Topology-Aware Distributed Multi-Robot Coordination

MSc Thesis

Jules Zwanen

Delft University of Technology

TU Delft

Topology-Aware Distributed Multi-Robot Coordination

by

Jules Zwanen

to obtain the degree of Master of Science
at the Delft University of Technology,

Thesis committee:

Chair:	Dr. L. Ferranti	
Supervisors:	Dr. L. Ferranti,	(TU Delft)
External Examiners:	Dr. M. Khosravi	(TU Delft)
	Prof.dr. J. Alonso-Mora	(TU Delft)
Project duration:	June, 2025 – January, 2026	
Student number:	4959361	

TU Delft

Abstract

Explicit trajectory communication can be used to coordinate multiple robots, but communicating at every planning iteration can lead to congestion of the communication network, increase message delays and message loss. At the same time, collision-free trajectory planning is often formulated as a nonconvex optimization problem, which can converge to different locally optimal solutions across consecutive planning iterations. When this happens, a robots planned motion can switch between distinct high-level avoidance behaviors, such as passing an obstacle on the left versus on the right, which can lead to inefficient or unsafe behavior. Topology-based motion planners address this by explicitly computing multiple candidate motion plans that represent these different passing decisions, each associated with a distinct homotopy class. This work asks how (changes in) homotopy-class representations can trigger communication to reduce communication load while maintaining safe and efficient behavior. Building on the topology-driven trajectory optimization (T-MPC) approach of [**?**], we propose T-DMPC, a topology-aware distributed motion planner in which each robot, computes multiple guidance trajectories in distinct homotopy classes and refines them via parallel local trajectory optimization within the corresponding homotopy classes, selects a solution using a consistent decision rule that prioritizes the previously executed homotopy class, and communicates the selected trajectory using an event-triggered policy. Communication is triggered by homotopy changes and complemented by geometric-deviation and time-based triggers to bound trajectory staleness. In addition, the robots communicate a fallback trajectory during planning failures (e.g. infeasibility). We evaluate T-DMPC on antipodal swap maneuvers with 2 and 3 robots in simulation and on physical robots, comparing against T-VMPC (no communication, constant-velocity predictions) and T-AMPC (always communicate). The experiments show that, T-DMPC achieves task duration and traveled distance comparable to T-AMPC and T-VMPC, while reducing communication to about 9.8% (2 robots) and 13.2% (3 robots) of planning iterations in simulation, and 17.8% (2 robots) and 13.2% (3 robots) in real-world experiments, with no observed physical collisions. Ablations however show that topology-change alone is insufficient for safety, motivating the combined trigger design.

Topology-Aware Distributed Multi-Robot Coordination

Jules Zwanen

Abstract*—Explicit trajectory communication can be used to coordinate multiple robots, but communicating at every planning iteration can lead to congestion of the communication network, increase message delays and message loss. At the same time, collision-free trajectory planning is often formulated as a nonconvex optimization problem, which can converge to different locally optimal solutions across consecutive planning iterations. When this happens, a robots planned motion can switch between distinct high-level avoidance behaviors, such as passing an obstacle on the left versus on the right, which can lead to inefficient or unsafe behavior. Topology-based motion planners address this by explicitly computing multiple candidate motion plans that represent these different passing decisions, each associated with a distinct homotopy class. This work asks how (changes in) homotopy-class representations can trigger communication to reduce communication load while maintaining safe and efficient behavior. Building on the topology-driven trajectory optimization (T-MPC) approach of [1], we propose T-DMPC, a topology-aware distributed motion planner in which each robot, computes multiple guidance trajectories in distinct homotopy classes and refines them via parallel local trajectory optimization within the corresponding homotopy classes, selects a solution using a consistent decision rule that prioritizes the previously executed homotopy class, and communicates the selected trajectory using an event-triggered policy. Communication is triggered by homotopy changes and complemented by geometric-deviation and time-based triggers to bound trajectory staleness. In addition, the robots communicate a fallback trajectory during planning failures (e.g. infeasibility). We evaluate T-DMPC on antipodal swap maneuvers with 2 and 3 robots in simulation and on physical robots, comparing against T-VMPC (no communication, constant-velocity predictions) and T-AMPC (always communicate). The experiments show that, T-DMPC achieves task duration and traveled distance comparable to T-AMPC and T-VMPC, while reducing communication to about 9.8% (2 robots) and 13.2% (3 robots) of planning iterations in simulation, and 17.8% (2 robots) and 13.2% (3 robots) in real-world experiments, with no observed physical collisions. Ablations however show that topology-change alone is insufficient for safety, motivating the combined trigger design.

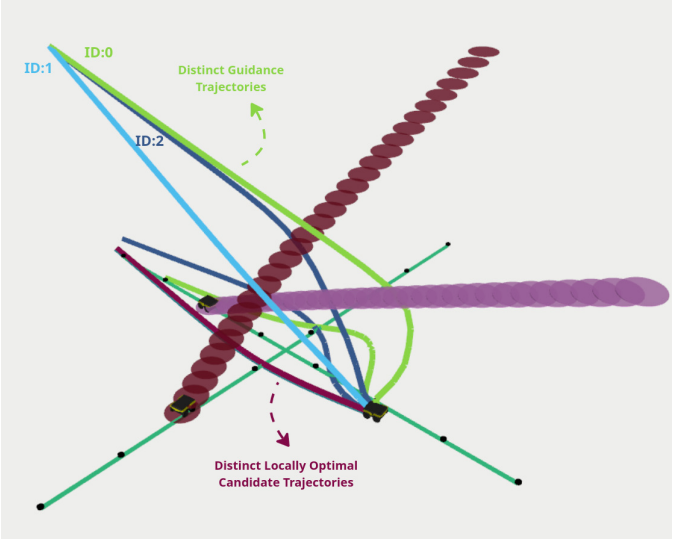


Fig. 1: Illustration of one planning iteration of T-DMPC from the perspective of Robot i. The guidance planner first generates multiple guidance trajectories (time visualized as increases upward), each belonging to a different homology class, denoted with ID: i. Each guidance trajectory initializes a (parallel) local optimization instance, producing a locally optimal candidate trajectory in the same homology class (shown **without** time upward), which move in different ways around obstacles(predicted obstacle motion from the other robots is visualized as cylinders). The candidate trajectory selected for execution in the current iteration (red) is then passed to the communication policy. The communication policy decides whether to communicate it to the other robots based on the homology class chosen in the previous iteration(h_i^{prev}) and the current iteration($h_i^{current}$).

I. Introduction

ROBOTS (and other autonomous systems) are increasingly integrated into daily life with applications in transportation, logistics, healthcare, hospitality and agriculture. Rather than operating in isolation, robots are increasingly deployed in settings, where multiple robots share environments and tasks. For example, small fleets of self-driving taxis operate in public traffic and autonomous robots move goods in large-scale warehouses. In such settings the main challenge is to have each robot reach their goal efficiently while avoiding collisions with other robots, humans and other obstacles. A way to approach this challenge is through the planning architecture. In *centralized planning*, a single computation unit computes trajectories for all robots, which can result in globally optimal plans but is limited by scalability, robustness and the need for complete system information. In *decentralized planning*, each robot plans trajectories independently and treats other robots as non-communicating moving obstacles, relying on local sensing and prediction to avoid them. This scales well but can lead to wrong beliefs and prediction errors. In *distributed planning*, robots still plan trajectories independently but can communicate information explicitly (e.g., intended trajectories). This can scale well and lead to less uncertainty about the intended motion of other robots when communication is fast and reliable. However, distributed approaches are also susceptible to communication delays, network congestion and packet loss. If these effects are not accounted for, coordination performance can degrade and lead to unsafe behavior and even collisions.

The three architectures above describe *how multi-robot planning can be organized*: whether trajectories for each robot are computed by a single computation unit, independently without communication or independently with explicit communication. However, these architectures do not specify *how* a motion planner computes safe and efficient trajectories(motion plans). Optimization-based motion planners are widely used to generate safe and efficient trajectories, but the underlying optimization problems are often nonconvex. As a result, the solution tends to converge to local optima. In order to solve this researchers have introduced concepts of topology into motion planning. *Topology-based motion planners* are of interest because they can compute multiple collision-avoidance trajectories that belong to different *homology classes*. Intuitively, trajectories belong to different homology classes if they pass obstacles in different topological ways (e.g. passing another robot on the left versus on the right). One can thus interpret a homology class as a *high-level interaction strategy* for navigating around other robots and obstacles. Topology-based motion planners obtain topologically distinct trajectories by reasoning about the topology of trajectories in relation to the collision-free space. While topological concepts have been used to compute multiple topologically distinct (high-level) motion plans, they are rarely used as an explicit coordination or communication mechanism in distributed multi-robot motion planning. This work proposes a topology-aware communication strategy in which robots communicate their predicted trajectories in each planning iteration only when a change in homology class occurs. A robot changing its homology class can be interpreted as a switch in its high-level interaction strategy(e.g. switching from passing an obstacle on the left to passing it on the right). By restricting communication to these topologically interesting events, the approach aims to reduce unnecessary communication traffic and reduce network congestion, while maintaining safe and efficient behavior.

Research question: *How can (a change in) homology class representations of trajectories be used to trigger (explicit) communication in distributed multi-robot motion planning, so as to reduce communication load while preserving safe and efficient coordination? Building on the recent topology-based trajectory optimization approach (T-MPC) presented in [1], this work develops and evaluates a topology-aware distributed planning framework (T-DMPC)(see Fig. 1). We quantitatively and empirically study its coordination and communication behavior under multi-robot interactions in both simulation and real-world experiments.*

II. Related Work

This section reviews prior work on (multi-robot) motion planning methods. It highlights key challenges in the distributed planning architecture, explains why the concept of topology is useful for optimization-based planners that have to deal with a nonconvex collision-free space and positions this thesis within the literature. The goal is not to list methods exhaustively, but to group common approaches by their main characteristics and to identify limitations that motivate this work.

Reactive methods. Reactive collision avoidance methods compute short-horizon motion plans from the current perceived state. Examples include artificial potential field methods [2] and velocity-obstacle methods [3], [4], with ORCA [5] as a widely used variant. These methods are computationally efficient, but they are short-sighted and can lead to oscillations and deadlocks in dense scenes. These methods are often executed within a decentralized coordination architecture. Other classical methods coordinate robots through a priority order (prioritized planning) [6]. Priority schemes can work well however however they often rely on a predetermined priority ordering, which is non-trivial and can result in overly conservative solutions.

Learning-based methods. Learning-based methods use learning to generate collision-free motion. Learning-based motion planners [7] can capture complex (interaction) patterns from data and can be efficient at run time once trained . However, they require representative training data, can generalize poorly outside the training distribution and typically provide limited interpretability and safety guarantees. These methods are also often executed within the decentralized coordination framework.

Optimization-based methods. A third group formulates motion planning as trajectory optimization [8], [9], [10], [11] , including Model Predictive Control (MPC) [12], [13], [14], [15], [16]. These planners can incorporate nonlinear dynamics, actuator limits and (hard) collision-avoidance constraints, while optimizing objectives such as tracking, smoothness and jerk. In multi-robot settings, the predicted motion of other robots is often included through collision avoidance constraints or reflected in the costs function. A difficulty is that collision avoidance constraints lead to a nonconvex collision-free space . As a result, the optimized solution may converge to different local optima depending on the initialization, can become temporarily infeasible and may oscillate between distinct local solutions across consecutive iterations. Consequently, high-level interaction decisions such as whether to pass another robot on the left or the right are often made implicitly rather than explicitly [1].

Centralized optimization methods. Some optimization-based planners formulate the optimization in a centralized way by formulating it as a joint optimization problem over all robots [17].

Distributed optimization methods with explicit trajectory exchange. Within optimization-based planning, a large body of work targets distributed settings in which robots plan locally but exchange information explicitly, for example by communicating their planned trajectories or state [12], [18], [19]. Compared to purely sensor prediction-based decentralized planning, explicit communication reduces reliance on onboard prediction via sensors. However, distributed methods must handle network effects such as delay, packet loss and congestion. When shared information becomes outdated due to these effects, robots may enforce collision-avoidance constraints against stale trajectories, which can degrade coordination performance and safety [20], [9], [21], [22]. This motivates communication policies that reduce communication load while still exchanging information when it matters.

Topology-based motion planning. Optimization-based (distributed) planners can, as already mentioned, converge to different local optima due to the nonconvex collision-free space induced by other robots and obstacles. Which local optimum is reached depends on the initial guess and the tuning of the cost function [1]. As a result, different initializations can produce different solutions that correspond to different high-level interaction strategies, such as passing another robot on the left versus on the right. The authors in [1] note that collision-avoidance related local optima are closely linked to the topology of trajectories through the collision-free space. Roughly speaking, two trajectories belong to the same *homotopy class* if one can be smoothly deformed into the other without intersecting obstacles [23] (they pass obstacles on the same side). However, computing homotopy classes of trajectories is difficult in general. Therefore to represent and distinguish homotopy classes of trajectories an approximation is used, called *homology classes*. Homology classes are often be calculated via the *H-signature* [23] and *winding numbers* [24]. Several authors exploit such topological representations in relation to optimization-based planning. A common pattern is to optimize over multiple candidate trajectories in parallel, each of which is in a different homotopy class and to select the best solution after optimization. For instance in [25], the authors use the H-signature to construct and optimize several trajectories online in distinct homotopy classes. Similarly in [26], the method applies the same idea in 3D by optimizing over multiple trajectories that are initialized from different topological paths through the collision-free space. More recently, topology has been integrated directly into MPC-based formulations. In [1], they run multiple MPC instances initialized in distinct homotopy classes and add linear topology constraints to keep each instance within its assigned class, enabling a consistent comparison across planning iterations. In an extension, [27], replaces the explicit cost-based selection with a learned selection rule. Instead of choosing the homotopy class based only on the optimized trajectory costs, the authors learn preferences from human data to bias the choice toward socially preferred interaction strategies. Parallel optimization over different (high-level) interaction strategies also appears in settings such as self driving cars. The authors in [28] initialize multiple MPC instances in parallel each in a different homotopy class (similar to [1]) but solve a *joint* optimization that includes other agents as decision variables, rather than treating them only through fixed predicted trajectories.

Other works use different topological concepts to encode and influence coordination outcomes. [29] incorporates a winding-number cost term into an MPC formulation to encourage passing progress and select the motion plan that realizes the maximum passing progress in relation to all obstacles. In a related direction, they use braid representations to describe multi-agent interaction strategies (e.g., ordering at intersections) and design a decentralized policy that reduces uncertainty over these modes to improve safety [30].

Finally, these topology-aware ideas have also been explored in distributed planners. [10] adapts the idea of optimizing multiple trajectories in distinct homotopy classes of [25] to a distributed trajectory-optimization setting and [22] similarly employs topological structure in a distributed multi-robot system.

Overall, topology-aware methods complement standard trajectory optimization methods by explicitly representing multiple distinct high-level motion strategies, by distinguishing trajectories via their topological relationship to obstacles and other robots. These methods expose high-level choices (e.g., pass left vs. pass right) that can be used to find diverse candidates, mitigate poor local minima and improve robustness in cluttered or crowded interactions.

Positioning of this work. This work builds on distributed optimization-based planning and uses homology classes to represent high-level interaction strategies of trajectories. Instead of communicating continuously or at a fixed rate, communication is triggered by changes in homology class, i.e., when the planner switches between distinct high-level interaction strategies. The aim is to reduce unnecessary communication while still communicating decisions that are most relevant during close interactions. The remainder of this work details the resulting planning framework and evaluates its behavior in simulation and real-world experiments.

A. Contributions

This work builds primarily on the method presented in [31] and [1] and extends the proposed motion planning framework to a distributed multi-robot setting. The main contributions of this work are as follows:

- An extension of the topology-driven motion planning method introduced in [1] to a distributed multi-robot system, leveraging explicit trajectory communication between robots.
- An extension of the homology classification framework to the non-guided motion planner, allowing its trajectories to be assigned to homology classes whenever possible using the same H-signature implementation as in [1]. This aims to increase consistency in high-level motion planning, even when the non-guided planner is selected.
- A novel topology-based communication strategy that reduces the amount of communication by triggering communication only upon changes in homology class.
- A trajectory consistency cost term is incorporated into the MPC formulation of [1], penalizing deviations from the previously computed trajectory within the same homology class and thereby encouraging the predictability of the distributed system. The cost is applied only to the MPC instance associated with the previously selected homology class. All other MPC instances exclude this term and remain unaffected by the consistency cost component.
- Evaluations of the proposed method in both simulation and real-world experiments.

III. Problem Formulation

This section defines the multi-robot motion planning problem studied in this work. It introduces the assumptions used throughout the remainder of the chapter, the objective and the workspace setting.

Terminology convention. Throughout this work, the term *robot* refers to a communicating robot that participates in explicit trajectory exchange. The term *Agent* refers to a non-communicating dynamic object whose motion is observed and predicted locally. The term *dynamic obstacle* refers to both robots and agents. The term *obstacle* is used as an umbrella term that includes dynamic obstacles and, when stated explicitly static obstacles.

A. Assumptions

The problem formulation uses the following assumptions:

- Each robot can estimate its own position and velocity.
- Each robot runs the same motion planning algorithm locally.
- Each robot can estimate the current position and velocity of agents.
- Communicating robots can exchange predicted trajectories with all other robots.
- All robots start at the same initial time, and each Robot has a given start state and a fixed reference path.

