Robotics and Autonomous Systems 180 (2024) 104773

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A decoupled solution to heterogeneous multi-formation planning and coordination for object transportation[☆]

Weijian Zhang, Charlie Street, Masoumeh Mansouri [*]

School of Computer Science at the University of Birmingham, UK

ARTICLE INFO

Keywords:
Formation control
Multi-robot systems
Multi-robot planning and coordination

ABSTRACT

Multi-robot formations have numerous applications, such as cooperative object transportation in intelligent warehouses. In this context, robots are tasked with delivering objects in formation while avoiding intra- and inter-formation collisions. This necessitates the development of solutions for multi-robot task allocation, formation generation, rigid formation route planning, and formation coordination. In this paper, we present a cooperative formation object transportation system for heterogeneous multi-robot systems which captures robot dynamics and avoids inter-formation collisions. Accounting for heterogeneous formations expands the applicability of the proposed robotic transport system. For formation generation, we propose an approach based on conflict-based search, which integrates high-level path planning with low-level trajectory optimisation. For heterogeneous formation planning, we present a two-stage iterative trajectory optimisation framework which adheres to the kinematic constraints of our heterogeneous multi-robot system while retaining formation rigidity. For multi-formation coordination, we use a loosely-coupled algorithm which can guarantee collision-free and deadlock-free formation navigation under minimal assumptions. We demonstrate the efficacy of our approach in simulation.

1. Introduction

A common approach for large object transportation in robotics involves the collective use of multiple robots to collaboratively push, cage, or grasp the objects [1]. This paper addresses cooperative transportation problems where multiple robot formations carry objects on top of them, as shown in Fig. 1. Unlike most cases where the object is rigidly attached to the mobile robots [2–6], our transportation method prevents the object from slipping using the friction between the surface of the object and the surface of the robots [7]. To expand the applicability of this robotic transportation system, our paper proposes a solution that accommodates *heterogeneous* robots, including those with car-like or differential drive kinodynamics. Heterogeneous robots improve the flexibility of collaborative transportation through the use of robots with different load capacities, sizes, and costs. To transport objects, formations must remain *rigid* to prevent the objects from falling, meaning the formation shape must remain as consistent as possible during execution [8].

Collaborative multi-robot object transportation has been widely studied from a control perspective [6,8–10]. Many studies have analysed the interaction between robots and objects from a dynamic perspective. These include coordination and consensus on the manipulation forces and torques that robots need to apply to the object during transportation [4,5,11], estimating the contact point positions of the robots and the inertia parameters of unknown objects [2], and the impact of load manipulation on robot wheels [3]. However, this paper focuses on the kinematic aspects of rigid formation planning and coordination without considering dynamic effects, similar to [8,9,12–15]. Many of these studies assume a single formation which is already formed at the start of the task. This work focuses on the more comprehensive, complex problem of coordinating multiple heterogeneous formations for object transportation. We refer to this problem as *Heterogeneous Multi-formation Planning and Coordination* (H-MFPC). A solution to H-MFPC requires generating and maintaining effective formations until a goal location is reached while avoiding collisions with other formations and obstacles.

☆ Charlie Street and Masoumeh Mansouri are UK participants in Horizon Europe Project CONVINCE, and supported by UKRI grant number 10042096. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising.

* Corresponding author.
E-mail addresses: wxz163@student.bham.ac.uk (W. Zhang), c.l.street@bham.ac.uk (C. Street), m.mansouri@bham.ac.uk (M. Mansouri).

https://doi.org/10.1016/j.robot.2024.104773

Available online 5 August 2024

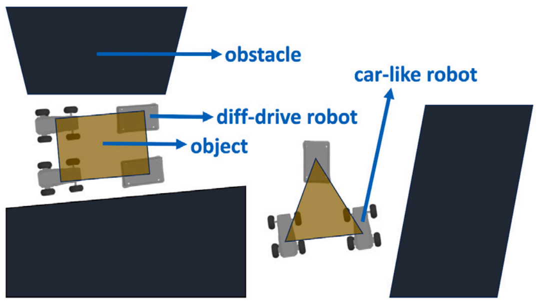


Fig. 1. Object transportation for multiple rigid formations of heterogeneous robots.

Solving the complete H-MFPC problem in a unified way is computationally challenging. Therefore, we propose decomposing H-MFPC into several major sub-problems: formation generation, planning, and coordination. We illustrate this decomposition alongside our proposed framework in Fig. 2. Though we solve each sub-problem independently, their solutions are highly interrelated. For instance, our multi-formation coordination solution heavily relies on the synthesised formation plans. Further, our unified representation of obstacle-free space admits a systematic approach to handling obstacle avoidance constraints in each sub-problem.

For formation generation, we sequentially assign different classes of robots to formations using the Hungarian algorithm [16]. To guide each robot to its formation location, we combine conflict-based search for multi-agent path finding [17] with an improved trajectory optimisation method which leverages the constraint splitting mechanism in [18]. Following formation generation, each formation must navigate towards its destination while maintaining rigidity and avoiding collisions with other formations and obstacles; this is the planning component of the H-MFPC. For this, we introduce an iterative two-stage trajectory optimisation method. In the first stage, one robot is assigned as a leader, and an initial cost-minimising trajectory is synthesised which ensures no robots in the formation collide with static obstacles. The second stage involves synthesising follower trajectories which maintain rigidity. These stages are repeated iteratively until suitable trajectories for the followers are determined which satisfy all kinematic constraints while minimising the cost for the formation. To avoid inter-formation collisions, we adopt a loosely coupled multi-formation coordination algorithm based on [19]; this represents the coordination component of H-MFPC. Finally, we use Lyapunov-based trajectory tracking controllers to follow the robot trajectories and complete object transportation [20, 21].

To the best of our knowledge, ours is the first work to investigate planning and coordination for object transportation using multiple formations of heterogeneous robots. We address the H-MFPC problem through a decoupled framework which solves each sub-problem through state-of-the-art methods. This paper extends our prior work presented in [22], which only considered homogeneous formations. Addressing heterogeneity introduces new challenges, and requires us to modify the solutions for three major sub-problems compared to our earlier work. The main contributions of this work are summarised as follows:

- A comprehensive H-MFPC framework which integrates formation generation, planning, and coordination techniques for heterogeneous formations.
- An efficient formation generation approach for heterogeneous multi-robot systems which synthesises collision-free and kinematically feasible trajectories in unstructured environments.
- A cost-optimal formation planning method that maintains rigidity for heterogeneous formations.
- A loosely-coupled multi-formation coordination algorithm for ensuring deadlock-free and collision-free navigation among formations.

We demonstrate the efficacy of our H-MFPC framework through extensive experiments in simulation. A video demonstrating our proposed framework can be found at https://youtu.be/A_S-e0mkLGY.

2. Related work

To the best of our knowledge, no works have attempted to simultaneously address the sub-problems illustrated in Fig. 2. In this section, we discuss previous work addressing these sub-problems, namely formation generation, formation planning, and formation coordination.

2.1. Formation generation

Formation generation requires a team of robots to organise into a predefined configuration [23], a process synonymous with multi-robot trajectory planning (MRTP). The MRTP problem involves planning safe and efficient paths for multiple robots in a given environment, enabling them to move from their starting positions to their target destinations while avoiding collisions with each other and obstacles. Current literature on solving the MRTP problem can be divided into coupled and decoupled methods. When considering nonlinear robot dynamics, MRTP can only be formulated as a non-convex nonlinear programming (NLP) problem, intensifying its complexity in obstacle-dense environments. To account for this complexity, a common approach is to use a centralised two-stage method that encompasses multi-agent path finding (MAPF) and trajectory optimisation [18,24,25]. For instance, in [24], a conflict-based search (CBS) solver is used to generate reference paths for each robot during optimisation. Model predictive control (MPC) is then applied to produce executable trajectories based on these paths. However, deviations from these predetermined paths may result in potential collisions among robots. Another example involves the introduction of conflict-based MPC (CB-MPC), which resolves conflicts similarly to CBS [18]. CBS is then integrated with MPC as the low-level planner to devise collision-free and executable motion plans for multiple robots in a receding horizon fashion. CB-MPC treats each robot as a circle for collision avoidance purposes, which is conservative and may result in failure when robots are highly concentrated around the target configuration. Conversely, in this paper, we consider the geometric shapes of the robots, which enables the usage of the full potential solution space. Additionally, heuristic terms are introduced to incrementally resolve conflicts among robots, ensuring scalability and robustness.

To enhance computational efficiency during MRTP, Li et al. [25] propose a prioritised and sequential trajectory optimisation method based on grouping strategies. In this method, lower-priority robots must avoid the trajectories of higher-priority robots that have already planned. This approach exhibits robust performance in environments with fewer constraints, but in densely constrained settings, inappropriate prioritisation may result in deadlocks or collisions. Wen et al. [26] present a car-like CBS (CL-CBS) approach that explicitly captures the non-holonomic kinematics of the agent. They introduce a conflict tree to record collisions arising from agent shapes, and propose a spatiotemporal hybrid A^* algorithm to mitigate those collisions. However, the trajectories generated by CL-CBS do not ensure curvature continuity. Additionally, Ouyang et al. [27] propose a two-stage approach in which a feasible homotopy class is identified from which a locally optimal solution is found. These latter two methods are tailored specifically for car-like robots.

MRTP can also be solved in a distributed manner. Reactive-based methods, such as artificial potential field methods [22] and optimal reciprocal collision avoidance [28] can avoid local collisions during navigation. These techniques are often computationally efficient and fairly

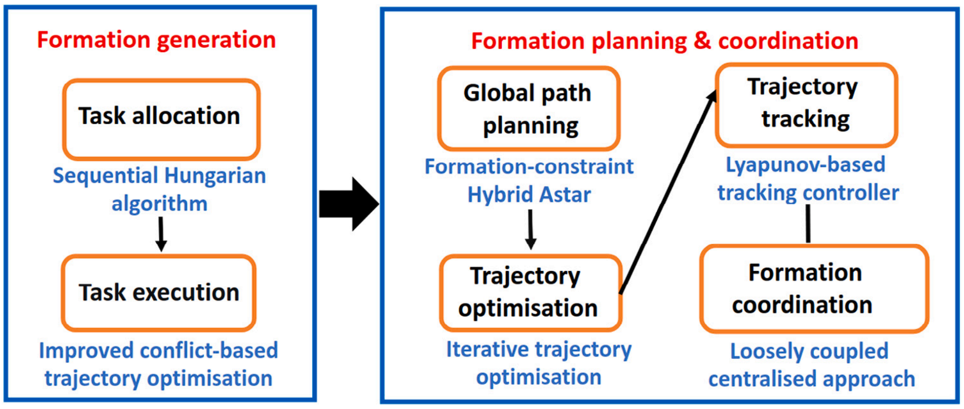

Formation generation
Task allocation
Sequential Hungarian algorithm
Task execution
Improved conflict-based trajectory optimisation
Formation planning & coordination
Global path planning
Formation-constraint Hybrid Astar
Trajectory optimisation
Iterative trajectory optimisation
Trajectory tracking
Lyapunov-based tracking controller
Formation coordination
Loosely coupled centralised approach

Fig. 2. The proposed problem decomposition for H-MFPC.

Table 1
A comparison of state-of-the-art formation planning techniques.

Planner	Robot type	Heterogeneous	Rigid	Global planning	Avoids inflation of formation footprint or obstacles
[6]	Holonomic	✗	✗	✓	✓
[8]	Diff-drive	✗	✓	✓	✗
[9]	Diff-drive	✗	✓	✗	✗
[10]	Car-like	✗	✗	✓	✓
[12]	Holonomic	✗	✓	✓	✓
[13]	Holonomic	✗	✓	✓	✗
[14]	Car-like	✗	✓	✗	✗
[15]	Car-like	✗	✓	✗	✓
[30]	Diff-drive	✗	✗	✗	✓
[31]	Holonomic	✗	✗	✗	✗
Our planner	Car-like & Diff-drive	✓	✓	✓	✓

scalable. However, due to their inability to predict future robot behaviour, they fail to synthesise high-quality trajectories and deadlock-free operation in densely populated environments. Further, in [29] a distributed trajectory planning method is proposed which leverages distributed model predictive control (DMPC). This approach enables each robot to execute real-time trajectory planning and resolve on-demand conflicts. Though efficient and scalable, it does not assure the absence of deadlocks, and the resulting trajectories are often suboptimal.

2.2. Formation planning

Many formation planning techniques leverage classical AI search methods. In [8], relaxed A^* [32] is used to build a spline-based formation planner, where obstacles are inflated by a circle which encompasses the formation's footprint. Such inflation is overly conservative, restricting the feasible solution space and causing planning to fail in the worst case. In this paper, we do not inflate or approximate obstacles, as the formation's convex hull is constrained to the obstacle-free convex region. Constraint optimisation techniques can be used to plan for non-rigid formations to allow for flexible obstacle avoidance [6,31]. These techniques assume holonomic robots however, making them unsuitable for the heterogeneous non-holonomic robots considered in this paper. Formation planning can also be addressed through sampling-based methods. In [12], an improved rapidly-exploring random tree (RRT) method is presented for rigid formation planning which explicitly considers the geometric constraints of the formation. Alternatively, in [13], a graph search-based global path planner is presented that can handle rigid or flexible formations of holonomic robots in complex static environments with dense obstacles. These methods are probabilistically complete, but do not consider trajectory optimisation. This may result in trajectories with discontinuities which cannot be accurately followed by non-holonomic robots. Formation planning has also been tackled through deep reinforcement learning. In [30], each robot learns how to independently adjust its position in the formation to avoid obstacles. This approach can scale to different numbers of robots, however it is challenging to obtain sufficient training data.

The formation planning problem has been formalised in terms of optimal control. In [14], a distributed adaptive trajectory optimisation method for car-like robots with state and input constraints is presented, but does not explicitly consider formation rigidity. A hierarchical quadratic programming approach is presented in [9]. This method can detect and avoid *a priori* unknown obstacles during execution while maintaining a high level of rigidity. However, the lack of a global planner may lead to local minima, reducing performance. In [15], the formation leader's information is used to compute the expected trajectories of car-like follower robots. These trajectories are then tracked through MPC. Though the leader's trajectory is guaranteed to be collision-free, this does not hold for the followers, where collision avoidance is not explicitly considered. In [10], nets are used to transport objects in formations of car-like robots. This allows for object transportation in unstructured environments. Under this net-based approach, formations are not rigid, making it unsuitable for the H-MFPC problem tackled in this paper.

In Table 1, we summarise the state-of-the-art in formation planning. Here, we compare methods based on the robots in the formation, whether formations are rigid, whether a global planner is used, and whether the formation footprint or obstacles are inflated during planning.

2.3. Formation coordination

In this paper, we explicitly address collision avoidance among multiple formations, which has received little attention in the literature. Multi-formation coordination is similar to the multi-robot multi-target motion planning problem. However, multi-formation coordination solutions must consider the constraints of the formation in addition to the kinematic, dynamic, and collision avoidance constraints imposed on each robot. Formation constraints significantly reduce the feasible

Table 2
Nomenclature.

$\mathcal{W}$	2D workspace.
$\mathcal{O}$	A set of static obstacles.
$\mathcal{W}_{free}$	Obstacle-free workspace.
C_g	A seed point used for IRIS.
$\mathbf{A}_g$, $\mathbf{b}_g$	A set of hyperparameters used to describe a convex region.
C_c	A continuous point inside a convex polygon.
$\mathcal{R}$	A team of heterogeneous robots.
Q_i	The ith robot's configuration space.
R_i	Robot i's footprint.
f_i^{type}	The dynamics model for robot i.
g_i^{type}	A set of kinematic constraints for robot i.
Q_i^{free}	The set of obstacle-free configurations for robot i.
$q_i(t)$	Robot i's configuration at time t.
$z_i(t)$	The state for the robot i at time t.
$\mathcal{F}$	A set of heterogeneous formations.
z_e	The eth formation.
$\mathbf{d}_i$	The relative distance of the ith robot to the formation centre.
ref_i^k	The kth waypoint of the discrete reference path for robot i.
SC_i	The corresponding safe corridors for the robot i's trajectory.
$v_{i,l}^k$	The four vertices of robot i.
T_i^{ref}	The initial trajectory for robot i.
$V_{e,c}$	A set of vertices representing formation e's footprint.
SC_e	The safe corridors for formation e.
T_i	The trajectory for robot i in Section 5.
$\mathbf{p}_i$	The sequence of configurations along robot i's trajectory.
$\varepsilon(\mathbf{p}_i)$	The spatial envelope for robot i.
σ	An arc length used to parameterise $\mathbf{p}_i$.
$F_i(\mathbf{p}_i(\sigma))$	The footprint of the formation led by robot i.
C	The critical section between two formations.
l_i^C	The instant immediately before a formation enters C.
u_i^C	The instant immediately after a formation leaves C.
$m_i(t)$	The precedence constraint applied to robot i at time t.
$\mathcal{T}(t)$	The set of precedence constraints active at time t.

motion of each robot, increasing the difficulty of inter-formation collision avoidance. A neighbour-based multi-formation control protocol is proposed for inter-formation collision avoidance in obstacle-free environments in [33]. However, its heavy reliance on inter-formation communication topology compromises scalability. In [34], an improved version of the optimal reciprocal collision avoidance algorithm (ORCA) is presented for non-holonomic robot formations which aims to avoid collisions with other formations and obstacles. However, the myopic nature of this method results in deadlocks, and robots do not consistently reach the correct orientation at the goal state. Additionally, the assumption of circular robots often causes each robot to occupy too much space in confined environments, leading to planning failures. In [22], a prioritised dynamic window approach is proposed for formation coordination. However, this method may also lead to deadlocks and collisions in narrow environments. In this paper, we adapt the loosely-coupled multi-robot coordination method in [19] to address inter-formation collisions rather than collisions among individual robots.

