MOCHA: Motion Camouflage-Based Homotopy-Aware Trajectory Planning

Jianqing Li, Hechao Zhang, and Zhaohui Song

***Abstract*—Trajectory planning for multi-agent systems in dynamic and complex environments is a significant challenge. Planning speed is often constrained by the high-dimensional and large-scale nature of the optimization problem, while non-convex optimization is highly susceptible to local minima, leading to reduced motion efficiency and safety. This paper introduces a novel trajectory planning framework, MOCHA (Motion Camouflage-Based Homotopy-Aware Trajectory Planning). Inspired by the motion camouflage phenomenon in nature, MOCHA uses a novel reparameterization method that transforms high-dimensional intermediate waypoints into a sequence of one-dimensional scalar parameters, thereby significantly reducing the decision dimensionality. The framework also integrates a multi-homotopy front-end path search module to suppress local minima, utilizing H-signature to distinguish topologically different path classes. For dynamic environments, a local replanning module based on 3D spatiotemporal projections is further proposed. In both simulation and real-world experiments, we demonstrate that this method substantially increases computation speed while ensuring the agents' motion efficiency and safety.**

***Index Terms*—Multi-agent systems, trajectory planning, homotopy awareness, motion camouflage, nonconvex optimization**

I. Introduction

ACHIEVING safe and flexible trajectory planning in complex environments remains a central challenge for aerial and ground robots, requiring planners to satisfy kinodynamic feasibility, collision avoidance, and smoothness under onboard real-time constraints [1], [2]. The difficulty escalates in multi-agent systems due to coupling between agents, uncertainty propagation, and communication limitations [3], [4]. These systems must generate dynamically feasible trajectories and update them frequently to accommodate changes in the environment or the behavior of neighboring agents. Optimization-based planners have shown strong performance by leveraging differential flatness and polynomial parameterizations to encode dynamics and smoothness [5], [6], [7]. However, the dimensionality of decision variables grows rapidly with the number of trajectory segments and agents. This amplifies the nonconvexity of the problem and causes computation latency to scale poorly, threatening responsiveness when replanning is required at high frequency [8], [9].

Jianqing Li is with the Space Information Research Institute and Zhejiang Key Laboratory of Space Information Sensing and Transmission, Hangzhou Dianzi University, Hangzhou, 310018, China.

Hechao Zhang was with the Department of Communication Engineering, Hangzhou Dianzi University, Hangzhou, Zhejiang 310018, China. e-mail: a0823bbdd@163.com

Zhaohui Song is with the Space Information Research Institute and Zhejiang Key Laboratory of Space Information Sensing and Transmission, Hangzhou Dianzi University, Hangzhou, 310018, China.

Beyond scalability, nonconvex optimization is highly sensitive to initialization and often converges to undesired local minima when multiple homotopy classes exist [10], [11]. Recent work combines global structure with local optimization to alleviate this issue by enumerating distinct homotopy routes [12], or constructing safe corridors from perception or search [1], [13]. Yet, accurate homotopy reasoning in space–time remains costly, and naive multi-start strategies struggle with real-time operation.

To address these challenges, we propose MOCHA, a framework that introduces a geometry-inspired reparameterization to reduce trajectory dimensionality and integrates homotopy-aware multi-candidate optimization for robustness in complex, dynamic scenes. The design naturally extends to distributed coordination, where each agent asynchronously shares minimal trajectory descriptors [3], [4], preserving scalability without sacrificing safety.

The contributions of this article are as follows:

1) A new reparameterization technique, inspired by motion camouflage, is presented. It converts the complex waypoint optimization problem into the task of optimizing a sequence of scalars for each trajectory segment. This approach maintains geometric interpretability while facilitating efficient gradient flow through the simplified representation.

2) We introduce an integrated, homotopy-aware planning framework that can manage both dynamic obstacles and the coordination of multiple agents. It combines a lightweight frontend planner with diverse topological options and an optimization backend that maintains structural integrity, utilizing a spatiotemporal homotopy approximation. This system naturally extends to distributed multi-agent systems through an asynchronous trajectory commitment process.

3) We develop an efficient local replanning module that perturbs only a short sliding time window of the current plan to avoid predicted dynamic obstacles and then smoothly rejoin the global trajectory at an anchor point. This targeted modification significantly cuts solve time and latency while preserving feasibility and consistency with the global plan. A simple check ensures that a full global replan is only triggered if the conflict extends beyond this window or feasibility is severely violated.

II. Related Work

A. Trajectory Planning

Two predominant paradigms address kinodynamically feasible trajectory planning: sampling-based search and trajectory optimization. Sampling-based planners, such as FMT* and

BIT* [14], [15], [16], exploit random geometric structures to scale exploration in cluttered environments. Geometric sampling-based planners (e.g., RRT* [17]) typically produce a geometry-only path that is subsequently time-parameterized to satisfy kinodynamic limits. By contrast, SST/SST* [18] and kinodynamic RRT* variants [19] reason directly in the state space and therefore do not require a separate retiming step; however, under high-frequency replanning and strong actuation constraints they can still suffer from temporal consistency and responsiveness issues.

Optimization-based approaches instead encode smoothness, dynamic limits, and obstacle clearance directly in the cost formulation. Representative methods such as CHOMP/STOMP [5], [6] and frameworks like TrajOpt [7] and GPMP2 [8] exploit differentiability and sparse structure for efficient gradient-based refinement. Convex relaxation through safe corridors or polytopic approximations (e.g., IRIS, Bézier convex hull) [20], [1] further improves safety guarantees, and GCS-based relaxations [21] yield tight convex relaxations with global optimality certificates under idealized assumptions. For aerial robots, approaches such as time-optimal retiming (e.g., TOPPRA [22]), robustness-oriented trajectory optimization in clutter [9], MINCO/GCOPTER, and ESDF-free optimization have achieved fast onboard performance.

However, these optimization-based planners remain highly sensitive to initialization and the scale of the decision space. Their efficiency relies heavily on the quality of the initial path provided by a front-end module, and waypoint-dense parameterizations aggravate nonconvexity, resulting in dimensionality issues, degraded gradient flow, and a high risk of local minima. Consequently, supporting global-scale planning and real-time replanning simultaneously remains challenging, motivating the need for stronger topological guidance and lower-dimensional parameterizations that stabilize convergence while reducing computational cost.

B. Homotopy-Aware Planning and Local Minima

When obstacles induce multiple feasible homotopy classes, gradient-based solvers can easily converge to locally valid but globally suboptimal trajectories. To overcome this, early studies introduced homology and H-signature representations [10], [11] to classify trajectories into distinct topological classes and to diversify optimization seeds. Subsequent systems extend this idea by optimizing multiple candidates in parallel across different homotopy classes [12], effectively improving robustness to local minima. Skeleton-based reasoning and corridor decomposition methods also reduce the dimensionality of topological representations, improving computational tractability.

Nevertheless, accurate homotopy classification in high-dimensional or time-varying spaces is computationally expensive, and naive multi-start strategies quickly become infeasible for real-time operation. Recent advances [12] have incorporated spatiotemporal reasoning by embedding time as an additional dimension, enabling planners to handle moving obstacles as static entities in a high-dimensional configuration space. Although this improves adaptability to dynamic environments, it significantly increases computational burden. This limitation motivates the use of practical, lightweight homotopy proxies that preserve class distinction while minimizing labeling and optimization cost. In this work, we adopt this direction by extending classical H-signature reasoning with an efficient 3D spatiotemporal projection to approximate topological distinctions in dynamic scenes.

C. Planning in Dynamic and Multi-Agent Environments

Multi-robot trajectory planning involves managing collision avoidance, reciprocal anticipation, and communication constraints among different agents [23]. Early approaches often relied on centralized optimization, such as model predictive control (MPC) formulations [3], which ensure global consistency but scale poorly with the number of agents and communication load. Their dependence on centralized computation makes them unsuitable for dynamic environments that demand rapid replanning and responsiveness.

In contrast, single-robot high-speed replanning frameworks such as FASTER [1] and Ego-Planner [13] demonstrate strong adaptability to dynamic obstacles through perception–planning co-design and continuous optimization. However, these methods operate on a single platform and therefore lack the coordination mechanisms required in multi-agent scenarios.

To improve scalability and flexibility, decentralized and asynchronous planning frameworks such as ORCA [23], DMPC [3], and RMADER [4] allow each agent to plan locally while maintaining safety through limited inter-agent communication. This distributed structure significantly reduces coupling and computational load, enabling multi-agent systems to operate effectively in dynamic and communication-limited environments.

Despite these advantages, frequent local replanning can still cause deviations from globally consistent trajectories and increase computational overhead due to continuous full-trajectory optimization. To mitigate these issues, this work adopts a distributed multi-agent planning scheme incorporating a sliding time-window mechanism. Each agent performs only minor local adjustments within a short temporal horizon and triggers global replanning only when deviations exceed this window, maintaining smooth trajectories, predictable latency, and coherence with the overall plan.

III. Preliminaries and Problem Formulation

A. Differential Flatness and Trajectory Representation

Consider a general nonlinear *control-affine* system

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}. \tag{1}$$

The system is *differentially flat* if there exists a flat output $\mathbf{z}$ and smooth mappings Ψ_x, Ψ_u such that

$$\mathbf{x} = \Psi_x(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(s-1)}), \tag{2}$$

$$\mathbf{u} = \Psi_u(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(s)}). \tag{3}$$

For multicopters, a widely adopted flat output is

$$\mathbf{z} = (p_x, p_y, p_z, \psi)^\top \tag{4}$$

where $(p_x, p_y, p_z)^\top$ denotes the position of the center of mass and ψ the yaw angle.