B. Problem formulation

We consider a team of communicating robots that move in a shared workspace containing static obstacles and dynamic obstacles. We distinguish two classes of dynamic obstacles: (i) *communicating* dynamic obstacles, which correspond to other robots that exchange predicted trajectories, and (ii) *non-communicating* dynamic obstacles, which include agents such as pedestrians, non-communicating robots, or other unknown dynamic objects. Each robot is assigned a fixed reference path from a given start position to a goal position. The objective of each robot is to make progress along its reference path while respecting a maximum reference speed. Each robot must avoid collisions with static obstacles, non-communicating dynamic obstacles, and other robots. An illustration of this problem is shown in Fig. 1.

Formal problem formulation. We consider a team of R communicating robots indexed by $I_R := \{1, \dots, R\}$. For a Robot $i \in I_R$, we denote any other communicating Robot by $j \in I_{R\setminus i} := I_R \setminus \{i\}$. In addition, we consider A non-communicating dynamic obstacles, also referred to as *agents*, indexed by $I_A := \{1, \dots, A\}$. Then the set of all dynamic obstacles considered by Robot i is defined by $I_{O\setminus i} := I_{R\setminus i} \cup I_A$, and we denote a generic dynamic obstacle by $o \in I_{O\setminus i}$. Each Robot $i \in I_R$ is described by a discrete-time nonlinear dynamical system

$$x_i(t_{k+1}) = f\big(x_i(t_k), u_i(t_k)\big), \quad \forall i \in I_R. \tag{1}$$

where $x_i(t_k) \in \mathbb{R}^{n_x}$ and $u_i(t_k) \in \mathbb{R}^{n_u}$ denote the state and control input at discrete time step k, respectively, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ denotes the (possibly nonlinear) dynamics. Planning is performed at the current planning time t_c over a horizon of length N with sampling time dt, which defines the *time-grid*

$$t_k := t_c + k\,dt, \quad k = 0, \dots, N, \qquad t_N := N dt.$$

For collision checking, Robot i is modeled as a disc of radius $r_i > 0$ centered at its 2D position $\mathbf{p}_i(t_k) \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$. Communicating Robot j is modeled as a disc of radius $r_j > 0$ centered at $\mathbf{p}_j(t_k) \in \mathbb{R}^2$, and Agent a as a disc of radius $r_a > 0$ centered at $\mathbf{p}_a(t_k) \in \mathbb{R}^2$. Robot i maintains predictions of all dynamic obstacles over the time-grid $\{t_k\}_{k=0}^N$ at each planning time t_c. For each Robot j, Robot i receives a predicted 2D position trajectory through explicit communication. These received predictions are denoted by $\{\hat{\mathbf{p}}^{\,j\to i}(t_k)\}_{k=0}^N$. Since robots do not communicate their trajectories at every planning iteration, the received predictions may be *time-misaligned* with the current planning time t_c of Robot i. Therefore, each received trajectory of a generic Robot j is shifted forward in time based on its last *update time* t_u^j to obtain a prediction that is consistent with the current time-grid. We denote the resulting *time-aligned prediction trajectory* of Robot j, available to Robot i at planning time t_c, by $\hat{\tau}^{\,j\to i}(t_u^j, t_c) := \big\{\hat{\mathbf{p}}^{\,j\to i}(t_k)\big\}_{k=0}^N$ (Section IV-A, for a detailed explanation).

The collection of all time-aligned trajectory predictions received by Robot i at planning time t_c is defined as *the (updated communicating) robot prediction set*:

$$\mathcal{P}_i(t_c) := \{\hat{\tau}^{\,j\to i}(t_u^j, t_c)\} \qquad \forall j \in I_{R\setminus i}. \tag{2}$$

In addition to the explicitly communicated Robot trajectories, Robot i locally predicts the motion of *Agent* a in the environment using a prediction module. The resulting predicted positions of these agents are denoted by $\{\hat{\mathbf{p}}^{\,a\to i}(t_k)\}_{k=0}^N$ and the respective predicted trajectory is denoted by $\hat{\tau}^{\,a\to i}(t_c) := \big\{\hat{\mathbf{p}}^{\,a\to i}(t_k)\big\}_{k=0}^N$. The collection of all agent trajectory predictions is defined as *the agent prediction set*:

$$\mathcal{S}_i(t_c) := \{\hat{\tau}^{\,a\to i}(t_c)\} \qquad \forall a \in I_A. \tag{3}$$

These predictions define a time-varying set of *dynamic obstacles* for Robot i over the horizon: at each planning time t_c, Robot i considers (i) the communicated robot predictions in $\mathcal{P}_i(t_c)$ and (ii) the locally predicted agent motion in $\mathcal{S}_i(t_c)$. Each element corresponds to the predicted motion of a single obstacle (either a communicating robot or a locally predicted agent). We denote the time-indexed prediction trajectory of a generic obstacle $o \in I_{O\setminus i}$ available to Robot i over the planning horizon by $\hat{\tau}^{\,o\to i}(t_c)$, which contains the predicted obstacle positions $\hat{\mathbf{p}}^{\,o\to i}(t_k)$ for $k = 0, \dots, N$. Dynamic obstacles make the collision-free workspace time-dependent. We therefore consider the finite space-time domain $\mathcal{X} := \mathbb{R}^2 \times [t_0, t_N]$. For Robot i, let $\mathcal{O}_i(t_k) \subset \mathbb{R}^2$ denote the area of the workspace occupied by the union of all obstacles for Robot i at time t_k (induced by the predicted obstacle positions at time t_k). The corresponding *dynamic obstacle prediction set* in space-time is defined by

$$\mathcal{O}_i(t_c) := \bigcup_{t_k \in [t_0, t_N]} \big(\mathcal{O}_i(t_k),\, t_k\big) \subset \mathcal{X}, \tag{4}$$

and the collision-free space-time for Robot i is $\mathcal{C}_i := \mathcal{X} \setminus \mathcal{O}_i(t_c)$. The planning objective of each robot is to make progress along a given reference path $\gamma_i : [0, 1] \to \mathbb{R}^2$ while avoiding collisions with obstacles and tracking a reference velocity v_{ref}. Deviations from the reference path are allowed.

C. Optimization

For a Robot i, to reach the planning objective we define the problem as a trajectory optimization problem over a horizon of N steps:

$$\min_{\mathbf{u}_i \in \mathbb{U},\, \mathbf{x}_i \in \mathbb{X}} \quad \sum_{k=0}^{N} J(\mathbf{x}_i(t_k), \mathbf{u}_i(t_k)) \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_i(t_{k+1}) = f(\mathbf{x}_i(t_k), \mathbf{u}_i(t_k)), \quad \forall k \tag{5b}$$

$$\mathbf{x}_i(t_0) = \mathbf{x}_{\text{init}} \tag{5c}$$

$$g\big(\mathbf{x}_i(t_k), \hat{\mathbf{p}}^{\,o\to i}(t_k)\big) \le 0, \quad \forall k,\, o \tag{5d}$$

where the cost function J in Eq. (5a) encodes the planning objectives (e.g., tracking a reference path). The system dynamics and initial condition are enforced by Eq. (5b) and Eq. (5c), respectively, while collision avoidance with dynamic obstacles is imposed through Eq. (5d). Since dynamic obstacles make the collision-free space nonconvex, the optimization problem can lead to multiple locally optimal solutions. The particular solution obtained depends on the initialization of Eq. (5).

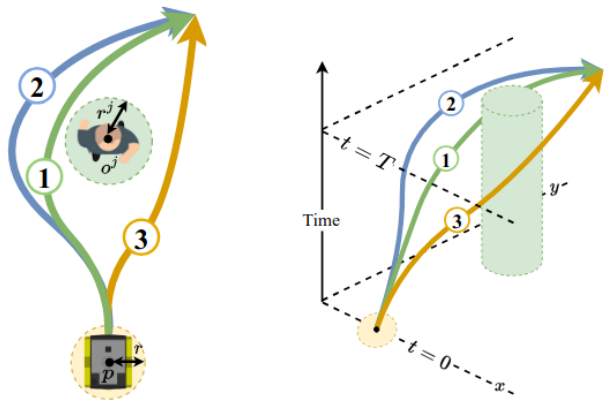
D. Homotopic Trajectories

The goal is to address the multiple local optimal solutions in Section III-C, which arise from the nonconvex collision-free space induced by obstacles. We exploit that these local optima are closely linked to the different ways how the resulting (optimal) trajectories pass obstacles. This can be captured by the topology of trajectories in space-time. By exploring multiple such locally optimal trajectories in parallel, each representing a different passing mode, it is possible to find a solution that is closer to the globally optimal trajectory than what would be obtained from a single initialization. We therefore rely on the concept of *homotopic trajectories* to find and compare candidate (initialization) trajectories. For a more detailed and theoretical explanation of homotopy-based trajectory classification the reader is referred to [1]. Here we provide a compact description relevant to this work.

To capture fundamentally different ways of navigating around obstacles, we rely on the notion of *homotopic trajectories*. Two trajectories that connect the same start and goal positions are said to be homotopic if one can be continuously deformed into the other while remaining entirely within the collision-free space. Intuitively, homotopic trajectories avoid obstacles in the same way and therefore belong to the same homotopy class. An illustrative example of different homotopy classes is shown in Fig. 2. To distinguish between trajectories belonging to different homotopy classes, we use a homotopy comparison function:

$$\mathcal{H}(\tau_i^m, \tau_i^n, \mathcal{O}_i) = \begin{cases} 1, & \tau_i^m, \tau_i^n \text{ same homotopy class} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Directly verifying whether two trajectories are homotopic according to the formal definition is, in general, computationally expensive and therefore impractical for online planning. Instead, we make use of the *H-signature* [1], which provides the efficient approximate representation called the homology class of the homotopy class of a trajectory and enables real-time homotopy comparison.



(a) (b)

Fig. 2: Illustration of trajectories in the same versus different homotopy classes: (a) top-down view and (b) space–time domain. Trajectories 1 and 2 share a homotopy class, whereas Trajectories 1 and 3 belong to different classes. Reproduced from [1], *licensed under CC BY 4.0.*

IV. Topology-Aware Distributed Model Predictive Control(T-DMPC) - Methodology

This section describes the proposed topology-aware distributed motion planning framework which we call T-DMPC. We first present the overall per-robot planning loop and a general description of the main modules, we then provide a more detailed description of each module.

Per-robot planning framework. We consider a team of robots indexed by $i \in I_R$. Each Robot runs the same planner locally and coordinates with other robots through explicit communication. Planning is performed asynchronously, such that each Robot executes its planning loop independently, without requiring synchronization with other robots. At the beginning of each planning iteration, Robot i updates its estimate of the environment and incorporates the most recent communicated trajectories of the other robots (Section IV-A). It then computes multiple candidate trajectories corresponding to distinct topological alternatives (Section IV-B) following [31], [1], refines them through local trajectory optimization Section (Section IV-C), selects one trajectory for execution (Section IV-D), and optionally communicates the selected trajectory to the other robots based on a communication policy Section (Section IV-E). An overview of the proposed planning framework can be seen in Fig. 3.

Module 1 - Dynamic obstacle representation. Robot i maintains a prediction trajectory of dynamic obstacles over a planning horizon N. These trajectories are updated in each planning iteration using a *constant velocity model* for the non-communicating dynamic obstacles or with the communicated trajectory of the other robots. Robots do not broadcast a new trajectory at every planning iteration. Therefore, when the most recently received trajectory from a Robot j is timestamped in the past, Robot i time-shifts it to the current planning time t_c. The output of this module is a set of prediction trajectories of all dynamic obstacles $\mathcal{O}_i(t_c)$, used by both the guidance and the local planner.

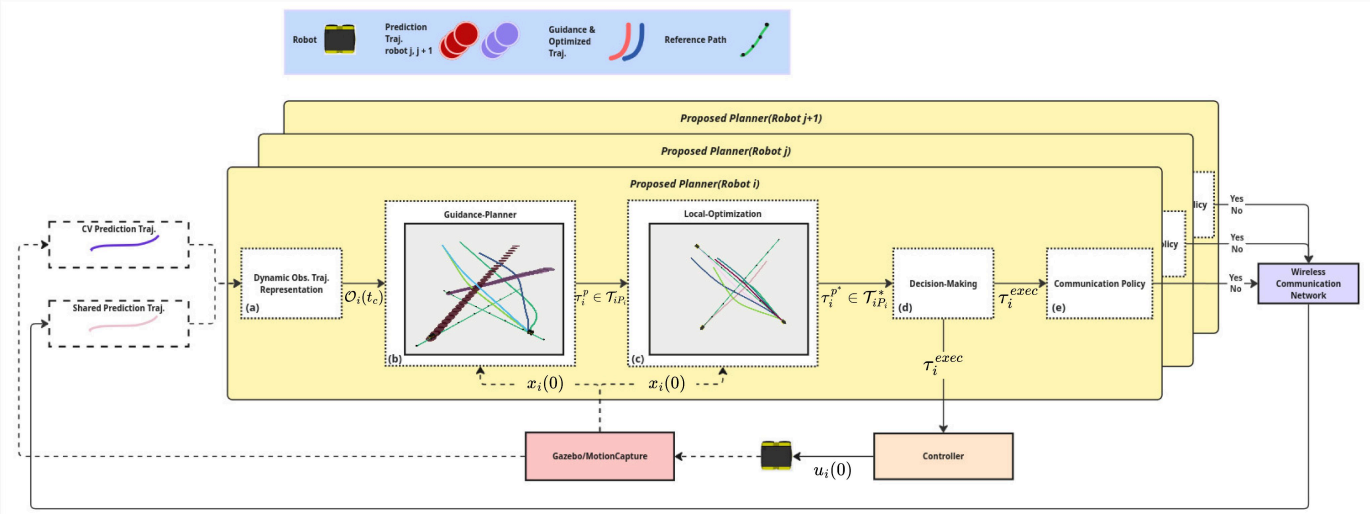


Fig. 3: Illustration of the complete motion planning framework of the proposed method T-DMPC. Each Robot i runs the same planner locally and replans asynchronously, without synchronization. At each iteration, (a) Robot i updates its environment estimate and incorporates the most recent communicated trajectories of the other robots$j$(Section IV-A), (b) generates multiple candidate trajectories corresponding to distinct topological alternatives (Section IV-B), (c) refines them via local trajectory optimization (Section IV-C), (d) selects one trajectory for execution (Section IV-D),and (e) optionally communicates it to the other robots based on a communication policy (Section IV-E)

Module 2 - Guidance planner. Given the current state $x_i(t_0)$, $\mathcal{P}_i^g$, which denotes a set of goal positions, and the current collision free space $\mathcal{C}_i$, the guidance planner G_i generates a set of topologically distinct guidance trajectories for Robot i:

$$G_i(x_i(t_0), \mathcal{P}_i^g, \mathcal{C}_i) = \{\boldsymbol{\tau}_i^1, \ldots, \boldsymbol{\tau}_i^{P_i}\} =: \mathcal{T}_{iP_i} \tag{7}$$

Here, $\boldsymbol{\tau}_i^p$ denotes the p-th guidance trajectory for Robot i, and P_i is the number of guidance candidates found for Robot i in the current plannings iteration (which may differ across robots and over time). In addition each guidance trajectory $\boldsymbol{\tau}_i^p$ is associated with a homology class label h_i^p.

Module 3 - Local trajectory optimization (one instance per guidance trajectory). For each guidance trajectory $\boldsymbol{\tau}_i^p$, Robot i initializes a local planner to obtain a locally optimized trajectory $\boldsymbol{\tau}_i^{p^*}$ within the same homology class. We denote this as a mapping:

$$L_i : \ \mathcal{X}^N \to \mathcal{X}^N, \qquad \boldsymbol{\tau}_i^{p^*} = L_i(\boldsymbol{\tau}_i^p) \tag{8}$$

The optimization accounts for (i) Robot dynamics and actuation limits and collision avoidance with dynamic obstacles. To ensure that each local planner instance remains within the intended homology class, the local planners also incorporate additional constraints from its assigned guidance trajectory $\boldsymbol{\tau}_i^p$. As a result, Robot i computes a set of locally optimal candidate trajectories $\mathcal{T}_{iP_i}^* := \{\boldsymbol{\tau}_i^{1^*}, \ldots, \boldsymbol{\tau}_i^{P_i^*}\}$, each trajectory $\boldsymbol{\tau}_i^{p^*}$ corresponding to a distinct topological alternative.

Module 4 - Decision making (trajectory selection). Robot i selects one trajectory from $\mathcal{T}_{iP_i}^*$ for execution based on comparing the cost of each trajectory. Denoting the selected index by p^*:

$$\boldsymbol{\tau}_i^{\text{exec}} := \boldsymbol{\tau}_i^{p^*}, \qquad p^* = \arg\min_{p^*} J_i(\boldsymbol{\tau}_i^{p^*})$$

Here $J_i(\cdot)$ is the cost function used by Robot i as defined in Section III-B.

Module 5 - Communication policy. A communication policy determines whether Robot i communicates the execution trajectory $\tau_i^{\text{exec}}(t_c)$ to the other robots. We introduce a discrete communication trigger variable:

$$\kappa_i \in \mathcal{K} \subset \mathbb{Z}^+ \tag{9}$$

where $\mathcal{K}$ is a finite set of *trigger states* encoding the reason for a communication decision. The trigger variable resembles high-level events detected by Robot i(e.g., feasibility changes, topological changes, or timing conditions). Based on the trigger state, a binary communication decision is taken:

$$\delta_i(\kappa_i) \in \{0, 1\}, \tag{10}$$

where $\delta_i(\kappa_i) = 1$ indicates that Robot i communicates its selected trajectory $\tau_i^{\text{exec}}(t_c)$. If $\delta_i(\kappa_c) = 0$, Robot i executes $\tau_i^{\text{exec}}(t_c)$ without communicating $\tau_i^{\text{exec}}(t_c)$ to the other robots.