3. Preliminaries

In this section, we define our assumptions over the workspace, robots, and formations. In Table 2, we describe the notation used throughout the paper.

3.1. Workspace

Let $\mathcal{W} \subset \mathbb{R}^2$ be a 2D workspace containing a set of static obstacles $\mathcal{O} \subset \mathcal{W}$, where the obstacle-free workspace is denoted $\mathcal{W}_{free} = \mathcal{W} \setminus \mathcal{O}$. We assume that the obstacles are convex polygons, which allows for a safe approximation of most irregularly shaped obstacles. In this paper, we represent the obstacle-free workspace as a union of convex polygons. For this, we employ IRIS [35], a convex decomposition technique which given an initial seed point alternately solves two

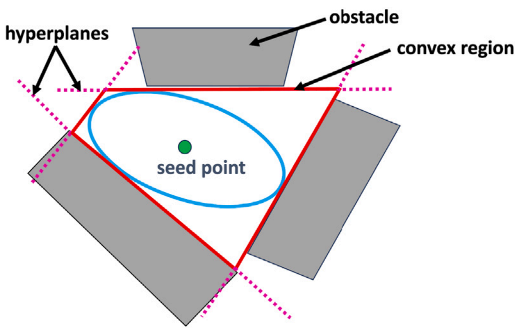


Fig. 3. The outcome of running IRIS in a simple environment with three obstacles and a seed point. Grey areas are obstacles, the red polygon is the convex region produced by IRIS, and the pink dashed lines show the separating hyperplanes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

convex optimisations: (1) finding a set of hyperplanes that separate an ellipse from $\mathcal{O}$ via quadratic optimisation; and (2) finding the largest ellipse within the polygon via a semi-definite programme. An example polygon synthesised by IRIS is shown in Fig. 3. To decompose the entire workspace into convex polygons, we run IRIS for a set of equally spaced seed points across the obstacle-free workspace, denoted $C_g \in \mathcal{W}_{free}$. For each seed point C_g, we obtain a convex polygon defined by a matrix $\mathbf{A}_g \in \mathbb{R}^{n_g \times 2}$ and a vector $\mathbf{b}_g \in \mathbb{R}^{n_g}$, where n_g is the number of hyperplanes, equivalent to the number of edges that depict this convex region. A continuous point C_c falls inside this polygon if $\mathbf{A}_g C_c \le \mathbf{b}_g$.

3.2. Robots

We consider a team of heterogeneous rectangular robots $\mathcal{R} = \{1, \dots, n\}$. Each robot i is described by a tuple $r_i = \langle Q_i, R_i, f_i^{type}, g_i^{type} \rangle$, where Q_i is the robot's configuration space. Robot i's configuration at time t is given by $q_i(t) = (x_i(t), y_i(t), \theta_i(t)) \in Q_i$, where $x_i(t), y_i(t) \in \mathbb{R}$ is the robot's position, and $\theta_i(t) \in [-\pi, \pi)$ is its orientation. $R_i(q_i(t))$ defines robot i's footprint at configuration $q_i(t)$, i.e. the obstacle-free space occupied by the robot. With this, we write $Q_i^{free} = \{q_i \in Q_i : R_i(q_i) \cap \mathcal{O} = \emptyset\}$ to denote the set of obstacle-free configurations for robot i. f_i^{type} is the dynamics model for robot i, which in this paper we consider to be car-like (a bicycle model) or differential drive (a unicycle model); we abbreviate these models to car and $diff$, respectively. g_{type} is a set of kinematic constraints for robot i. For a car-like robot, f_i^{car} and g_i^{car} are given by:

$$f_i^{car}: \frac{d}{dt}\begin{bmatrix} x_i(t) \\ y_i(t) \\ v_i(t) \\ \phi_i(t) \\ \theta_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t) \cdot cos\theta_i(t) \\ v_i(t) \cdot sin\theta_i(t) \\ a_i(t) \\ \omega_i(t) \\ v_i(t) \cdot tan\phi_i(t)/L \end{bmatrix}, \quad i \in \mathcal{R}, t \in [0, t_{f_i}], \tag{1}$$

$$g_i^{car}: \begin{bmatrix} -v_m^{car} \\ -\omega_m^{car} \\ -a_m^{car} \\ -\phi_m^{car} \end{bmatrix} \le \begin{bmatrix} v_i(t) \\ \omega_i(t) \\ a_i(t) \\ \phi_i(t) \end{bmatrix} \le \begin{bmatrix} v_m^{car} \\ \omega_m^{car} \\ a_m^{car} \\ \phi_m^{car} \end{bmatrix}, \quad i \in \mathcal{R}, t \in [0, t_{f_i}], \tag{2}$$

where $\phi_i(t)$, $v_i(t)$, $\omega_i(t)$, and $a_i(t)$ are the steering angle, velocity, angular velocity, and acceleration applied at time t, respectively. Further, L denotes the wheelbase of the car-like robot, signifying the distance between its front and rear axle, and t_{f_i} is the finite planning and coordination horizon. From this, the *state* for a car-like robot at time t is given by $z_i(t) = [q_i(t), \phi_i(t), v_i(t)]$, and the control input is given by $u_i(t) = [a_i(t), \omega_i(t)]$. g_i^{car} is constrained by the car-like robot's maximum velocity v_m^{car}, maximum angular velocity ω_m^{car}, maximum acceleration

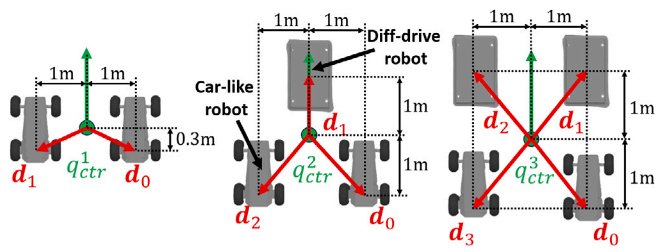


Fig. 4. Linear (2 car-like robots), triangular (1 diff-drive robot + 2 car-like robots), and rectangular (2 car-like robots + 2 diff-drive robots) formations. In each formation, each robot is positioned according to its state vector $\mathbf{d}_i$ relative to the centroid state q_{ctr} of the formation, with respect to the body frame.

a^{car}_m, and maximum steering angle ϕ^{car}_m. For differential drive robots, f^{diff}_i and g^{diff}_i are given by:

$$f^{diff}_i : \frac{d}{dt}\begin{bmatrix} x_i(t) \\ y_i(t) \\ v_i(t) \\ \theta_i(t) \\ \omega_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t)\cdot cos\theta_i(t) \\ v_i(t)\cdot sin\theta_i(t) \\ a_i(t) \\ \omega_i(t) \\ \Omega_i(t) \end{bmatrix}, \quad i \in \mathcal{R}, t \in [0, t_{f_i}], \tag{3}$$

$$g^{diff}_i : \begin{bmatrix} -v^{diff}_m \\ -\omega^{diff}_m \\ -a^{diff}_m \\ -\Omega^{diff}_m \end{bmatrix} \leq \begin{bmatrix} v_i(t) \\ \omega_i(t) \\ a_i(t) \\ \Omega_i(t) \end{bmatrix} \leq \begin{bmatrix} v^{diff}_m \\ \omega^{diff}_m \\ a^{diff}_m \\ \Omega^{diff}_m \end{bmatrix}, \quad i \in \mathcal{R}, t \in [0, t_{f_i}], \tag{4}$$

where $\Omega_i(t)$ is the angular acceleration, and all other variables are the same as for car-like robots. The *state* of a differential drive robot is written as $z_i(t) = [q_i(t), v_i(t), \Omega_i(t)]$, and the control input is $u_i(t) = [a_i(t), \omega_i(t)]$. g^{diff}_i is constrained by the diff-drive robot's maximum velocity v^{diff}_m, maximum angular velocity ω^{diff}_m, maximum acceleration a^{diff}_m, and maximum angular acceleration Ω^{diff}_m.

3.3. Formations

In this paper, we decompose our set of robots $\mathcal{R}$ into a set of heterogeneous formations $\mathcal{F} = \{1, \ldots, n_f\}$. We consider three formation types: linear formations which contain two car-like robots; triangular formations which contain one differential drive robot and two car-like robots; and rectangular formations, which contain two car-like robots and two differential drive robots (see Fig. 4). Car-like robots can only maintain a rigid formation if they are driving side by side or in a straight line [36]. This is due to the car-like robots' inability to turn in place [37]. Therefore, in this paper we assume all car-like robots in the same formation are aligned in parallel. Let formation e contain n_{f_e} robots, the set of which is denoted $\mathcal{F}_e$. The eth formation is given by the tuple $\mathcal{Z}_e(t) = \langle q^e_{ctr}(t), \{\mathbf{d}^{n_{f_e}-1}_{i=0}\}\rangle$, where $q^e_{ctr}(t) = (x^e_{ctr}(t), y^e_{ctr,}(t), \theta^e_{ctr})$ is the configuration at the centre of the formation at time t, and $\mathbf{d}_i = (d_{x,i}, d_{y,i})^T$ is the relative distance of the ith robot to the formation centre, which should remain fixed. Note that the formation geometry can be altered by adjusting $\mathbf{d}_i$.

4. Formation generation

In this section, we address formation generation, the first sub-problem of H-MFPC in Fig. 2.

4.1. Problem formulation

The formation generation problem has two components. First, each robot i must be allocated a formation location q^g_i given its initial configuration q^s_i. For formation e, the formation location q^g_i can be deduced from q^e_{ctr} and $\mathbf{d}_i$ (see Fig. 4). To avoid collisions between robots at start or goal positions, we assume $R(q^s_i) \cap R(q^s_j) = \emptyset$ and $R(q^g_i) \cap R(q^g_j) = \emptyset, \forall i, j \in 1, \ldots, n,\ i \neq j$, i.e. no robot footprints overlap at the start or goal configurations. Following task allocation, each robot $i \in \mathcal{R}$ must synthesise a trajectory T_i which begins at q^s_i and finishes at q^g_i. Upon reaching q^g_i, robot i should maintain its position. Robot trajectories should be collision free, i.e. $R(q_i(t)) \cap \mathcal{O} = \emptyset,\ t \in [0, t_{f_i}]$, and $R(q_i(t)) \cap R(q_j(t)) = \emptyset$. We refer to this problem as heterogeneous multi-robot trajectory planning (H-MRTP).

4.2. Task allocation

For multi-robot task allocation, we use the optimal Hungarian algorithm [16] with a Euclidean distance cost function. Each formation type described in Section 3.3 requires different numbers of each robot type. For example, triangular formations require two car-like robots and one differential drive robot. To enforce these constraints during task allocation, we apply the Hungarian algorithm twice, first to allocate the car-like robots, and then to allocate the differential drive robots.

4.3. Heterogeneous multi-robot trajectory planning (H-MRTP)

We now present a two-stage approach for H-MRTP. First, we solve the multi-agent path finding problem on an approximate representation of the environment to synthesise a conflict-free reference path for each robot. Using these reference paths, we generate safe corridors for each robot, and then solve a trajectory optimisation problem within these corridors.

4.3.1. Discrete reference path generation

To synthesise a reference path for each robot, we discretise the workspace into a 2D grid map, where each cell is either free or occupied. We then apply the enhanced conflict-based search (ECBS) algorithm [38] for multi-agent pathfinding to synthesise a collision-free path for each robot on the grid map. ECBS is a centralised, complete, and suboptimal algorithm that resolves conflicts in a high-level search tree to synthesise collision-free paths. The discrete reference path for robot i is an ordered sequence of K waypoints, where the kth waypoint is denoted ref^k_i.

4.3.2. Trajectory optimisation

We now describe how to synthesise robot trajectories from the ECBS reference paths. We begin by defining the collision avoidance constraints all trajectories must satisfy.

Obstacle Avoidance Constraints. To avoid collisions with obstacles, we compute a safe convex region SC_i for robot i which its trajectory must remain inside. Given a trajectory T_i for robot i, we compute K discrete waypoints by taking the robot location at time $k\Delta t$ for $k \in \{1, \ldots, K\}$, where $K\Delta t = t_{f_i}$. These waypoints are then used as seed points for IRIS, as described in Section 3.1. The resulting convex regions then form the safe corridor, i.e. $SC_i = \bigcup_{k=\{1,\ldots,K\}} SC^k_i$. Each convex region in the safe corridor must intersect with the next region, i.e. $SC^k_i \cap SC^{k+1}_i \neq \emptyset, \forall k \in \{1, \ldots, K-1\}$. If this condition is not met, we use uniform interpolation along the robot's trajectory to generate further IRIS seed points. To avoid collisions with obstacles in $\mathcal{O}$, robot i must keep its trajectory within the safe corridor. Formally, let $p^k_i = (x^k_i, y^k_i)$ be robot i's position at the kth waypoint. Robot i's footprint at waypoint k can be described with the four vertices $\{V^k_{i,l} = (x^k_{i,l}, y^k_{i,l}), l = 1 \ldots 4\}$. Further, let $\mathbf{l}_{i,l}$ be the relative coordinate of the lth vertex within the body frame. The collision avoidance constraint can be written as:

$$V^k_{i,l} \in \mathbb{R}^2 : V^k_{i,l} = \{p^k_i + \mathbf{R}(\theta^k_i)\cdot \mathbf{l}_{i,l}\} \in SC^k_i, \\ \forall i \in \mathcal{R},\ l \in \{1, \ldots, 4\},\ k \in \{1, \ldots, K\}, \tag{5}$$

where $\mathbf{R}(\theta^k_i)$ is the rotation matrix representing the orientation of the body frame relative to the world frame.

Inter-Robot Collision Avoidance Constraints. To avoid collisions between robots, the footprints of two or more robots should never intersect, i.e. $R(q_i(t)) \cap R(q_j(t)) = \emptyset,\ \forall t \in [0, t_{f_i}]$. In practice, we can implement this collision avoidance constraint using the inequality constraints presented in [39] for collision checking between convex polygons. Briefly, the inequality constraints in [39] check that each vertex of one robot's footprint lies outside the footprint of the other robots.

A Conflict-Based Approach to Trajectory Optimisation. The first step for trajectory optimisation is to transform the discrete reference path ref_i for robot i into a trajectory T_i^{ref}. We do this by assigning a time value $t = k\Delta t$ to the kth waypoint in the path, i.e. $T_i^{ref} = \{ref_i^k, k\Delta t\}_{k=1}^{K}$. Given the initial trajectories for each robot $\{T_i^{ref}\}_{i=1}^{n}$ and the corresponding safe corridors for those trajectories $\{SC_i\}_{i=1}^{n}$, trajectory optimisation for robot i can be formulated as a nonlinear optimisation problem:

$$\min_{q_i, u_i} \quad J_i = \sum_{k=1}^{K} w_e {\Delta q_i^k}^T \Delta q_i^k + \sum_{k=1}^{K-1} w_s {\Delta u_i^k}^T \Delta u_i^k + w_p \Phi_i \tag{6a}$$

$$\text{s.t.} \quad q_i^{k+1} = f(q_i^k, u_i^k), \forall k \tag{6b}$$

$$(5) \tag{6c}$$

$$\underline{u} \le u_i^k \le \overline{u}, \forall k \tag{6d}$$

$$z_i^0 = z_i^{initial} \tag{6e}$$

$$R(q_i(k)) \cap R(q_j(k)) = \emptyset, \forall j \neq i, k, \tag{6f}$$

where w_e, w_s and w_p are positive parameters. The objective function is shown in (6a). The first term imposes a penalty on any deviation from the initial reference path, represented by $\Delta q_i^k = q_i^k - ref_i^k$. This aims to mitigate robot collisions, as the reference trajectories are collision free, but may be suboptimal. The second objective function term penalises large changes in the control input to improve the smoothness of the synthesised trajectory. In formation generation, robots should start at their initial location and finish at their formation location. This is typically enforced as a hard constraint. However, car-like robots cannot rotate in place, and there may be other robots near a particular robot's formation location. This makes it difficult for car-like robots to stop perfectly at their formation location, making optimisation problem (6) challenging to solve. To mitigate this, we instead use a soft constraint which penalises robots for deviating from their formation location. This is captured in the third term in (6), where $\Phi_i = (z_i^N - z_i^{final})^2$ measures the deviation from robot i's final state. Φ_i can also be viewed as a proxy for the feasibility of the optimised trajectory. We use this to validate our formation generation approach in Section 8.1. Constraint (6b) constrains robot i to follow its discrete-time dynamics model, which is either (1) for car-like robots, or (3) for differential drive robots. Similarly, (6d) enforces the kinematic constraints of the robot, dependent on its type. Constraint (6c) enforces collision avoidance with static obstacles by constraining the robot to its safe corridor. Constraint (6e) sets constraints on the robot's initial state. Finally, (6f) enforces collision avoidance with other robots.

The trajectory optimisation problem for each robot i is intractable for large teams due to the non-convex constraints in (6f). To address this, we employ a CBS-like algorithm for H-MRTP, as proposed in [18]. The core idea is to apply inter-robot collision avoidance constraints incrementally in a high-level search tree as they are needed. This is motivated by the fact that the paths of many robots are naturally far apart, foregoing the need for collision avoidance constraints. A potential collision between robot i and j is represented by the tuple $\langle i, j, t \rangle$, where t is the timestep at which (6f) is violated.

