2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Abstraction-based Motion Coordination Control for Multi-Robot Systems

Zhuo-Rui Pan, Wei Ren, Xi-Ming Sun

Abstract— **This paper studies the motion coordination control problem for multiple mobile robots under a common workspace and reach-avoid tasks. Using abstraction-based techniques, we combine the offline and online control methods to propose a distributed motion coordination control strategy. In the offline control strategy, we partition the workspace to derive the graph for the offline planning, and construct the symbolic abstraction for each robot to design the individual controller offline. In the online control strategy, we provide a detection mechanism to check the existence of the potential robot collision, and implement the constructed symbolic abstraction and graph-searching techniques to resolve the robot collision. The combination of the offline and online control strategies results in the overall motion coordination control strategy for all robots.**

I. Introduction

Multi-robot systems (MRS) are defined as a group of robots, which are systematized in the form of a multi-agent architecture and work towards the same or different goals [1]. A key problem for MRS is how to coordinate and control the interactive activities of different robots such that different goals can be achieved, including collision avoidance, connectivity maintenance, tracking control and reach-stay task [2]. This problem is generally addressed in two directions. The first direction is to combine the motion coordination and control together, whose essence is to unify the motion coordination and control via appropriate mechanisms like energy function based approaches [3]–[5]. However, these approaches depend heavily on the existence of energy functions, and it becomes difficult to design the combination mechanisms when the tasks and environment of MRS are complex. The second direction is to separate the motion coordination and control. In this direction, a motion planning is derived first for each robot to achieve its task, and then the inter-robot collision is solved. In this paper we follow the second direction to address the motion coordination control problem of MRS.

Along the second direction, many approaches have been proposed to deal with the motion coordination control problems. Since the motion planning is derived first, many existing techniques can be applied, including optimization methods [6], sampling methods [7], learning-based methods [8] and symbolic methods [9], [10]. The comparisons among different planning techniques can be found in [11]. With the motion planning, the motion control can be treated as a tracking control while avoiding the inter-robot collision, and can also be solved via many methods, such as safe corridor methods [12] and model predictive control (MPC) based methods [13]. The motion planing and control can be considered together offline to resolve all potential inter-robot collisions, which however is inefficient since not all inter-robot collisions can be detected *a priori* and the priorly-detected inter-robot collisions may not appear in real time. Hence, the motion planning is usually studied offline while the motion control is discussed online. In this respect, the motion planning is assumed to be well-designed such that only the online motion control is investigated [14]. However, since the motion control is based on the motion planning, we need to consider the effects of the motion planning on the motion control, which motivates us to consider the motion planning and control in a unified framework.

In this paper, we address the motion control problem of multi-robot systems via abstraction-based techniques. We propose a novel abstraction-based motion control strategy, which consists of both offline and online motion control strategies. Specifically, the first step is to address the motion control problem for each robot individually. Since all robots move in a common workspace, the workspace is covered by generating finite zonotopes, whose intersection relations result in a graph [15], [16]. Using the graph theory, we establish an offline motion planning for the desired task of each robot, and further design an abstraction-based offline controller for each robot via the abstraction construction. This step is to consider each robot separately since initially each robot has no information on other robots. If all robots apply the offline controllers to move simultaneously, then the inter-robot collisions are inevitable. Hence, the second step is to design the online motion control strategy to address the motion control problem for all robots. Based on the abstraction construction, a prediction-based detection mechanism is established to check the potential inter-robot collision. For each robot involved in an inter-robot collision, we apply the constructed symbolic abstraction and forward searching techniques to design a local abstract controller to ensure the collision avoidance. Once all potential internal collisions are resolved, the desired tasks for all robots are accomplished via the proposed motion control strategy.

Problem formulation is stated in Section II. The offline and online control strategies are designed in Sections III and IV, respectively. A numerical example is given in Section V. Conclusion and future work are given in Section VI.

This work was supported by the Fundamental Research Funds for the Central Universities under Grant DUT22RT(3)090, and the National Natural Science Foundation of China under Grants 61890920 and 61890921.

Z.-R. Pan, W. Ren and X.-M. Sun are with Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, Dalian University of Technology, Dalian 116024, China. Email: panzhuorui@mail.dlut.edu.cn, wei.ren@dlut.edu.cn, sunxm@dlut.edu.cn.