A. Dynamic Obstacle Representation - Detailed Description

This section details how Robot i maintains a consistent dynamic obstacle prediction set $\mathcal{O}_i(t_c)$ in the presence of communicating robots and non-communicating agents. As defined in Section III-B, Robot i evaluates prediction trajectories of robots and agents on the time-grid $\{t_k\}_{k=0}^N$ at each planning time t_c.

Communicated robot trajectories. For each other communicating Robot $j \in I_{R\setminus i}$, Robot i receives a prediction trajectory at irregular times, and consecutive messages may be separated by multiple planning iterations (e.g., when Robot j does not broadcast because its intended behavior remains unchanged). We denote the most recently received trajectory from Robot j by $\hat{\tau}^{j\to i}(t_r)$, where t_r^j is the time at which Robot i received

the trajectory. Additionally t_u^j, where $t_r^j \leq t_u^j \leq t_c$ is the time at which Robot i last updated the prediction trajectory of Robot j. The *prediction age* is:

$$\Delta t := t_c - t_u^j \geq 0. \tag{11}$$

If $\Delta t > 0$, Robot i time-shifts the received trajectory forward by Δt such that it can be used on the current time-grid. The resulting *Robot prediction set* $\mathcal{P}_i(t_c)$ on the current time-grid is defined by Eq. (2).

Non-communicating dynamic obstacles. As mentioned in Section Section III-B, Robot i does not receive prediction trajectories for agents $a \in I_A$. Instead, at each planning iteration Robot i propagates the current estimated agent state forward over the horizon N using a *constant-velocity model*. The resulting *agent prediction set* $\mathcal{S}_i(t_c)$ on the current time-grid is defined by Eq. (3).

Dynamic obstacles. The combined dynamic obstacle prediction set, $\mathcal{O}_i(t_c)$ of Robot i at planning time t_c is defined by Eq. (4). This dynamic obstacle set is used by both the guidance layer and the local trajectory optimization layer.

Alignment of Robot i's last communicated traj. Finally, Robot i also time-aligns its own last communicated trajectory to the current time-grid by propagating it forward, this is denoted by $\tau_i^{\text{comm}}(t_u^i, t_c)$. Here t_u^i is the time at which Robot i last updated its own communicated trajectory (so it is **not** the time when it communicated the trajectory). This trajectory represents what the other robots currently believe Robot i will execute until it communicates a new trajectory. After solving the new optimization, Robot i compares the newly selected trajectory $\boldsymbol{\tau}^{\text{exec}}(t_c)$ against this time-aligned last communicated trajectory to check whether the planned motion deviates significantly from what other robots expect. This will be explained in more detail in Section IV-A.

B. The Guidance Planner - Detailed Description

This section details how Robot i generates a set of topology-distinct guidance trajectories that serve as initializations for the local trajectory optimization layer. The guidance planner operates on the time dependent collision free space $\mathcal{C}_i$ induced by the dynamic obstacle prediction set $\mathcal{O}_i(t_c)$(Section IV-A) and follows the Visibility-PRM-based approach of [31], [1]. We summarize the essential steps here and refer to [31], [1] for full implementation details.

Visibility-PRM. At planning time t_c, Robot i constructs (or updates) a sparse roadmap $\mathcal{G}_i$ in the collision-free space $\mathcal{C}_i$. The roadmap is built in space-time and connects the intial Robot state $x_i(t_0)$ to a set of goal nodes $x_i(t_N)$ by randomly sampling positions. These goals are placed around the reference path near the end of the planning horizon. A set of goals nodes is used instead of a single goal to increase robustness when a single goal becomes unreachable.

DepthFirstSearch and FilterAndSelect. Given the roadmap $\mathcal{G}_i$, Robot i extracts candidate paths to each goal node using depth-first search, yielding a set $\mathcal{T}_{iF_i}$ of feasible paths. The procedure *FilterAndSelect* then removes homotopy-equivalent candidates using a homotopy comparison function $\mathcal{H}(\cdot)$ and returns a set of filtered guidance trajectories $\mathcal{T}_{iF_i}$ that satisfies

$$\mathcal{H}(\tau_i^m, \tau_i^n, \mathcal{O}_i) = 0,\ \forall m, n,\ m \neq n,\ \tau_i^m, \tau_i^n \in \mathcal{T}_{iF_i} \tag{12}$$

The P_i lowest-cost trajectories in $\mathcal{T}_{iF_i}$ make up the selected guidance set $\mathcal{T}_{iP_i}$. In this work, $\mathcal{H}(\cdot)$ is implemented *only* using the H-signature Eq. (23). We refer to [1] for the precise definition and implementation details.

Re-identification and propagation across iterations. To maintain behavioral consistency across planning iterations, the guidance planner links new trajectories to trajectories from the previous planning iteration. Let $\mathcal{T}_{iP_i}^-$ denote the set of previously found guidance trajectories. For each $\tau_i^{m-} \in \mathcal{T}_{iP_i}^-$, we check whether there exists a trajectory $\tau_i^n \in \mathcal{T}_{iP_i}$ such that:

$$\exists \tau_i^n \in \mathcal{T}_{iP_i},\ \mathcal{H}(\tau_i^{m-}, \tau_i^n) = 1. \tag{13}$$

If such a match exists, the homology class of τ_i^{m-} is propagated to τ_i^n. This enables us to refer to consistent topological alternatives over time.

C. Local planner - Detailed Description

This section details how Robot i refines the set of guidance trajectories by initializing multiple local trajectory optimizers each with its own guidance trajectory in parallel. Each local planner operates in the time dependent collision free space $\mathcal{C}_i$ induced by the dynamic obstacle set $\mathcal{O}_i(t_c)$. The overall procedure follows the approach of [1], with several adjustments.

To refine the guidance trajectories generated by the guidance planner of Robot i, we initialize P_i local planners in parallel. Each local planner optimizes one guidance trajectory τ_i^p and produces a dynamically feasible trajectory that satisfies all imposed constraints (e.g., dynamics, actuation limits, and collision avoidance). In this work, each local planner is instantiated as a trajectory optimization problem Eq. (5) with two modifications to ensure that the optimized solution remains in the same homotopy class as its associated guidance trajectory. First, the guidance trajectory τ_i^p is used as the initial guess. Second, we augment the optimization with additional topology-preserving constraints that keep the optimized trajectory in the same homology class as indicated by the guidance trajectory. To keep each optimized trajectory in the same homology class as its associated guidance trajectory, we augment every local planner with an additional set of linear topology constraints To keep each optimized trajectory in the same homology class as its associated guidance trajectory, we augment every local planner with linear topology-preserving constraints $g_H\big(\mathbf{x_i}(t_k), \hat{\mathbf{p}}^{\,o\to i}(t_k), \mathbf{p}_i^p(t_k)\big)$. Here, $\mathbf{p}_i^p(t_k)$ denotes the 2D position of the *guidance trajectory* associated with guidance candidate p at time t_k, and should not be confused with the position which is part of $\mathbf{x_i}(t_k)$. Following [1], these constraints are written as $\mathbf{A}(t_k)\,\mathbf{x}_i(t_k) \leq \mathbf{b}(t_k)$. For each time index $k = 0, \ldots, N$ and each predicted dynamic obstacle position $\hat{\mathbf{p}}^{o\to i}(t_k)$ (from $\mathcal{O}_i(t_c)$), we construct a separating half-plane with respect to the guidance trajectory τ_i^p:

$$\begin{aligned} \mathbf{A}(t_k) &= \frac{\hat{\mathbf{p}}^{o\to i}(t_k) - \mathbf{p}_i^p(t_k)}{\|\hat{\mathbf{p}}^{o\to i}(t_k) - \mathbf{p}_i^p(t_k)\|}, \\ \mathbf{b}(t_k) &= \mathbf{A}(t_k)^\top \big(\hat{\mathbf{p}}^{o\to i}(t_k) - \mathbf{A}(t_k)(\beta\,(r_i + r_o))\big)\,. \end{aligned} \tag{14}$$

Here $\beta \approx 0$ is a relaxation factor. These constraints prevent the optimizer from switching sides relative to obstacles while

remaining compatible with standard collision-avoidance constraints. We refer to [1] for a more detailed derivation and discussion.

The resulting local planner for guidance candidate p can be written as:

$$
\begin{aligned}
J_i^{p^*} = \min_{u_i,\, x_i} \; & \sum_{k=0}^{N} J\big(\mathbf{x_i}(t_k), \mathbf{u_i}(t_k)\big) && \text{(15a)}\\
\text{s.t.} \quad & \mathbf{x_i}(t_{k+1}) = f\big(\mathbf{x_i}(t_k), u_i(t_k)\big), \quad \forall k, && \text{(15b)}\\
& \mathbf{x_i}(t_0) = \mathbf{x}_{init}, && \text{(15c)}\\
& g\big(\mathbf{x_i}(t_k), \hat{\mathbf{p}}^{o\to i}(t_k)\big) \le 0, \quad \forall k, o, && \text{(15d)}\\
& g_H\big(\mathbf{x_i}(t_k), \hat{\mathbf{p}}^{o\to i}(t_k), \mathbf{p}_i^p(t_k)\big) \le 0 \quad \forall k, o && \text{(15e)}
\end{aligned}
$$

where $g(\cdot)$ denotes the collision-avoidance constraints[1] and $g_H(\cdot)$ encodes the topology-preserving constraints constructed from the guidance trajectory as in Eq. (14). The P_i local planners yield a set of locally optimal candidate trajectories, which are passed to the decision-making layer for selection.

Trajectory consistency cost. In addition to the objective function defined in Eq. (15a), which follows the formulation of [1], we introduce an additional *trajectory consistency cost*. This term encourages spatial-temporal consistency when the planner continues to execute trajectories within the same homology class across consecutive planning iterations. Specifically, if Robot i selected a guided trajectory in homology class h_i^{prev} at the previous planning time, the corresponding guided local planner is penalized for deviating from the previously executed trajectory. Let $\tau_i^{\text{prev}} := \{\mathbf{p}_i^{\text{prev}}(t_k)\}_{k=0}^{N}$ denote the time-aligned trajectory executed by Robot i in the previous planning iteration. The additional cost term is defined as:

$$
J_{\text{prev}} = w_\tau \sum_{k=1}^{N-1} \left\| \mathbf{p}_i(t_k) - \mathbf{p}_i^{\text{prev}}(t_k) \right\|^2, \qquad \text{(16)}
$$

where $\mathbf{p}_i(t_k) \subset \mathbf{x_i}(t_k)$ denotes the position decision variable of the current local optimization at time step t_k and $w_\tau \ge 0$ is a weighting parameter. This cost is applied only to the guided local planner associated with the homology class selected in the previous iteration. For all other guided planners, the consistency weight is set to $w_\tau = 0$, making Eq. (16) inactive.

Non-guided Local Planner in Parallel. While the topology-preserving constraints Eq. (14) enforce consistency with the guidance trajectories, they also restrict the feasible set of the local optimization problem. As a result, there may be situations in which the local optimization becomes suboptimal or even infeasible. In such cases, the guidance constraints themselves can thus limit the performance of the local planner. To mitigate this effect, we augment the set of P_i (guided) local planners with an additional *non-guided* local planner. This planner solves the same trajectory optimization problem as in Eq. (15), but without the topology-preserving constraints Eq. (15e) and without relying on a guidance trajectory as initialization. As such, the non-guided planner is less restricted and can discover solutions with lower costs than the guided local planners. In addition to the approach of [1], we also attempt to associate the trajectory produced by the non-guided local planner with one of the P_i homology classes identified by the guidance module. To this end, the non-guided trajectory τ_i^{NG} is projected onto the guidance roadmap $\mathcal{G}i$, after which its homology class is determined by evaluating the mapping in Eq. (12) for all $\tau_i^p \in \mathcal{T}_{iP_i}$. An illustration of this mapping is shown in Fig. 4. The inclusion of a non-guided local planner provides two key benefits. First, it improves robustness by allowing the guided planning architecture to recover from infeasibility. Second, it guarantees that the proposed parallel planning architecture does not perform worse than the non-guided local planner in isolation. When a matching homology class is identified for the non-guided trajectory, the trajectory is assigned to the corresponding homology class and included in the same decision-making and communication logic as the guided candidates associated with that class.

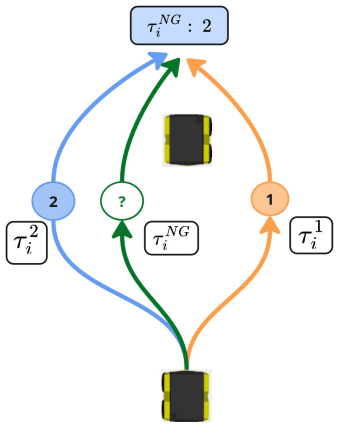


Fig. 4: Illustration of how the optimized trajectory τ_i^{NG} produced by the non-guided local planner is assigned a homology label. The trajectory τ_i^{NG} (green) is initially unlabeled. Using Eq. (12), it is matched to homology ID 2, corresponding to the guided optimized trajectory $\boldsymbol{\tau}_i^2$.

D. Decision Making Layer - Detailed Description

This section details how Robot i decided on which one of the $\boldsymbol{\tau}_i^{p^*}$ optimized trajectories from the set of optimized trajectories $\mathcal{T}_{iP_i}^*$ to execute in the current planning iteration.

Since Robot i can execute only a single trajectory at each planning iteration, the set of candidate solutions must be reduced to a single execution decision. The set of optimized trajectories produced by the P_i guided local planners and the non-guided local planner, with corresponding optimal costs $J_i^{p^*}$ is:

$$
\mathcal{T}_{iP_i}^* := \{\boldsymbol{\tau}_i^{1^*}, \ldots, \boldsymbol{\tau}_i^{P_i^*}\} \qquad \text{(17)}
$$

For the guided local planner associated with the homology class selected in the previous planning iteration, the local optimization includes an additional trajectory consistency J_{prev}(Eq. (15a)). To ensure a fair comparison between candidate trajectories during decision making, this consistency term is removed prior to selection.

Since all local planners minimize the same nominal objective function, the trajectory with the lowest cost $J_i^{p^*}$ represents the

[1]A compact formulation of Eqs. (15) and (15d) is provided in Section VIII-B.

best solution under the specified objective. The minimal-cost decision is therefore given by:

$$\boldsymbol{\tau}_i^{\text{exec}} := \boldsymbol{\tau}_i^{p^*}, \quad p^* = \arg\min_{p^*} \; J_i^{p^*}, \tag{18}$$

As described in [1], frequent switching between topologically distinct trajectories across consecutive planning iterations can degrade navigation performance. Following their approach, we apply a consistency rule that prefers the trajectory in the same homology class as the one executed in the previous iteration. This consistent decision is defined by:

$$\tau_i^{p^*}, \quad p^* = \arg\min_{p} \; w_p J_i^{p^*}, \tag{19}$$

Here, $w_p = c_p \in [0, 1]$, if the corresponding homology class was selected in the previous iteration, provided that a feasible trajectory in that class exists. Intermediate values of c_p trade off minimization of the "normal" trajectory cost against topological consistency across consecutive iterations.

E. Communication Policy - Detailed Description

This section details how Robot i decides whether to broadcast $\tau_i^{\text{exec}}(t_c)$, based on the current value of the trigger variable $\kappa_i(t_c)$.

After the decision-making layer selects a single execution trajectory $\tau_i^{\text{exec}}(t_c)$ at planning time t_c, Robot i decides whether this trajectory should be communicated to other robots. Since planning is asynchronous and communication is event-triggered, other robots may base their predictions on a previously received plan for multiple planning iterations. The communication policy therefore aims to (i) limit bandwidth usage, while trying to (ii) prevent other robots from relying on stale predictions of Robot i's motion. When communication is triggered meaning $\delta_i(\kappa_i(t_c)) = 1$, Robot i communicates the selected execution trajectory τ_i^{exec}. The trigger variable $\kappa_i(t_c)$ is evaluated in a fixed priority order (highest priority first). The following triggers are used:

1) *Infeasibility*. If none of the $P_i + 1$ local trajectory optimizations results in a feasible solution, then other robots must be informed to avoid relying on an outdated prediction of Robot i.
2) *Non-guided / unmapped* . If the selected execution trajectory cannot be assigned to any homology class, then the other robots must be informed because a topology switch might happend. This can happen when the chosen trajectory is the non-guided trajectory but the mapping failed.
3) *Topology-change*. Robot i triggers communication if the homology label of the selected execution trajectory differs from the one executed in the previous planning iteration, i.e., $h_i^{prev} \neq h_i^{exec}$.
4) *Geometric-deviation* . Even if topology remains unchanged, the executed trajectory may deviate significantly from what other robots currently believe, this can result in a collision. Robot i therefore triggers communication if the deviation between $\tau_i^{\text{exec}}(t_c)$ and the time-aligned last communicated plan exceeds a threshold $\epsilon > 0$:

$$D\big(\tau_i^{\text{exec}}(t_c), \tau_i^{\text{comm}}(t_u^i, t_c)\big) > \epsilon \tag{20}$$

In this work we have chosen for:

$$D(\tau, \bar{\tau}) := \max_{k \in \{0,\dots,N\}} \|\mathbf{p}(t_k) - \bar{\mathbf{p}}(t_k)\|. \tag{21}$$

We use the maximum position deviation over the horizon N, because it captures the worst-case discrepancy between what other robots enforce as collision-avoidance constraints (based on $\hat{\tau}_i^{\text{comm}}(t_i^{\text{last}}, t_c)$) and what Robot i is actually executing. In other words, even if most of the trajectory matches, a single large deviation at any horizon index can invalidate another Robot j's prediction and potentially lead to a collision.