Our CBS-based H-MRTP algorithm is presented in Alg. 1. We begin with the reference paths, footprint, dynamics model, and kinematic constraints for each robot. Alg. 1 runs a high-level best-first search, where each node H stores a joint trajectory $H.traj$, the corresponding cost $H.cost$, and any inter-robot collision avoidance constraints $H.constraints$. In line 4, we generate initial reference trajectories from the reference paths for each robot. In line 6, each robot then solves

Algorithm 1: H-MRTP

```
   Input: ${ref_i}_{i=1}^n$, ${r_i}_{i=1}^n$, $map$
   Output: Collision-free trajectories $T$
1  $Q \leftarrow PriorityQueue()$;
2  $H \leftarrow SearchNode()$;
3  foreach each $i \in \mathcal{R}$ do
4      $H[i].traj \leftarrow$ GenerateTrajectory($ref_i$);
5      $SC_i \leftarrow$ GenerateSafeCorridors($H[i].traj$, $map$);
6      $[solution, \Phi] \leftarrow$ SolveNLP($H[i]$, $r_i$, $SC_i$);
7      $H[i].traj \leftarrow solution$;
8      $H[i].cost \leftarrow$ CalcCost($r_i$, $\Phi$, $H$);
9  $Q.Push(H)$;
10 while $!Q.Empty()$ and time not exceeded do
11     $H \leftarrow Q.Pop()$; //Lowest cost node
12     if $H.traj$ has no conflicts then
13         return $H.traj$;
14     $\{i, j, z, t\} \leftarrow$ FindFirstConflict($H.traj$);
15     foreach $c \in \{i, j\}$ do
16         $H' \leftarrow Copy(H)$;
17         $H'.constraints \cup \langle c, z, t \rangle$;
18         $[solution, \Phi_c] \leftarrow$ SolveNLP($H'$, $r_c$, $SC_c$);
19         $H'[c].traj \leftarrow solution$;
20         $SC_c \leftarrow$ GenerateSafeCorridors($H'[c].traj$, $map$);
21         $H'[c].cost \leftarrow$ CalcCost($r_c$, $\Phi_c$, $H'[c].traj$);
22         $Q.Push(H')$;
```

(6) to obtain a kinematically feasible trajectory without considering the other robots. The cost of the resulting trajectories are then computed in line 8, where CalcCost() computes $J_{t_i} = J_{l_i} + J_{t_i} + J_{f_i}$ for each robot i. Here, J_{l_i}, J_{t_i} and J_{f_i} represent the normalised trajectory length for robot i, a value based on robot i's type (1 for car-like and 0 for differential drive), and the normalised Φ_i for all robots, i.e. the deviation from the start and goal positions. In line 10, we begin the main search loop. Here, we maintain a priority queue where nodes with lower cost have higher priority. In each iteration, we pop the lowest cost node from the priority queue (line 11). If this trajectory is collision-free, i.e. it satisfies (6f), then we return the solution, which will have minimal cost (lines 12–13). Otherwise, we find the earliest conflicting robot pair (i, j) in the current node, alongside the conflict state z and conflict time t. From this potential collision, we generate two child nodes with different constraints, one in which the first robot must avoid the second, and vice versa (lines 16–17). For each child node, we then synthesise a new trajectory for the conflicting robot by solving (6), update the safe corridor and cost, and then push the new node to the priority queue (lines 18–22). We update the robot's safe corridor as the revised trajectory may change the corresponding homotopy class and thus the safe corridor.

Our approach follows the same high-level CBS search procedure as [18]. However, [18] assume robots have a circular footprint, which can cause conservative behaviour in confined spaces, or failure to find a solution in the worst case. We adapt the low-level trajectory optimisation to handle robots with rectangular footprints, and demonstrate how this improves performance in Section 8.

5. Formation planning

After each robot has reached its formation location, we must solve the formation planning problem. Here, we must synthesise a safe and kinematically feasible trajectory for each robot such that its formation moves from an initial configuration to the target configuration while maintaining rigidity and obstacle avoidance. Maintaining a perfectly rigid formation is extremely difficult due to the non-holonomic nature of robots, the necessity of obstacle avoidance, and actual control errors. In optimisation problems, capturing formation rigidity as a hard constraint would drive the solver to significantly slow the robots to achieve more precise control. Such control would involve frequently adjusting speed and direction to maintain rigidity, ultimately leading to an ill-conditioned optimisation problem. In the worst case, imposing hard formation constraints can result in the failure of the optimisation solver. Therefore, our formation planner captures rigidity as a soft constraint, allowing for a certain degree of deviation. In this section, to balance rigidity, task completion time, and the smoothness of control inputs, we present an iterative two-stage optimisation framework that continuously adjusts the weight parameters of the cost function until the solution meets a predetermined acceptable formation error. We begin by formalising the formation planning problem.

5.1. Problem formulation

Efficient trajectories for heterogeneous formations must be smooth, fast, and maintain formation rigidity. To achieve this, we formulate the formation trajectory planning problem for each robot y in the eth formation as follows:

$$\min_{z_y(t),u_y(t),t_{f_y}} \quad J_y(z_y(t), u_y(t)) \tag{7a}$$

$$\text{s.t.} \quad \dot{z_y}(t) = f(z_y(t), u_y(t)), \tag{7b}$$

$$z_y(0) = z_y^{initial}, \; u_y(0) = u_y^{initial}, \tag{7c}$$

$$z_y(t_{f_y}) = z_y^{final}, \; u_y(t_{f_y}) = u_y^{final}, \tag{7d}$$

$$\mathcal{G}(z_y(t), u_y(t), t) \preceq 0, \; \forall t \in [0, t_{f_y}],$$

$$\forall y \in \mathcal{F}_e. \tag{7e}$$

Here, (7a) is the trajectory cost function, which we describe later in this section, and t_{f_i} is the trajectory duration. Eq. (7b) is the dynamic constraint associated with the robot. Eqs. (7c) and (7d) define the initial and final state constraints of the robot, in addition to the initial control and final control input constraints. Function $\mathcal{G}$ encompasses the robot's obstacle avoidance constraints and kinematic constraints. In the remainder of this section, we elaborate on how we solve this optimal control problem.

5.2. Stage one: Trajectory optimisation for the formation leader

Our first step towards formation planning is to select a single robot in the formation as leader. The leader will then be used to synthesise collision-free, kinematically feasible trajectories for all robots in the formation. The follower robots must follow these trajectories to maintain formation rigidity. In this paper, we select a car-like robot as leader. If we selected a differential drive robot as leader, car-like robots would have to maintain a constant relative distance to a differential drive trajectory. This is hard to achieve for car-like robots, as their inability to rotate in place restricts their ability to make tight turns or precise directional adjustments [37] In contrast, differential drive robots can easily follow a trajectory generated for a car-like robot due to their increased maneuverability. Car-like followers can also accurately maintain a relative distance to a car-like trajectory, assuming they are aligned in parallel with the leader [36].

5.2.1. Cost function

To synthesise a trajectory for leader robot i, we optimise the following cost function:

$$J_i = w_s \int_{\tau=0}^{t_{f_i}} (a_i^2(\tau) + u_i(\tau)u_i{}^T(\tau))d\tau + w_t t_{f_i}, \tag{8}$$

where w_s and w_t are positive weight parameters. The first cost term encourages the robot to follow a smooth trajectory towards the target by penalising high control inputs and acceleration. If the control input and acceleration is high, the robot is more likely to experience sharp changes in movement. The second cost term captures the trajectory duration.

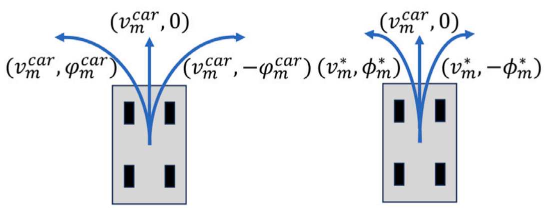


Fig. 5. Left: The motion primitives proposed in [42]. Right: The motion primitives used in this paper.

5.2.2. Initial guess

The optimal control problem in (7) can be reformulated into a nonlinear optimisation problem. It is common to start exploring the highly complex search space with an initial guess derived from the approximation of some constraints. The initial guess can also determine the homotopy class of the optimal trajectory [40], aiding in the identification of topologically equivalent paths, and thus reducing computational complexity. We also use the initial guess to construct the robot's safe corridor [41], which defines a safe boundary for collision-free navigation in a way that simplifies the obstacle avoidance constraints, improving computational efficiency. To synthesise an initial guess, we extend hybrid A* [42] to handle the curvature and collision avoidance constraints of the formation. We refer to this as formation-constrained hybrid A* (FCHA*). We now describe two key components of FCHA*: the motion primitives which form the actions of our search tree, and the collision checking mechanism which tests if a search node is reachable.

Motion Primitives. In hybrid A^* [42], search nodes contain the continuous state of the car-like leader. Nodes are expanded by applying a set of motion primitives, such as turn left, go forward, or turn right (as shown on the left in Fig. 5). To maintain rigidity when the leader executes a turn, followers on the outside of the leader may need to exceed their maximum velocity, and followers on the inside may need a below-minimum turning radius, causing the robot to move backward. The transition from forward to backward motion would initiate a jerk in the robot's movement, which is detrimental to maintaining formation rigidity [8]. To avoid violating the kinematic constraints of the followers or causing the followers to move backward, we integrate the followers' kinematic constraints into the leader's motion primitives, as shown on the right in Fig. 5. The minimum turning radius of the jth follower robots $|r_{m,j}^{type}|$ depends on their type, with car-like robots and differential drive robots defined as $|r_m^{car}| = \frac{L}{tan(\phi_m^{car})}$ and $|r_m^{diff}| = 0$, respectively. If we consider the desired formation as a rigid body when the leader robot executes a circular trajectory, based on Ackermann steering geometry [43] and the definition of curvature, the minimum turning radius of leader robot i which prevents follower j from reversing can be derived as follows:

$$r_{m,j}^* = \begin{cases} r_{1,j} & |r_{m,j}^{type}| < \|\mathbf{d}_j\| \cdot cos(\psi), \; d_{x,j} \neq 0, \\ r_{2,j} & |r_{m,j}^{type}| \geq \|\mathbf{d}_j\| \cdot cos(\psi), \; d_{x,j} \neq 0, \\ r_{3,j} & d_{x,j} = 0, \end{cases} \tag{9}$$

where:

$$\begin{aligned} r_{1,j} &= \|\mathbf{d}_j\| \cdot sin(\psi), \\ r_{2,j} &= r_{1,j} + \delta \cdot \sqrt{(r_{m,j}^{type})^2 - (\|\mathbf{d}_j\| \cdot cos(\psi))^2}, \\ r_{3,j} &= r_{m,j}^{type} + d_{y,j}, \\ \psi &= atan2(\frac{d_{y,j}}{d_{x_j}}). \end{aligned} \tag{10}$$

In (10), the constant value δ is set to -1 if the formation rotates clockwise, and 1 otherwise. With this, the adjusted minimum turning radius of the leader robot is given by $r_m^* = \min_j\{r_{m,j}^*, r_m^{car}\}$ if the formation rotates clockwise, and $r_m^* = \max_j\{r_{m,j}^*, r_m^{car}\}$ otherwise, $\forall j \in \{0, 1, \dots, n_{f_e} - 1\}$. Given r_m^*, the maximum steering angle of the leader

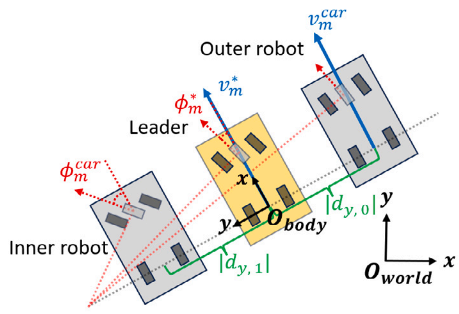


Fig. 6. The steering angle velocity constraints imposed by the rigidity of parallel formations.

robot i is $\phi_m^* = \arctan(\frac{L}{r_m^*})$. Further, the desired linear velocity of the jth follower, with respect to the formation leader, can be expressed as:

$$|v_j^d| = |\frac{v_{leader}}{r_m^*}|\sqrt{(r_m^* + r_{1,j})^2 + (\|\mathbf{d}_i\| \cdot cos(\psi))^2}, \quad \forall j \in \{0, 1, \ldots, n_{f_e}\}. \tag{11}$$

In order not to violate the kinematic constraints of all follower robots, the maximum velocity of the leader robot must satisfy:

$$|v_m^*| = \min_j \frac{|v_{m,j}^{type} \cdot r_m^*|}{\sqrt{(r_m^* + r_{1,j})^2 + (\|\mathbf{d}_i\| \cdot cos(\psi))^2}}, \quad \forall j \in \{0, 1, \ldots, n_{f_e}\}, \tag{12}$$

where $v_{m,j}^{type}$ is the maximum velocity of the jth follower robot. As an example, consider a linear formation, as in Fig. 6. The revised motion primitives of the leader are given by $\phi_m^* = \arctan(\frac{L+|d_{y,1}|\cdot(\tan(\phi_m^{car}))}{L\cdot\tan(\phi_m^{car})})$ and $v_m^* = \frac{v_m^{car}\cdot L}{L+|d_{y,0}|\cdot\tan(\phi_m^*)}$, where $|d_{y,0}|$ and $|d_{y,1}|$ are the lateral offsets between the car-like follower and the leader relative to the leader's exterior and interior, respectively.

Collision Check. During node expansion, hybrid A* checks if a potential successor node causes a robot's footprint to intersect with an obstacle. If an intersection occurs, the node is not added to the search tree. For formation planning, we must check the footprint of the entire formation during collision detection. For this, we first define the centre of the leader's rear axle and its orientation as the origin and x-axis of the body frame for the formation, respectively (see Fig. 6). Let robot i be the leader of the eth formation. From the formation centre position $p_{ctr}^e(t) = (x_{ctr}^e(t), y_{ctr}^e(t))^T$ and its orientation $\theta_{ctr}^e(t)$, we can compute a set of vertices which capture the convex hull representing the formation's footprint at time t:

$$V_{e,c}(t) \in \mathbb{R}^2 : V_{e,c}(t) = p_{ctr}^e(t) + \mathbf{R}(\theta_{ctr}^e(t)) \cdot \mathbf{l}_{e,c}, \quad \forall e \in \mathcal{F},\ c = 1, \ldots, n_{e,c},\ t \in [0, t_{f_i}], \tag{13}$$

where $n_{e,c}$ is the number of vertices and $\mathbf{l}_{e,c}$ is the relative coordinate of the cth vertex in the body frame. With this, during node expansion we check whether the convex hull representing the formation's footprint intersects with any obstacles in the environment.

5.2.3. Obstacle avoidance constraints

After synthesising the initial guess, the convex hull formulation in (13) can be used to enforce obstacle-avoidance in the leader's trajectory. Similar to Section 4, we can take a trajectory, extract a number of discrete waypoints from it, and then use these as seed points for IRIS to generate a set of safe convex regions (see 5). However, here we use the footprint of the formation rather than that of a single robot. We denote the safe corridors for formation e as SC_e:

$$V_{e,c}(t) \in SC_e(t), \quad \forall e \in \mathcal{F},\ c = 1, \ldots, n_{e,c},\ t \in [0, t_{f_i}]. \tag{14}$$

Algorithm 2: Trajectory optimisation for the leader i.

```
   Input: $r_i$, $\mathcal{Z}_e$, $map$, $w_t$, $w_s$
   Output: $T_i$, $\phi_i$, $SC_e$
1  $SC_e \leftarrow \emptyset$;
2  if $T_{guess} = \emptyset$ then
3      $\chi_{guess} \leftarrow$FCHA*($map$);
4      $T_{guess} \leftarrow$GenerateTrapezoidalProfile($\chi_{guess}$);
5  $SC_e \leftarrow$GenerateSafeCorridors($T_{guess}$, $\mathcal{Z}_e$, $map$);
6  GenerateOCP($r_i$, $SC_e$, $\mathcal{Z}_e$, $w_t$, $w_s$);
7  $T_i \leftarrow$Solve (15);
8  return [$T_i$, $SC_e$]
```

Algorithm 3: Trajectory optimisation for follower j.

```
   Input: $r_j$, $T_i$, $map$, $SC_e$
   Output: $T_j$
1  $T_j^{ref} \leftarrow$GenerateRefTraj($\mathbf{d}_i$, $\mathbf{d}_j$, $T_i$);
2  GenerateOCP($r_j$, $SC_e$, $T_j^{ref}$);
3  $T_j \leftarrow$Solve (19);
4  return $T_j$
```

With this, we can rewrite the optimal control problem for the leader robot i using the safe corridor; the initial and terminal constraints; the car-like dynamics models; and kinematic constraints with the modified maximum/minimum steering angles introduced in Section 5.2.2:

$$\begin{aligned} \min_{z_i(t), u_i(t), t_{f_i}} \quad & (8) \\ \text{s.t.} \quad & (1), \\ & (7c), (7d), \\ & (2), (14). \end{aligned} \tag{15}$$

We summarise our procedure for synthesising the leader's trajectory in Alg. 2. Here, we synthesise an initial guess trajectory from the FCHA* path using a trapezoidal velocity profile (lines 3–4) [44]. This is then used in line 5 to construct the safe corridor.