II. Preliminaries and Problem formulation

$\mathbb{R} := (-\infty, +\infty)$; $\mathbb{R}^+ := [0, +\infty)$; $\mathbb{N} := \{0, 1, \ldots\}$; $\mathbb{N}^+ := \{1, 2, \ldots\}$. Given two sets $\mathbb{A}, \mathbb{B} \subset \mathbb{R}^n$, the cardinality of $\mathbb{A}$ is denoted as $|\mathbb{A}|$, and $\mathbb{B} \setminus \mathbb{A} := \{x : x \in \mathbb{B}, x \notin \mathbb{A}\}$. Given $x \in \mathbb{R}^n$, x_i is the i-th element of x, $|x|$ is the Euclidean norm of x, and $\|x\|$ is the infinity norm of x. The closed ball centered at $x \in \mathbb{R}^n$ with radius $\varepsilon \in \mathbb{R}^+$ is defined by $\mathbf{B}(x, \varepsilon) = \{y \in \mathbb{R}^n : \|x - y\| \leq \varepsilon\}$. Given a compact set $\mathbb{A} \subset \mathbb{R}^n$ and $\varepsilon > 0$, $\mathbf{E}_\varepsilon(\mathbb{A}) := \{y \in \mathbb{R}^n : \|y - x\| \leq \varepsilon, x \in \mathbb{A}\}$ is the ε-expansion of $\mathbb{A}$. A set $\mathbf{Z} \subset \mathbb{R}^n$ is a *zonotope*, if there exists $(\mathbf{c}, \mathbf{G}) \in \mathbb{R}^n \times \mathbb{R}^{n \times n_g}$ such that $\mathbf{Z} := \{\mathbf{c} + \mathbf{G}\xi : \|\xi\| \leq 1\}$, where $\mathbf{c} \in \mathbb{R}^n$ is the center and $\mathbf{G} \in \mathbb{R}^{n \times n_g}$ is the generator matrix with each column being a generator. We denote any zonotope by $\{\mathbf{c}, \mathbf{G}\}$. A *graph* is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$ and the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. An edge from node i to node j is denoted by (i, j), and implies that node i can receive information from node j. The *adjacency matrix* is denoted by $A = [a_{ij}]_{N \times N}$, where $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. The graph $\mathcal{G}$ is *undirected* if $a_{ij} = a_{ji}$ for all $i, j \in \mathcal{V}$.

A. Problem Formulation

Consider the multi-robot system with $N \in \mathbb{N}$ mobile robots, whose dynamics are presented below:

$$\Sigma_i : \quad \dot{x}_i(t) = f_i(x_i, u_i), \tag{1}$$

where $i \in \mathcal{N} := \{1, \ldots, N\}$. For each robot, $x_i := (\mathsf{p}_i, \eta_i) \in \mathbb{R}^n$, $\mathsf{p}_i \in \mathbb{R}^{n_1}$ is the position state, $\eta_i \in \mathbb{R}^{n_2}$ is the non-position state, and $n = n_1 + n_2$. $u_i \in \mathbb{R}^m$ is the control input. For each robot, the following constraints are imposed.

$$x_i \in \mathbb{X} \times \Gamma_i \subset \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}, \quad u_i \in \mathbb{U}_i \subset \mathbb{R}^m, \tag{2}$$

where $\mathbb{X}, \Gamma_i, \mathbb{U}_i$ are compact sets, and the origin is contained in $\mathbb{U}_i$. From (2), we can see that all robots share the same workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$. For the i-th robot, a curve $\mathbf{x}_i : \mathbb{R}_+ \to \mathbb{X}_i$ is called a *trajectory*, if there exists a controller $u_i : \mathbb{R}_+ \to \mathbb{U}_i$ such that (1) holds for all $t \in \mathbb{R}_+$. From the robot model (1), the trajectory can be denoted as $\mathbf{x}_i(t) := (\mathfrak{p}_i(t), \boldsymbol{\eta}_i(t))$, where $\mathfrak{p}_i(t)$ is called the *position trajectory* of the i-th robot. In addition, $\mathbf{x}_i(t, x, u)$ denotes the state reached at $t \in \mathbb{R}^+$ under $u \in \mathbb{U}_i$ from $x \in \mathbb{X} \times \Gamma_i$.

For all robots, their sensing abilities are assumed to be the same. That is, the sensing radii of all robots are the same and denoted as $\mathsf{R} > 0$. Hence, the communication graph of all robots are denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, which is time-varying and undirected. For each robot, its neighbor set at time $t \in \mathbb{R}^+$ is defined as $\mathcal{N}_i(t) = \{j \in \mathcal{V} : |\mathfrak{p}_i(t) - \mathfrak{p}_j(t)| \leq \mathsf{R}, j \neq i\}$. That is, the neighbor set is time-varying, and a robot is a neighbor of the i-th robot only when it is sensed.