5) *Time-based trigger*. To avoid unbounded silence and to reduce uncertainty in the true trajectory Robot i is following, Robot i also communicates a periodic "heartbeat" update: if no other trigger fires for ΔT_{hb} seconds, it transmits its current trajectory. Here t_i^{last} denotes the time at which Robot i last communicated a trajectory.

$$t_c - t_i^{last} \geq T_{hb} \tag{22}$$

If none of the above conditions hold, $\delta_i(\kappa_i) = 0$ and Robot i executes $\tau_i^{\text{exec}}(t_c)$ without transmitting.

The resulting (event-triggered) communication policy communicates irregularly during "steady-state" motion (unchanged topology and small deviation from the last communicated plan). It reacts immediately to interaction-critical events, such as infeasibility, non-guided/undetermined topological plans, or large deviations from the belief trajectory. The time-based trigger can refresh a Robot j's predictions even when no relevant events occurred.

V. Simulation Experiments

This section describes the simulation setup and evaluation metrics used to compare the proposed topology-aware distributed motion planner (T-DMPC) against baseline methods in multi-robot scenarios. It then presents and analyzes both the overall results and the behavior of the communication triggers, and includes an ablation study to isolate the impact of each individual component of the proposed method.

A. Planner Implementation and Parameters

The planner is implemented in C++/ROS1. For the optimization-based planner we use Local Model Predictive Contouring Control (LMPCC) [32][2] with second-order unicycle dynamics. The parallel local optimization problems are solved using acados [33]. For more details on the LMPCC objective function and collision avoidance constraints we refer the reader to [32], [1]. All planner parameters are summarized in Table I. The guidance and local-planner weights are the same once used in [1]. The communication-policy parameters are manually tuned on a small set of pilot runs until no collisions occured, and then held fixed across all experiments (they are not optimally tuned to trade off safety against the amount of communication; this is left as future work). All simulations (and real-world) experiments are executed on a laptop with an Intel i9 CPU

[2]A compact formulation is provided in Section VIII-B.

(2.4 GHz, 16 cores). The implementation runs with $P+1$ CPU threads, where P is the number of guided local planners executed in parallel(for 1 robot). Each planning thread is stopped if it runs longer than the control period of $50\,\mathrm{ms}$. The robot then executes the best trajectory among the local optimizations that finished within this time. If none of the local optimizations finishes in time or returns a feasible solution, we apply a fallback command corresponding to a braking maneuver with $3.0\,\mathrm{m/s^2}$. In this case, the resulting **braking trajectory** is always communicated to the other robots as described in Section IV-E.

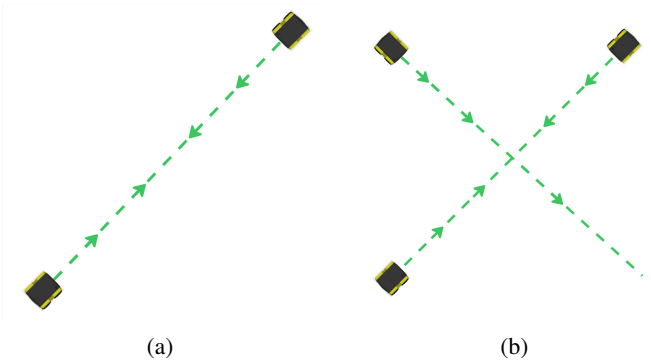
Note: All robots are simulated on a single laptop, meaning that all planning threads must share the same CPU. As a result, some planning threads may occasionally receive less processing time and fail to finish within the $50\,\mathrm{ms}$ *control period. This can introduce compute-induced artifacts that would be unlikely in truly distributed deployment, where each robot runs its planner on its own onboard computer. To mitigate this effect and keep compute availability fair and realistic, we run the simulation in slow motion by scaling the simulation clock by a factor* 0.5.

TABLE I: Experimental settings.

Parameter	Value	Description
N	30	Global and local planner horizon
ΔT	$0.2\,\mathrm{s}$	Integration time step
h	$0.05\,\mathrm{s}$	Planning time step
n	30	Visibility-PRM sample limit
$T_{\max}$	$10\,\mathrm{ms}$	Visibility-PRM time limit
Eqs. (12) and (23)	H-signature	Homotopy comparison function
P	4	Number of distinct guidance trajectories
G	5×5	Grid of goals (longitudinal $\times$ lateral)
r_i	$0.850\,\mathrm{m}$	Combined radius of two robots
w_c	0.05	Optimization contouring weight
w_l	0.75	Optimization lag weight
w_v	0.55	Optimization velocity tracking weight
w_ω	0.85	Optimization rotational velocity weight
w_a	0.34	Optimization acceleration weight
w_τ	0.4	optimization previous trajectory weight
Decision	Eq. (18)	Type of decision-making
c_i	0.75	Discount factor for previously followed homotopy class
Homology ID(non-guided)	True	Assign a homology ID to the non-guided trajectory (False for baselines)
Communication policy	True	Enable communication policy (False for baselines)
ϵ	$1.1\,\mathrm{m}$	Max deviation threshold (see Eq. (20)
T_{hb}	$3.0\,\mathrm{s}$	Heartbeat period

B. Scenario

We consider scenarios with $R \in \{2,3\}$ communicating robots. In each run, the robots perform an antipodal swap maneuver on their assigned reference path. The robot tries to follow the reference path with a reference velocity of $2\,\mathrm{m/s}$ and is controlled at $20\,\mathrm{Hz}$. Each robot has a radius of $0.325\,\mathrm{m}$ but the planners work with a more conservative radius of $0.425\,\mathrm{m}$ for increased distance between the robots. A simulation experiment starts from the initialized robot and agent states; a trial is successful when all robots reach the end of the reference path within a specified goal tolerance. After reaching the path endpoint, each robot rotates by π and waits until all other robots have reached their respective endpoints. Once all robots have arrived, each robot repeats the same maneuver on the reversed reference path. This cycle is repeated for a fixed number of trials, resulting in multiple swaps per experiment. Even though all planner settings are identical for a given scenario, trials can still differ due to small variations in each robot's starting pose and the sensitivity of the nonconvex motion-planning pipeline in combination with the communication policy. Small perturbations, noise, and sampling effects(Section IV-B) can lead to different outcomes in the guidance module and make the local optimization converge to different local optima, which changes the executed trajectory and therefore the outcome of the communication policy. Since our communication policy is based on the executed trajectory, this directly affects when communication is triggered. We therefore perform 11 trials per experiment to capture this variability and to avoid conclusions from a single favorable or unfavorable trial (i.e., a "lucky" or "unlucky" outcome). A visualization of the scenarios is shown in Fig. 5.



(a) (b)

Fig. 5: Illustration of the simulation scenarios. (a) $R=2$, two robots perform an antipodal swapping maneuver. (b) $R=3$, three robots perform the same swapping maneuver.

C. Baselines

We compare T-DMPC against two baselines that separate the benefits of having communication at all from the benefits of our (event-triggered) communication policy. All methods use identical guidance and local-planner weights Table I. The prediction and communication mechanisms differ across methods. In addition, the baselines disable the non-guided trajectory mapping step (Section IV-C) and the trajectory-consistency cost term (Eq. (16)).

1) *T-MPC + Local CV Predictions (No Communication) – T-VMPC*: Uses T-MPC as the motion planner and predicts all dynamic obstacles locally at each planning iteration using a constant-velocity (CV) model over the horizon, without communicating trajectories. Apart from treating the other robots as the dynamic obstacles (instead of

pedestrians following a social-forces model), this baseline matches the implementation in [1] and uses the same local constant-velocity prediction model.

2) *T-MPC + Always-Communicate Trajectory (No Policy) – T-AMPC*: Uses T-MPC as the motion planner and communicates the newly computed execution trajectory at every planning iteration, without applying a communication policy.

D. Evaluation metrics

We evaluate the methods at both the individual robot level and the system level. System-level results summarize team performance per trial by aggregating the per-robot metrics across all robots (e.g., by averaging across robots within a trial), after which we report summary statistics over the 11 trials. Each method is evaluated using the following metrics:

1) *Task duration:* time required to reach the end of the reference path.
2) *Traveled distance:* total path length executed before reaching the end of the reference path.
3) *Runtime:* average computation time per control-loop iteration (including trajectory update, prediction alignment, and optimization solve).

To quantify communication efficiency, we additionally report for, T-DMPC:

1) *Communication count:* the total number of messages communicated by a robot per experiment.
2) *Critical communication count:* The number of messages robot i communicated per experiment during *critical* planning iterations. A plannings iteration is considered critical if robot i's distance to all other robots is decreasing (i.e. the robots are moving closer to each other). The total number of plannings iterations is always greater than the number of *critical* iterations.

Note: The communication metrics are not reported for T-VMPC because it is not communication based, and not for T-AMPC because it communicates at every planning iteration.

E. Analysis

The results of the 11 trials are reported per scenario and per robot in Table II and Table III. System-level results, obtained by aggregating the per-robot metrics, are summarized in Table IV. Communication metrics (only for T-DMPC) per robot and on a system-level are reported in Table V and Table VI. For the 2-robot scenario, the per-robot results show no meaningful differences in task duration or distance traveled across methods. At the system level, there is also no meaningful difference across methods for the 2-robot scenario. For the 3-robot scenario, the distance traveled per robot is similar across all methods. For task duration, however, we observe small variation in both the mean duration and the duration standard deviation across methods. In particular, T-VMPC shows for Robot 1 a mean duration of $13.6\,\mathrm{s}$ with a standard deviation of $1.5\,\mathrm{s}$, compared to $13.1\,\mathrm{s}(0.3\,\mathrm{s})$ and $12.8\,\mathrm{s}(0.5\,\mathrm{s})$ for T-AMPC and T-DMPC, respectively, where values in parentheses denote the standard deviation. The same patterns holds for the other 2 robots. At the system level, T-VMPC also shows the largest duration standard deviation $1.5\,\mathrm{s}$ and the highest maximum duration $17.7\,\mathrm{s}$. Fig. 6 visualizes the system paths over the 11 trials for both scenarios. As can be seen, T-AMPC and T-DMPC yield smoother paths in both scenarios and track the reference path more closely in the 3-robot scenario. We quantitatively capture this smoothness by computing, for each method, the system-level mean squared integral jerk of the acceleration and angular-acceleration components across trials. Specifically, we report $J_a(R{=}2, R{=}3)$ for the acceleration jerk and $J_\alpha(R{=}2, R{=}3)$ for the angular-acceleration jerk. The integral jerk values for T-VMPC are higher in both scenarios $J_a = (28.2, 35.5)$ and $J_\alpha = (51.1, 40.3)$, than for T-AMPC with $J_a = (24.8, 19.8)$ and $J_\alpha = (2.3, 7.1)$, and for T-DMPC, with $J_a = (20.53, 19.2)$ and $J_\alpha = (5.2, 10.45)$. This indicates less smooth (i.e., more oscillatory) motion under T-VMPC in both the 2-robot and 3-robot scenarios. This was also empirically observed during the experiments. Fig. 7 shows the minimum pairwise distance between robots over time. No trial resulted in a physical collision, the distance never drops below the collision threshold of $0.65\,\mathrm{m}$. However, in the 2 robot scenario(Fig. 7a) all methods slightly violate the (more conservative) intrusion radius used in the local optimization, indicating close interactions even when collisions are avoided. In the 3-robot scenario(Fig. 7b), intrusions occur for T-VMPC, whereas T-AMPC and T-DMPC, remain above the intrusion threshold. *However, a small caveat is that although no intrusions were observed in the recorded experiment, intrusions can still occur for T-AMPC and T-DMPC in the 3-robot scenario (see Fig. 8). In particular, T-DMPC can occasionally show large intrusions.* Overall, closer passes (smaller minimum distances) occur more often in the 2-robot scenario, likely because the interaction is less constrained than with three robots. Finally, T-AMPC and T-DMPC achieve a similar distance traveled and task duration, both per robot and at the system level. However, T-DMPC attains this performance with less communication, only about 9.8% of the critical plannings iterations triggers communication in the 2-robot scenario and about 13.7% in the 3-robot scenario.

F. Communication-trigger behavior

To better understand how T-DMPC communicates in the $R \in \{2, 3\}$ robot scenarios, we analyze where in the workspace the communication triggers occur. We additionally relate these trigger events to the objective values of the $P + 1$ local planners, to assess how trigger activations relate to changes in the local optimization outcomes.

1) Spatial distribution of communication triggers Fig. 9a shows the robot positions at which communication is triggered. Most *critical* communication events occur near the initial states of the reference paths and around the middle of the maneuver. This pattern becomes clearer when separating the triggers by type. The topology-change triggers in Fig. 9b, occur predominantly near the start of the reference paths, suggesting that robots initially switch between homology classes while resolving the interaction and selecting a consistent passing

TABLE II: Quantitative simulation Results: 2 robots of Section V-E over 11 trials. Task duration(Dur.), distance(Dist.) and runtime are reported as "mean(std.dev.)". The results indicate similar performance across methods.

Robot	Method	Dur. [s]	Dist. [m]	Runtime [ms]
Robot1	T-VMPC (CV)	13.0(0.2)	11.6(0.4)	8.1(4.0)
	T-AMPC (AC)	12.9(0.2)	11.5(0.4)	4.5(0.7)
	T-DMPC (ours)	12.6(0.2)	11.6(0.4)	5.4(0.8)
Robot2	T-VMPC (CV)	13.0(0.2)	11.6(0.4)	4.5(0.8)
	T-AMPC (AC)	12.9(0.2)	11.5(0.5)	6.8(3.4)
	T-DMPC (ours)	12.6(0.2)	11.5(0.4)	5.5(0.8)

TABLE III: Quantitative simulation Results: 3 robots of Section V-E over 11 trials. Task duration(Dur.), distance(Dist.) and runtime are reported as "mean(std. dev.)". The results indicate similar performance across methods, however T-VMPC shows a larger duration standard deviation compared to T-AMPC and T-DMPC.

Robot	Method	Dur. [s]	Dist. [m]	Runtime [ms]
Robot1	T-VMPC (CV)	13.6 (1.5)	11.7 (0.5)	5.0 (1.6)
	T-AMPC (AC)	13.1 (0.3)	11.6 (0.4)	4.8 (1.2)
	T-DMPC (ours)	12.8 (0.5)	11.6 (0.4)	6.3 (1.7)
Robot2	T-VMPC (CV)	13.1 (1.2)	11.7 (0.5)	9.5 (5.5)
	T-AMPC (AC)	13.1 (0.3)	11.6 (0.4)	5.9 (2.2)
	T-DMPC (ours)	12.8 (0.2)	11.7 (0.4)	11.3 (4.9)
Robot3	T-VMPC (CV)	12.4 (1.6)	11.7 (0.6)	13.2 (4.2)
	T-AMPC (AC)	13.7 (0.6)	11.6 (0.5)	9.5 (4.5)
	T-DMPC (ours)	13.0 (0.3)	11.6 (0.4)	5.8 (1.3)

TABLE IV: Quantitative simulation system results: 2(Fig. 6a) and 3(Fig. 6b) robots of Section V-E over 11 trials. Task duration(Dur.), distance(Dist.) and runtime are reported as "mean(std. dev.)". Results indicate similar system-level performance, however in the 3 robot scenario we see for T-VMPC a higher duration standard deviation compared to T-AMPC and T-DMPC.

Scenario	Method	Dur. [s]	Dur. min-max [s]	Dist. [m]	Dist. min-max [m]	Runtime [ms]
2 Robots	T-VMPC (CV)	13.0(0.2)	$[12.7 - 13.5]$	11.6(0.4)	$[11.2 - 12.9]$	6.3(3.4)
	T-AMPC (AC)	12.9(0.2)	$[12.7 - 13.6]$	11.5(0.4)	$[11.1 - 12.9]$	5.6(2.7)
	T-DMPC (ours)	12.6(0.2)	$[12.4 - 13.2]$	11.6(0.4)	$[11.2 - 12.8]$	5.4(0.8)
3 Robots	T-VMPC (CV)	13.0(1.5)	$[8.8 - 17.7]$	11.7(0.5)	$[11.2 - 13.3]$	9.1(5.3)
	T-AMPC (AC)	13.3(0.5)	$[12.7 - 15.2]$	11.6(0.4)	$[11.1 - 12.8]$	6.8(3.6)
	T-DMPC (ours)	12.8(0.3)	$[12.2 - 13.6]$	11.6(0.4)	$[11.2 - 12.8]$	7.7(3.7)

side. In contrast, the geometric triggers in Fig. 9c, cluster around the middle of the reference paths, where robots are already committed to a homology class but still adapt their motion plans within that class due to nearby interactions. In this phase, deviations from the time-aligned last communicated trajectory can become large enough to exceed the geometric threshold (Eq. (20), Eq. (21)), which leads to communication.

