time complexity and resulting agent makespan.

References

[1] P. vSvestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[2] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

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

[4] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium*. Springer, 2013, pp. 203–216.

[5] 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, 2017.

[6] 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.

[7] M. Cáp, P. Novák, M. Selecký, J. Faigl, and J. Vokffnek, "Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3822–3829.

[8] J. Li, M. Ran, and L. Xie, "Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 405–412, 2020.

[9] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.

[10] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 237–11 245.

[11] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

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

[13] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robotics*, vol. 38, no. 2, pp. 922–938, 2021.

[14] J. Park and H. J. Kim, "Fast trajectory planning for multiple quadrotors using relative safe flight corridor," in *Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2019, pp. 596–603.

[15] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *arXiv preprint arXiv:2305.01072*, 2023.

[16] H. Sira-Ramirez and S. K. Agrawal, *Differentially flat systems*. Crc Press, 2018.

[17] L. Greco, H. Mounier, and M. Bekcheva, "An approximate characterisation of the set of feasible trajectories for constrained flat systems," *Automatica*, vol. 144, p. 110484, 2022.

[18] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *ASME Int. Mech. Eng. Congress and Exposition*, 1995.

[19] M. Greiff, A. Vinod, S. Nabi, and S. Di Cairano, "Quadrotor motion planning in stochastic wind fields," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 4619–4625.

[20] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The international journal of robotics research*, vol. 5, no. 3, pp. 72–89, 1986.

[21] M. vCáp, J. Gregoire, and E. Frazzoli, "Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5113–5118.

[22] T. Siméon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE transactions on robotics and automation*, vol. 18, no. 1, pp. 42–49, 2002.

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

[24] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Info. Proc. Lett.*, vol. 1, pp. 132–133, 1972.

[25] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.