Building on *differential flatness*, MINCO [2] introduces an efficient parameterized trajectory representation. For a given flat output $\mathbf{z}(t)$, it formulates the problem of a constrained minimum control effort as follows

$$\min_{\mathbf{z}(t),T} \quad J = \int_0^T \|\mathbf{z}^{(s)}(t)\|^2 \, dt \tag{5}$$

and shows that, given a fixed set of segment durations, boundary derivatives up to order $s-1$ and specified intermediate knots, there exists a unique optimal solution, which takes the form of a piecewise polynomial of degree $2s-1$.

This result enables efficient spatiotemporal decoupling: the piecewise polynomial over M segments is parameterized by intermediate waypoints $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_{M-1})$ and durations $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_M)^\top$, while the polynomial coefficients $\mathbf{c}$ can be directly constructed with linear complexity $O(M)$ by solving a nonsingular banded system:

$$\mathbf{A}(\boldsymbol{\tau})\,\mathbf{c}(\mathbf{q}, \boldsymbol{\tau}) = \mathbf{b}(\mathbf{q}) \tag{6}$$

Then we can address the original problem that involves complex constraints. Leveraging *differential flatness*, requirements such as dynamic feasibility and obstacle avoidance are transformed into the flat output space. Since the trajectory is parameterized, these constraints are then handled by optimizing over the parameter space$\{\mathbf{q}, \boldsymbol{\tau}\}$. Constraint elimination techniques are used to ultimately reformulate the problem as an unconstrained optimization.

B. Motion Camouflage Principle

Motion camouflage is a strategy observed in nature, where a predator moves towards its prey in such a manner that it appears to remain stationary from the prey's perspective against a distant background [24]. This illusion is achieved by constraining the predator's movement to follow a specific path. Specifically, its bearing is restricted to a fixed line, known as the **camouflage line**, which continuously connects the prey and a selected stationary reference point. The relative motion relationships and the geometric principle of this phenomenon are illustrated in Fig. 1.

The geometric relationship among the predator's position $\mathbf{p}_{\text{predator}}$, the prey's position $\mathbf{p}_{\text{prey}}$, and the reference point $\mathbf{p}_{\text{ref}}$ can be mathematically formulated as [25]:

$$\mathbf{p}_{\text{predator}} = \mathbf{p}_{\text{ref}} + \lambda(\mathbf{p}_{\text{prey}} - \mathbf{p}_{\text{ref}}) \tag{7}$$

where λ is a one-dimensional path control parameter. The value of λ dictates the predator's position along the camouflage line. In particular, when $\lambda = 1$, the positions of predator and prey coincide, signifying a successful interception [25]. By manipulating the parameter λ and the location of $\mathbf{p}_{\text{ref}}$, a wide array of trajectories can be generated, forming the theoretical basis for our proposed dimensionality reduction method.

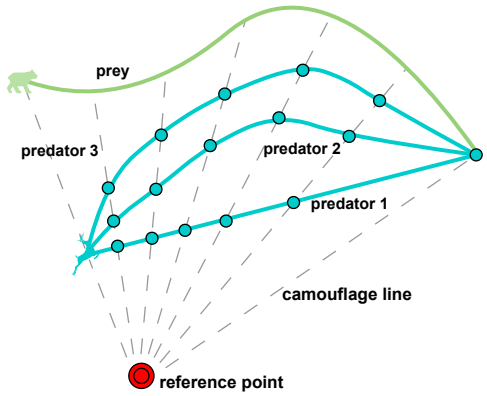


Fig. 1: Illustration of the motion camouflage phenomenon. The light green line represents the virtual prey trajectory (p_{prey}), while the different cyan lines (predator 1, 2, 3) represent different actual trajectories generated by adjusting the one-dimensional control parameter (λ). All points on the trajectories lie on the "camouflage line" connecting the reference point (p_{ref}) and the corresponding points on the prey's trajectory.

C. Problem Formulation

We consider N agents collaboratively executing tasks in a d-dimensional workspace $\mathcal{W} \subseteq \mathbb{R}^d$ with $d \in \{2, 3\}$. As established in Sec. III-A, all agents considered here are *differentially flat*. For each agent $\varkappa$, we optimize the translational flat-output vector $\mathbf{p}(t)$ and its derivatives up to order s. This simplification is justified as the yaw component, $\psi(t)$, is largely decoupled from the translational dynamics and can be planned separately, which significantly reduces the dimensionality.

We assume a cluttered, dense environment containing static and dynamic obstacles represented by closed sets $\mathcal{O}_{\text{stat}} \subset \mathcal{W}$ and $\mathcal{O}_{\text{dyn}}(t) \subset \mathcal{W}$, respectively. Let $\mathcal{F}(t) \subseteq \mathcal{W}$ denote the obstacle-free region available to the agent at time t.

The task is to generate a trajectory $\mathbf{p}(t)$ that is smooth, dynamically feasible, and collision-free. Following the principles of optimal control, we formulate the trajectory planning problem for each agent as a continuous-time constrained optimization problem that minimizes a weighted sum of control effort and flight duration. The problem is formally stated as:

$$\begin{aligned} \min_{\mathbf{p}(t),T} \quad & J = \int_0^T \|\mathbf{p}^{(s)}(t)\|^2 \, dt + \rho \cdot T \\ \text{s.t.} \quad & \mathbf{p}^{(0:s-1)}(0) = \bar{\mathbf{p}}_0, \\ & \mathbf{p}^{(0:s-1)}(T) = \bar{\mathbf{p}}_f, \\ & \mathcal{H}(\mathbf{p}(t), \ldots, \mathbf{p}^{(s)}(t)) \le 0, \ \forall t \in [0, T]. \end{aligned} \tag{8}$$

Here, $\mathbf{p}^{(0:s-1)}(t) = \left(\mathbf{p}^\top(t), \dot{\mathbf{p}}^\top(t), \ldots, \mathbf{p}^{(s-1)\top}(t)\right)^\top$ denotes the stacked flat state up to order $s-1$. The boundary conditions $\bar{\mathbf{p}}_0$ and $\bar{\mathbf{p}}_f$ define the fixed initial and terminal states, ρ is the time regularization parameter, and T denotes the total trajectory duration.

The term $\mathcal{H}(\cdot)$ encapsulates the set of continuous-time inequality constraints that are the primary source of complexity in multi-agent planning. These constraints include:

1) Obstacle Avoidance: The position $\mathbf{p}(t)$ must remain within the defined obstacle-free region for all time.
2) Inter-Agent Collision Avoidance: A minimum safe distance d_s must be maintained between any pair of agents.

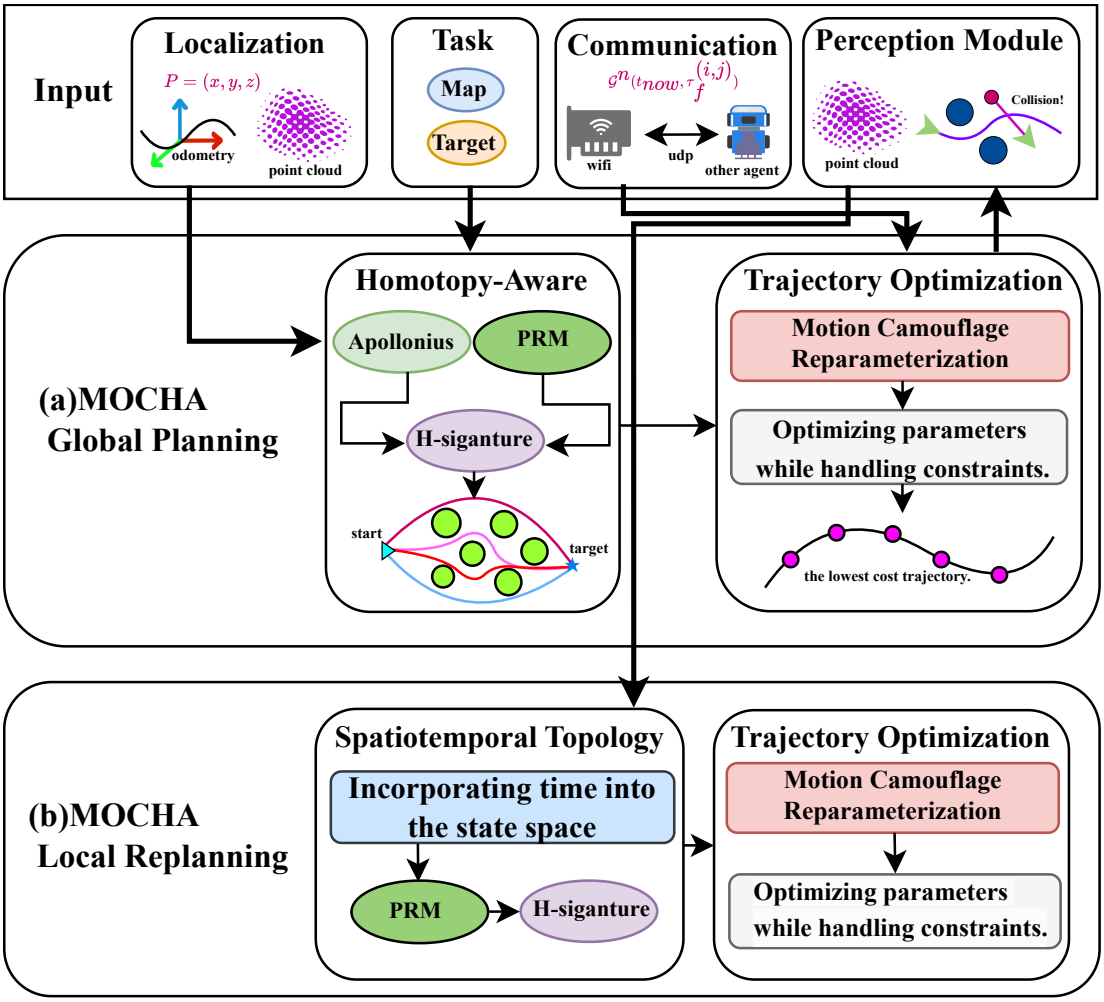


Fig. 2: System architecture of the MOCHA framework. The system comprises two main components: **(a) MOCHA Global Planning**, which utilizes a Homotopy-Aware front-end (Apollonius/PRM and H-signature) to generate multiple topologically distinct path candidates, followed by the Motion Camouflage Reparameterization and Trajectory Optimization to select the lowest-cost trajectory. **(b) MOCHA Local Replanning**, which incorporates time into the state space for Spatiotemporal Topology search (using PRM and H-signature approximation) and then employs the same efficient Motion Camouflage Reparameterization and Optimization modules for rapid dynamic obstacle avoidance and smooth return to the global path.