5.3. Stage two: Trajectory optimisation for the followers

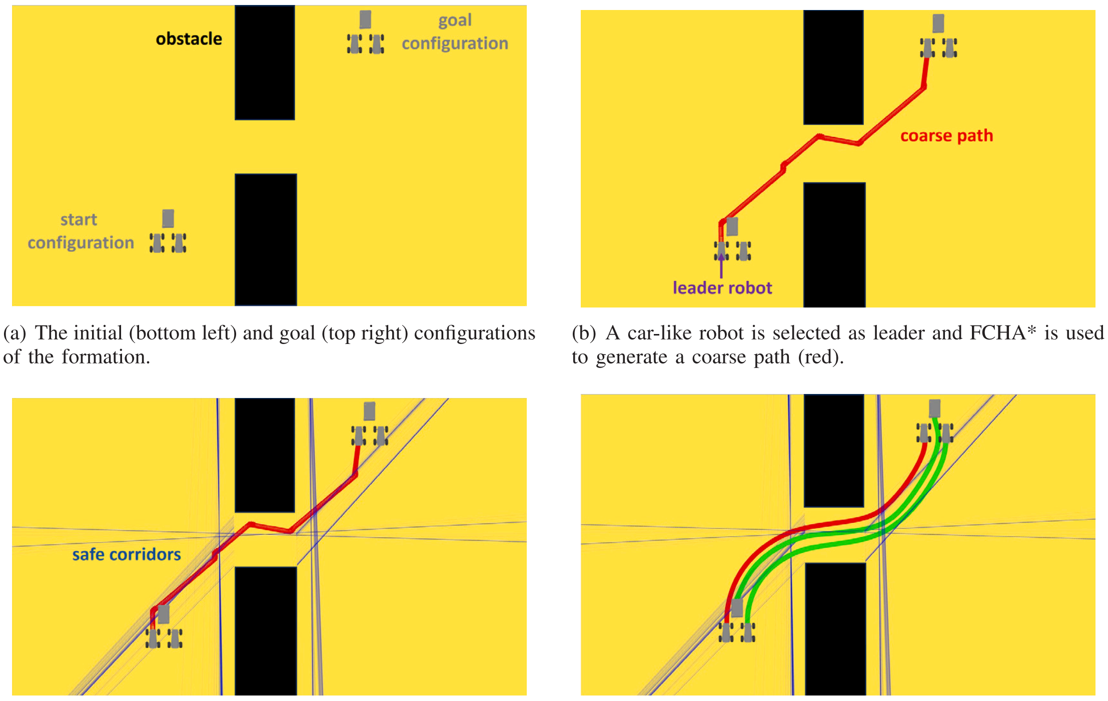
After synthesising the leader's trajectory, we must synthesise trajectories for the followers which maintain rigidity by sustaining a constant relative position with respect to the leader. We present our approach for follower trajectory optimisation in Alg. 3.

We begin with `GenerateRefTraj` in line 1, which extracts a reference trajectory T_j^{ref} for follower j using the leader's trajectory T_i and the known formation geometry. Concretely, the reference position of follower j at time t is derived from the leader i's position:

$$p_j(t) = p_i(t) + (\mathbf{d}_j - \mathbf{d}_i) \cdot \mathbf{R}^l(\theta_i(t)), \tag{16}$$

where $\mathbf{d}_i$ and $\mathbf{d}_j$ represent the position vectors of leader i and follower j relative to the centre of the formation, as shown in Fig. 4. To transform the world frame to the body frame, we use a rotation matrix $\mathbf{R}^l(\theta_i(t)) = [\sin(\theta_i(t))\ \cos(\theta_i(t));\ -\cos(\theta_i(t))\ \sin(\theta_i(t))]$. Note that our framework can be applied to irregular formation shapes by adjusting the value of $\mathbf{d}_i$, which is the distance of the ith robot to the formation centre. With this, $FCHA^*$ and the subsequent trajectory optimisation keep the formation within the safe corridors generated by IRIS [35] to guarantee collision avoidance. Using the reference trajectory we then formulate trajectory optimisation for follower j as an optimal control problem similar to (7) with the following cost function:

$$J_j = \int_{\tau=0}^{t_{f_i}} a_j^2(\tau) + u_j(\tau)u_j^T(\tau) + w_e \Delta p_j(\tau) \Delta p_j^T(\tau) d\tau, \tag{17}$$



(a) The initial (bottom left) and goal (top right) configurations of the formation.

(b) A car-like robot is selected as leader and FCHA* is used to generate a coarse path (red).

(c) An initial guess for the leader is generated alongside the safe corridor (blue).

(d) (15) and (19) are solved iteratively until a kinematically feasible, safe, and time-optimal trajectory is found for the leader (red) and followers (green).

Fig. 7. An illustrative formation planning example for a triangular formation.

where w_e is a weight parameter. The first term encourages smoother trajectories, as in (8), and the second term encourages the follower to minimise its positional error with respect to the reference trajectory, denoted Δp_j. Here, we use soft formation constraints as the non-holonomic nature of car-like robots makes it challenging to maintain rigidity while avoiding collisions. In the worst case, using hard formation constraints would make the optimisation problem unsolvable. To guarantee collision avoidance with obstacles, we use the formation safety corridor SC_e constructed during leader trajectory optimisation to guarantee that the follower's footprint remains in the safe corridor at each timestep:

$$V_{j,l}(t) \in \mathbb{R}^2 : \; V_{j,l}(t) = \{p_j(t) + \mathbf{R}(\theta_j(t)) \cdot \mathbf{l}_{j,l}\} \in SC_e(t), \quad \forall j \in \mathcal{R},\, l \in \{1,\dots,4\},\, t \in [0, t_{f_j}]. \tag{18}$$

The kinematic constraints of the follower are defined using (1) and (2), or (3) and (4), depending on the type of the follower. With this, we formulate the optimal control problem for follower j as follows:

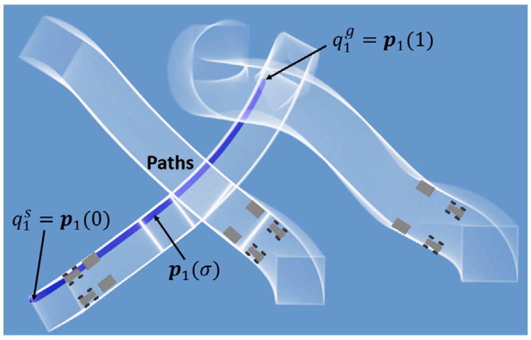
$$\begin{aligned} \min_{z_j(t), u_j(t), t_{f_j}} \quad & (17) \\ \text{s.t.} \quad & (1)/(3), \\ & (7c), (7d), \\ & (4)/(2), (18). \end{aligned} \tag{19}$$

5.4. Iterative framework

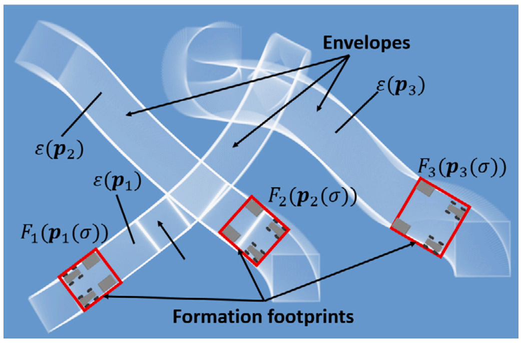
By executing the two formation planning stages described above, we can synthesise trajectories for all robots in a formation. However, during trajectory planning for the leader, we ignore the kinodynamic constraints of the follower. This may make the reference trajectories for the followers impossible to execute, leading to significant formation errors. To address this, in Alg. 4 we present a framework for trajectory planning which iterates between trajectory optimisation for the leader and for the followers. We iterate between these steps (lines 4–9) until the average and maximum formation errors fall below a predetermined threshold (line 2), or an iteration threshold is met (line 14). After each iteration, we multiply the weight parameter w_s in the leader's cost function (8) and weight w_e in the follower's cost function (17) by α and β respectively (lines 11–12), where $\alpha, \beta > 1$. Lower values of α and β will result in more efficient plans at the cost of increased planning time. By increasing w_s for the leader, we begin to prioritise the trajectory's smoothness over its duration, making it easier to execute. By increasing w_e for the followers, we begin to prioritise the tracking error with respect to the reference trajectory over smoothness, penalising deviations away from the reference trajectory. Note that in each subsequent iteration, the leader's initial guess is set to the trajectory from the previous iteration (line 6).

Algorithm 4: Our complete formation planning method.

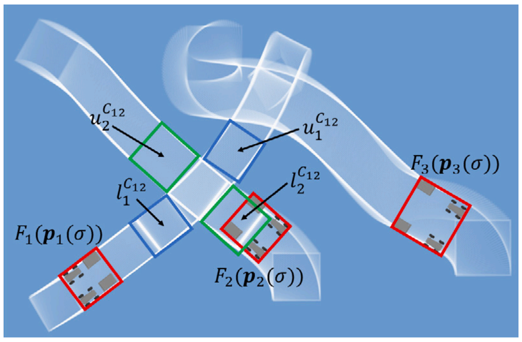
```
   Input: $\{r_f\}_{f\in\mathcal{F}_e}$, $\mathcal{Z}_e$, $map$
   Output: $T$
1  $T \leftarrow \emptyset$, $T_{guess} \leftarrow \emptyset$, $iter \leftarrow 0$, $err_{max} \leftarrow +\infty$, $\overline{err} \leftarrow +\infty$;
2  while $err_{max} > \mu_{max}$ or $\overline{err} > \mu_{avg}$ do
3      select robot $i$ as a leader;
4      // 1st stage
5      $[T_i, SC_e] \leftarrow$CalculateLeaderTraj($r_i$, $T_{guess}$, $\mathcal{Z}_e$, $map$, $w_s$);
6      $T_{guess} \leftarrow T_i$;
7      // 2nd stage
8      foreach $j \neq i \vee j \in \mathcal{F}_e$ do
9          $T_j \leftarrow$CalculateFollowerTraj($r_j$, $r_i$, $T_i$, $map$, $w_e$, $SC_e$);
10     $[\overline{err}, err_{max}]$ =CalculateFormationError($T$, $\mathcal{Z}_e$);
11     $w_s \leftarrow \alpha \cdot w_s$;
12     $w_e \leftarrow \beta \cdot w_e$;
13     $iter \leftarrow iter + 1$;
14     if $iter > iter_{max}$ then
15         return $\emptyset$;
16 return $T$
```


$q_1^g = \boldsymbol{p}_1(1)$
Paths
$q_1^s = \boldsymbol{p}_1(0)$
$\boldsymbol{p}_1(\sigma)$

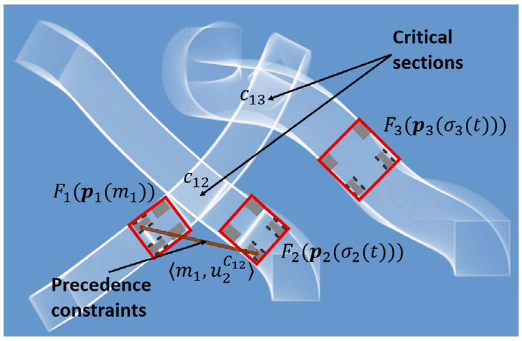
(a) Dark blue denotes the path of leader robot 1. $\mathbf{p}_1(0)$ and $\mathbf{p}_1(1)$ denote the start and end point of the path, respectively, while $\mathbf{p}_1(\sigma)$ represents any point along the path.


Envelopes
$\varepsilon(\boldsymbol{p}_3)$
$\varepsilon(\boldsymbol{p}_2)$
$\varepsilon(\boldsymbol{p}_1)$
$F_3(\boldsymbol{p}_3(\sigma))$
$F_2(\boldsymbol{p}_2(\sigma))$
$F_1(\boldsymbol{p}_1(\sigma))$
Formation footprints

(b) Using leader robot 1 as an illustration, the red bounding box symbolises its formation's footprint $F_1(\mathbf{p}_1(\sigma))$, and the white denotes its spatial envelope $\varepsilon(\mathbf{p}_1)$.


$u_2^{C_{12}}$
$u_1^{C_{12}}$
$l_2^{C_{12}}$
$l_1^{C_{12}}$
$F_3(\boldsymbol{p}_3(\sigma))$
$F_1(\boldsymbol{p}_1(\sigma))$
$F_2(\boldsymbol{p}_2(\sigma))$

(c) The critical section $\mathcal{C}_{12}$ of formation 1 and formation 2 is depicted. The blue and green bounding boxes represent the instantaneous footprints of formations 1 and 2 as they enter and exit $\mathcal{C}_{12}$, respectively.


Critical sections
C_{13}
$F_3(\boldsymbol{p}_3(\sigma_3(t)))$
$F_1(\boldsymbol{p}_1(m_1))$
C_{12}
$F_2(\boldsymbol{p}_2(\sigma_2(t)))$
$\langle m_1, u_2^{C_{12}}\rangle$
Precedence constraints

(d) Since formation 1 yield to formation 2, formation 1 is not allowed to exceed $F_1(\mathbf{p}_1(m_1))$ until formation 2 crosses $\mathcal{C}_{12}$, indicating that it has left the critical section.

Fig. 8. An illustrative example of the coordination approach in [19].

We demonstrate our formation planner in Fig. 7. In Fig. 7(a), we present a scenario where a triangular formation must navigate to the top right of the map, with black areas indicating obstacles. Fig. 7(b) displays the initial coarse path for the leader robot. The safe corridors along this coarse path are depicted in Fig. 7(c)). Fig. 7(d) shows the final trajectories synthesised by iteratively solving (15) and (19). These trajectories are kinematically feasible, safe, and time-optimal.

Alg. 4 returns a feasible solution if at each time step t there exists at least one feasible formation configuration within the current convex region which satisfies constraint (14). If no feasible configuration can be found within a convex region, optimisation problem (15) will fail without a solution. Therefore, if our algorithm returns a solution, it is guaranteed to be feasible. IRIS is not guaranteed to find the largest possible convex region in the environment [35]. The success rate of our algorithm is highly dependent on the initial safe convex region. The initial convex region is generated based on the result of $FCHA^*$ in Section 5.2.2. During $FCHA^*$, we explicitly consider the formation's convex hull during collision detection to ensure a collision-free initial guess. By considering the convex hull during $FCHA^*$, we hope to maximise the chance of finding a solution in Alg. 4.

6. Formation coordination

In this section, we propose a loosely-coupled approach for inter-formation collision avoidance which forms the coordination component of our H-MFPC framework (see Fig. 2).

6.1. Problem formulation

Consider a formation where robot i is the leader. Let q_i^s and q_i^g be the start and goal configurations of robot i, and $\mathbf{p}_i : [0,1] \to Q_i^{free}$ represent the sequence of configurations $q_i \in Q_i^{free}$ along robot i's trajectory, where $\mathbf{p}_i(0) = q_i^s$, $\mathbf{p}_i(1) = q_i^g$ (see Fig. 8(a)). Here, $\mathbf{p}_i$ is parameterised by an arc length $\sigma \in [0,1]$. Given the workspace $\mathcal{W}$, obstacles $\mathcal{O}$, and the trajectories $\{T_i\}_{i=1}^n$ obtained in Section 5, the multi-formation coordination problem is to synthesise a revised set of trajectories $\{T_i^{coord}\}_{i=1}^n$ that satisfy the following criteria:

(1) At no point does the footprint of two or more robots intersect, i.e. $\forall (i, j \neq i) \in \mathcal{R} \times \mathcal{R}$, $\forall t \in [0, t_{f_i}]$, $R_i(\mathbf{p}_i(t)) \cap R_i(\mathbf{p}_j(t)) = \emptyset$.
(2) Each formation maintains rigidity throughout execution.
(3) All robots eventually reach their destination, i.e. $\forall i \in \mathcal{R}$, $\exists\, t < +\infty$, $q_i(t) = q_i^g$.

6.2. Multi-formation coordination algorithm

To solve the multi-formation coordination problem, we adapt the coordination framework in [19] for formations. In [19], the velocity profile of mobile robots is adjusted online based on local information while maintaining kinematic feasibility and safety. For multi-formation coordination, we begin with the formation trajectories synthesised in Section 5.4, and then revise core concepts in [19] for formations.

Spatial envelopes. For the formation led by robot i, its footprint is associated with the leader's pose $\mathbf{p}_i(\sigma)$, denoted as $F_i(\mathbf{p}_i(\sigma))$. The spatial envelope $\varepsilon(\mathbf{p}_i)$ is the total area covered by the formation's geometry along its path, which can be described as the union of successive footprints of the formation such that $\cup_{\sigma\in[0,1]} F_i(\mathbf{p}_i(\sigma)) \subseteq \varepsilon(\mathbf{p}_i)$; also, let $\varepsilon(\mathbf{p}_i(\sigma))^{\{\sigma_1,\sigma_2\}} = \cup_{\sigma\in[\sigma_1,\sigma_2]} F_i(\mathbf{p}_i(\sigma))$. We visualised the above definition in Fig. 8(b).