2) Objective-gap and topology-switching behavior Fig. 10a and Fig. 10b relate topology switches to the *objective gap* between the best solutions returned by the $P+1$ local planners. When this gap is small, multiple homology classes yield nearly identical costs. In that case, small noise or minor changes in the local environment can make a different homology class become optimal at the next planning iteration, which increases the likelihood of a topology switch (and thus a communication event). As the robots approach each other, the objective gap typically increases, the interaction pattern converges and one homology class becomes preferable. Therefore, topology switches become less frequent during close interactions.

G. Ablation Studies

To investigate the contribution of each component of our method and the communication policy, we evaluate variants in which components and triggers are enabled step by step. Across all ablations, the motion-planning pipeline and all planner parameters remain fixed. The triggers for infeasibility and non-guided/unmapped execution remain enabled in all variants, since they represent explicit fallback behaviors that other robots must be aware of Section IV-E. The performance and the communication results are shown in Table VII. *Note: We restrict the ablation study to the $R = 3$ scenario because it is the smallest team size where each robot can be influenced by more than one other robot at the same time. This makes prediction errors and stale trajectory information more likely to affect safety than in the largely symmetric $R = 2$ scenario, while still covering the same pairwise interactions. Given the number of ablation variants and trials, we therefore focus on $R = 3$ as the most informative setting.*

1) Only topology-change trigger This ablation isolates the core idea of topology-aware communication, a robot communicates only when its selected homology class changes between consecutive planning iterations. It therefore quantifies how much communication can be reduced by relying solely on these high-level "topology switches," and shows the main failure mode. The main failure mode is substantial changes *within* a homology class can remain uncommunicated, causing robots to maintain wrong predictions of each other. At the system level in the 3-robot scenario, the resulting task duration and distance traveled are comparable to T-DMPC, indicating that the overall efficiency is not affected. However, the system-level communication amount decreases from approximately $13.2\,\%$ to $8.6\,\%$ because communication is triggered less often. This reduction comes at the cost of safety: the minimum-distance plot in Fig. 11 shows that the topology-only trigger leads to more intrusion of the conservative radius and even leads to collisions. This is caused by a mismatch between robot i's prediction of robot j's trajectory and robot j's actual executed trajectory. In other words, topology-change alone is an insufficient communication trigger, additional triggers are required to capture interaction relevant deviations within the same homology class.

TABLE V: Simulation communication results: 2 robots. The results are reported for each individual robot and for the complete system where the communication amount for Robot 1 and Robot 2 are aggregated. The results show that T-DMPC uses about $9.8\,\%$ of communication to complete the experiment with comparable performance results as T-DMPC.

Robot	Total comm.	Total iter.	Comm/Iter [%]	Critical comm.	Critical iter.	Crit. comm/ Crit. iter [%]
Robot1	153	2789	5.5	93	1001	9.3
Robot2	160	2785	5.7	103	1003	10.3
System	313	5574	5.6	196	2004	9.8

TABLE VI: Simulation communication results: 3 robots. The results are reported for each individual robot and for the complete system where the communication amount for Robot 1, Robot 2 and Robot 3 are aggregated. The results show that T-DMPC uses about $13.2\,\%$ of communication to complete the experiment with comparable performance results as T-DMPC.

Robot	Total comm.	Total iter.	Comm/Iter [%]	Critical comm.	Critical iter.	Crit. comm/ Crit. iter [%]
Robot1	193	2815	6.9	135	1089	12.4
Robot2	204	2835	7.2	141	1071	13.2
Robot3	213	2861	7.4	153	1097	13.9
System	610	8511	7.2	429	3257	13.2

TABLE VII: Ablation study results for 3 robots: system-level performance results and system-level critical communication results. Task duration(Dur.), distance(Dist.) and runtime are reported as "mean(std. dev.)".

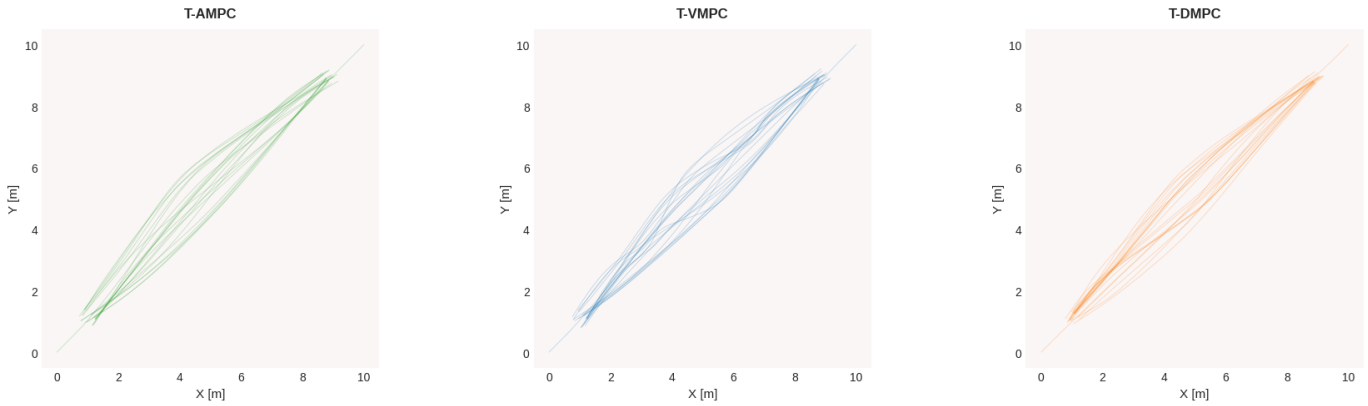
Method	Duration [s]	Dur. min–max [s]	Distance [m]	Dist. min–max [m]	Runtime [ms]	Critical iter.	Critical comm.	Comm [%]
topology(Section V-G1)	13.0(0.4)	[11.9 - 13.9]	11.6(0.4)	[11.1 - 12.9]	13.7(6.3)	3222	278	8.6
topology_consistency(Section V-G2)	12.8(0.3)	[12.3 - 13.7]	11.6(0.4)	[11.1 - 12.8]	12.7(6.3)	3210	264	8.2
topology_consistency_geo(Section V-G3)	12.8(0.3)	[12.4 - 13.8]	11.6(0.4)	[11.1 - 12.8]	14.4(6.5)	3205	448	14
topology_consistency_time(Section V-G4)	12.8(0.3)	[12.3 - 13.6]	11.6(0.4)	[11.2 - 12.8]	8.0(4.0)	3205	291	9.1
topology_consistency_time_2(Section V-G4	12.7(0.7)	[9.2 - 14.1]	11.6(0.4)	[11.1 - 12.8]	15.8(6.5)	3265	363	11.1
topology_geo_time(Section V-G5)	13.1(0.4)	[12.7 - 14.0]	11.6(0.4)	[11.2 - 12.9]	11.9(6.1)	3257	461	14.2

2) Topology-change trigger + trajectory consistency In addition to topology-change communication, this variant activates the trajectory-consistency mechanism Eq. (16) that discourages deviations from the *previously executed trajectory* when staying in the same homology class. The goal is to improve predictability and reduce within homology class trajectory drift and therefore reduce the amount of communication. This ablation tests whether increasing *behavioral continuity* can reduce the need for additional communication while maintaining safety and efficiency. At the system level, performance remains comparable to T-DMPC, while communication is reduced. However, a collision and multiple intrusions of the conservative radius still occur, although less frequently than with the topology-only trigger. Topology-change plus trajectory consistency is therefore still insufficient. The main reason is that the consistency term is enforced with respect to robot i previously executed trajectory, which may differ from what the other robots believe robot i will do (i.e., the last communicated trajectory time-aligned to the current planning iteration). As a result, robot i can remain "consistent" locally while still deviating from the trajectory that others are predicting, leading to mismatched plans and collisions.

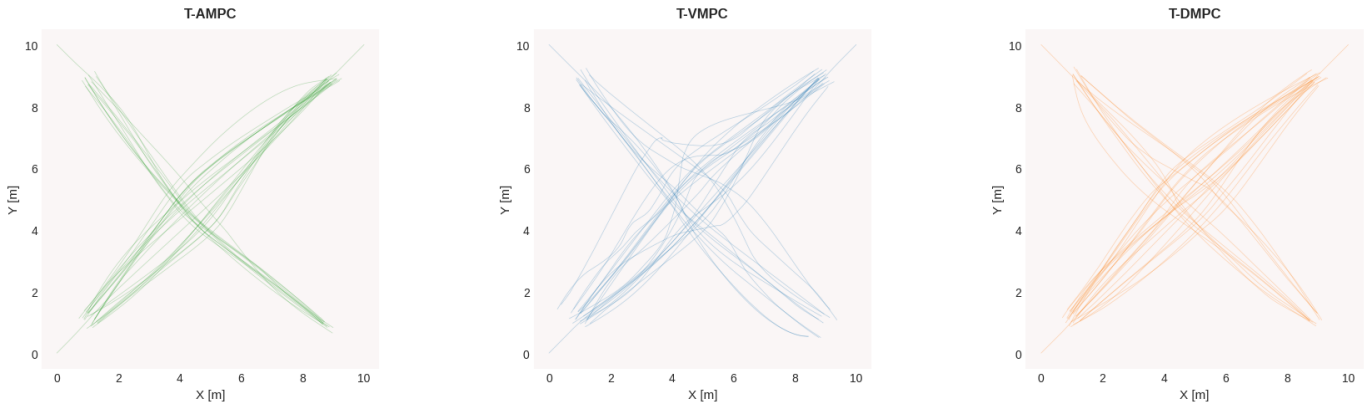
3) Topology-change trigger + trajectory consistency + geometric trigger Even when the homology class stays the same and trajectory consistency is encouraged, Robot i can still change its motion in a way that is not shown in the last trajectory communicated to the other robots. The geometric trigger addresses this by communicating whenever the newly selected trajectory deviates too much from the time-aligned last communicated trajectory Eq. (20). This ablation therefore highlights the increased safety of preventing *wrong predictions*. At the system level, performance is comparable to T-DMPC. Communication increases relative to the previous ablations (Section V-G2) to $14\,\%$ and becomes comparable to T-DMPC. Importantly, Fig. 11 shows no collisions. This is because robots now communicate when their actual behavior no longer matches what the other robots are predicting.

4) Topology-change trigger + trajectory consistency + time-based trigger Only using the topology-change and geometric trigger can result in periods of no communication. The time-based (heartbeat) trigger therefore enforces periodic updates. This ablation shows the effect of occasional refreshes, and the amount of communication. At the system level, performance remains comparable to T-DMPC. The amount of communication is lower $9.1\,\%$ compared to the geometric-trigger variant and also slightly lower than T-DMPC, because the chosen $T_{hb} = 3\,\mathrm{s}$ triggers less frequently than the geometric trigger. However, Fig. 11 shows that $T_{hb} = 3\,\mathrm{s}$ leads to collisions, the amount of communication is too low and is not able to capture mismatches between predicted and executed behavior fast enough. Reducing the heartbeat to $T_{hb} = 2\,\mathrm{s}$ avoids collisions, but still produced visibly unsafe behavior in practice.

5) Topology-change trigger + geometric trigger + time-based trigger To isolate the effect of the trajectory-consistency term in Eq. (16), we disable it while keeping the topology-change, geometric, and time-based triggers enabled. At the



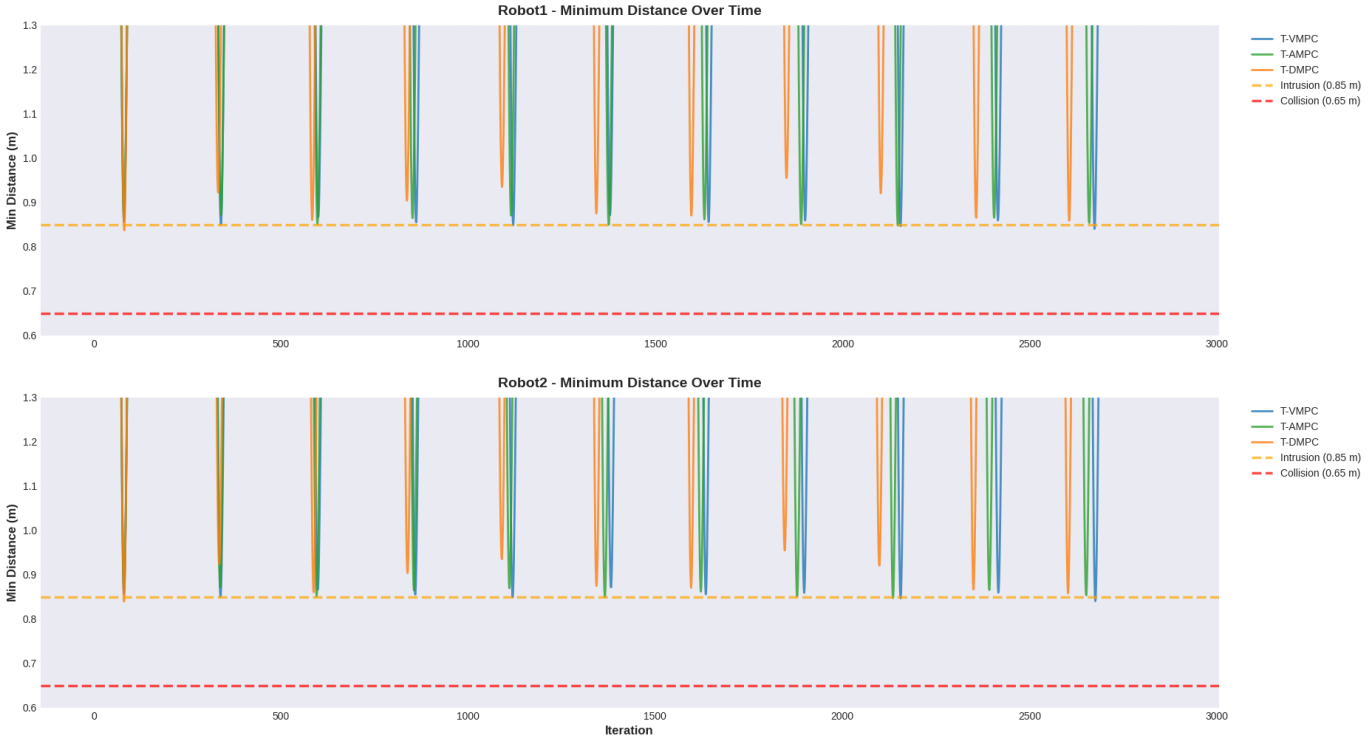
(a)



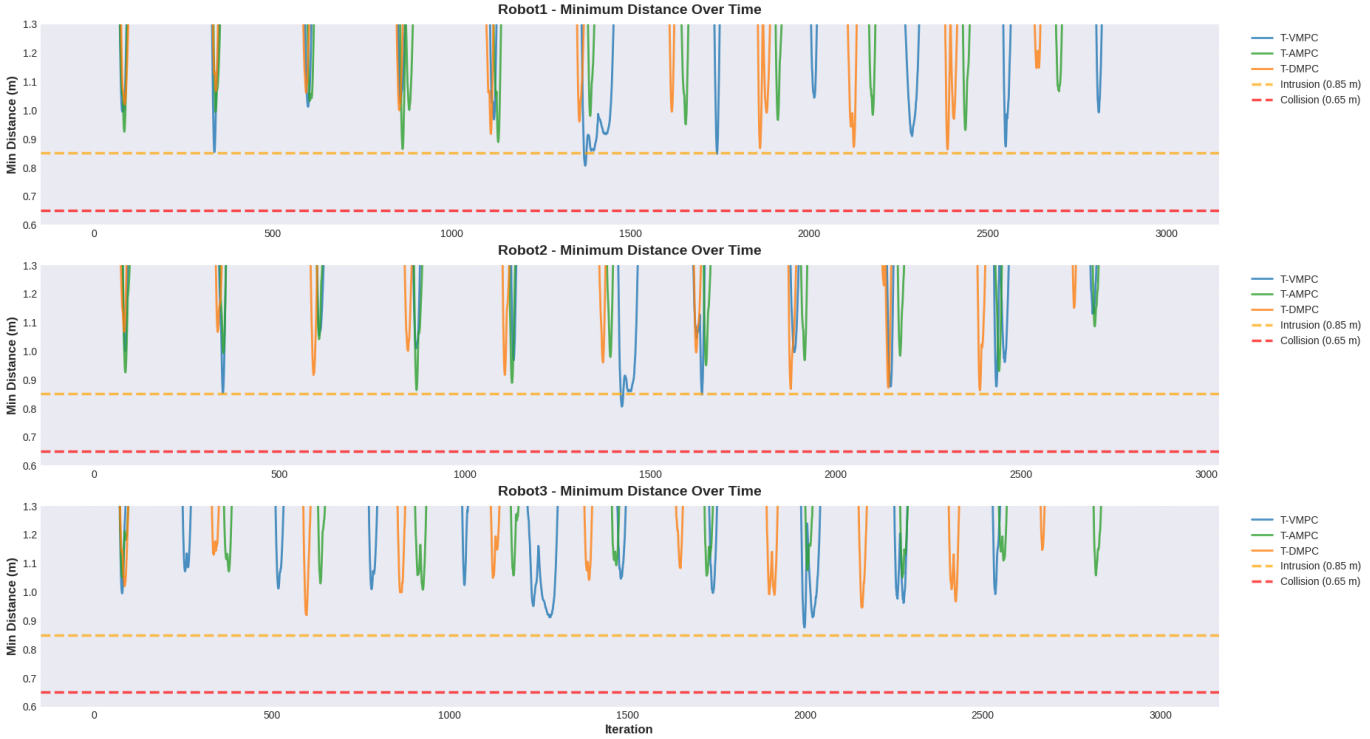
(b)

Fig. 6: Executed paths during the **simulation** experiments for (a) 2 robots and (b) 3 robots. The paths indicate that T-VMPC produces less smooth trajectories and tracks the reference path less closely than T-AMPC and T-DMPC. In the 3-robot scenario, T-AMPC also shows slightly more variability around the reference path than T-DMPC, whereas T-DMPC remains more consistent across runs.

system level, performance remains comparable to T-DMPC. However, the amount of communication increases slightly relative to T-DMPC(from $13.2\,\%$ to $14.2\,\%$). This suggests that without Eq. (16) the planner exhibits a marginally larger within-class trajectory variation, which causes the geometric trigger to trigger more frequently. However the observed differences are (really) minor indicating a limited effect of the trajectory-consistency term.