3) Dynamic Feasibility: The magnitudes of velocity and acceleration are constrained by the physical limits of the vehicle, $v_{\max}$ and $a_{\max}$. These are expressed as constraints on the derivatives of the trajectory, i.e., $\|\dot{\mathbf{p}}(t)\|^2 \leq v_{\max}^2$ and $\|\ddot{\mathbf{p}}(t)\|^2 \leq a_{\max}^2$.

As discussed in Sec. III-A, we can use a compact, sparse parameterization to solve this optimization problem. However, as the number of agents N and the segment count per trajectory grow, this increase leads to slower solve times. Therefore, we introduce a novel and more efficient parameterization in the next section.

IV. THE PROPOSED MOCHA METHOD

Before delving into the technical details of our approach, we first provide an overview of the MOCHA framework's structure and workflow, which is divided into Global Planning and Local Replanning modules, as illustrated in Fig. 2.

A. Motion Camouflage Reparameterization

The MINCO framework transforms the trajectory optimization problem described in (8) into a sparse parameter optimization at intermediate waypoints $\mathbf{q} \in \mathbb{R}^{d\times(M-1)}$ and segment durations $\boldsymbol{\tau}$. However, the high dimensionality of the intermediate waypoints can still become a significant bottleneck. Specifically, a large number of decision variables rapidly amplifies the non-convexity of the objective function, increasing the risk of convergence to undesired local minima, and simultaneously causes the computation latency of gradient-based solvers to scale poorly, thus threatening the real-time responsiveness required for frequent replanning.

To overcome this limitation and further accelerate computation, we draw inspiration from the motion camouflage principle to propose a novel reparameterization. Optimization-based algorithms often rely on an initial path provided by a front-end planner. We use this path as the virtual prey path $\mathbf{p}_{\text{prey}}$, and by selecting an additional reference point $\mathbf{p}_{\text{ref}}$, we reduce the high-dimensional waypoints $\mathbf{q}$ to a sequence of one-dimensional scalar parameters.

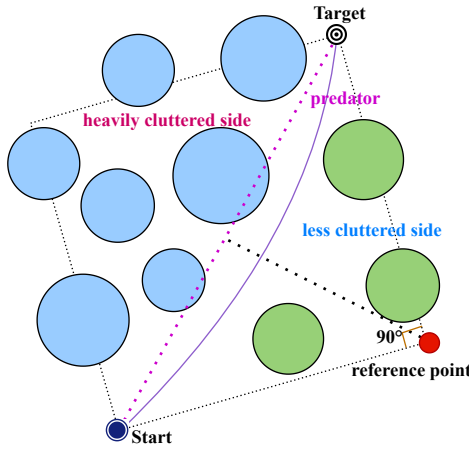


Fig. 3: Strategy for selecting the reference point $\mathbf{p}_{\text{ref}}$. The position is based on a right-angled triangle formed by the start-end line, and the direction is selected towards the side with lower obstacle density to enhance optimization flexibility.

Specifically, the position $\mathbf{q}_k$ of each intermediate knot $k \in \{1, \dots, M-1\}$ is now defined by a scalar parameter λ_k:

$$\mathbf{q}_k = \mathbf{p}_{\text{ref}} + \lambda_k(\mathbf{p}_{\text{prey}}(k) - \mathbf{p}_{\text{ref}}), \tag{9}$$

where $\mathbf{p}_{\text{prey}}(k)$ is the position on the virtual prey path corresponding to the k-th knot. The selection strategy for the reference point $\mathbf{p}_{\text{ref}}$ and its resulting influence on the state space are significant, especially the choice of direction. Therefore, our prior work [25] investigated this selection strategy in depth. As illustrated in Fig. 3, the position is determined based on a right-angled triangle formed by the line connecting the start and end points, while the direction is chosen towards the side with sparser obstacle distribution density to enhance optimization flexibility.

In this method, the path $\mathbf{p}(t)$ is described as a piecewise polynomial of degree $2s-1$ across M distinct segments. Optimization is carried out over the prey path parameters $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_{M-1})$ and segment durations $\boldsymbol{\tau} = (\tau_1, \dots, \tau_M)^\top$. The polynomial coefficients $\mathbf{c} = (\mathbf{c}_1^\top, \dots, \mathbf{c}_M^\top)^\top$ are directly constructed by solving

$$\mathbf{A}(\boldsymbol{\tau})\mathbf{c}(\boldsymbol{\lambda}, \boldsymbol{\tau}) = \mathbf{b}(\boldsymbol{\lambda}). \tag{10}$$

The i-th segment trajectory is expressed as

$$\mathbf{p}_i(t) = \mathbf{c}_i^\top \boldsymbol{\beta}(t), \qquad t \in [0,\, \tau_i), \tag{11}$$

where $\boldsymbol{\beta}(t) = (1, t, t^2, \dots, t^{(2s-1)})^\top$ denotes the polynomial basis, $\mathbf{c}_i \in \mathbb{R}^{(2s)\times d}$ is the i-th segment coefficient matrix.

This reparameterization substantially decreases the number of decision variables and endows them with a clear physical interpretation: $\boldsymbol{\lambda}$ implies the magnitude of the trajectory's deviation from the virtual prey path at the corresponding knot.

We have thus reformulated the trajectory optimization problem into a lower-dimensional and more structured parameter space, paving the way for scalable, real-time navigation for multi-agent systems.

B. Problem Reformulation

To efficiently solve the constrained trajectory optimization problem presented in (8), we reformulate it into an unconstrained nonlinear program. This is achieved by first parameterizing the trajectory $\mathbf{p}(t)$ using our proposed method, where the decision variables are reduced to the one-dimensional scalar sequence $\boldsymbol{\lambda}$ and the segment durations $\boldsymbol{\tau}$.

Since $\boldsymbol{\lambda}$ and $\boldsymbol{\tau}$ are constrained, we introduce $\mathcal{C}^1$ bijective maps from unconstrained variables. Specifically, we optimize over two unconstrained variables, $\boldsymbol{\eta} \in \mathbb{R}^M$ for time and $\boldsymbol{\xi} \in \mathbb{R}^{M-1}$ for space.

For the temporal parameters, each segment duration τ_i is mapped from an unconstrained variable η_i using a piecewise smooth function $\Psi(\cdot)$ that guarantees positivity:

$$\tau_i = \Psi(\eta_i) = \begin{cases} (\frac{1}{2}\eta_i + 1.0)\eta_i + 1.0, & \text{if } \eta_i > 0 \\ \frac{1}{(\frac{1}{2}\eta_i - 1)\eta_i + 1}, & \text{if } \eta_i \le 0 \end{cases}. \tag{12}$$

For the spatial parameters, each scalar λ_k is mapped from an unconstrained variable ξ_k using a scaled logistic function, which bounds its value within a predefined range $(\lambda_{\min}, \lambda_{\max})$:

$$\lambda_k = \Phi(\xi_k) = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \sigma(\xi_k), \tag{13}$$

where $\sigma(\xi_k) = 1/(1 + e^{-\xi_k})$ is the logistic function.

Subsequently, by sampling a fixed number of points within each trajectory segment, all continuous-time inequality constraints $\mathcal{H}(\cdot)$ are transcribed into the objective function as penalty terms. The final optimization problem is thus stated as:

$$\min_{\boldsymbol{\xi}, \boldsymbol{\eta}} \quad J_{\text{total}} = w_e J_e + w_t J_t + J_p, \tag{14}$$

where w_e and w_t are positive weights, J_e denotes control effort, J_t is the total flight time $T = \sum_{i=1}^{M} \tau_i$, and J_p includes dynamic feasibility limits, static obstacle avoidance, and inter-agent collision avoidance. By solving this NLP problem, we can efficiently generate trajectories that are smooth, dynamically feasible, and collision-free.

C. General Cost Function and Gradient Propagation

The MINCO framework provides an efficient method to map gradients of any cost functional J^* from the space $(\mathbf{c}, \boldsymbol{\tau})$ to the waypoint–time space $(\mathbf{q}, \boldsymbol{\tau})$. We denote the resulting gradients by $\partial J^*/\partial \mathbf{q}$ and $\partial J^*/\partial \boldsymbol{\tau}$.

Our motion-camouflage reparameterization inserts one additional layer into this chain: the optimization variables are the scalars $\boldsymbol{\lambda}$ instead of $\mathbf{q}$. From (9),

$$\frac{\partial \mathbf{q}_k}{\partial \lambda_k} = (\mathbf{p}_{\text{prey}}(k) - \mathbf{p}_{\text{ref}}) \tag{15}$$

and by the chain rule the gradient with respect $\boldsymbol{\lambda}$ to is the vector–Jacobian product

$$\frac{\partial J^*}{\partial \lambda_k} = \frac{\partial J^*}{\partial \mathbf{q}_k} \cdot \frac{\partial \mathbf{q}_k}{\partial \lambda_k} \tag{16}$$

The gradient with respect to the segment durations passes through unchanged.