Critical sections. A critical section is an area of intersection between the spatial envelopes of two formations. Let C_{ij} be the decomposition of the set $\{q_i \in Q_i,\, q_j \in Q_j \mid F_i(q_i) \cap \varepsilon(\mathbf{p}_j) \neq \emptyset \vee F_j(q_j) \cap \varepsilon(\mathbf{p}_i) \neq \emptyset\}$

Algorithm 5: The multi-formation coordination algorithm, adapted from [19].

```
   Input: ${T_i}_{i=1}^n$, ${r_i}_{i=1}^n$, ${\mathcal{F}_e}_{e=1}^{n_f}$, map
   Output: $T_{coord}$
1  $\mathcal{P} \leftarrow \emptyset$, $\mathcal{C} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $t \leftarrow 0$;
2  while true do
3      foreach leader robot $i \in \mathcal{R}$ do
4          $\mathbf{p}_i \leftarrow$ExtractPathFromTraj($T_i$);
5          $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{p}_i\}$;
6          foreach pair of leader robots $(\mathbf{p}_i, \mathbf{p}_{i' \neq i}) \in \mathcal{P}^2$ do
7              $\mathcal{C} \leftarrow \mathcal{C} \cup$ getIntersections($\varepsilon(\mathbf{p}_i)$, $\varepsilon(\mathbf{p}_{i'})$);
8          $\mathcal{T} \leftarrow$ reviseConstraints($\mathcal{P}$, $\mathcal{C}$, $\mathcal{T}$, $t$);
9          foreach $\mathbf{p}_i \in \mathcal{P}$ do
10             $\mathcal{T}_i = \{\exists j : \langle \mathbf{p}_i, \mathbf{p}_j, m_i, u_j^C \rangle \in \mathcal{T}\}$;
11             if $\mathcal{T}_i \neq \emptyset$ then
12                 $q_i^{closest} \leftarrow \underset{q_i \in \mathcal{T}_i, \sigma \in [0,1]}{\arg\min} \ \sigma = \mathbf{p}_i^{-1}(q_i)$;
13                 $\mathbf{p}_i \leftarrow$ InsertWaypoint($\mathbf{p}_i$, $q_i^{closest}$, $t$);
14             else
15                 $\mathbf{p}_i \leftarrow$ StitchPathToGoal($\mathbf{p}_i$, $\mathbf{p}_i(1)$, $t$);
16     if $\mathcal{T} = \emptyset$ then
17         break;
18     $t \leftarrow t + \Delta t$;
19 foreach $\mathbf{p}_i \in \mathcal{P}$ do
20     $T_i^{ref} \leftarrow$ GenerateInitialGuess($\mathbf{p}_i$);
21     $T_i^{coord} \leftarrow$CalculateFollowerTraj($r_i$, $r_i$, $T_i^{ref}$, map, $\emptyset$);
22     foreach $i \in \mathcal{F}_e$, $j \neq i \wedge j \in \mathcal{F}_e$ do
23         $T_j^{coord} \leftarrow$ CalculateFollowerTraj($r_j$, $r_i$, $T_i^{coord}$, map, $\emptyset$);
24 return $T^{coord}$
```

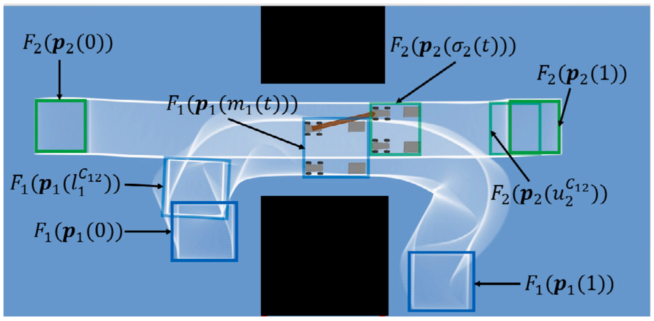

$F_2(\mathbf{p}_2(0))$
$F_2(\mathbf{p}_2(\sigma_2(t)))$
$F_2(\mathbf{p}_2(1))$
$F_1(\mathbf{p}_1(m_1(t)))$
$F_1(\mathbf{p}_1(l_1^{C_{12}}))$
$F_2(\mathbf{p}_2(u_2^{C_{12}}))$
$F_1(\mathbf{p}_1(0))$
$F_1(\mathbf{p}_1(1))$

Fig. 9. $m_1(t)$ is continually updated to represent the current state of formation 2. This instigates a following behaviour, whereby formation 1 persists in advancing as formation 2 progresses along $\mathbf{p}_2$.

into its largest contiguous subsets. Each of these subsets $C \in \mathcal{C}_{ij}$ is a critical section. For each critical section $C \in \mathcal{C}_{ij}$, let $l_i^C \in [0,1]$ and $u_i^C \in [0,1]$ denote the instants immediately before the formation enters C and immediately after the formation leaves C (see Fig. 8(c)). Formally, $\mathbf{p}_i(l_i^C) \notin C$, $\mathbf{p}_i(u_i^C) \notin C$, and $\mathbf{p}_i(\sigma) \in C$, $\forall \sigma \in (l_i^C, u_i^C)$. A critical section C is active if at least one robot is present in C. We denote the set of active critical sections at time t with $\mathcal{C}(t)$.

Precedence constraints and critical points. Given a critical section $C \in \mathcal{C}(t)$, we can define a precedence constraint $\langle \mathbf{p}_i, \mathbf{p}_j, m_i, u_j^C \rangle$, which enforces that leader robot i cannot navigate beyond configuration $\mathbf{p}_i(m_i(t))$, $m_i(t) \in [0,1]$ at time t until leader robot j has surpassed configuration $\mathbf{p}_j(u_j^C), u_j^C \in [0,1]$ (see Fig. 8(d)). This constraint defines which formation should yield, where it should wait, and when until. m_i is time-dependent, as the location robot i can reach is dependent on robot j's progress. This allows for robot i to follow robot j through the critical section whenever feasible, rather than waiting for robot j to completely exit. We illustrate the following behaviour in Fig. 9. Formally, let $\sigma_i(t)$ and $\sigma_j(t)$ be the positions of leaders i and j at time t, respectively. The precedence constraint states that robot i cannot pass $m_i(t)$ at time t unless robot j has left the critical section, where:

$$m_i(t) = \begin{cases} \max\{l_i^C, r_{ij}(t)\} & \text{if } \sigma_j(t) < u_j^C \\ 1 & \text{otherwise.} \end{cases} \tag{20}$$

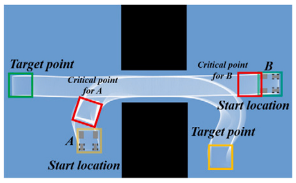
Here, $r_{ij}(t)$ is the last configuration that robot i can access along its path $\mathbf{p}_i$ at time t without overlapping with robot j's footprint. Formally, $r_{ij}(t) = \sup_{\sigma}\{\sigma \in [\sigma_i(t), u_i^C] : \varepsilon(\mathbf{p}_i)^{\{\sigma_i(t),\sigma\}} \cap \varepsilon(\mathbf{p}_j)^{\{\sigma_j(t),u_j^C\}} = \emptyset\}$. (20) implies that robot i is constrained from surpassing robot j's current configuration while robot j remains in the critical section. Let $\mathcal{T}(t)$ be the set of precedence constraints active at time t. With this, if $C \in \mathcal{C}(t)$, then $\langle m_i, u_j^C \rangle$ or $\langle m_j, u_i^C \rangle \in \mathcal{T}(t)$, i.e. if a critical section is active, one of the conflicting robots must wait. The precedence constraints enforce an order for robots to cross a critical section, ensuring collision-free trajectories. We define a *critical point* as the last reachable configuration for robot i which adheres to the current set of precedence constraints $\mathcal{T}(t)$. The multi-formation coordination problem can then be reduced to computing and updating the set of critical points such that collisions do not occur. For a critical section C, we assume that $l_i^C > 0$ and $u_i^C < 1$ for all leaders i, i.e. no robot begins or ends in a critical section.

We present our multi-formation coordination algorithm adapted from [19] in Alg. 5. We begin by extracting the path $\mathbf{p}_i$ for each leader robot i from their trajectory synthesised in Section 5 (lines 4–5). Next, we compute the critical sections for each pair of spatial envelopes and add them to the set $\mathcal{C}$ (lines 6–7). In line 8, we then convert the critical sections into a series of precedence constraints for each formation. We follow the priority ordering in [19], where robots closer to the critical section have higher priority. Moreover, the priority ordering must be dynamically feasible to guarantee safety, i.e. waiting robots must be able to stop before the critical point. In lines 9–15, we adjust the paths of each formation leader i given the precedence constraints. We do this by finding the configuration $q_i^{closest}$ which the leader i is currently forbidden to pass, and forcing the robot to wait there until the critical section is free. If there are no constraints applied to a formation, it can navigate directly to the target. After updating the paths for each formation leader, in lines 19–23, we generate the updated trajectories for each formation. For this, we use the updated paths to synthesise an initial guess for each leader, and then solve the optimal control problem (19) in Section 5.3 without the safe corridor constraints in (18) to compute the updated trajectories for the leader and followers. We do not consider obstacle avoidance constraints in Alg. 5, as the updated paths remain on the spatial envelope of the formation's original path, which is guaranteed to avoid obstacles. Further, as the original path considered the kinematic constraints of both the leader and the followers, the revised path can still be tracked in a straightforward way.

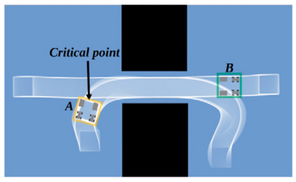
We illustrate the critical-section based coordination approach in Fig. 10. In Fig. 10(a), two formations must navigate through a narrow passage which can only accommodate one formation at a time. Formation B is closer to its critical point, and thus enters the critical section first. As a result, formation A reaches a critical point and must wait for formation B to leave the critical section before continuing (Fig. 10(b)). Once the path is clear, formation A can then proceed towards its target (Fig. 10(c)), and both robots complete navigation without collisions (Fig. 10(d)).

7. Trajectory tracking

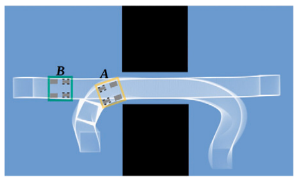
In this section, we implement a trajectory tracking controller for closed-loop control of the trajectories synthesised for each robot. For robot i, the deviations in the x direction, y direction, and orientation between the reference trajectory and the actual tracking trajectory within the world frame are denoted as $x_{i,e} = x_{i,r} - x_{i,c}$, $y_{i,e} = y_{i,r} - y_{i,c}$, and $\theta_{i,e} = \theta_{i,r} - \theta_{i,c}$, respectively. For differential drive robots, we apply a control law adapted from [20] to accommodate a predefined target



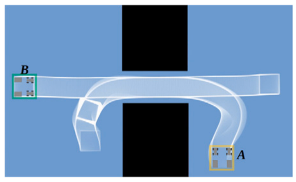
(a) **[Our methods]:** Formations A and B are on track to reach their respective target points. Each formation's critical points are highlighted in red.



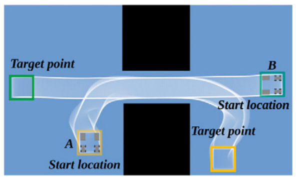
(b) **[Our methods]:** Formation A stops at the critical point and waits until Formation B leaves the critical section.



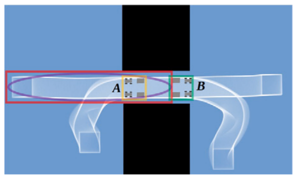
(c) **[Our methods]:** Once formation A is clear ahead, proceed to the target point.



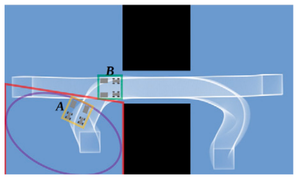
(d) **[Our methods]:** The two formations reach their respective target points.



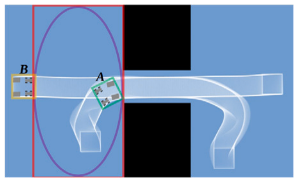
(e) **[Prioritised DWA]:** Formations A and B are on track to reach their respective target points.



(f) **[Prioritised DWA]:** Formation A treats formation B as an obstacle and computes its convex region.



(g) **[Prioritised DWA]:** Formation A backs off to prevent a collision and computes the updated convex region.



(h) **[Prioritised DWA]:** Formation B is eventually clear of A, and A returns to trajectory tracking.

Fig. 10. A comparative illustration showcasing the differences between our methods and prioritised DWA in a narrow corridor. Fig. 10(a) through 10(d) depict our proposed loosely coupled centralised approach, while Fig. 10(e) to 10(h) illustrate the prioritised DWA as presented in [22]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

trajectory. This control law calculates a control input $(v_{i,c}, \omega_{i,c})$ for each follower robot based on the control error e_0:

$$v_{i,c} = v_{i,r}\cos(\theta_{i,e}) + K_x x_{i,e}, \tag{21}$$

$$\omega_{i,c} = \omega_{i,r} + v_{i,r}(K_y y_{i,e} + K_\theta \sin(\theta_{i,e})), \tag{22}$$

where K_x, K_y, K_θ are positive parameters. $v_{i,r}$ and $\omega_{i,r}$ respectively denote the target velocity and target angular velocity of robot i's open-loop trajectory, as obtained from Section 5. For car-like robots, the control law is implemented as in [21]:

$$v_{i,c} = v_{i,r}\cos(\theta_{i,e}) + k_x x_{i,e}, \tag{23}$$

$$\phi_{i,c} = \tan^{-1}\left(\frac{(\omega_{i,r} + k_y y_{i,e} + k_\theta \theta_{i,e})L}{v_{i,c}}\right), \tag{24}$$

where k_x, k_y, k_θ are positive parameters which can significantly affect the efficacy of the tracking controller. However, in this paper, we are only concerned with minimising the tracking error under possibly suboptimal parameters, rather than searching for the optimal parameters.

8. Experiments

In this section, we demonstrate the efficacy of our approach in simulation. All experiments are run on Ubuntu 20.04 with an Intel Core i9 processor @ 3.2 GHz and 16 GB of RAM.[1] All software is implemented in C++. To solve the optimal control problem, we use the explicit Runge–Kutta method [45] to form the nonlinear programming (NLP) problem, employ ADOL-C [46] for automatic differentiation, and utilise CasADi [47] with the primal–dual interior-point solver IPOPT [48] for solving the nonlinear programming. We use Boost [49] to generate spatial envelopes and identify critical intersections for formation coordination. We list all solver parameter values in Table 3. For all experiments, we simulate a 384 $m \times 384m$ environment in Gazebo which contains obstacles, narrow corridors, and tight corners. We show this map in Fig. 11. We model the footprint of each robot as a $1m \times 0.8m$ rectangle, and consider linear, triangular, and rectangular formations, as shown in Fig. 4.

[1] The source code is available: https://github.com/HyPAIR/Heterogeneous-formation-controller

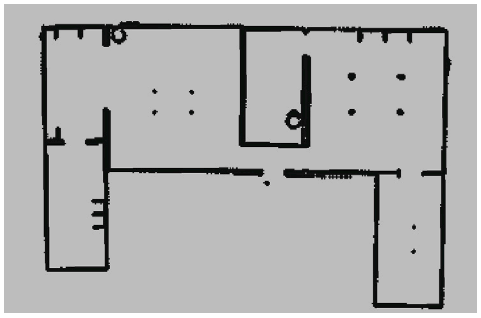


Fig. 11. The simulation environment used throughout our experiments.

Table 3
Experimental parameter values.

Parameter	Value	Description
L	0.65 m	Car-like robot wheelbase
v_m^{car}, v_m^{diff}	1.0 m/s, 1.0 m/s	Linear velocity limit
a_m^{car}, a_m^{diff}	1.0 m/s^2	Linear acceleration limit
ϕ_m^{car}	0.68 rad	Steering angle limit
ω_m^{car}, ω_m^{diff}	0.2 rad/s, 1.5 rad/s	Angular velocity limit
Ω_m^{car}	2.5 rad/s^2	Angular acceleration limit
–	0.2 m	IRIS grid size
Δt	0.15 s	Time between control inputs
w_e, w_s, w_p	0.1, 1.0, 10	Weights for cost function (6a)
w_s, w_t	0.1, 1.0	Weights for cost function (8)
w_e	0.5	Weights for cost function (17)
μ_{avg}, μ_{max}	0.01 m, 0.02 m	Algorithm 4 convergence threshold
$iter_{max}$	25	Maximum iterations in Algorithm 4
α, β	1.1, 1.1	Weights for Algorithm 2
K_x, K_y, K_θ	0.3, 0.1, 0.7	Trajectory tracking parameters
k_x, k_y, k_θ	2, 4, 1.4	Trajectory tracking parameters

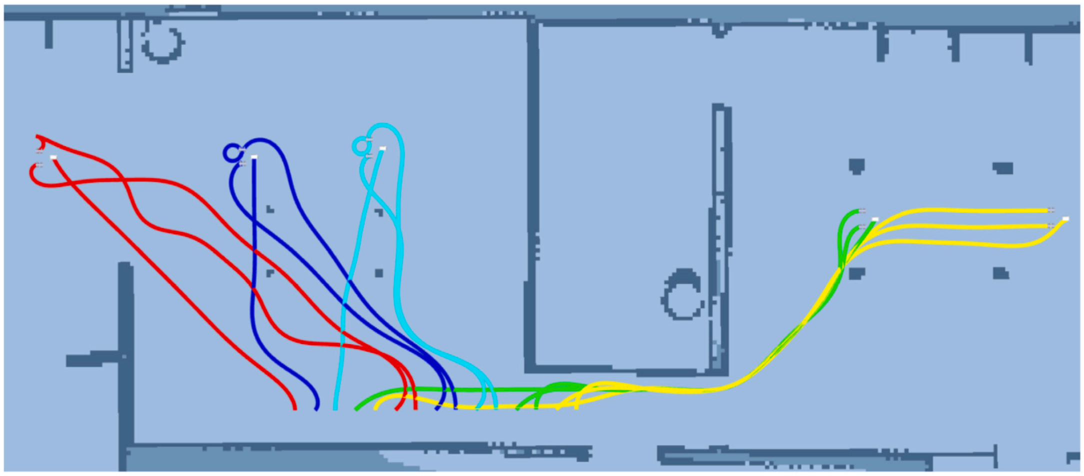


Fig. 12. The 15(5 ∗ 3)-robot formation generation problem.