(a) $R = 2$



(b) $R = 3$

Fig. 7: Minimum inter-robot distance over time in simulation across all methods. (a) $R = 2$ robots and (b) $R = 3$ robots.

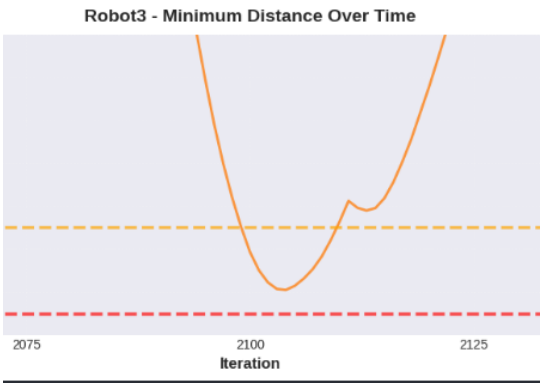
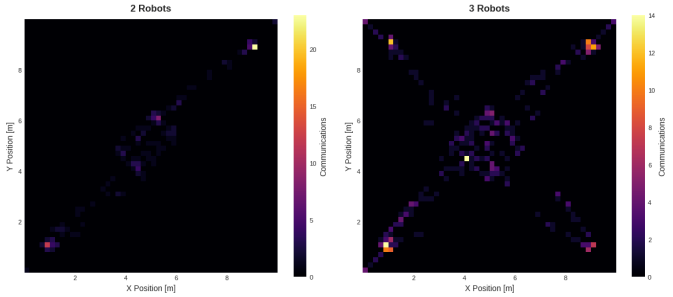
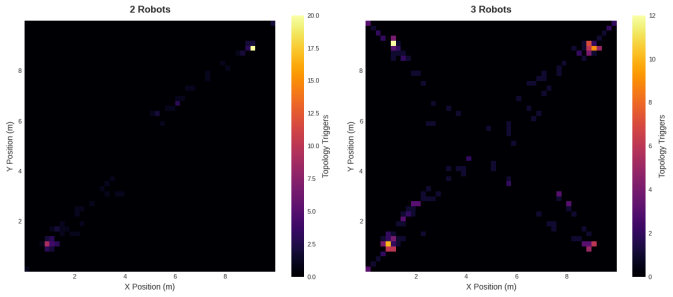


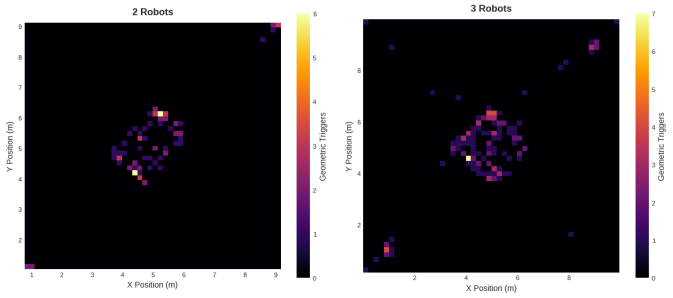
Fig. 8: Illustration of the variation in the violation of the conservative radius(orange dotted line) exhibited by T-DMPC. For the x, y-axis and line color definitions, see Fig. 7; they are omitted here to reduce figure size and avoid repetition.



(a) Heatmap of (all) **critical** communication events over the **simulation** workspace. Brighter regions indicate higher communication density.

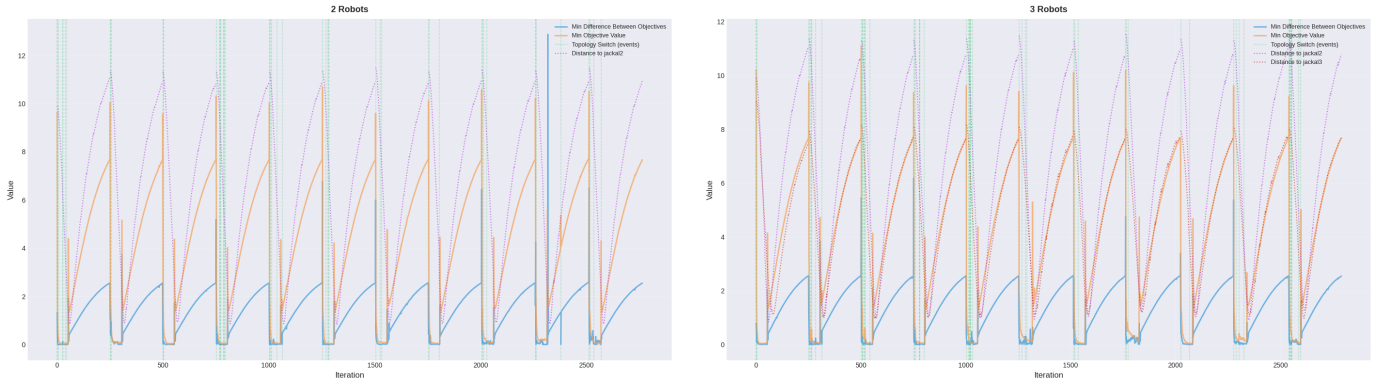


(b) Heatmap of topology-trigger **critical** communication events over the **simulation** workspace. Brighter regions indicate higher communication density.



(c) Heatmap of geometric-trigger **critical** communication events over the **simulation** workspace. Brighter regions indicate higher communication density.

Fig. 9: Heatmaps of **critical** communication events over the **simulation** workspace. The color intensity indicates how frequently communication was triggered at each (x, y) position, with brighter (more yellow) regions corresponding to higher communication density.



(a) (b)

Fig. 10: Objective-gap and topology-switching behavior in simulation for (a) $R = 2$ and (b) $R = 3$, shown from the perspective of Robot 1. At each planning iteration, Robot 1 compares the objective values of the $P+1$ parallel local planners (each associated with a distinct homology class) and evaluates the minimum objective gap between the best and second-best candidate. When this gap is small, multiple homology classes yield nearly identical costs, so minor noise or small changes in the perceived environment can change which class is optimal, increasing the likelihood of a topology switch. As the robots approach each other and interactions become more constrained, the objective gap typically increases and one homology class becomes preferable, after which topology switching becomes less frequent.

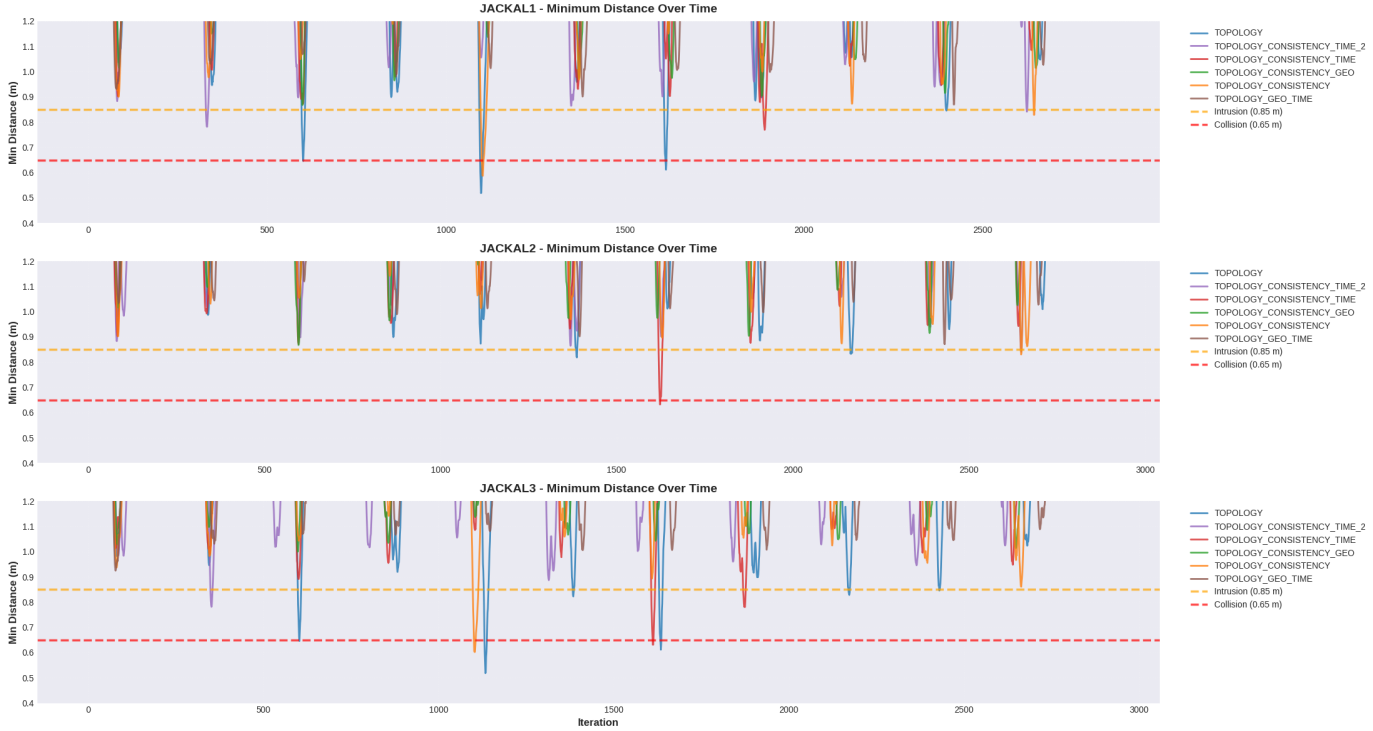


Fig. 11: Minimum inter-robot distance across for all ablation study experiments. The yellow dashed line denotes the conservative intrusion radius used in the optimization, and the red dashed line denotes the robots true physical radius

VI. Real-World Experiments

This section describes the real-world experimental setup used to validate the proposed topology-aware distributed motion planner (T-DMPC) on physical robots. It then reports and analyzes the experimental results, and the observed communication-trigger behavior, and demonstrates additional real-world variants such as static and dynamic obstacle experiments to show that the method is applicable to multiple scenarios.

A. Planner Implementation and Parameters

The implementation used in the real-world experiments closely follows the simulation implementation described in Section V. Each robot is assigned a reference path between two opposite corners of the arena. Upon reaching a corner, the robot rotates by π and waits until the other robots have also reached their respective corners. Once all robots have arrived, the maneuver is repeated on the reversed reference paths. Two implementation aspects differ from simulation. First, robot and agent states are obtained from a motion-capture system at $20\,\mathrm{Hz}$. These measurements are filtered using a Kalman filter to estimate velocities and to provide constant-velocity predictions of the non-communicating agent for the motion planner. Second, the trajectory communication mechanism differs due to the distributed ROS setup: in contrast to simulation, where all nodes operated under a single ROS master, each physical robot runs its own *roscore*. To enable trajectory communication across multiple ROS masters, we run the motion planner of each robot inside a dedicated Docker container on a single computer and add an external communication layer based on ZeroMQ [34]. This layer transports planned trajectories between containers, enabling communication of the prediction trajectories, while being on separate ROS networks. The resulting communication architecture is illustrated in Fig. 12.

B. Real-world Scenario

We evaluate the real-world performance of the planner in the same antipodal swap maneuvers as in simulation, with $R \in \{2, 3\}$ communicating robots and one additional non-communicating agent $A = 1$. The agent is used in two variants: (i) as a moving obstacle, and (ii) as a static obstacle. All experiments take place in a $5\,\mathrm{m}\times 8\,\mathrm{m}$ environment. Each robot is assigned a reference path between two opposite corners of the arena. A trial starts from the initialized robot (and agent) states and ends when all robots reach the end of their respective reference paths within a goal tolerance. To ensure safe and real-time execution, we adjust the the following parameters depending on the team size. For $R = 2$, the reference velocity is reduced to $1.5\,\mathrm{m/s}$. For $R = 3$, all planners still run on the same computer, which increases the computational load. To ensure that all planning iterations finish within the control period, we reduce the control frequency to $10\,\mathrm{Hz}$, giving each planning thread $100\,\mathrm{ms}$ instead of $50\,\mathrm{ms}$. We also lower the reference velocity to $1\,\mathrm{m/s}$ to compensate for the lower control frequency. A visualization of the real-world scenario for 3 robots is shown in Fig. 13.

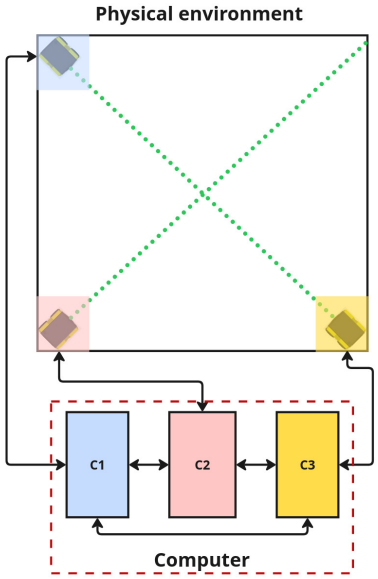


Fig. 12: Real-world communication architecture for the $R = 3$ scenario. Three Docker containers (C1–C3) run on a single physical computer but appear as separate networked devices. The containers exchange data via ZeroMQ, and each container connects to its color-matched robot through that robot's isolated ROS network and corresponding ROS master to send the computed motion commands in each planning iteration.

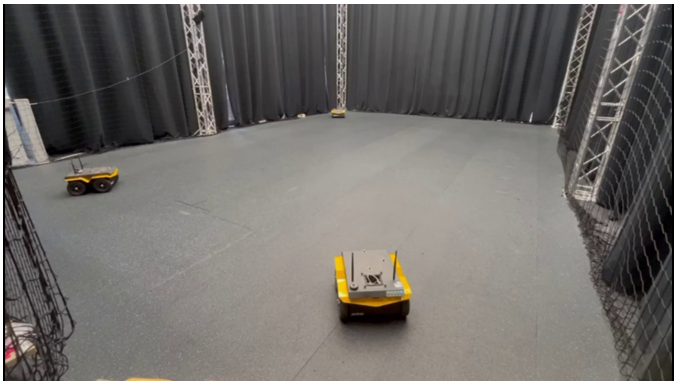


Fig. 13: Real-world $R = 3$ scenario, each robot performs an antipodal swap, moving across the arena to the opposite corner.

TABLE VIII: Quantitative real-world system results: 2 and 3 robots of Section VI-B over 11 trials. Task duration(Dur.), distance(Dist.) and runtime are reported as "mean(std. dev.)". The results indicate similar performance across methods, however T-VMPC and T-DMPC show a larger duration standard deviation in the 3 robot scenario.

Scenario	Method	Dur. [s]	Dur. min-max [s]	Dist. [m]	Dist. min-max [m]	Runtime [ms]
2 Robots	T-VMPC(CV)	8.0(0.6)	[7.0 – 9.2]	9.1(0.8)	[8.0 – 10.1]	11.4(5.0)
	T-AMPC(AC)	8.1(0.5)	[7.3 – 8.9]	9.0(0.6)	[8.0 – 10.1]	12.9(5.8)
	T-DMPC(ours)	8.4(0.5)	[7.8 – 9.4]	9.1(0.6)	[8.3 – 9.8]	12.9(5.1)
3 Robots	T-VMPC(CV)	10.7(1.0)	[9.0 – 12.3]	8.6(0.5)	[7.7 – 9.6]	20.5(11.7)
	T-AMPC(AC)	11.3(0.6)	[9.9 – 12.3]	8.7(0.4)	[8.0 – 9.3]	18.0(8.9)
	T-DMPC(ours)	11.2(1.1)	[9.4 – 14.7]	8.7(0.5)	[7.7 – 9.5]	15.8(7.6)

TABLE IX: Real-world communication results 2 robots. The results are reported for each robot and for the complete system where the communication amount for Robot 1 and Robot 2 are aggregated. The results show that T-DMPC uses about $9.8\,\%$ of communication to complete the experiment.

Robot	Total comm.	Total iter.	Comm/Iter [%]	Critical comm.	Critical iter.	Crit. comm/ Crit. iter [%]
Robot1	224	1793	12.5	111	637	17.4
Robot2	228	1935	11.8	114	625	18.2
System	452	3728	12.1	225	1262	17.8

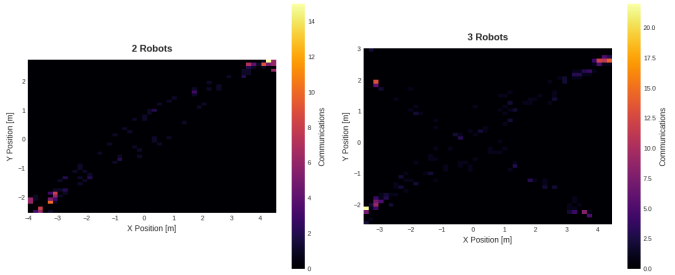
TABLE X: Real-world communication results 3 robots. The results are reported for each robot and for the complete system where the communication amount for Robot 1, Robot 2 and Robot 3 are aggregated. The results show that T-DMPC uses about $13.2\,\%$ of communication to complete the experiment.