Finally, optimization is carried out in the unconstrained variables $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$. Applying the chain rule yields

$$\frac{\partial J^*}{\partial \xi_k} = \frac{\partial J^*}{\partial \lambda_k}\,\Phi'(\xi_k), \qquad k = 1, \dots, M-1, \tag{17}$$

$$\frac{\partial J^*}{\partial \eta_i} = \frac{\partial J^*}{\partial \tau_i}\,\Psi'(\eta_i), \qquad i = 1, \dots, M. \tag{18}$$

where

$$\Phi'(\xi_k) = (\lambda_{\max} - \lambda_{\min})\,\sigma(\xi_k)\big(1 - \sigma(\xi_k)\big), \tag{19}$$

$$\Psi'(\eta_i) = \begin{cases} \eta_i + 1, & \eta_i > 0, \\ \dfrac{1 - \eta_i}{\left(\frac{1}{2}\eta_i^2 - \eta_i + 1\right)^2}, & \eta_i \le 0, \end{cases} \tag{20}$$

This hierarchical gradient flow preserves the structured parameterization while keeping the overall computation efficient.

Any time-integral penalty functional J_p^* can be approximated by introducing a general penalty function $\mathcal{P}^*$ and sampling $\kappa + 1$ points in each segment:

$$J_p^* \approx \sum_{i=1}^{M} \frac{\tau_i}{\kappa} \sum_{j=0}^{\kappa} \omega_j\ \mathcal{P}^*(\mathbf{c}, \boldsymbol{\tau}, \alpha_j \tau_i), \tag{21}$$

where $\omega_0 = \omega_\kappa = \frac{1}{2}$, $\omega_j = 1$ for $j = 1, \dots, \kappa - 1$, and $\alpha_j = j/\kappa$. From (11), we can form:

$$\begin{aligned} &\mathcal{P}^*(\mathbf{c}, \boldsymbol{\tau}, \alpha_j \tau_i) = \mathcal{P}^*\big(\mathbf{y}_i(\alpha_j \tau_i), \tau_f^{(i,j)}\big), \\ \text{s.t.} \quad &\tau_f^{(l,j)} = \sum_{m=1}^{l-1} \tau_m + \alpha_j \tau_l \\ &\mathbf{y}_i(t) = \big(\mathbf{p}_i(t), \dots, \mathbf{p}_i^{(s-1)}(t)\big) \end{aligned} \tag{22}$$

If the constraints are independent of absolute time, the argument $\tau_f^{(l,j)}$ can be omitted. We then require the gradients of J_p^* with respect to the polynomial coefficients $\mathbf{c}$ and the segment durations $\boldsymbol{\tau}$.

The gradient with respect to c_i is given by the chain rule:

$$\frac{\partial J_p^*}{\partial \mathbf{c}_i} = \frac{\tau_i}{\kappa} \sum_{j=0}^{\kappa} \omega_j\, \frac{\partial \mathcal{P}^*}{\partial \mathbf{y}_i}\, \frac{\partial \mathbf{y}_i}{\partial \mathbf{c}_i} \tag{23}$$

where $\partial \mathbf{y}_i / \partial \mathbf{c}_i$ can be expressed using the derivatives of the polynomial basis $\boldsymbol{\beta}(t)$.

The gradient with respect to τ has two cases. If $\mathcal{P}^*$ does not depend on the absolute time:

$$\frac{\partial J_p^*}{\partial \tau_i} = \frac{1}{\kappa} \sum_{j=0}^{\kappa} \omega_j\, \mathcal{P}^* + \frac{\tau_i}{\kappa} \sum_{j=0}^{\kappa} \omega_j\, \frac{\partial \mathcal{P}^*}{\partial \mathbf{y}_i}\, \frac{\partial \mathbf{y}_i}{\partial t}\, \alpha_j, \tag{24}$$

If $\mathcal{P}^*$ depends on absolute time, a perturbation of τ_i affects the current segment and all subsequent $l > i$. The sensitivity of the absolute time is:

$$\frac{\partial \tau_f^{(l,j)}}{\partial \tau_i} = \begin{cases} \alpha_j, & \text{if } l = i \\ 1, & \text{if } l > i \\ 0, & \text{if } l < i \end{cases} \tag{25}$$

which yields the additional contribution

$$B = \sum_{l=i}^{M} \frac{\tau_l}{\kappa} \sum_{j=0}^{\kappa} \omega_j\, \frac{\partial \mathcal{P}^*}{\partial \tau_f^{(l,j)}}\, \frac{\partial \tau_f^{(l,j)}}{\partial \tau_i} \tag{26}$$

The full gradient is the sum of (24) and (26).

D. Specific Cost Function and Gradients

This section will specify the concrete penalties; the gradients of the penalty functions and the general gradient propagation from $(\mathbf{c}, \boldsymbol{\tau})$ to $(\boldsymbol{\xi}, \boldsymbol{\eta})$ follows Sec.IV-C.

1) Control Effort J_e: We sum the s-th control effort on each trajectory segment to obtain:

$$\begin{aligned} J_e &= \sum_{i=1}^{M} \int_0^{\tau_i} \|\mathbf{p}_i^{(s)}(t)\|^2\, dt \\ &= \sum_{i=1}^{M} \operatorname{tr}\left(\mathbf{c}_i^\top \left[\int_0^{\tau_i} \boldsymbol{\beta}^{(s)}(t) \boldsymbol{\beta}^{(s)\top}(t) dt \right] \mathbf{c}_i \right). \end{aligned} \tag{27}$$

The gradients follow directly from the properties of quadratic forms and the Leibniz rule:

$$\frac{\partial J_e}{\partial \mathbf{c}_i} = 2 \left(\int_0^{\tau_i} \boldsymbol{\beta}^{(s)}(t) \boldsymbol{\beta}^{(s)\top}(t)\, dt \right) \mathbf{c}_i \tag{28}$$

$$\frac{\partial J_e}{\partial \tau_i} = \operatorname{tr}\left(\mathbf{c}_i^\top \boldsymbol{\beta}^{(s)}(\tau_i) \boldsymbol{\beta}^{(s)\top}(\tau_i) \mathbf{c}_i \right) \tag{29}$$

2) Obstacle Avoidance $\mathcal{P}_o$: We consider R obstacles. To ensure collision-free motion, we define a penalty function that activates when the trajectory encroaches upon the safety margin of any obstacle:

$$\mathcal{P}_o\left(\mathbf{p}_i(t)\right) = \sum_{r=1}^{R} \phi\left(d_s^2 - d_r^2(\mathbf{p}_i(t)\right) \tag{30}$$

$$\phi(x) = \max\{0, x\}^3 \tag{31}$$

where d_s is the prescribed safety distance and $d_r^2\left(\mathbf{p}_i(t)\right)$ is the squared minimum Euclidean distance from $\mathbf{p}_i(t)$ to obstacle r.

3) Dynamic Feasibility $\mathcal{P}_f$: To ensure the trajectory respects the agent's physical limits, we take velocity and acceleration as examples and introduce a penalty as follows:

$$\begin{aligned} \mathcal{P}_f(\mathbf{y}_i(t)) &= \mathcal{P}_v(\dot{\mathbf{p}}_i(t)) + \mathcal{P}_a(\ddot{\mathbf{p}}_i(t)) \\ \mathcal{P}_v(\dot{\mathbf{p}}_i(t)) &= \phi\left(\|\dot{\mathbf{p}}_i(t)\|^2 - v_{\max}^2\right), \\ \mathcal{P}_a(\ddot{\mathbf{p}}_i(t)) &= \phi\left(\|\ddot{\mathbf{p}}_i(t)\|^2 - a_{\max}^2\right), \end{aligned} \tag{32}$$

4) Inter-Agent Collision Avoidance $\mathcal{P}_s$: To achieve scalable multi-agent navigation, we employ a distributed and asynchronous planning framework that does not rely on centralized computation. In this architecture, each agent broadcasts its current trajectory to neighboring agents as a "committed trajectory" during execution. Other agents then utilize these received broadcasted trajectories for their own collision avoidance planning.

Specifically, we utilize a global time base t_{now} and trajectory sampling offset time $\tau_f^{(i,j)}$ to calculate the positions $\mathcal{G}^n(t_{\text{now}}, \tau_f^{(i,j)})$ of other agents based on their committed trajectories. Then we introduce a penalty as follows:

$$\begin{aligned} \mathcal{P}_s\left(\mathbf{p}_i(\alpha_j \tau_i), \tau_f^{(i,j)}\right) &= \sum_{n \neq \varkappa} \phi\left(d_s^2 - d_n^2\right), \\ d_n^2 &= \|\mathbf{p}_i(\alpha_j \tau_i) - \mathcal{G}^n(t_{now}, \tau_f^{(i,j)})\|^2 \end{aligned} \tag{33}$$

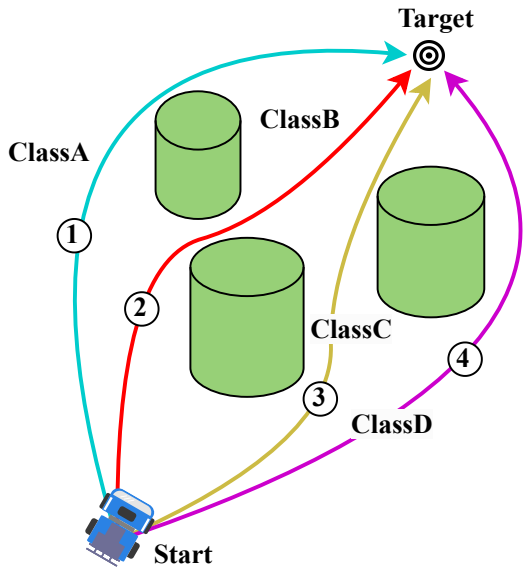


Fig. 4: Four representative homotopy classes (A–D) between the same endpoints.

Where n denotes any agent other than $\varkappa$, d_s is the safety distance threshold.