8.1. *Formation generation performance*

First, we evaluate formation generation performance as the number of robots increases. We compare our formation generation method against the following state-of-the art approaches:

- **Conflict-based model predictive control (CB-MPC)** [18]: CB-MPC models the footprint of each robot as a circle and resolves potential collisions incrementally using conflict-based search [17] with MPC as the lower-level planner.
- **Prioritised trajectory optimisation for multiple non-holonomic robots (MNHP)** [25]: Similar to CB-MPC, MNHP also models robot footprints as a circle. MNHP partitions the robots into a group, and assigns a priority to each group. Joint trajectories are then synthesised for each group, where lower-priority groups must then avoid the trajectories of robots in higher-priority groups.
- **Distributed model predictive control (DMPC)** [29]: DMPC is a decentralised approach where each robot synthesises its trajectory independently while anticipating the actions of neighbouring robots.

We consider multi-robot systems ranging from 4 to 24 robots, where the robot types per formation are defined as in Fig. 4. For each number of robots, we randomly generate 30 non-overlapping, obstacle free start and goal locations. We show an example problem instance in Fig. 12. A method is successful if each robot reaches its respective target without collisions within 30 s. For all baselines except DMPC, we use ECBS [38] to synthesise the reference trajectories. In this experiment, we measure the success rate, computation time, average travel distance, and the makespan, i.e. the time for the last robot to reach its formation location. Failed instances are excluded when measuring the computation time, travel distance, and makespan. We present our results in Fig. 13, which specify the formation configuration for each number of robots. For example, 8(1 ∗ 2 + 2 ∗ 3) indicates 8 robots arranged in 1 linear formation and 2 triangular formations.

In Fig. 13(a), our approach consistently achieves the highest success rate, which is often above 80%. This is because our approach precisely models robot footprints as a rectangle. This results in less conservative inter-robot collision avoidance constraints, improving the chances of finding a valid solution. Conversely, the baselines model each robot as a circle. This reduces the robot's feasible space when approaching the target configuration, and limits the solution space. Under DMPC, each robot plans independently without any explicit coordination. This causes frequent deadlocks; for 24 robots, 50% of the problem instances were unsolvable. The success rates of MNHP and CB-MPC decrease with the number of robots due to their use of a circle as a footprint model, which limits the solution space. In particular, when the robot formation locations are close together, MNHP struggles to find feasible paths for lower priority robots within the allotted time.

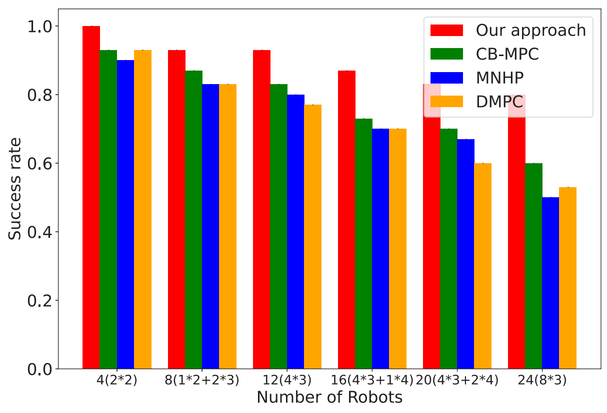
In Fig. 13(b), the computation time of each approach increases approximately linearly in the number of robots. Each approach either decouples inter-robot collision constraints, or plans in a distributed manner, simplifying planning. CB-MPC models the robot footprint as a circle, which simplifies the inter-robot collision avoidance constraint to a distance check between the centre of two circles. This improves computational efficiency at the cost of a lower success rate. Though MNHP plans for each group sequentially to improve scalability, lower-priority groups are subject to high numbers of constraints, which increases the complexity of planning for that group. The DMPC results show the sum of computation times for each robot. Though this time would decrease if run in parallel, the individual computation times for each robot are still relatively high due to the complexity of handling predictions of robot behaviour during trajectory optimisation [50].

In Fig. 13(c) and Fig. 13(d), there is little difference in the average travel distance between our approach, CB-MPC, and MNHP. This is because each of these approaches use reference paths generated by ECBS on a coarse grid representation of the environment. DMPC only considers local information from neighbouring robots during planning, which often causes robots to take long detours in densely populated with obstacles and other robots, increasing the makespan and travel distance. Though our approach produces trajectories with similar makespans and travel distances to CB-MPC and MNHP, its success rate is still superior.

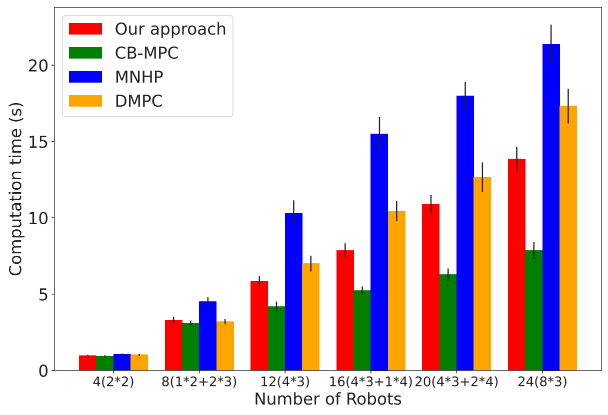
Next, we investigate the impact of softening the terminal constraints in cost function (6a). For the previous experiment, we record the terminal position error, i.e. the average Euclidean distance between each robot's final and target position; and the terminal orientation error, i.e. the average deviation between each robot's final and target orientation, for each experimental run. We present the results of this analysis in Fig. 13(e) and Fig. 13(f). The average position and orientation error are consistently lower than $0.0053m$ and $0.0043rad$ respectively. This demonstrates that the soft terminal constraints have negligible impact on the quality of the synthesised trajectories.

8.2. *Formation planning performance*

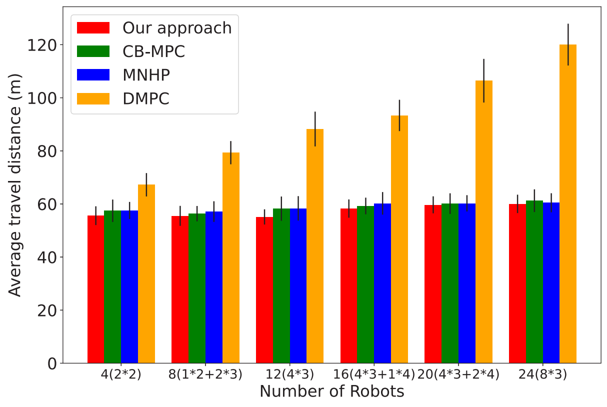
Next, we evaluate our formation planner. For each type of formation, we randomly generate 200 start and target locations, where the start and target are at least $5m$ apart. For each pair of start and target locations, we record the trajectory length and the corresponding planning time, which is the sum of times spent generating safe corridors, planning coarse paths, and optimising trajectories. In Fig. 14 we show that the planning time increases roughly linearly with the



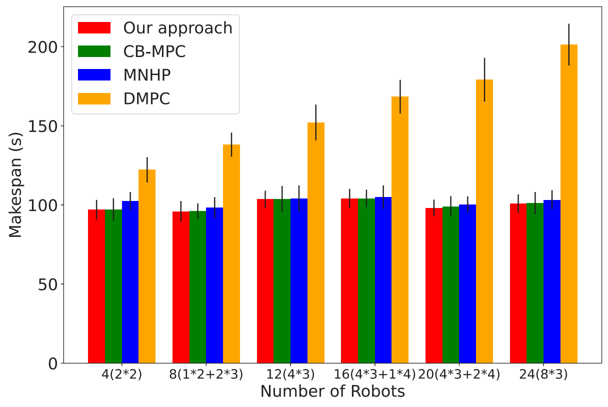
(a) Success rate.



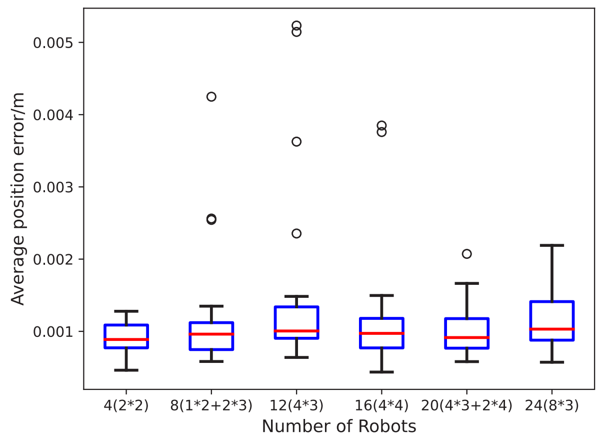
(b) Computation time.



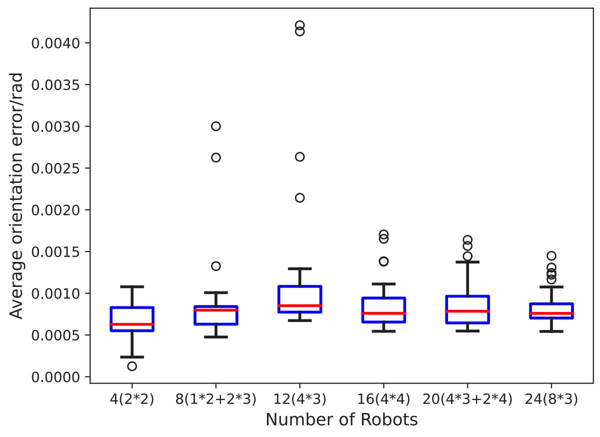
(c) Average travel distance.



(d) Makespan.

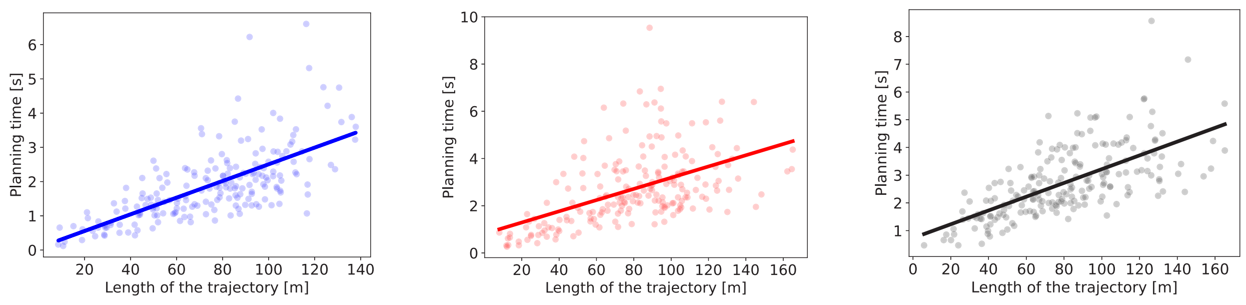


(e) Average position error.



(f) Average orientation error.

Fig. 13. The results of the formation generation experiment.



(a) Linear formation. (b) Triangular formation. (c) Rectangular formation.

Fig. 14. Planning time as the trajectory length increases.

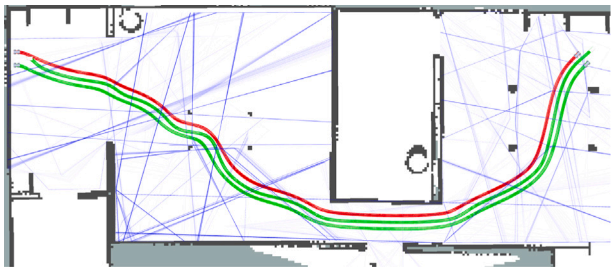


Fig. 15. An example trajectory planned for a triangular formation of two car-like robots and one differential drive robot. The leader trajectory is in red, and the follower trajectories are green. Black areas are obstacles, and blue areas represent the safe convex regions generated using IRIS [35]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

trajectory length. In Table 4 and Fig. 17 we break down the planning time into its constituent components We also show the proportion of the planning time. For each formation type, safe corridor generation takes a few milliseconds, and initial path planning takes around 75 milliseconds on average. The majority of planning time is then allocated to trajectory optimisation. We break down the trajectory optimisation time for the leaders and followers in Table 5. Trajectory optimisation takes longer for the leader, as the collision avoidance constraints for the entire formation are considered during planning, which reduces the solution space. Conversely, each follower only has to consider their own obstacle avoidance constraints, which alongside a quadratic cost function allows for efficient trajectory optimisation.

Next, we evaluate the quality and feasibility of the previously generated trajectories for each formation type. Our planner produces collision-free trajectories across all problem instances. We present an example trajectory of a triangular formation consisting of 2 car-like robots and 1 differential drive robot in Fig. 15, where the formation must move an object from the initial point $(73.0\ m, 39.0\ m)$ to the target point $(57.0\ m, 39.0\ m)$. In Table 6, we show the steering angles, velocities, travel time, and formation error across all problem instances. These results show that our synthesised trajectories keep the maximum steering angle and velocity within the limits detailed in Table 3, where the formation error is often below $0.1m$. Moreover, the maximum velocity often approaches the upper limit of 1 m/s. These results demonstrate that our formation planner synthesises efficient trajectories without violating kinematic constraints.

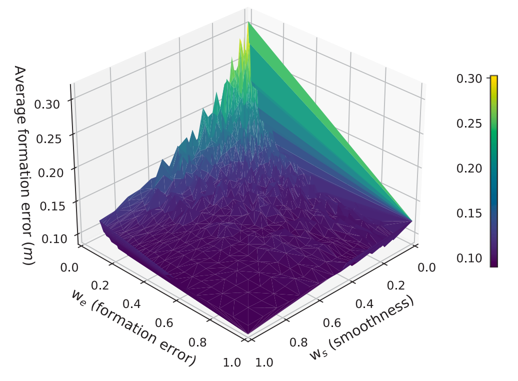
To provide insight into our parameter choices, we conducted additional experiments to show how the initial values of w_s in (8) and w_e in (17) affect formation planning performance. Recall that w_s is used to penalise the control effort in the leader's cost function, and w_e penalises the deviation from the reference trajectory in the followers' cost function. For this, we randomly generate 35 initial and target configurations for a rectangular formation in the environment in Fig. 11. For each configuration pair, we set w_s and w_e to $0.05 \times 1.1^k, k \in \{1, \dots, 32\}$. This results in 1024 combinations. For each combination, we measure the average formation position error, maximum formation position error, average speed, average acceleration, average angular velocity, and task completion time. We present the results in Fig. 16.

Fig. 16(a) and Fig. 16(b) show that formation error decreases significantly as our parameters increase. As w_s increases, the leader's trajectory and the corresponding reference trajectory for the followers becomes smoother. This is reflected in the reduction in average acceleration and angular velocity in Fig. 16(d) and Fig. 16(e), which makes the trajectory easier for followers to track. As w_e increases, followers will attempt to track the reference trajectory more closely. With this, increasing the parameters leads to higher rigidity. However, increased rigidity requires more precise control which leads to longer task completion times, as illustrated in Fig. 16(f). This imposes a trade-off between rigidity and completion time when selecting w_s and w_e. For our experiments, we choose $w_s = 0.5$ and $w_e = 0.1$, as this is where the decrease in formation error begins to slow.

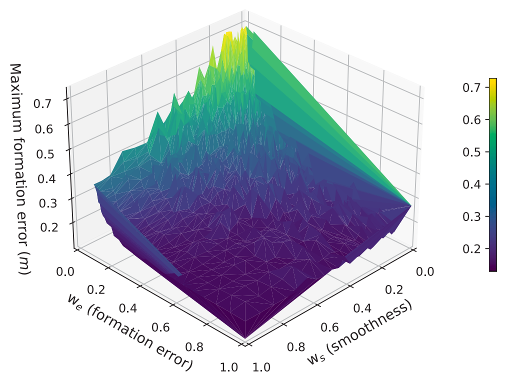
8.3. Formation coordination performance

Finally, we evaluate the efficacy of our multi-formation coordination approach. We demonstrate scalability up to 23 formations and 92 robots. For each number of formations ranging from 2 to 23, we generate 30 random problems, where each problem is a set of start and target positions for each formation. Across all problem instances, no robot begins or terminates its motion in any critical section, as discussed in Section 6.2. In Fig. 18(a), we detail the computation time for our approach to achive coordination, i.e. lines 3–18 of Alg. 5. Our approach scales exponentially in the number of formations, as the robots are coupled during coordination. However, the trajectory optimisation time (lines 19–23 in Alg. 5) scales linearly in the number of robots if we compute each robot's trajectory in sequence. We illustrate this in Fig. 18(b). In practice, each robot's trajectory can be computed in parallel after coordination.