Robot	Total comm.	Total iter.	Comm/Iter [%]	Critical comm.	Critical iter.	Crit. comm/ Crit. iter [%]
Robot1	154	1281	12	116	781	14.9
Robot2	154	1321	11.7	122	924	13.2
Robot3	110	1121	9.8	88	759	11.6
System	418	3723	11.2	326	2464	13.2

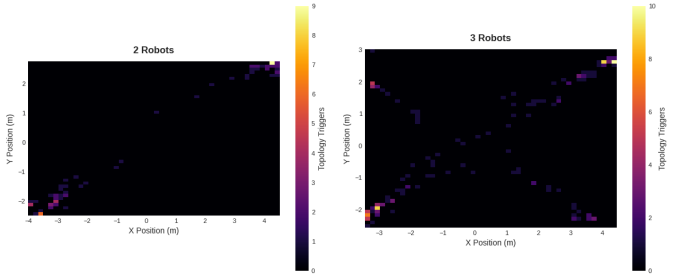
C. Analysis

The results of the 11 trials are reported per scenario. System-level results, obtained by combining all robots, are summarized in Table VIII for all methods. Communication metrics per robot are reported in Table IX and Table X.

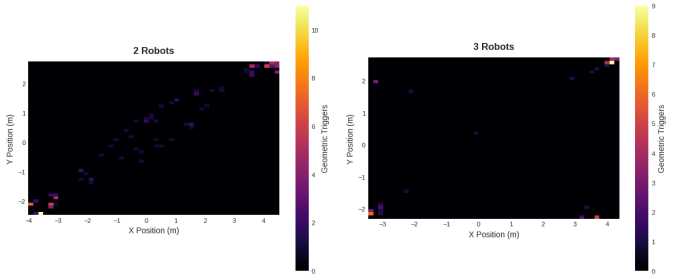
Overall, the real-world experiments indicate that the robots can execute the maneuver using our method while requiring, on average, about $17.8\,\%$ of communication in the $R = 2$ scenario and about $13.2\,\%$ of communication in the $R = 3$ scenario, while achieving comparable performance (see Table VIII). Across all trials and methods, no physical collisions were observed. This is consistent with the minimum pairwise distance analysis in Fig. 16, where the inter-robot distance remains above the collision threshold and also above the more conservative intrusion threshold in both scenarios. That said, in all methods the robots occasionally pass close to the intrusion threshold, which matches the close interactions observed during the experiments. When inspecting the paths followed by the robots for each method and scenario in Fig. 15, we observe that T-VMPC produces less smooth paths and tracks the reference path not as close as T-AMPC and T-DMPC. This matches our (empirical) experimental observations: T-VMPC exhibited the most oscillatory motion, with occasional "ping-pong" behavior in some trials. Visually, T-AMPC appeared the smoothest, with slightly less oscillation than T-DMPC. A likely reason is that T-AMPC updates and exchanges information every planning iteration, allowing it to respond even to small changes, whereas T-DMPC only communicates when its triggers fire. To better understand when communication is triggered in the real-world experiments, Fig. 14a visualizes the robot positions at which communication occurs. Most critical communication events occur near the initial states of the reference paths and around the middle of the maneuver. This pattern becomes clearer when separating triggers by type. The topology-change triggers in Fig. 14b occur predominantly near the start of the reference paths, suggesting that robots initially switch between homology classes while resolving the interaction and committing to a consistent passing side. In contrast to the simulation results, the geometric-deviation triggers in Fig. 14c show a different pattern, they occur more frequently near the beginning of the reference paths. This indicates that, in the real-world experiments, robots often deviate from the last communicated trajectory earlier in the maneuver.



(a) Heatmap of (all) **critical** communication events over the **real-world** workspace. Brighter regions indicate higher communication density.

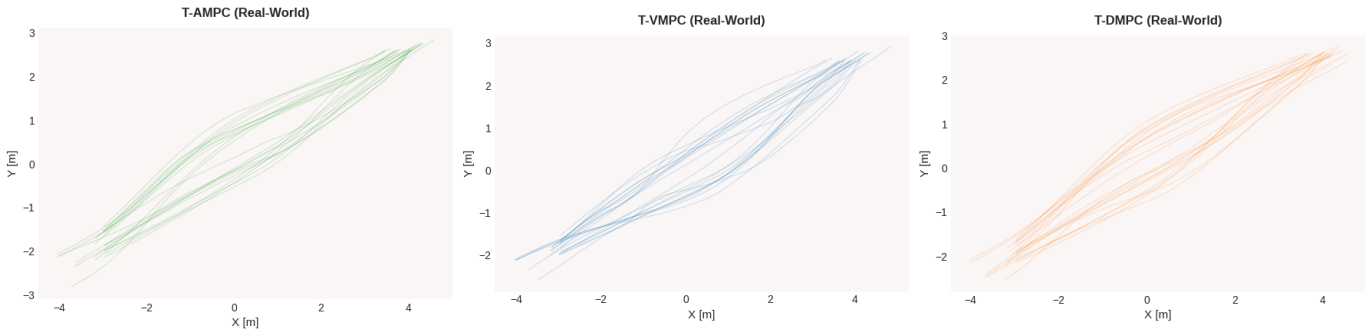


(b) Heatmap of topology-trigger **critical** communication events over the **real-world** workspace. Brighter regions indicate higher communication density.

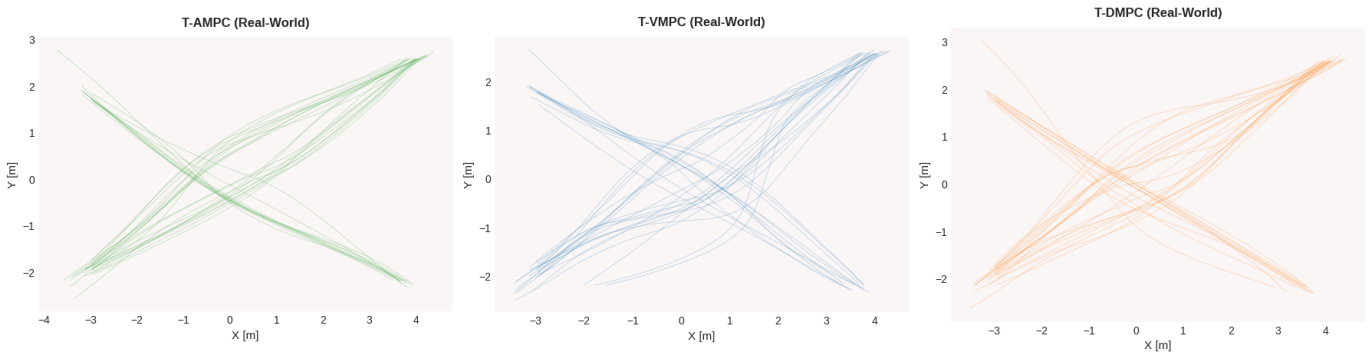


(c) Heatmap of geometric-trigger **critical** communication events over the **real-world** workspace. Brighter regions indicate higher communication density.

Fig. 14: Heatmaps of **critical** communication events over the **real-world** workspace. The color intensity indicates how frequently communication was triggered at each (x, y) position, with brighter (more yellow) regions corresponding to higher communication density.



(a) $R = 2$



(b) $R = 3$

Fig. 15: Executed paths during the **real-world** experiments for (a) 2 robots and (b) 3 robots. The paths indicate that T-VMPC produces less smooth trajectories and tracks the reference path less closely than T-AMPC and T-DMPC. In the 3-robot scenario, T-AMPC also shows slightly more variability around the reference path than T-DMPC, whereas T-DMPC remains more consistent across runs.

D. Static Obstacle

We include an additional demonstration in which a static obstacle is placed near the middle of the reference paths, illustrating that T-DMPC can incorporate static obstacles. The results show that the maneuver can still be executed in the real-world with less communication, which was also observed empirically during the experiments. Fig. 17a visualizes the scenario and the collision free paths followed by each robot for a given trial. Compared to the case without a static obstacle, the $R = 2$ scenario required around $24\,\%$ of communication. For $R = 3$, the overall communication amount was approximately $16.1\,\%$. Interesting is that the topology-change triggers concentrate even more strongly near the beginning of the reference paths, suggesting that the static obstacle forces the robots to commit earlier to a consistent homology class.

E. Dynamic Agent

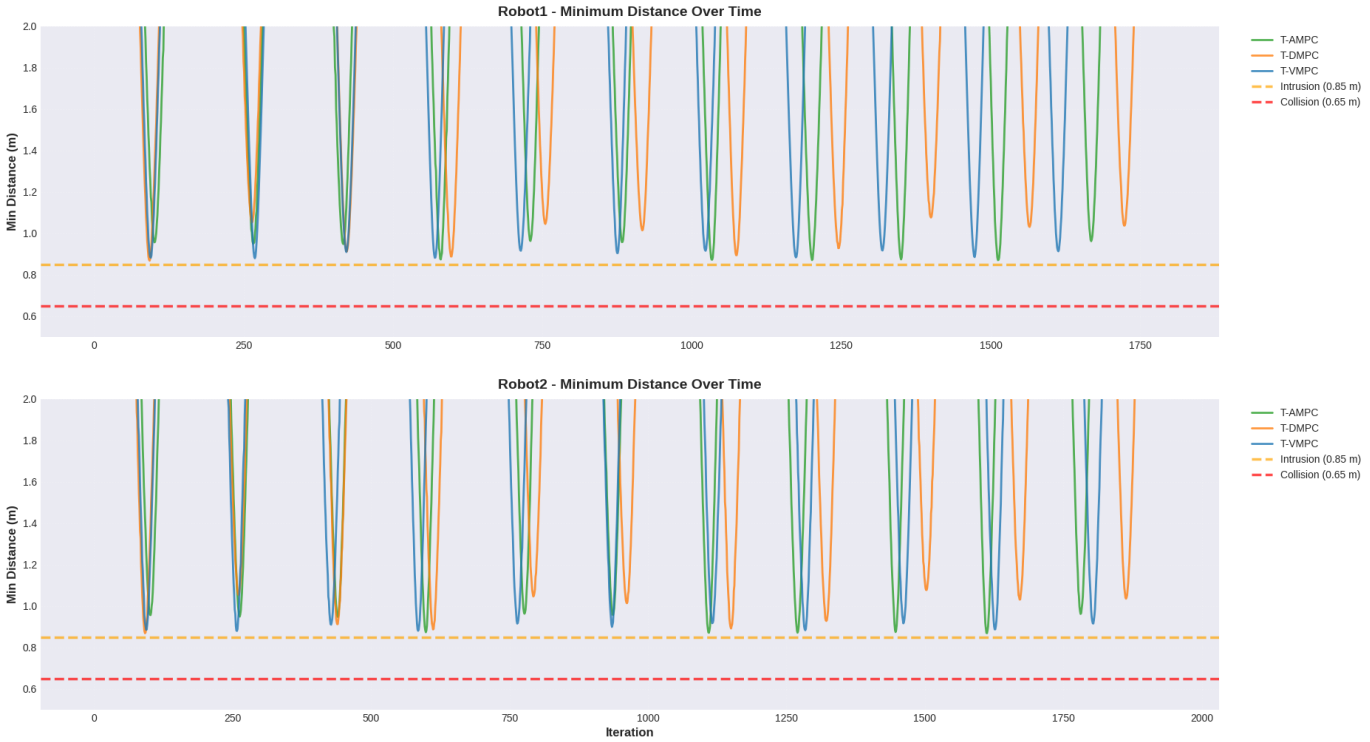
Finally, we demonstrate T-DMPC in a scenario with a dynamic obstacle that performs a swapping maneuver as well. While we do not analyze and draw conclusions from this experiment, it illustrates that the proposed framework can incorporate non-communicating dynamic obstacles, see Fig. 17b.

VII. Discussion

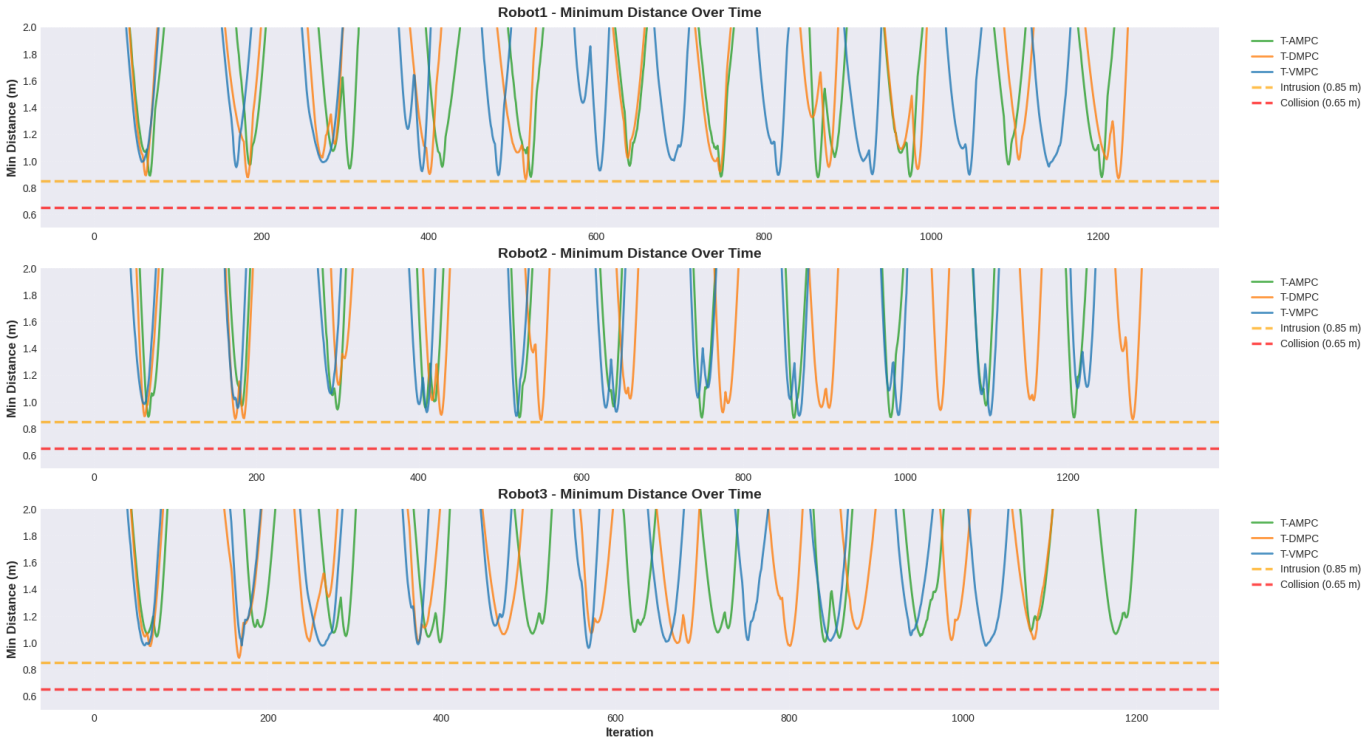
This section discusses the proposed topology-aware distributed motion planner (T-DMPC), the main limitations, the assumptions under which the reported results were obtained and ideas for future work.

A. Main Findings

Overall, the results indicate that, for the **tested scenarios**, T-DMPC can achieve performance comparable to the always-communicating baseline T-AMPC while needing less communication by making use of topological (homology-class) information. The results further suggest that relying on constant-velocity predictions of other robots can result in more oscillatory and less consistent motion than using trajectory predictions derived from a robots communicated planned trajectory. Possible reasons are that constant-velocity predictions do not capture changes in speed and heading along the executed trajectory. As a result, the predicted trajectory can differ from what is actually executed. In addition, small changes in the state of a robot over consecutive iterations (especially orientation) can lead to large changes in the predicted constant-velocity trajectory over consecutive iterations. Together, this can cause more oscillatory (ping-pong like) motion. The presented findings come with several caveats.

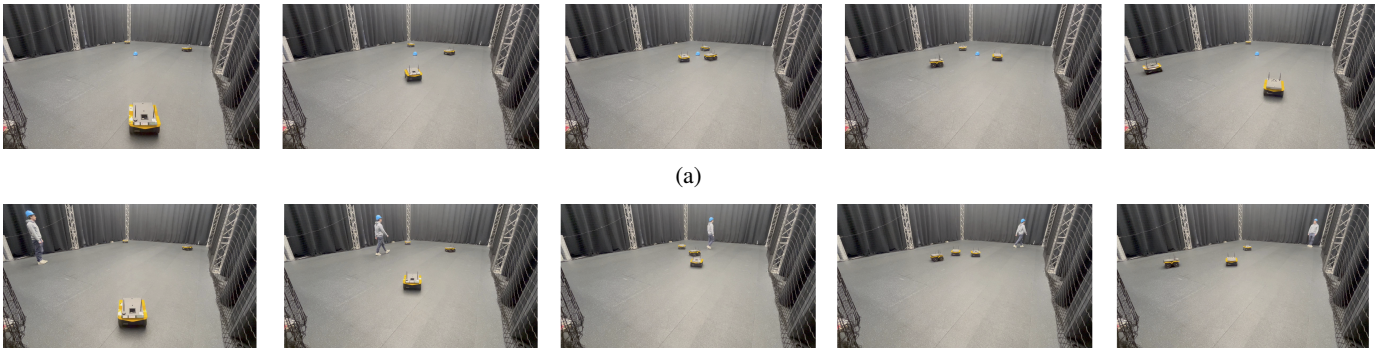


(a) $R = 2$



(b) $R = 3$

Fig. 16: Minimum inter-robot distance across all real-world experiments. (a) $R = 2$ robots and (b) $R = 3$ robots. The yellow dashed line denotes the conservative intrusion radius used in the optimization, and the red dashed line denotes the robots' true physical radius.