5) Total Time J_t: A key consideration for trajectory optimization to balance smooth and aggressiveness is minimizing the total time cost, which is defined as follows:

$$J_t = \sum_{i=1}^{M} \tau_i \ , \frac{\partial J_t}{\partial \mathbf{c}_i} = 0 \ , \frac{\partial J_t}{\partial \tau_i} = 1 \tag{34}$$

E. Global Path Multi-Homotopy Topology

As shown in Fig.4, in cluttered environments, two continuous trajectories are *homotopic* if one can be smoothly deformed into the other without crossing obstacles or altering endpoints; otherwise they belong to different *homotopy classes*.

Our motion camouflage reparameterization relies on a virtual prey path $\mathbf{p}_{\text{prey}}$ generated by a front-end planner. To mitigate local minima in non-convex optimization, we employ multi-homotopy path topology into the front-end planning and then parallel optimize different prey path to select the optimal solution.

The process begins by constructing a searchable sparse graph of the free space using methods like the Probabilistic Roadmap (PRM) or skeletonization (e.g., Apollonius diagram).

To filter redundant paths within the same class, we employ the H-signature: a lightweight integer vector that serves as a topological discriminator for a path relative to a set of obstacles.

Let a path γ be represented by a sequence of vertices z_k, and each obstacle by a point a_r. For each obstacle, we compute its corresponding integer winding number $h_r(\gamma)$, which represents the net number of counter-clockwise turns the path makes around it.

This is calculated by summing the continuous angle changes for each path segment (z_k, z_{k+1}) relative to the obstacle a_r:

$$\begin{aligned} \Theta_r(\gamma) &= \sum_{k=0}^{K-1} \widetilde{\angle}\left(\frac{z_{k+1} - a_r}{z_k - a_r}\right), \\ h_r(\gamma) &= \left\lfloor \frac{\Theta_r(\gamma)}{2\pi} \right\rceil . \end{aligned} \tag{35}$$

Here, $\widetilde{\angle}(\cdot)$ is the path-continuous argument function that accumulates angle changes without 2π jumps, and $\lfloor\cdot\rceil$ is the nearest-integer operator.

The H-signature of the path, $\mathbf{h}(\gamma)$, is the vector composed of these individual winding numbers for all R obstacles:

$$\mathbf{h}(\gamma) = (h_1(\gamma), h_2(\gamma), \ldots, h_R(\gamma)). \tag{36}$$

Thus, H-signature provide a direct method to distinguish between homotopy classes. In the three-dimensional setting, the essence of the H-signature is to generalize the point-based abstraction of obstacles in 2D to representing each obstacle as a spatial curve, and to determine whether two paths belong to different homotopy classes by evaluating the Gauss linking number or line-integral-based linkage of a closed path around that curve. This provides sound but not complete homotopy discrimination, following practical strategies adopted in topology-guided dynamic planning.

V. Local Replanning under Spatiotemporal Topology

A. Triggering Conditions and Assumptions

The agent obtains a high-quality, feasible global trajectory from the global planner described in the previous section. However, dynamic obstacles still exist in the environment. To ensure motion safety, a local replanning algorithm is required. We assume the agent can predict the trajectories of dynamic obstacles via sensors such as radar, which leads to our proposed method.

For compactness, we utilize the global time base t_{now} and the future offset time δ to calculate the agent's global trajectory positions $\tilde{\mathbf{p}}_g(t_{now}, \delta)$. Likewise, the predicted position of the r-th dynamic obstacle is $\tilde{\mathbf{p}}_o^r(t_{now}, \delta)$.

We then check in real-time whether the trajectory will conflict with any dynamic obstacle r within a forward time horizon μ_{pre}:

$$\min_{\delta \in [0, \mu_{pre}]} \|\tilde{\mathbf{p}}_g(t_{now}, \delta) - \tilde{\mathbf{p}}_o^r(t_{now}, \delta)\| < d_s \tag{37}$$

where d_s is the prescribed safety margin. If condition (37) is met, meaning a collision risk is predicted within the next μ_{pre} time, the local replanning is triggered.

In practice, computing the distance over a continuous time offset is computationally expensive. We approximate this check by discretely sampling time offsets Δt at a sufficiently high resolution within the time horizon $[0, \mu_{pre}]$ and evaluating the minimum distance over these sample points.

B. Local Objective and Dynamic-Obstacle Penalty

If the local planner is activated, it will generate a new trajectory $\mathbf{p}_{\mathrm{L}}(t)$ that avoids the predicted collision while remaining as close as possible to the original global path. This problem is formulated as an optimization that re-uses the efficient motion camouflage parameterization that is introduced in Sec. IV.

First, we should specify the boundary conditions of the optimization problem. The initial state is set to the agent's current state at the moment t_σ when the local planner is triggered, and the terminal state is "anchored" to the original global trajectory.

$$\begin{aligned} \bar{\mathbf{p}}_0 &= \tilde{\mathbf{p}}_g^{(0:s-1)}(t_\sigma, 0) \\ \bar{\mathbf{p}}_f &= \tilde{\mathbf{p}}_g^{(0:s-1)}(t_\sigma, \mu) \end{aligned} \tag{38}$$

Where μ is the time horizon for the local replanning, and is set larger than the prediction window μ_{pre} to ensure sufficient time to smoothly navigate around obstacles and rejoin the global path. The objective function for the local problem is structured similarly to the global objective (14), but with added penalty and modified time-cost term.

The new penalty term $\mathcal{P}_d$ is for the dynamic obstacles avoidance. It is formulated analogously to the inter-agent collision avoidance term (33) and leverages the absolute time $\tau_f^{(i,j)}$ to query the predicted position of dynamic obstacles $\tilde{\mathbf{p}}_o^r(t_{now}, \tau_f^{(i,j)})$ at each sampled point along the local trajectory.

$$\begin{aligned} \mathcal{P}_d\left(\mathbf{p}_i(\alpha_j\tau_i), \tau_f^{(i,j)}\right) &= \sum_{r=1}^{R_d} \phi\left(d_s^2 - d_r^2\right), \\ d_r^2 &= \|\mathbf{p}_i(\alpha_j\tau_i) - \tilde{\mathbf{p}}_o^r(t_{now}, \tau_f^{(i,j)})\|^2 \end{aligned} \tag{39}$$

where R_d is the number of relevant dynamic obstacles; the gradient calculation follows the same logic as (26).

To encourage the duration of the local trajectory closely matches the time horizon μ, we replace the simple linear time cost J_t with the following soft constraint:

$$J_t^l = \left|\sum_{i=1}^{M} \tau_i - \mu\right|^3 \tag{40}$$

The gradients with respect to $\mathbf{c}_i$ and τ_i are:

$$\frac{\partial J_t^l}{\partial \mathbf{c}_i} = 0, \frac{\partial J_t^l}{\partial \tau_i} = 3(\sum_{i=1}^{M} \tau_i - \mu)^2 \cdot \mathrm{sgn}(\sum_{i=1}^{M} \tau_i - \mu) \tag{41}$$

Finally, a safety protocol is enforced after the local optimization converges. We define a maximum permissible time window T_ϵ, if the duration of the local trajectory exceeds this window,

$$\sum_{i=1}^{M} \tau_i > T_\epsilon, \tag{42}$$

the agent will still execute the computed local trajectory $\mathbf{p}_{\mathrm{L}}(t)$ to ensure dynamic collision avoidance. However, during the execution of this local trajectory, the global planner described in Sec. IV is asynchronously triggered.

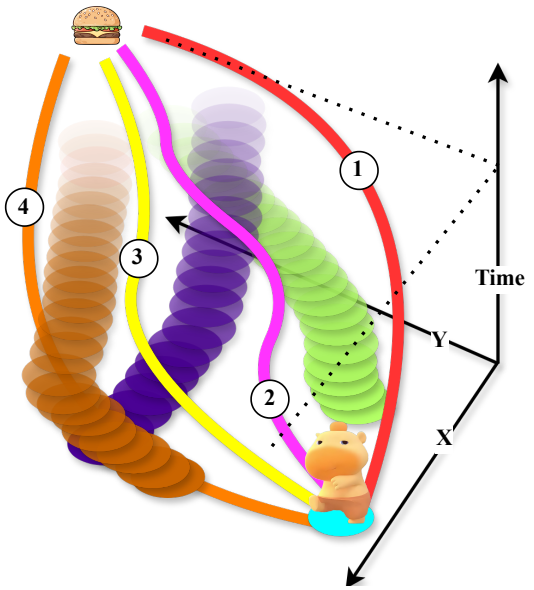


Fig. 5: Dynamic obstacles are represented as static "cylinders" in the 2D+Time state space. A sampling-based planner finds multi-homotopy paths (e.g., 1, 2, 3, 4)

C. Homotopy Approximation via 3D Spatiotemporal Projections

Local optimization also requires a virtual prey path. However, different from the global topology, the local planner must handle topology in a dynamic environment.

To address this, we adopt and generalize the topology-driven concept from De Groot et al.[12] This approach handles moving obstacles by incorporating time into the state space. Fig.5 illustrates this concept using the $d = 2$ case.

While the implementation in [12] focused on 2D, the principles of sampling-based search and topological analysis can extend to higher dimensions.

For the $d = 3$ case, we construct a simplified spatiotemporal state space $\mathcal{X} := \mathbb{R}^3 \times [0, \mu]$, where μ is our local planning horizon.

In this 4-dimensional state space, the predicted future motion of dynamic obstacles are treated as static hyper-tubes.

Finding paths in this space using PRM is algorithmically feasible, the challenge lies in distinguishing different homotopy classes. H-signature in a 4D space is computationally complex and challenging for real-time applications.

Therefore, we adopt a practical approximate method. We project the 4D paths and obstacle hyper-tubes onto three distinct 3D spatiotemporal subspaces: (x, y, t), (x, z, t), and (y, z, t). We then compute the H-signature for any two paths independently in each of these three projections. If the H-signature differ between the two paths in any of these three projections, we consider them to be in different homotopy classes.