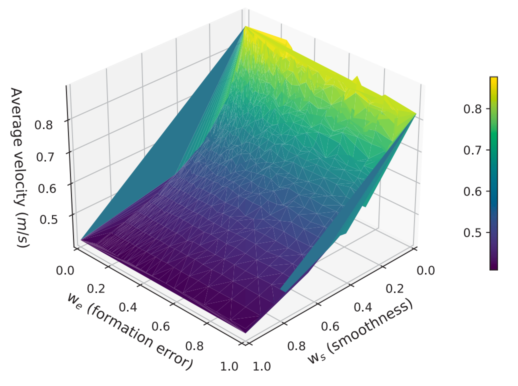
We also compare our coordination approach to the prioritised DWA proposed in [22], where coordination is only initiated when one formation detects another. We demonstrate this approach using the example in Fig. 10(e)- Fig. 10(h). Here, formation A detects formation B, where formation B has a higher priority. Formation A then creates a new safe convex region using IRIS which avoids formation B and nearby obstacles. This causes formation A to avoid formation B by deviating from its original trajectory. This is repeated until formation B has cleared formation A's path, at which point formation A can continue to track its trajectory towards the target. This ad-hoc approach often avoids collisions, but causes robots to travel further than our approach, where robots wait. Moreover, unlike our approach collision avoidance is not guaranteed. For the formation coordination problems generated above, we measure the success rate, total running time (i.e. the total time to run Alg. 5), makespan, and average travel distance of the two approaches. Upon reaching 15 formations, the success rate of prioritised DWA drops to zero, whereas our method retains a 100% success rate,



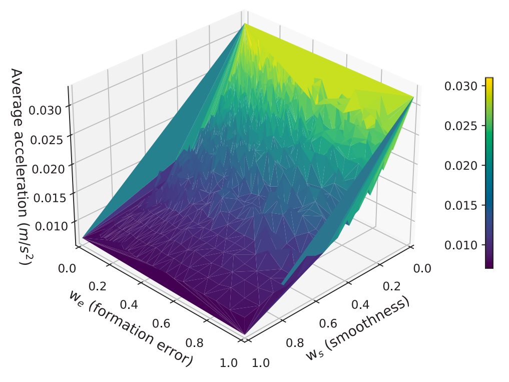
(a) Average formation error over cost weight w_e and w_s.



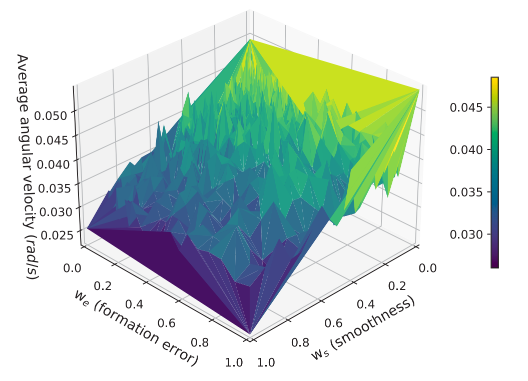
(b) Maximum formation error over cost weight w_e and w_s.



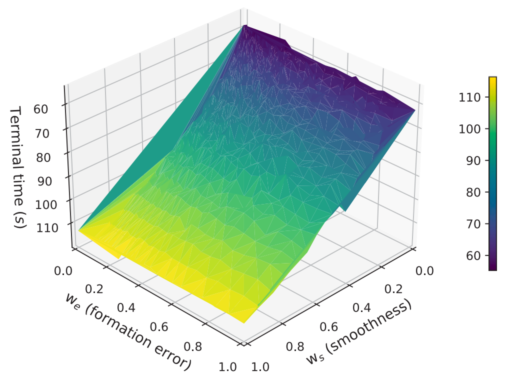
(c) Average velocity over cost weight w_e and w_s.



(d) Average acceleration over cost weight w_e and w_s.



(e) Average angular velocity over cost weight w_e and w_s.



(f) Terminal time over cost weight w_e and w_s.

Fig. 16. The performance of formation planning under varying values of the smoothness cost weight w_s in (8) and the formation error cost weight w_e in (17). As w_s increases, smoother trajectories are synthesised. As w_e increases, the formation receives heavier penalties for breaking rigidity.

Table 4
The computation time for each formation planning component.

Number of robots	Safe corridor generation	Coarse path planning	Trajectory optimisation
Linear formation (2 car-like)	2.904 ms ± 1.451 ms	65.204 ms ± 39.085 ms	1820.948 ms ± 1018.799 ms
Triangular formation (1 diff-drive + 2 car-like)	2.978 ms ± 1.264 ms	84.241 ms ± 54.553 ms	2617.272 ms ± 1498.528 ms
Rectangular formation (2 diff-drive + 2 car-like)	2.965 ms ± 1.628 ms	75.953 ms ± 47.587 ms	2647.546 ms ± 1217.983 ms

Table 5
The time to optimise trajectories for leader and follower robots.

Number of robots	Trajectory optimisation (leader)	Trajectory optimisation (followers)
Linear formation (2 car-like)	1555.389 ms ± 915.513 ms	265.559 ms ± 121.393 ms
Triangular formation (1 diff-drive + 2 car-like)	2070.89 ms ± 1238.453 ms	546.382 ms ± 295.349 ms
Rectangular formation (2 diff-drive + 2 car-like)	1954.89 ms ± 945.326 ms	692.656 ms ± 313.688 ms

Table 6
The maximum steering angles ϕ_{max}, maximum velocities v_{max}, average velocities $\overline{v}$, maximum formation errors e_{max}, and average formation error $\overline{e}$ for different formations. Units are shown in square brackets.

Number of robots	ϕ_{max} [rad]	v_{max} [$\frac{m}{s}$]	$\overline{v}$ [$\frac{m}{s}$]	e_{max} [mm]	$\overline{e}$ [mm]
Linear formation (2 car-like)	0.575 ± 0.148	0.952 ± 0.021	0.776 ± 0.094	104.591 ± 13.339	77.097 ± 7.431
Triangular formation (1 diff-drive + 2 car-like)	0.579 ± 0.081	0.954 ± 0.034	0.755 ± 0.104	172.278 ± 24.151	89.403 ± 8.919
Rectangular formation (2 diff-drive + 2 car-like)	0.567 ± 0.135	0.955 ± 0.022	0.762 ± 0.089	178.444 ± 15.086	92.988 ± 6.733

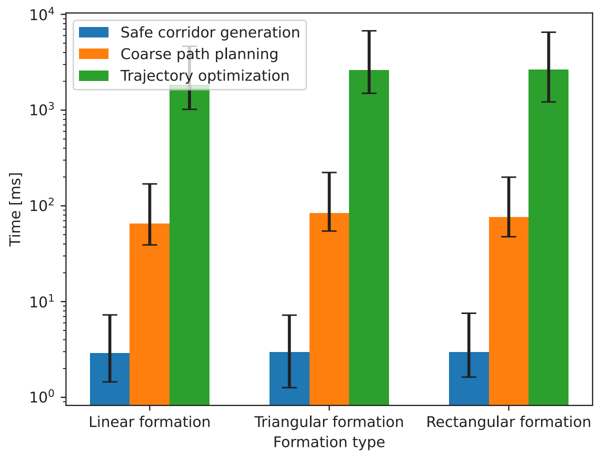


Fig. 17. The computation time for each formation planning component.

as shown in Fig. 18(c). Our approach is guaranteed to be collision-free so long as the formations adhere to their updated paths, i.e. the precedence constraints are satisfied. Conversely, prioritised DWA is myopic, and can lead to frequent deadlocks in tightly constrained environments. Our approach is also more computationally efficient than prioritised DWA, as shown in Fig. 18(d), as prioritised DWA requires convex region updates at each time step, which is expensive. We present the makespan and travel distance results in Fig. 18(e) and Fig. 18(f), respectively. Our approach produces lower makespans and shorter trajectories. Under our method, low priority formations wait at critical sections rather than deviate from their original path. This means the distance travelled is unchanged from the original trajectories. In contrast, under prioritised DWA [22], robots repeatedly deviate from their original trajectory to prevent collisions, which increases the travel distance and makespan, as illustrated in Fig. 10.

8.4. Demonstration of the H-MFPC framework

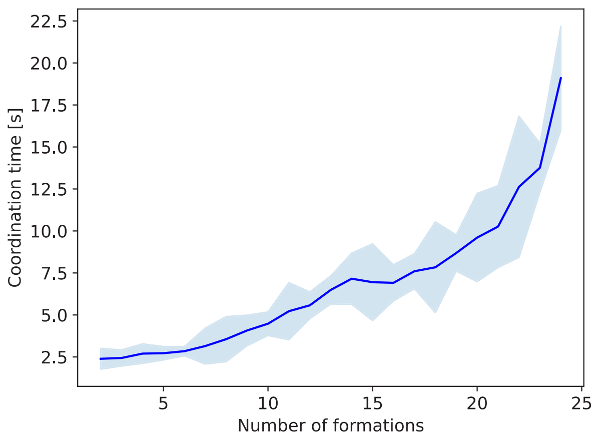
In this section, we provide a complete demonstration of our H-MFPC framework in Gazebo [51]. The simulation results are recorded in our supplementary video, with snapshots shown in Fig. 19(f). Here, we use the Agilex Hunter 2.0 [52] for car-like robots, and the MiR-100 transport robot [53] for differential-drive robots. We use a random environment, where each obstacle is a convex polygon generated by randomly selecting four points on the circumference of a randomly generated circle. We use the control law in Section 7 to control car-like robots using the front wheel steering angle and rear wheel velocity, and to control differential-drive robots using the velocity and angular velocity. The simulation task involves six car-like robots and four differential-drive robots. The robots must form two triangular formations and one rectangular formation, as shown by the dashed shapes in Fig. 19(a). We apply the formation generation approach in Section 4 to drive each robot to their formation locations, as shown in Fig. 19(b). After the formations are generated, each formation must navigate to its target position, as shown by the solid shapes in Fig. 19. We apply the formation planner in Section 5 to synthesise feasible trajectories for each formation. In Fig. 19(d), the green rectangular formation is close to colliding with the yellow triangular formation. Therefore, we apply the formation coordination method in Section 6 such that the green rectangular formation yields to the yellow triangular formation. Formation coordination is applied again in Fig. 19(e), where the red triangular formation must yield to the green rectangular formation. In Fig. 19(f), each formation successfully reaches their target configuration.

9. Conclusion

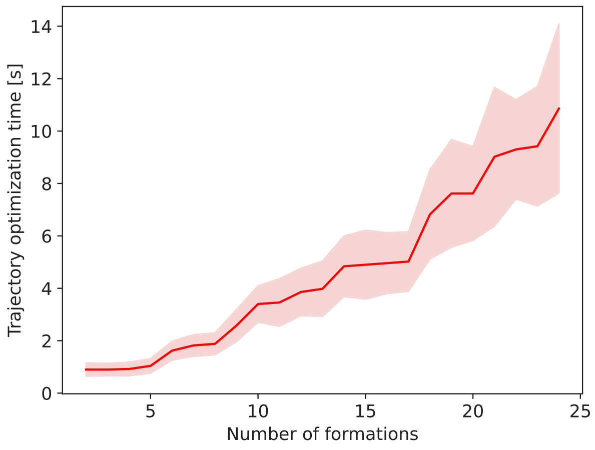
In this paper, we have proposed a comprehensive framework for H-MFPC. During the formation generation phase, we integrate a conflict-based multi-agent path finding with enhanced trajectory optimisation, thereby ensuring the algorithm's robustness and scalability. In the context of formation planning, we propose an iterative two-stage trajectory optimisation approach, aiming to satisfy kinematic constraints while concurrently minimising the total time cost of the entire formation. Regarding formation coordination, we present a loosely coupled multi-formation coordination algorithm, specifically designed to resolve deadlock and inter-formation collisions. In future work, we will adapt our framework to handle dynamic and human-populated environments. Such environments have multiple sources of uncertainty, such as unexpected obstacle appearance, or robot delays during execution. Therefore, we will capture the effects of uncertainty on robot execution to synthesise robust formation behaviour. We will also relax the rigid formation constraints to allow flexible and efficient object transportation which can navigate through obstacles by adjusting the object height. Additionally, to achieve safer and more reliable transportation we plan to incorporate dynamic characteristics (such as the object's mass and inertia) into the proposed kinematic-based control framework, dynamically adjusting the robots' trajectories to account for load variations caused by the terrain or other factors.

CRediT authorship contribution statement

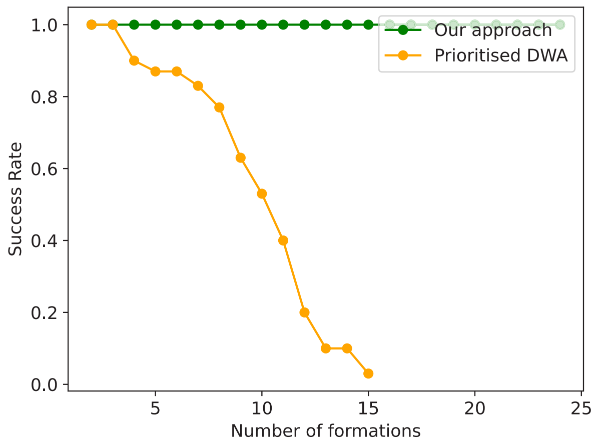
Weijian Zhang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Charlie Street:** Writing – review & editing, Visualization, Validation, Supervision, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Masoumeh Mansouri:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.



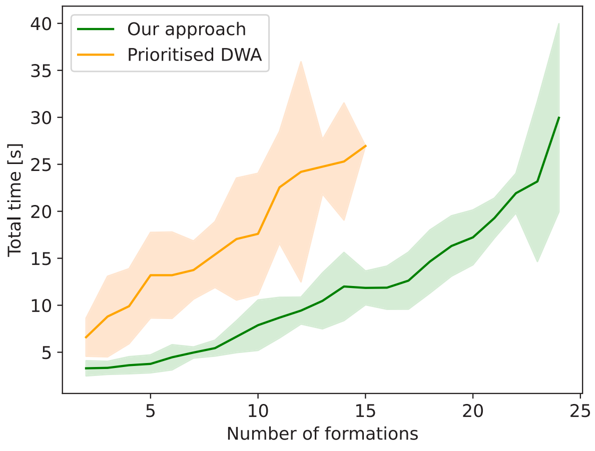
(a) Coordination time.



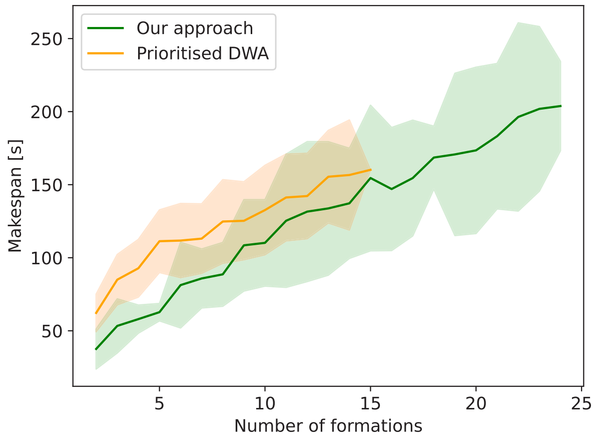
(b) Trajectory optimisation time.



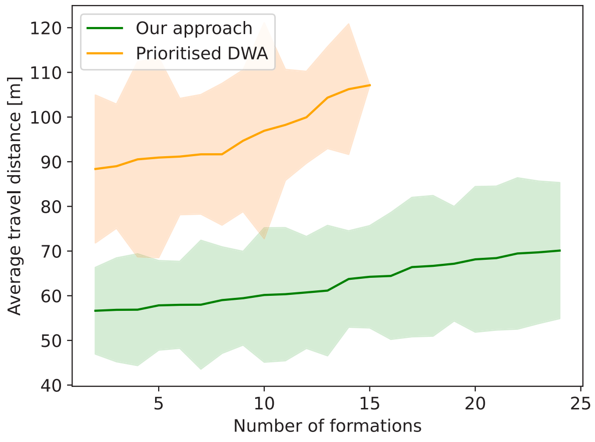
(c) Success rate against prioritised DWA [22].



(d) Total running time (coordination time + trajectory optimization time) against prioritised DWA [22].

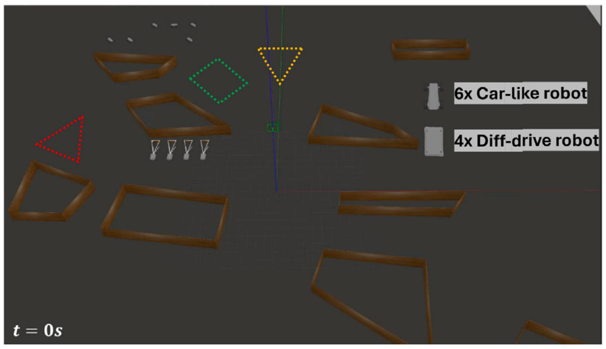


(e) Makespan against prioritised DWA [22].

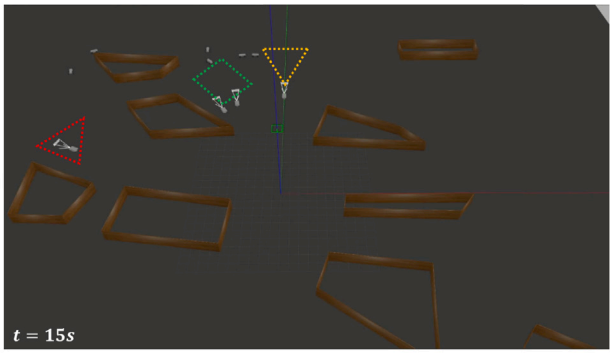


(f) Travel distance against prioritised DWA [22].