(a)

(b)

Fig. 17: Illustration of two additional real-world variants for $R = 3$. (a) Antipodal swap maneuver with an added *static obstacle* placed near the middle of the reference paths, demonstrating that T-DMPC can incorporate static obstacles. (b) Antipodal swap maneuver with one additional *non-communicating agent* ($A = 1$), whose motion is predicted locally using a constant-velocity model, demonstrating that T-DMPC can incorporate non-communicating dynamic obstacles. Time progresses from left to right.

B. Evaluation Scope and Scalability.

The evaluation is limited to antipodal swap maneuvers with $R \in \{2, 3\}$ communicating robots and a single additional non-communicating agent. While these scenarios capture interaction behavior, the approach has not yet been validated with larger team sizes or in more difficult environments. It could be that with larger team sizes robots keep on switching between homology classes which can results in oscillatory movement and even collisions. In addition, both computation and communication scale with the number of robots. Each robot solves multiple local optimization problems in parallel, and all-to-all trajectory exchange can become a bottleneck as R grows. Extending the approach to larger teams will likely require additional structure (e.g., neighborhood-based communication and collision-avoidance constraints).

C. Simulation and Real-World Experiment Realism

In the simulation and real-world experiments, the current evaluation uses state information from Gazebo or a motion-capture system (Vicon), both of which provide low uncertainty robot states. In real deployments, each robot must estimate its own state onboard, which introduces additional uncertainty that can degrade performance and safety. Additionally for the simulation, we slowed down the clock, so the results should be interpreted with this in mind. For the real-world experiments with $R = 3$, we reduced the control-loop frequency and the reference velocity to stay within the available computation time budget, this should also be considered when interpreting these results.

D. Communication Realism

The reported results do not explicitly evaluate the impact of communication delay or packet loss. In addition, the real-world experiments use an external communication layer to exchange trajectories across multiple ROS masters, with the planners running in separate containers on a single computer. This setup enables controlled trajectory exchange, but it does not capture the full variability and impact of onboard wireless communication. As a result, the observed trigger behavior and safety margins may change under realistic network conditions, and additional mechanisms may be required to handle delayed or dropped messages.

E. Conservative Safety Margin (inflated robot radius)

In both simulation and real-world experiments, each robot is modeled with an increased planning radius of $0.425\,\mathrm{m}$. This additional margin provides a buffer for differences between (i) the trajectory that other robots *believe* robot i is executing (i.e., the time-aligned last communicated plan) and (ii) the trajectory robot i actually executes. While this improves robustness, it also makes the optimization more conservative and may affect how closely robots can pass in reality. Moreover, **no collisions** should be interpreted in context of this increased radius. Results showed that trails still contain close interactions that temporarily violate the conservative radius used by the optimization (intrusions), even if physical collisions are avoided. Using the true radius would reduce the safety buffer and could increase the risk of collisions, so the no collision result should be interpreted in the context of the inflated planning radius.

F. Topology-Only Communication and Safety

Topological information alone is not sufficient to decide when robots should communicate. Collision-avoidance constraints are space-time dependent. Therefore if the prediction of another robot becomes stale, the local planner may enforce constraints against a wrong trajectory, which can lead to unsafe situations. This motivated incorporating an additional space-time communication mechanism, such as the geometric-deviation trigger and the time-based (heartbeat) trigger, to bound prediction staleness. In real deployments, onboard sensing (e.g., camera, LiDAR and radar) would likely also be required to validate and correct beliefs from communicated information. Perhaps a different view could be using communication as an additional, irregular *sensor* that can reinforce the predictions made with the sensors.

G. Trigger Parameter Tuning

The effectiveness of the communication policy depends on different thresholds (e.g., the geometric-deviation threshold and the heartbeat period). These thresholds trade off safety against amount of communication. Too high thresholds will make the amount of communication too low, allowing prediction mismatches to exist long enough to cause collisions. Too small thresholds will increase the amount of communication, making the communication policy less effective. Tuning these parameters is not straightforward, and they were not optimized in this work. Here, "optimal" refers to the trade-off between minimizing communication and avoiding collisions. Instead, we tuned them empirically until no collisions were observed. Finding this optimal balance is left as future work.

H. Planning failure modes

A limitation of the framework is its behavior under failure modes. In some iterations, the guidance planner may return no guidance trajectories (e.g., due to the computation time budget). In that case, the robot switches to the fallback braking maneuver and communicates the resulting braking trajectory to prevent other robots from relying on a stale prediction. When the guidance set is empty, the guided local planners are disabled and only the non-guided local planner can produce a feasible trajectory. However, the non-guided solution cannot be mapped to a homology class, the communication policy triggers (since a topology change cannot be ruled out), even though communication might have been unnecessary. Similarly, the local trajectory optimizations can become infeasible (e.g., due to restrictive topology-preserving constraints or challenging interactions) or fail to return a feasible solution within the time limit. This again triggers fallback behavior and additional communication to avoid stale predictions at other robots. Overall, these fallback mechanisms improve safety during undefined or degraded behavior, but they do not provide a formal safety guarantee.

I. Topology Classification Errors

Topology switches may be detected even when the executed trajectory is still in the same homology class as the previously executed trajectory. This can happen due to infeasibility fallbacks and approximation errors in the H-signature-based homology representation (mainly in symmetric low obstacle situations). As a result, the measured topology-change events can overestimate *actual* behavioral switches.

J. Future Work

Based on the discussion several directions can extend this work. First, the method should be evaluated in a more realistic distributed setting, where each robot runs its planner onboard and communicates over a wireless network. This would allow studying how communication delay and message loss affect system performance and safety. If such network effects degrade behavior, a natural extension is to incorporate delay and loss robust mechanisms, for example along the lines of [16], [21], [22]. Note, however, that the approaches in [21], [22] are formulated in a plan-and-track style and do not continuosly replan, whereas our framework relies on continuous replanning, and adapting these ideas would therefore require additional design choices. In addition, the method should be tested with more robots to assess scalability and to study whether frequent switching between homology classes occurs as interaction complexity increases. A second direction is to investigate how to integrate irregular communication with sensor-based prediction in a single prediction module. One approach could be to treat communicated trajectories as intermittent, higher-certainty information that can inform and reinforce sensor-based predictions, while ensuring the planner remains robust when communication is temporarily unavailable or delayed. A third direction is to explore distributed parallel joint optimization with communication, where each robot explicitly includes other agents as decision variables in its local optimization(along the lines of [28]). Such a formulation would explicitly capture how the robots decisions influence each other, and it would enable a more direct analysis of these interaction effects. For example, one could quantify how much a robots optimized trajectory causes other robots to deviate from their previously intended trajectories to maintain safety, and whether these induced deviations change the homology class of the interaction (e.g., switching the passing side) relative to the previous iteration or the last communicated plan. Finally, it would be interesting to study how communication can be used to negotiate and converge on an explicit system-level decision about the homology class of an interaction, for example along the lines of [30]. At the same time, such a mechanism should still allow the robots to adapt quickly to unexpected events in the environment, such as a pedestrian changing direction. Other approaches could range from simple distributed agreement mechanisms to more advanced approaches, like game-theoretic formulations in which robots exchange information in a distributed way and reach a consistent topology decision.

VIII. Conclusions

This work presented a topology-aware distributed motion planner that uses topological information as a communication trigger to reduce communication load in multi-robot coordination. For the **tested scenarios**, the results indicate that T-DMPC can achieve system-level performance comparable to the always-communicating baseline T-AMPC while requiring less communication. At the same time, the results show that restricting communication to topology changes alone is not sufficient for safe motion planning. Because collision-avoidance constraints are space-time dependent, additional space-time mechanisms are needed capture significant deviations between what a robot is executing and what the other robots are predicting, such as the geometric-deviation trigger or a time-based trigger. In our experiments, we additionally used a more conservative radius which served as safety buffer, the reported outcomes should be interpreted in that context. Finally, we noted that communication should probably not be used as the only information source in a real-world deployment. Since communication can suffer from message loss and delays. Communication-based predictions could perhaps be combined

with onboard sensing and sensor-based prediction. In that view, communicated trajectories can act as an additional, information source that can reinforce the robots belief about what other robots intend to do.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Laura Ferranti, for her valuable insights and for the time she dedicated to discussions on robotics, education, and their implications for society. I also thank Khaled Mustufa for taking the time to discuss directions and ideas for my thesis. Additionally, I would like to thank Diego Martínez Baselga for explaining one of his research papers in detail and for a valuable follow-up discussion that helped shape the direction of my thesis. Finally, I would like to thank my fellow students in the Cognitive Robotics department at Delft University of Technology for their insights, as well as the practical tips and tricks they shared along the way.

REFERENCES

[1] 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.

[2] N. B. Hui, "Coordinated motion planning of multiple mobile robots using potential field method," in *2010 International Conference on Industrial Electronics, Control and Robotics*, Dec. 2010, pp. 6–11.

[3] 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, Jul. 1998.

[4] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 360–366.

[5] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," in *Robotics Research*, B. Siciliano, O. Khatib, F. Groen, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 70, pp. 3–19.

[6] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[7] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 285–292.

[8] H. Zhou and C. Liu, "Distributed Motion Coordination Using Convex Feasible Set Based Model Predictive Control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8330–8336.

[9] J. Tordesillas and J. P. How, "MADER: Trajectory Planner in Multiagent and Dynamic Environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, Feb. 2022.

[10] Y. M. Chung, H. Youssef, and M. Roidl, "Distributed Timed Elastic Band (DTEB) Planner: Trajectory Sharing and Collision Prediction for Multi-Robot Systems," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 10 702–10 708.

[11] S. Wu, G. Chen, M. Shi, and J. Alonso-Mora, "Decentralized Multi-Agent Trajectory Planning in Dynamic Environments with Spatiotemporal Occupancy Grid Maps," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 7208–7214.

[12] Y. Zhang, J. Yang, S. Chen, and J. Chen, "Decentralized cooperative trajectory planning for multiple UAVs in dynamic and uncertain environments," in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec. 2015, pp. 377–382.

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

[14] 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, Apr. 2020.

[15] J. Xin, Y. Qu, F. Zhang, and R. Negenborn, "Distributed Model Predictive Contouring Control for Real-Time Multi-Robot Motion Planning," *Complex System Modeling and Simulation*, vol. 2, no. 4, pp. 273–287, 2022.

[16] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed Nonlinear Trajectory Optimization for Multi-Robot Motion Planning," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 809–824, Mar. 2023.

[17] F. Bertilsson, M. Gordon, J. Hansson, D. Möller, D. Söderberg, Z. Zhang, and K. Åkesson, "Centralized versus Distributed Nonlinear Model Predictive Control for Online Robot Fleet Trajectory Planning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, Aug. 2022, pp. 701–706.

[18] J. Filho, E. Lucet, and D. Filliat, "Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, pp. 657–663.

[19] Y. Zhou, H. Hu, Y. Liu, S.-W. Lin, and Z. Ding, "A Real-Time and Fully Distributed Approach to Motion Planning for Multirobot Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2636–2650, Dec. 2019.

[20] J. Gielis, A. Shankar, and A. Prorok, "A Critical Review of Communications in Multi-robot Systems," *Current Robotics Reports*, vol. 3, no. 4, pp. 213–225, Dec. 2022.

[21] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, "Robust MADER: Decentralized Multiagent Trajectory Planner Robust to Communication Delay in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1476–1483, Feb. 2024.

[22] X. Liu, Z. Miao, and Y. Wang, "Distributed Drone Swarm Trajectory Planner Using Topology Planning Under Communication Delay," in *2024 IEEE International Conference on Unmanned Systems (ICUS)*, Oct. 2024, pp. 1629–1634.

[23] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, Oct. 2012.

[24] M. A. Berger, "Topological Invariants in Braid Theory," *Letters in Mathematical Physics*, vol. 55, no. 3, pp. 181–192, Mar. 2001.

[25] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, Feb. 2017.

[26] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-Planner: An ESDF-free Gradient-based Local Planner for Quadrotors," Dec. 2020.

[27] D. Martinez-Baselga, O. de Groot, L. Knoedler, L. Riazuelo, J. Alonso-Mora, and L. Montano, "SHINE: Social homology identification for navigation in crowded environments," *The International Journal of Robotics Research*, p. 02783649251344639, Jun. 2025.

[28] Y. Chen, S. Veer, P. Karkus, and M. Pavone, "Interactive Joint Planning for Autonomous Vehicles," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 987–994, Feb. 2024.

[29] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, and S. S. Srinivasa, "Winding Through: Crowd Navigation via Topological Invariance," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 121–128, Jan. 2023.

[30] C. Mavrogiannis, J. A. DeCastro, and S. Srinivasa, "Implicit Multiagent Coordination at Uncontrolled Intersections via Topological Braids," in *Algorithmic Foundations of Robotics XV*, S. M. LaValle, J. M. O'Kane, M. Otte, D. Sadigh, and P. Tokekar, Eds. Cham: Springer International Publishing, Feb. 2023, vol. 25, pp. 368–384.

[31] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso–Mora, "Globally Guided Trajectory Planning in Dynamic Environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 10 118–10 124.

[32] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[33] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[34] ZeroMQ. (2026) Pyzmq: Python bindings for zeromq. GitHub repository. [Online]. Available: https://github.com/zeromq/pyzmq

APPENDIX

This appendix provides additional details for the homotopy comparison function in Eq. (6). In particular, we use the H-signature, which compares trajectories via a homology-based approximation. It also provides a compact formulation of the local planner objective in Eq. (15a) and the collision-avoidance constraint in Eq. (15d).

A. H-signature

The H-signature maps a trajectory to a scalar value based on a virtual magnetic field induced by the predicted obstacle motion. Each predicted obstacle trajectory is modeled as a virtual current-carrying wire in the space-time domain, which induces a magnetic field $\mathbf{B}(\mathbf{r})$(for details on $\mathbf{B}(\mathbf{r})$, the reader is referred to [1]). For two trajectories τ_1 and τ_2 that share the same start and end points, we form the closed loop $\tau_1 \cup -\tau_2$, where $-\tau_2$ denotes τ_2 traversed in reverse direction. The H-signature is then defined as the line integral of the magnetic field $\mathbf{B}(\mathbf{r})$ along this loop:

$$\mathcal{H}(\tau_1 \cup -\tau_2) = \int_{\tau_1 \cup -\tau_2} \mathbf{B}(\mathbf{r})\, d\mathbf{r}. \tag{23}$$

Intuitively, this integral indicates whether the looped trajectories enclose the space-time skeleton of an obstacle prediction trajectory: it evaluates to 1 if the obstacle is enclosed by $\tau_1 \cup -\tau_2$, and to 0 otherwise. Since we represent obstacle predictions as piecewise-linear (discrete-time) trajectories, the integral can be evaluated by summing the contributions of the individual prediction segments over the time-grid. *Note: If τ_1 and τ_2 do not share the same end point, we close the loop by adding a straight line segment that connects their end points at t_N of the planning horizon.*

B. Optimization-based local planner (LMPCC)

Cost function. In our implementation, each local planner is an optimization-based planner based on Local Model Predictive Contouring Control LMPCC [32] with second-order unicycle dynamics. At each time step t_k over the horizon N the cost is defined by

$$\begin{aligned} J(\mathbf{x}_i(t_k), \mathbf{u}_i(t_k)) = w_c J_c(t_k) + w_l J_l(t_k) \\ + w_v \left\| v_i(t_k) - v_{\text{ref}} \right\|_2^2 + w_\omega \left\| \omega_i(t_k) \right\|_2^2 \\ + w_a \left\| a_i(t_k) \right\|_2^2 . \end{aligned} \tag{24}$$

Here, $J_c(t_k)$ and $J_l(t_k)$ denote contour and lag error w.r.t. the reference path.

Collision avoidance. We model Robot i and obstacle o as circles with radii r_i and r_o, respectively, and define $r := r_i + r_o$. Collision avoidance with a predicted obstacle position $\hat{\mathbf{p}}^{\,o\to i}(t_k)$ is imposed by $g\big(\mathbf{x}_i(t_k), \hat{\mathbf{p}}^{\,o\to i}(t_k)\big) \le 0$. Following the formulation in [1], the constraint is written as

$$\begin{aligned} g\big(\mathbf{x}_i(t_k), \hat{\mathbf{p}}^{\,o\to i}(t_k)\big) = 1 - \Delta\hat{\mathbf{p}}^{\,o\to i}(t_k)^\top \mathbf{R}\big(\phi_i\big)^\top \\ \begin{bmatrix} \frac{1}{r^2} & 0 \\ 0 & \frac{1}{r^2} \end{bmatrix} \mathbf{R}\big(\phi_i\big) \Delta\hat{\mathbf{p}}^{\,o\to i}(t_k). \end{aligned} \tag{25}$$

where $\Delta\hat{\mathbf{p}}^{\,o\to i}(t_k) := \mathbf{p}_i(t_k) - \hat{\mathbf{p}}^{\,o\to i}(t_k)$ and $\mathbf{R}$ is rotation matrix with ϕ_i the orientation of Robot i (i.e., the two circles do not overlap).