We acknowledge that this method, designed for computational simplicity, has inherent limitations and cannot fully distinguish all homotopy classes. However, for our multi-homotopy path optimization scenario, this approximation is acceptable.

Proposition 1 (Soundness of the 3D projection test)**.** *Let $X = \mathbb{R}^3 \times [0, \mu]$. Obstacles $O \subset X$, free space $F = X \setminus O$. A causal path is a continuous map $\gamma : [0, 1] \to F$ with nondecreasing time. Denote the coordinate projections $\pi_{xyt}, \pi_{xzt}, \pi_{yzt}$ and the projected free spaces $F_\alpha = \pi_\alpha(F)$. On each F_α we compute an H-signature h_α that separates endpoint-fixed homotopy classes along the region we traverse.*

For two causal paths γ_1, γ_2 with the same endpoints, if there exists $\alpha \in \{xyt, xzt, yzt\}$ such that

$$h_\alpha(\pi_\alpha \circ \gamma_1) \neq h_\alpha(\pi_\alpha \circ \gamma_2),$$

then γ_1 and γ_2 are not homotopic in F.

Proof. The projection π_α induces a map on endpoint-fixed homotopy classes. If γ_1 and γ_2 were homotopic in F, then $\pi_\alpha \circ \gamma_1$ and $\pi_\alpha \circ \gamma_2$ would be homotopic in F_α, hence their H-signature would coincide. □

Remark 1 (Implementation Note)**.** *In practice we use obstacle shadows $\widetilde{F}_\alpha = \mathbb{R}^3 \setminus \pi_\alpha(O)$. This is conservative and keeps the proposition sound when the compared curves have clearance $c > 0$ to $\partial\widetilde{F}_\alpha$. A sufficient sampling rule in time: if the spatial motion is L-Lipschitz, choose step Δt with $L\,\Delta t < c$.*

Remark 2 (Limitation)**.** *The test is not complete: equal signatures in all three projections does not guarantee equivalence in 4D. We treat it as a certificate of difference.*

VI. Experiments and Results

This section presents the validation of our algorithm through both simulation and real-world experiments, accompanied by a systematic quantitative analysis of the outcomes.

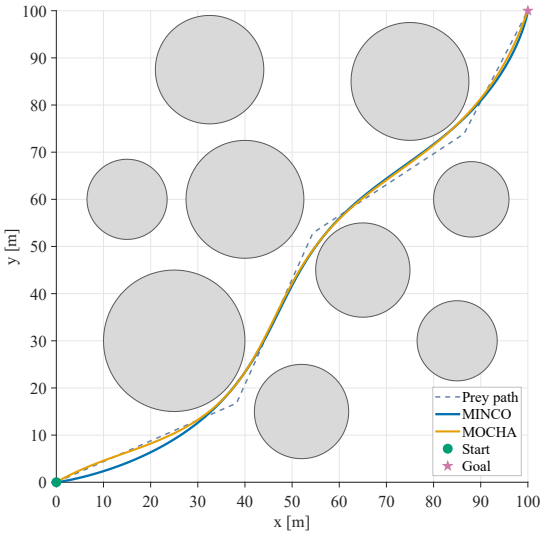
A. Implementation Setup

All simulations were performed on a workstation equipped with an Intel Core i7-14650HX CPU (2.2GHz). This environment utilized two software platforms: numerical simulations were run in MATLAB 2023b, while system-level simulations were conducted using ROS2-Humble. For real-world experiments, the experimental platform was a UGV equipped with a Raspberry Pi 4 and an M10P lidar, and the entire physical setup also operated on the ROS2-Humble.

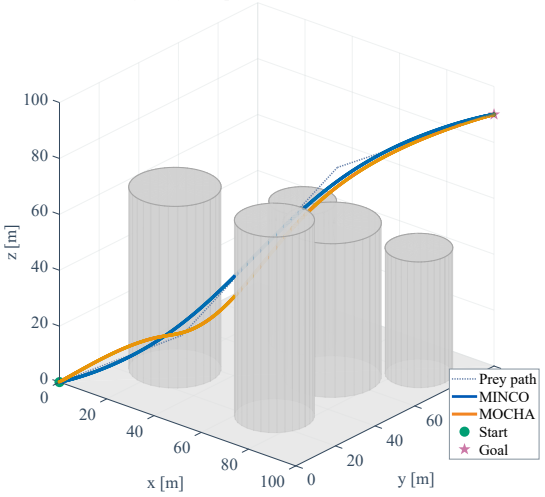
B. Simulation Experiments

To validate the effectiveness of our proposed algorithm's dimensionality reduction, we first conducted a numerical comparison between our MOCHA algorithm and the MINCO algorithm within MATLAB. The experiments were set in two representative static environments: a 2D environment with circular obstacles and a 3D forest environment with cylindrical obstacles.

In our experiments, the trajectories were uniformly discretized with a segment length of $1\,\mathrm{m}$, and each segment contained $\kappa = 20$ sampling points. The optimization weights were set as follows: the smoothness (energy) term $w_e = 1$, the temporal regularization term $w_t = 1$, the obstacle avoidance penalty $w_o = 100$, and the dynamic feasibility penalty $w_d = 100$. For both MOCHA and MINCO, the dynamic



(a) Trajectory comparison in the 2D environment.



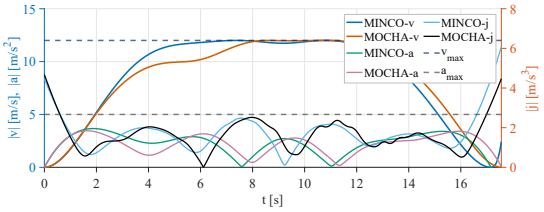
(b) Trajectory comparison in the 3D environment.

Fig. 6: Trajectories generated by MOCHA and MINCO in 2D (a) and 3D (b) environments.

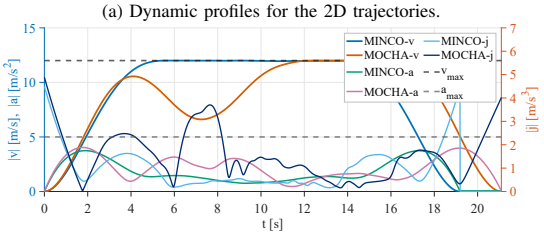
constraints were consistently enforced with $v_{\max} = 12\,\mathrm{m/s}$ and $a_{\max} = 5\,\mathrm{m/s^2}$. All optimization problems were solved using *fminunc*.

Fig. 6a and Fig. 6b show that both MOCHA and MINCO successfully generate smooth and collision-free trajectories in the 2D and 3D environments, respectively. The corresponding dynamic profiles in Fig. 7a and Fig. 7b confirm that all trajectories strictly satisfy the imposed dynamic limits.

The specific quantitative results of the optimization are summarized in Table I. In the 2D scenario, our MOCHA algorithm reduced the planning time by 25.9% while maintaining nearly identical trajectory quality. The computational advantage was even more pronounced in the 3D scenario, where our algorithm achieved a reduction of 28.4% in planning time. It is worth noting that in the 3D case (Table I), MOCHA's trajectory quality ($J = 494.34$) is slightly lower than that of MINCO ($J = 419.06$). Specifically, MOCHA yields a slightly longer trajectory length (increased by $\approx 1.65\%$) and a slower arrival time (increased by $\approx 10.0\%$), which represents a reasonable trade-off for its significantly reduced optimization runtime.



(a) Dynamic profiles for the 2D trajectories.



(b) Dynamic profiles for the 3D trajectories.

Fig. 7: Dynamic profiles for the trajectories in 2D (a) and 3D (b) environments, confirming adherence to dynamic constraints.

TABLE I: Validation of Dimensionality Reduction Effectiveness.

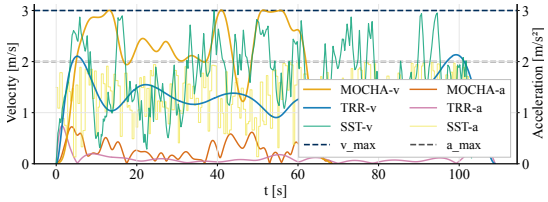
Method	t_{plan} (s)	T(s)	L (m)	J
MOCHA(2D)	**1.86**	**17.51**	**146.24**	**409.57**
MINCO(2D)	2.51	17.15	146.89	400.12
MOCHA(3D)	**1.74**	**21.11**	**178.3**	**494.34**
MINCO(3D)	2.43	19.19	175.4	419.06

This is an expected trade-off: by reducing the high dimensional waypoint search space to a single dimension, our method makes a small sacrifice in trajectory optimality in exchange for a significant gain in computational speed. As subsequent comparisons show, this trajectory quality remains significantly superior to other SOTA(state-of-the-art) algorithms. Therefore, the experimental results demonstrate the effectiveness of our algorithm's dimensionality reduction.

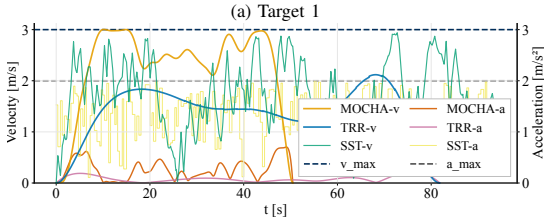
Subsequently, and prior to real-world validation, we deployed our complete planning framework—comprising the front-end topologizer and the MOCHA local planner—within the ROS2-Humble environment. Specifically, for 2D global planning, our front-end topologizer utilizes an H-signature topology check on a skeleton graph generated from an Apollonius diagram, which is highly efficient in 2D. For local replanning, the front-end topology is generated using a Probabilistic Roadmap (PRM). To evaluate the system in this more realistic setting, we benchmarked it against two other SOTA planners: TRR [26] and SST [27].