In this work, each robot has its own specification defined in the workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$, where the obstacle set is denoted as $\mathbb{O} := \{\mathbb{O}_k : k \in \mathbb{K}\} \subset \mathbb{X}$ with the index set $\mathbb{K} \subseteq \mathbb{N}$. Each specification is assumed to be a reach-avoid task and is expressed as a linear temporal logic formula φ_i; see [17] for more details. Let $\mathfrak{p}_i \models \varphi_i$ mean the satisfaction of φ_i for the system (1). Given multiple robots and their specifications φ_i, *our goal* is to design distributed controllers to guarantee the satisfaction of φ_i for each robot, which avoiding the collisions among all robots. This goal is further reformulated in the following optimization problem for each robot.

$$\begin{aligned}
\min \quad & \mathcal{J}_i(\mathbf{x}_i, u_i) && \text{(3a)}\\
\text{s.t.} \quad & (1) - (2), \quad \mathfrak{p}_i \models \varphi_i, && \text{(3b)}\\
& \mathfrak{p}_i(t) \cap \mathbb{O} = \varnothing, \quad \forall t \in \mathbb{R}^+, && \text{(3c)}\\
& \mathfrak{p}_i(t) \cap \mathfrak{p}_j(t) = \varnothing, \quad \forall j \in \mathcal{N}_i(t), \forall t \in \mathbb{R}^+, && \text{(3d)}
\end{aligned}$$

where $\mathcal{J}_i : \mathbb{X} \times \Gamma_i \times \mathbb{U}_i \to \mathbb{R}$ is a predefined cost function, which is assumed to be locally Lipschitz with respect to the first argument. For each robot, (3b) shows the constraints and specification, (3c) is to ensure the obstacle avoidance, and (3d) is to avoid the collisions with other robots.

III. Zonotope-based Offline Control

In this section, we first cover the workspace via zonotope techniques, then generate an undirected graph via the applied covering, and finally apply abstraction-based techniques to design the offline controller for each robot.

A. Zonotope-based Covering

The covering strategy is presented in Alg. 1, which is to generate finite zonotopes (which are called *cells*) to cover the workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$. The generation mechanism in Alg. 1 is explained below in detail. First, we set an integer $M > n_1$ *a priori* and choose M points $\mathbf{c}_i \in \mathbb{X}$ arbitrarily, where $i \in \{1, \ldots, M\}$ (line 1). In line 2, we connect each point with at least n_1 neighbour points such that the matrix $\mathbf{G}_i$ in (5) is full-rank. By taking the point $\mathbf{c}_i$ and the matrix $\mathbf{G}_i$ as the center and the generator matrix respectively, we construct the zonotopes in line 3. Here we would like to emphasize that each zonotope is well-constructed due to the assumptions on M and $\mathbf{G}_i$. If either $M \leq n_1$ or $\mathbf{G}_i$ is not full-rank, then the generated zonotopes cannot be applied in the control design afterwards, which should be avoided in the generation mechanism. Second, if these zonotopes cannot cover the workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$ (i.e., line 4), then a larger number $M \in \mathbb{N}$ is chosen (line 5) and lines 1-3 are re-implemented to generate more zonotopes. This loop is repeated until the workspace is covered by the union of all generated zonotopes. Note that with the larger $M \in \mathbb{N}$, we can choose different connections of each center with its neighbour centers, which results in different choices of the matrix $\mathbf{G}_i$ and further affects the generation of all zonotopes. Finally, if the union of all generated zonotopes covers the workspace, then each generated zonotope is expanded slightly in line 7. In lines 1-3, the zonotope generation may result in the intersection relation between two neighbor zonotopes, and thus line 7 is to ensure the existence of the intersection relation between any two neighbor zonotopes. This step is to enhance the intersection relation, which will play an important role in the controller design afterwards. The parameter $\varepsilon > 0$ comes from the desired precision, which will be discussed in the following subsections.