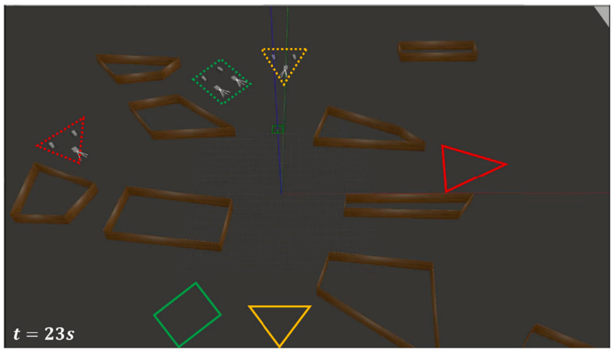
Fig. 18. Formation coordination performance across different metrics. Solid lines represent means, where the bands around each line show one standard deviation.



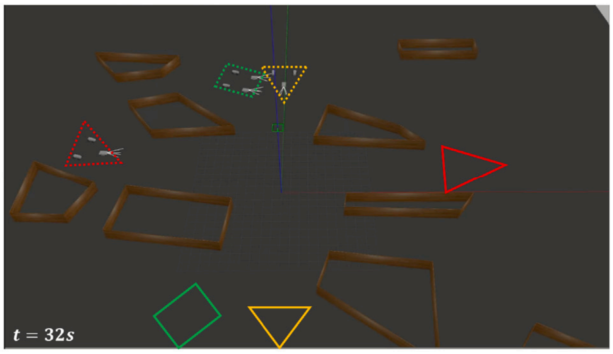
(a) A set of six car-like robots and four differential-drive robots must form two triangular formations and one rectangular formation.



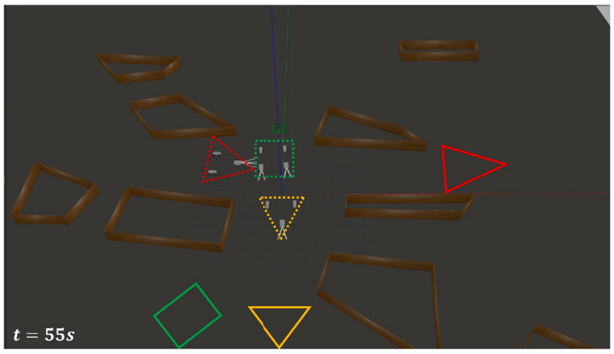
(b) The robots navigate to their formation locations.



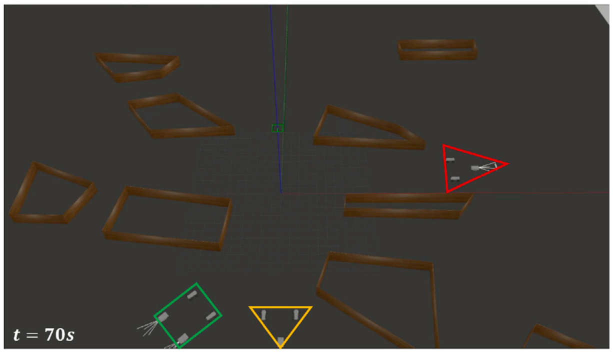
(c) The robots reach their formation locations and begin to navigate towards their target configurations.



(d) The green rectangular formation yields to the yellow triangular formation.



(e) The red triangular formation yields to the green rectangular formation.



(f) Each formation successfully reaches its target configuration.

Fig. 19. An end-to-end demonstration of our H-MFPC framework.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to the source code is mentioned in the paper.

References

[1] E. Tuci, M.H. Alkilabi, O. Akanyeti, Cooperative object transport in multi-robot systems: A review of the state-of-the-art, Front. Robotics AI 5 (2018) 59, http://dx.doi.org/10.3389/frobt.2018.00059.

[2] A. Franchi, A. Petitti, A. Rizzo, Distributed estimation of the inertial parameters of an unknown load via multi-robot manipulation, in: 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 6111–6116, http://dx.doi.org/10.1109/cdc.2014.7040346.

[3] F. Kennel-Maushart, S. Coros, Payload-aware trajectory optimisation for non-holonomic mobile multi-robot manipulation with tip-over avoidance, IEEE Robot. Autom. Lett. (2024) http://dx.doi.org/10.1109/lra.2024.3427555.

[4] M.A. Neumann, C.A. Kitts, A hybrid multirobot control architecture for object transport, IEEE/ASME Trans. Mechatronics 21 (6) (2016) 2983–2988, http://dx.doi.org/10.1109/tmech.2016.2580539.

[5] B. Hichri, J.-C. Fauroux, L. Adouane, I. Doroftei, Y. Mezouar, Design of cooperative mobile robots for co-manipulation and transportation tasks, Robot. Comput.-Integr. Manuf. 57 (2019) 412–421, http://dx.doi.org/10.1016/j.rcim.2019.01.002.

[6] J. Alonso-Mora, S. Baker, D. Rus, Multi-robot formation control and object transport in dynamic environments via constrained optimization, Int. J. Robot. Res. 36 (9) (2017) 1000–1021, http://dx.doi.org/10.1177/0278364917719333.

[7] C.C. Loh, A. Traechtler, Cooperative transportation of aload using nonholonomic mobile robots, Procedia Eng. 41 (2012) 860–866, http://dx.doi.org/10.1016/j.proeng.2012.07.255.

[8] T. Recker, H. Lurz, A. Raatz, Smooth spline-based trajectory planning for semi-rigid multi-robot formations, in: Proceedings of the IEEE International Conference on Automation Science and Engineering, CASE, IEEE, 2022, pp. 1417–1422, http://dx.doi.org/10.1109/case49997.2022.9926604.

[9] D. Koung, O. Kermorgant, I. Fantoni, L. Belouaer, Cooperative multi-robot object transportation system based on hierarchical quadratic programming, IEEE Robot. Autom. Lett. 6 (4) (2021) 6466–6472, http://dx.doi.org/10.1109/lra.2021.3092305.

[10] L. Pei, J. Lin, Z. Han, L. Quan, Y. Cao, C. Xu, F. Gao, Collaborative planning for catching and transporting objects in unstructured environments, IEEE Robot. Autom. Lett. (2023) http://dx.doi.org/10.1109/lra.2023.3335770.

[11] Z. Wang, M. Schwager, Multi-robot manipulation with no communication using only local measurements, in: 2015 54th IEEE Conference on Decision and Control, CDC, IEEE, 2015, pp. 380–385, http://dx.doi.org/10.1109/cdc.2015.7402230.

[12] S.K.A. De Sousa, R.C.S. Freire, E.Á.N. Carvalho, L. Molina, P.C. Santos, E.O. Freire, Two-layers workspace: A new approach to cooperative object transportation with obstacle avoidance for multi-robot system, IEEE Access 10 (2022) 6929–6939, http://dx.doi.org/10.1109/access.2022.3140857.

[13] W. Liu, J. Hu, H. Zhang, M.Y. Wang, Z. Xiong, A novel graph-based motion planner of multi-mobile robot systems with formation and obstacle constraints, IEEE Trans. Robot. (2023) http://dx.doi.org/10.1109/tro.2023.3339989.

[14] A. Gil-Pinto, P. Fraisse, R. Zapata, Decentralized strategy for car-like robot formations, in: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007, pp. 4176–4181, http://dx.doi.org/10.1109/iros.2007.4399434.

[15] X. Yan, D. Xu, Y. Chen, J. Lv, H. Hong, Multi-robot formation based on trajectory design and model predictive control, in: 2022 IEEE International Conference on Mechatronics and Automation, ICMA, IEEE, 2022, pp. 692–697, http://dx.doi.org/10.1109/icma54519.2022.9855942.

[16] H.W. Kuhn, The hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1955) 83–97, http://dx.doi.org/10.1002/nav.20053.

[17] G. Sharon, R. Stern, A. Felner, N.R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence 219 (2015) 40–66, http://dx.doi.org/10.1016/j.artint.2014.11.006.

[18] A. Tajbakhsh, L.T. Biegler, A.M. Johnson, Conflict-based model predictive control for scalable multi-robot motion planning, 2023, arXiv preprint arXiv:2303.01619.

[19] F. Pecora, H. Andreasson, M. Mansouri, V. Petkov, A loosely-coupled approach for multi-robot coordination, motion planning and control, in: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 28, 2018, pp. 485–493, http://dx.doi.org/10.1609/icaps.v28i1.13923.

[20] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Noguchi, A stable tracking control method for an autonomous mobile robot, in: Proceedings., IEEE International Conference on Robotics and Automation, IEEE, 1990, pp. 384–389, http://dx.doi.org/10.1109/robot.1990.126006.

[21] C.B. Low, Design, implementation, and experimental validation of a cascaded trajectory tracking controller for nonholonomic car-like wheeled mobile robots with velocity and steering controllers in the loops, in: 2017 IEEE Conference on Control Technology and Applications, CCTA, IEEE, 2017, pp. 1452–1459, http://dx.doi.org/10.1109/ccta.2017.8062663.

[22] W. Zhang, C. Street, M. Mansouri, Multi-formation planning and coordination for object transportation, in: 2023 European Conference on Mobile Robots, ECMR, IEEE, 2023, pp. 1–7, http://dx.doi.org/10.1109/ecmr59166.2023.10256314.

[23] Y. Mohan, S. Ponnambalam, An extensive review of research in swarm robotics, in: Proceedings of the World Congress on Nature & Biologically Inspired Computing, NABIC, IEEE, 2009, pp. 140–145, http://dx.doi.org/10.1109/nabic.2009.5393617.

[24] R.R. Negenborn, B. De Schutter, J. Hellendoorn, Multi-agent model predictive control: A survey, 2009, arXiv preprint arXiv:0908.1076.

[25] J. Li, M. Ran, L. Xie, Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization, IEEE Robot. Autom. Lett. 6 (2) (2020) 405–412, http://dx.doi.org/10.1109/lra.2020.3044834.

[26] L. Wen, Y. Liu, H. Li, CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints, Robot. Auton. Syst. 150 (2022) 103997, http://dx.doi.org/10.1016/j.robot.2021.103997.

[27] Y. Ouyang, B. Li, Y. Zhang, T. Acarman, Y. Guo, T. Zhang, Fast and optimal trajectory planning for multiple vehicles in a nonconvex and cluttered environment: Benchmarks, methodology, and experiments, in: 2022 International Conference on Robotics and Automation, ICRA, IEEE, 2022, pp. 10746–10752, http://dx.doi.org/10.1109/icra46639.2022.9812126.

[28] J. Alonso-Mora, P. Beardsley, R. Siegwart, Cooperative collision avoidance for nonholonomic robots, IEEE Trans. Robot. 34 (2) (2018) 404–420, http://dx.doi.org/10.1109/tro.2018.2793890.

[29] C.E. Luis, M. Vukosavljev, A.P. Schoellig, Online trajectory generation with distributed model predictive control for multi-robot motion planning, IEEE Robot. Autom. Lett. 5 (2) (2020) 604–611, http://dx.doi.org/10.1109/lra.2020.2964159.

[30] C. Bai, P. Yan, W. Pan, J. Guo, Learning-based multi-robot formation control with obstacle avoidance, IEEE Trans. Intell. Transp. Syst. 23 (8) (2021) 11811–11822, http://dx.doi.org/10.1109/tits.2021.3107336.

[31] Q. Tang, Y. Zhang, F. Yu, J. Zhang, An obstacle avoidance approach based on system outlined rectangle for cooperative transportation of multiple mobile manipulators, in: 2018 IEEE International Conference on Intelligence and Safety for Robotics, ISR, IEEE, 2018, pp. 533–538, http://dx.doi.org/10.1109/iisr.2018.8535773.

[32] A. Ammar, H. Bennaceur, I. Châari, A. Koubâa, M. Alajlan, Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments, Soft Comput. 20 (2016) 4149–4171, http://dx.doi.org/10.1007/s00500-015-1750-1.

[33] T. Han, Z.-H. Guan, M. Chi, B. Hu, T. Li, X.-H. Zhang, Multi-formation control of nonlinear leader-following multi-agent systems, ISA Trans. 69 (2017) 140–147, http://dx.doi.org/10.1016/j.isatra.2017.05.003.

[34] Y. Sirineni, P. Verma, K. Karlapalem, Traffic management strategies for multi-robotic rigid payload transport systems, in: 2019 International Symposium on Multi-Robot and Multi-Agent Systems, MRS, IEEE, 2019, pp. 225–227, http://dx.doi.org/10.1109/mrs.2019.8901082.

[35] R. Deits, R. Tedrake, Computing large convex regions of obstacle-free space through semidefinite programming, in: Proceedings of Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics, Springer, 2015, pp. 109–124, http://dx.doi.org/10.1007/978-3-319-16595-0_7.

[36] Y. Wang, M. Shan, D. Wang, A comparative analysis on rigid and flexible formations of multiple differential-drive mobile robots: a motion capability perspective, in: 2019 18th European Control Conference, ECC, IEEE, 2019, pp. 2077–2082, http://dx.doi.org/10.23919/ecc.2019.8795632.

[37] M. Saska, J.S. Mejia, D.M. Stipanovic, K. Schilling, Control and navigation of formations of car-like robots on a receding horizon, in: 2009 IEEE Control Applications,(CCA) & Intelligent Control, ISIC, IEEE, 2009, pp. 1761–1766, http://dx.doi.org/10.1109/cca.2009.5281107.

[38] M. Barer, G. Sharon, R. Stern, A. Felner, Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 5, No. 1, 2014, pp. 19–27, http://dx.doi.org/10.1609/socs.v5i1.18315.

[39] B. Li, Z. Shao, A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles, Knowl.-Based Syst. 86 (2015) 11–20, http://dx.doi.org/10.1016/j.knosys.2015.04.016.

[40] C. Rösmann, F. Hoffmann, T. Bertram, Integrated online trajectory planning and optimization in distinctive topologies, Robot. Auton. Syst. 88 (2017) 142–153, http://dx.doi.org/10.1016/j.robot.2016.11.007.

[41] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, C. Tomlin, Tunnel-milp: Path planning with sequential convex polytopes, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 7132, http://dx.doi.org/10.2514/6.2008-7132.
[42] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments, Int. J. Robot. Res. 29 (5) (2010) 485–501, http://dx.doi.org/10.1177/0278364909359210.
[43] W.C. Mitchell, A. Staniforth, I. Scott, Analysis of Ackermann Steering Geometry, Tech. Rep., SAE Technical Paper, 2006, http://dx.doi.org/10.4271/2006-01-3638.
[44] T. Chettibi, M. Haddad, H. Lehtihet, W. Khalil, Suboptimal trajectory generation for industrial robots using trapezoidal velocity profiles, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 729–735, http://dx.doi.org/10.1109/iros.2006.282621.
[45] P.J. van der Houwen, B.P. Sommeijer, Explicit Runge–Kutta (–Nyström) methods with reduced phase errors for computing oscillating solutions, SIAM J. Numer. Anal. 24 (3) (1987) 595–617, http://dx.doi.org/10.1137/0724041.
[46] A. Walther, A. Griewank, Getting started with ADOL-C, Comb. Sci. Comput. 1 (06.02) (2009) http://dx.doi.org/10.1201/b11644-8.
[47] J.A. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, CasADi: a software framework for nonlinear optimization and optimal control, Math. Program. Comput. 11 (2019) 1–36, http://dx.doi.org/10.1007/s12532-018-0139-4.
[48] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (2006) 25–57, http://dx.doi.org/10.1007/s10107-004-0559-y.
[49] B. Schäling, The Boost C++ Libraries, vol. 3, XML press Laguna Hills, 2014.
[50] B. Li, Y. Ouyang, Y. Zhang, T. Acarman, Q. Kong, Z. Shao, Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization, IEEE Robot. Autom. Lett. 6 (2) (2021) 1511–1518, http://dx.doi.org/10.1109/lra.2021.3056346.
[51] Gazebo, Open Source Robotics Foundation, 2014, [Online]. http://gazebosim.org/. (Accessed 04 April 2021).
[52] A. Villemazet, A. Durand-Petiteville, V. Cadenat, Autonomous navigation strategy for orchards relying on sensor-based nonlinear model predictive control, in: 2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM, IEEE, 2022, pp. 744–749, http://dx.doi.org/10.1109/AIM52237.2022.9863243.
[53] A.J. Moshayedi, G. Xu, L. Liao, A. Kolahdooz, Gentle survey on MIR industrial service robots: review & design, J. Mod. Process. Manuf. Prod. 10 (1) (2021) 31–50.

Weijian Zhang received the B.Eng. degree in Automation from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, in 2021, and the M.Sc. degree in Robotics from the University of Birmingham, UK, in 2022. He is currently working toward the Ph.D. degree with the School of Computer Science, University of Birmingham, UK. His research interests include multi-robot motion planning, cooperative transportation and formation control.

Charlie Street is currently a postdoctoral research fellow in the School of Computer Science at the University of Birmingham, UK. Previously, he was a postdoctoral research assistant at the Oxford Robotics Institute, University of Oxford, UK, where he also received his DPhil degree in 2022. He received the M.Sci. degree in computer science from the University of Birmingham in 2018. His research interests include robot planning under uncertainty, multi-robot coordination, formal methods applied to robotics, and continuous-time Markov models.

Masoumeh Mansouri is currently an associate professor in the School of Computer Science at the University of Birmingham, UK. Previously, she was a researcher at the Center for Applied Autonomous Sensor Systems at Örebro University, Sweden, where she received her Ph.D. as well. She was also a visiting researcher at the Oxford Robotics Institute and had a research stay in Sven Koenig's lab at the University of Southern California. Her research interest includes hybrid methods that integrate automated task/motion/coverage planning, scheduling, as well as temporal and spatial reasoning.