This selection provides a comprehensive comparison: TRR, much like our method, is an iterative spatio-temporal optimization algorithm, while SST is a prominent discrete sampling-based algorithm. The experiment was conducted in an environment with dozens of obstacles. For this simulation, the robot radius was set to 0.3m, and the dynamic limits were set to $v_{\max} = 3$ m/s and $a_{\max} = 2$ m/s^2. We selected three different goal locations for comparison to comprehensively evaluate performance.

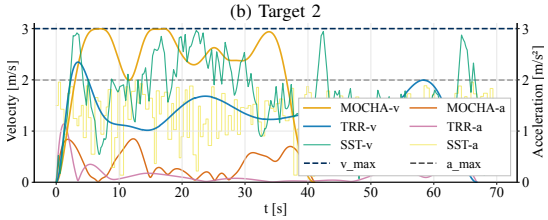
The trajectories in Fig. 9 and the dynamics profiles in Fig. 8 are quantitatively substantiated in Table II. MOCHA's planning time t_{plan} of 23–29 ms is exceptionally efficient, proving up to 8.8× faster than TRR (48–229 ms) and consistently over



(a) Target 1



(b) Target 2



(c) Target 3

Fig. 8: Dynamics corresponding to the trajectories in Fig. 9. MOCHA's velocity sustains a high average level, whereas TRR and SST are visibly more conservative, resulting in the longer flight times quantified in Table II.

TABLE II: Quantitative Comparison with SOTA Planners across Three Different Targets.

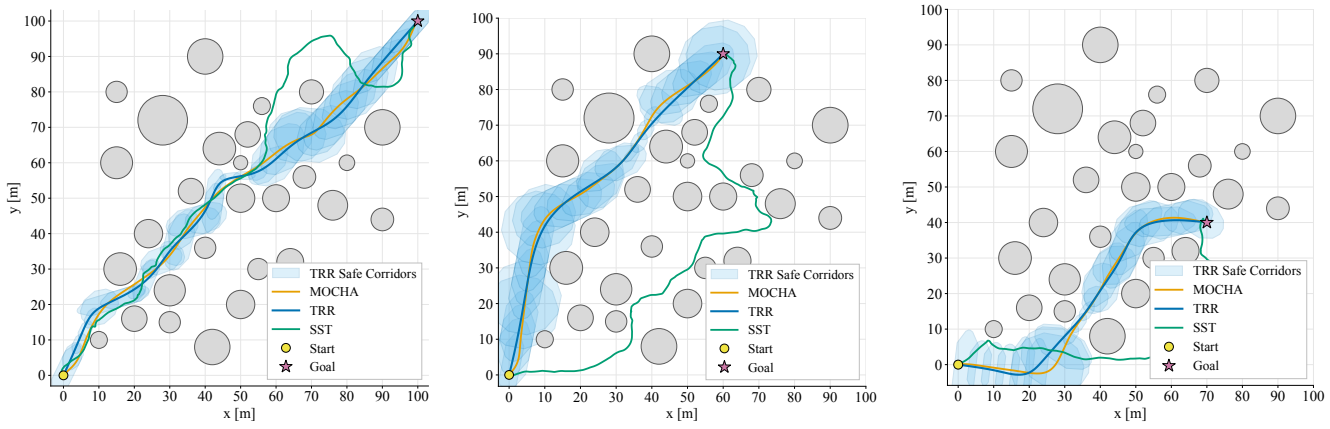
Target	Method	t_{plan} (s)	L (m)	T (s)	$v_{\max}$ (m/s)	v_{avg} (m/s)
Target 1	MOCHA	**0.029**	**144.57**	**68.45**	**3.00**	**2.11**
	TRR	0.187	145.80	108.93	2.14	1.38
	SST	1.506	177.97	103.05	2.99	1.72
Target 2	MOCHA	**0.026**	**114.67**	**50.48**	**3.00**	**2.27**
	TRR	0.229	113.04	81.79	2.11	1.38
	SST	1.071	159.39	93.65	2.94	1.71
Target 3	MOCHA	**0.023**	**93.21**	**41.03**	**2.98**	**2.27**
	TRR	0.048	91.11	66.82	2.34	1.36
	SST	1.059	115.96	69.75	2.96	1.66

an order of magnitude (36×–51×) faster than SST (1059–1506 ms). This computational speed, enabled by our motion camouflage reparameterization. MOCHA achieves the shortest total flight time (T) in all scenarios, completing tasks 37–39% faster than TRR and 34–46% faster than SST. This is achieved by generating aggressive trajectories with the highest average velocity v_{avg} (avg. 2.22 m/s vs TRR's 1.37 m/s) while adhering to $v_{\max}$ limits and maintaining competitive path lengths L.

C. Advanced features

Finally, we evaluate the advanced features of MOCHA: local replanning for dynamic obstacles and multi-agent scalability.

We first test the local replanning module, as demonstrated in Figure 10. In this scenario, the agent first plans a global trajectory (shown as the dashed blue line) and begins execution. During its movement, it continuously checks for potential



(a) Target 1 (b) Target 2 (c) Target 3

Fig. 9: Comparison of resulting trajectories from all three methods. MOCHA (ours) consistently finds smooth, efficient, and direct paths to the goal. TRR first constructs safe flight corridors and then performs iterative spatio-temporal optimization within them. SST explores the space and often produces paths that are noticeably longer and less direct.

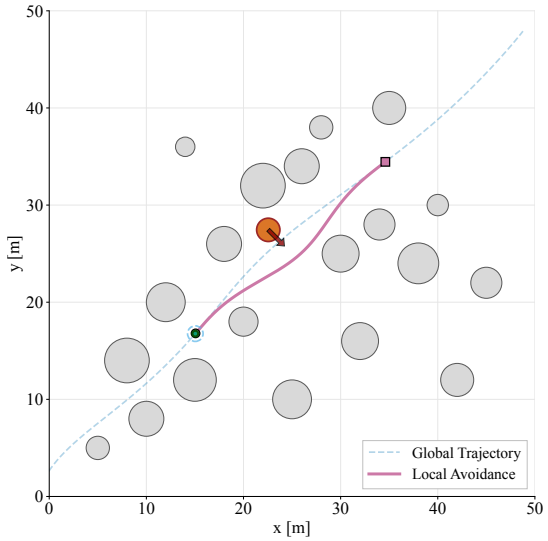


Fig. 10: Local replanning maneuver for a dynamic obstacle. The agent, while following its global path, detects a conflict with an obstacle. It triggers a local planner to generate a new avoidance trajectory that rejoins the global path at a future anchor point.

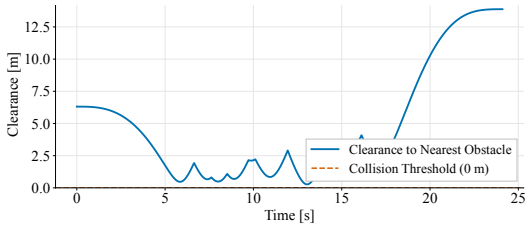


Fig. 11: Obstacle clearance data for the maneuver in Figure 10. The plot shows the agent's minimum distance to the nearest obstacle over time, confirming it remains safely above the collision threshold throughout the avoidance.

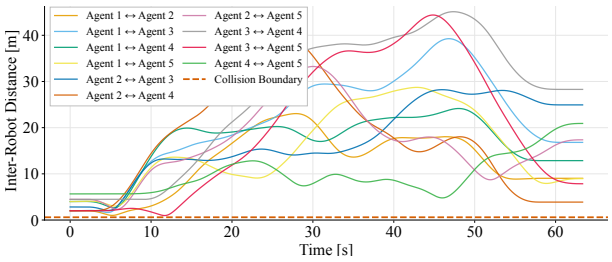


Fig. 12: Inter-agent distances for the 5-agent simulation. The plot shows the minimum distance between all pairs of agents over time. The minimum safety distance (dashed red line) is respected by all agents, proving the effectiveness of the inter-agent collision avoidance penalty.

conflicts with dynamic obstacles within a forward prediction horizon of $\mu_{pre} = 3s$.

As visualized in Fig. 10, upon detecting a collision risk with an approaching dynamic obstacle (orange circle), the local planner is triggered. It generates a new, smooth avoidance trajectory (solid red line) to safely navigate around the obstacle. This local trajectory is optimized to rejoin the original global path at a future "anchor point" (red square), which is set at $\mu = 5s$ along the global path from the moment of replanning. The corresponding clearance data in Fig. 11 quantitatively validates this behavior. It confirms that the agent maintains a safe distance from all obstacles (both static and the moving one), and crucially, the clearance never drops below the safety threshold.

We demonstrate the scalability of our framework by simulating 5 agents navigating to distinct goals in the same cluttered environment. Specifically, we placed the agents at nearby starting coordinates - (0,0), (2,0), (0,2), (4,0) and (0,4) - and assigned them distinct randomly generated goal locations, as shown in Fig. 13.

The effectiveness of our inter-agent collision avoidance penalty is proven in Fig. 12. The dashed line in the figure represents the set safety distance of 0.8m, which includes an additional 0.2m safety margin on top of the 0.6m minimum separation (twice the robot radius) to mitigate potential communication and planning delays. The plot confirms that all inter-agent distances remained safely above this 0.8m threshold throughout their entire motion.

Table III further details the performance metrics for this test. In this context, N represents the number of initial homotopy

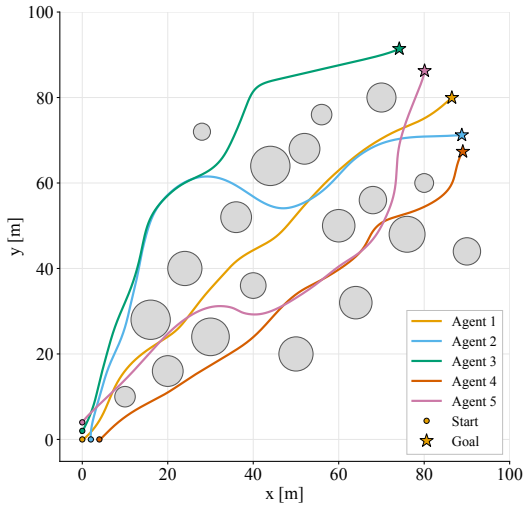


Fig. 13: Multi-agent scalability demonstration. Five agents, starting from nearby locations, successfully generate safe and efficient trajectories to distinct goals in a cluttered environment using our distributed and asynchronous planning approach.

TABLE III: Performance metrics for the 5-agent scalability test.

Start	t_{plan} (s)	N (Paths)	D_{min} (m)
(0,0)	0.027	6	1.002
(2,0)	0.053	6	1.002
(0,2)	0.052	6	0.969
(4,0)	0.079	5	2.001
(0,4)	0.083	5	0.969

paths found by our front-end planner, and t_{plan} is the *total* time taken to optimize all these N paths in parallel. The results show that the optimization time for any single trajectory is extremely low (e.g., for agent (0,0), 6 paths were optimized in parallel in just 0.027s). This multi-homotopy approach, combined with parallel optimization, effectively mitigates the problem of the optimizer getting trapped in local minima. This advantage is particularly evident given the scenario: the agents' start points are clustered closely together, as are their end points. A traditional, single-topology optimization algorithm would struggle to produce such diverse, high-quality, and collision-free trajectories. The data confirms the real-time performance and scalability of MOCHA.

D. Real-world Experiments

To validate the practical applicability and robustness of the MOCHA framework, we conducted real-world experiments using two ground vehicles in a cluttered indoor environment, as shown in Fig. 14 and Fig. 15.

The experimental scene was set up in a long corridor, populated with 12 cylindrical obstacles with radii ranging from 36–44 cm. The MOCHA algorithm was deployed in a distributed manner on the two vehicles, which were equipped with LiDAR for indoor localization and real-time position acquisition. The agents used UDP to broadcast their committed trajectories to each other.

In the scenario depicted, the two agents (Agent 1 and Agent 2) start at nearby initial positions of $(0, 0)$ and $(-1, 0)$,

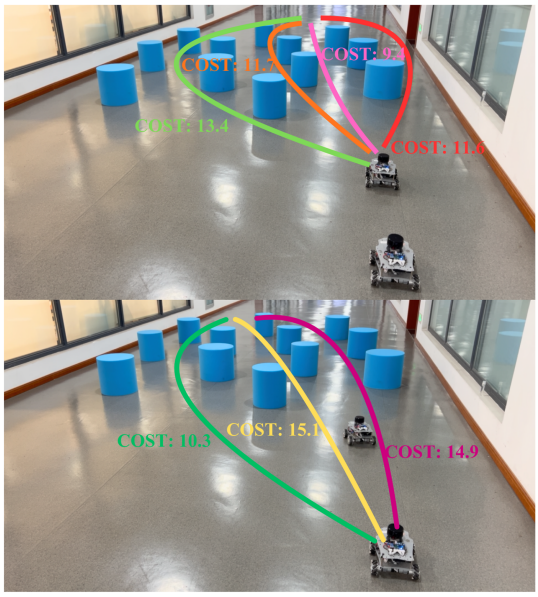


Fig. 14: Real-world multi-agent experiment setup. **Top:** Agent 1 (further robot) computes multiple homotopy-aware paths, with the lowest cost path (pink, 9.7) selected. **Bottom:** Agent 2 (closer robot) computes its paths. Due to Agent 1's committed trajectory, the costs of conflicting paths (yellow, 15.1; magenta, 14.9) increase, leading Agent 2 to select the safer, non-conflicting green path (10.3).

respectively. They are tasked with navigating to two distinct, randomly selected goal points at the far end of the corridor. Both agents first perform multi-homotopy path planning to find several topologically distinct, feasible routes.

As shown in the top panel of Fig. 14, Agent 1 first plans and finds multiple paths, selecting the one with the lowest cost (the pink path with cost 9.7) to execute. It commits to this trajectory and broadcasts it. Subsequently, Agent 2 performs its optimization. As shown in the bottom panel of Fig. 14, Agent 2 also finds multiple paths. However, when calculating its costs, it incorporates Agent 1's committed trajectory via the inter-agent collision penalty $\mathcal{P}_s$. This significantly increases the cost of its yellow (15.1) and magenta (14.9) paths, which would otherwise conflict with Agent 1. As a result, Agent 2 selects the green path (cost 10.3), which provides a higher safety margin and is now the optimal choice.

Fig. 15 shows a time-lapse of the successful execution of this plan. Agent 1 follows its chosen path, while Agent 2 executes the safe green path. Both vehicles successfully navigate the dense obstacle field and maintain safe separation, ultimately reaching their respective target points. This experiment demonstrates the effectiveness of our distributed, homotopy-aware framework in coordinating multiple agents safely and efficiently in a real-world, cluttered environment.

VII. Discussion and Limitations

In this work, we introduced MOCHA, a homotopy-aware trajectory planning framework that combines a motion camouflage–inspired reparameterization with multi candidate optimization to achieve fast and safe motion in dynamic and cluttered environments. Extensive simulation and real-world

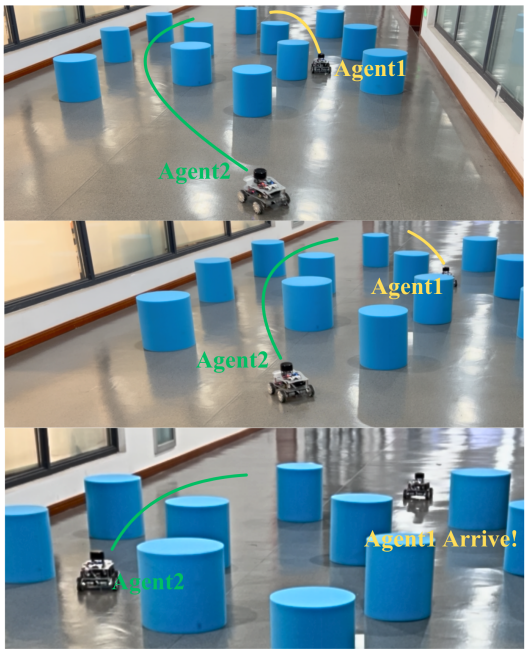


Fig. 15: Time-lapse of the real-world experiment execution. Agent 1 (following its chosen path, indicated in yellow) and Agent 2 (green path) execute their trajectories. Both agents successfully navigate the dense obstacle field and avoid each other, demonstrating the effectiveness of the distributed MOCHA framework.

experiments demonstrate that the method significantly accelerates computation while maintaining smoothness, dynamic feasibility, and multi-agent scalability.

Despite these advantages, several limitations remain. First, the current dynamic-obstacle handling relies on deterministic constant-velocity predictions, which may lead to suboptimal behavior in highly interactive or uncertain scenarios. Second, our homotopy reasoning is based on three spatiotemporal projections rather than full 4D topology, providing efficiency but only approximate discriminability in complex 3D entanglements. Additionally, extremely aggressive maneuvers may require increased scalar resolution to ensure full expressiveness of the compressed representation.

In future work, we plan to incorporate probabilistic behavior prediction, explore more rigorous space–time homotopy characterizations, and develop adaptive parameterization strategies. We believe these enhancements will further strengthen MOCHA's reliability and applicability in large-scale, real-world multi-agent deployment.

References

[1] F. Gao, B. Liu, Y. Lin, and S. Shen, "Fast and safe trajectory planning in unknown environments using hierarchical graph search," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Paris, France, 2020, pp. 100–106.

[2] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[3] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[4] K. Kondo, J. Tordesillas, R. Figueroa, J. Rached, J. Merkel, P. C. Lusk, and J. P. How, "Robust MADER: Decentralized and asynchronous multiagent trajectory planner robust to communication delay," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1687–1693.

[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 489–494.

[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 4569–4574.

[7] J. Schulman, J. Ho, A. Lee, A. Awwal, H. Bradlow, and P. Abbeel, "TrajOpt: A trajectory optimization framework for robotics," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014, pp. 1–8.

[8] M. Mukadam, F. Dellaert, and B. Boots, "GPMP2: A factor graph based approach for motion planning in arbitrary environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 1002–1009, 2018.

[9] K. Hauser, "Robust trajectory optimization in the presence of obstacles," in *Proc. Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

[10] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *AAAI Conference on Artificial Intelligence*, 2010, h-signature based topological labeling.

[11] S. Bhattacharya, M. Likhachev, and V. Kumar, "Identification and representation of homotopy classes of trajectories for search-based path planning in 3d," in *Robotics: Science and Systems (RSS)*, 2011.

[12] O. de Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, vol. 41, pp. 110–126, 2025.

[13] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[14] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[15] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed asymptotically optimal anytime search: BIT*," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[16] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, 2024.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," in *Proc. Robotics: Science and Systems (RSS)*, Los Angeles, USA, 2011.

[18] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016, pp. 4107–4113.

[19] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[20] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Seattle, USA, 2015, pp. 436–443.

[21] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science Robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[22] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *arXiv preprint arXiv:1707.07239*, 2017. [Online]. Available: https://arxiv.org/abs/1707.07239

[23] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research*, vol. 70, pp. 3–19, 2011.

[24] M. V. Srinivasan and M. Davey, "Strategies for active camouflage of motion," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 259, pp. 19 – 25, 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:131341953

[25] J. Li, Y. Zhu, C. Li, and Z. Song, "A motion camouflage-inspired path planning method for uavs based on reinforcement learning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 61, no. 2, pp. 4105–4114, 2025.

[26] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[27] K. E. Bekris and R. Shome, “Asymptotically optimal sampling-based planners,” in *Encyclopedia of Robotics*. Springer, 2021, pp. 1–12.