Algorithm 1: Workspace Covering

Input: the workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$
Output: the covering of the workspace $\mathbb{X} \subset \mathbb{R}^{n_1}$

1 Set $M > n_1$ and choose $\{\mathbf{c}_i \in \mathbb{X} : i = 1, \ldots, M\}$
2 Connect these points such that for each $\mathbf{c}_i \in \mathbb{R}^n$, there exists a full-rank matrix

$$\mathbf{G}_i = (\mathbf{c}_{k_1} - \mathbf{c}_i, \ldots, \mathbf{c}_{k_i} - \mathbf{c}_i), \quad k_i \geq n_1 \tag{4}$$

3 Construct the zonotope

$$\mathbf{Z}_i = \{\mathbf{c}_i + 0.5\mathbf{G}_i\xi : \|\xi\| \leq 1\} \tag{5}$$

4 **while** $\mathbb{X} \setminus (\cup_{i=1}^{M} \mathbf{Z}_i) \neq \varnothing$ **do**
5 Choose a larger number $M \in \mathbb{N}$
6 Re-implement lines 1-3
7 Expand all $\mathbf{Z}_i$ via a predefined parameter $\varepsilon > 0$

$$\bar{\mathbf{Z}}_i = \mathbf{E}_\varepsilon(\mathbf{Z}_i) = \{\mathbf{c}_i, 0.5(1+\varepsilon)\mathbf{G}_i\} \tag{6}$$

8 **return** $\mathbb{Z} = \cup_{i=1}^{M} \bar{\mathbf{Z}}_i$

Since $\mathbb{X}$ is bounded, Alg. 1 can be terminated in finite time, and the union set $\mathbb{Z}$ is derived. In this way, the covering of the workspace is denoted as $\mathcal{P}(\mathbb{X}) := \mathbb{Z}$.

B. Intersection-based Graph

To show the intersection relation among all zonotopes, we construct a graph to verify the realizability of each specification. To this end, each zonotope $\bar{\mathbf{Z}}_i$ is labeled via a symbol from the set $\Pi := \{\mathsf{v}_1, \ldots, \mathsf{v}_M\}$. To connect each zonotope with the corresponding symbol, we denote $\mathbf{R}(\mathsf{v}_i) = \bar{\mathbf{Z}}_i$ for all $\mathsf{v}_i \in \Pi$ and $i \in \mathcal{M} := \{1, \ldots, M\}$.

For two sets $\mathbb{A}, \mathbb{B} \subset \mathbb{X}$, their intersection $\mathbb{A} \cap \mathbb{B}$ is *admissible* if $(\mathbb{A} \cup \mathbb{B}) \setminus (\mathbb{A} \cap \mathbb{B} \cap \mathbb{O})$ is connected; see [2, Ch. 2, Sec. 5]. With this definition, we can check if the intersection region of any two zonotopes is admissible, which is summarized as Alg. 2. In this way, we can derive all admissible intersection regions, and an admissible intersection regions means the existence of an edge between the corresponding two zonotopes. That is, the adjacency matrix A is obtained in Alg. 2 and results in an undirected graph $\mathcal{G}_\mathsf{c} := (\mathcal{V}_\mathsf{c}, \mathcal{E}_\mathsf{c})$, where $\mathcal{V}_\mathsf{c} = \Pi$ and $\mathcal{E}_\mathsf{c} \subseteq \Pi \times \Pi$ with $(\mathsf{v}_i, \mathsf{v}_j) \in \mathcal{E}_\mathsf{c}$ if $a_{ij} = 1$.

For the specification φ_i of each robot, the region of interest from φ_i is denoted as $\mathbf{R}(\pi_i) \subset \mathbb{X} \setminus \mathbb{O}$ with the symbol π_i. All symbols consist of the set $\pi := \{\pi_i : i \in \mathcal{N}\}$, and thus $\mathbf{R}(\pi) = \cup_{i \in \mathcal{N}} \mathbf{R}(\pi_i)$. In addition, the initial regions $\mathbb{X}_{i0}$ of all robots are labeled into the set $\delta := \{\delta_i : i \in \mathcal{N}\}$ with $\mathbf{R}(\delta_i) = \mathbb{X}_{i0}$. By including the sets π and δ, the graph $\mathcal{G}$ is generalized as $\bar{\mathcal{G}}_\mathsf{c} = (\bar{\mathcal{V}}_\mathsf{c}, \bar{\mathcal{E}}_\mathsf{c})$. In particular, $\bar{\mathcal{V}}_\mathsf{c} = \Pi \cup \pi \cup \delta$ and $\bar{\mathcal{E}}_\mathsf{c} \subseteq \bar{\mathcal{V}}_\mathsf{c} \times \bar{\mathcal{V}}_\mathsf{c}$ including the edges in $\mathcal{G}$ and the edges from the connections among $\mathbb{Z}$, $\mathbf{R}(\pi)$ and $\mathbf{R}(\delta) = \cup_{i \in \mathcal{N}} \mathbf{R}(\delta_i)$.

C. Offline Planning for Each Robot

With the graph $\bar{\mathcal{G}}_\mathsf{c}$, we can check if the specification of each robot can be realized and further derive the initial offline planning for each robot. Each path in $\bar{\mathcal{G}}_\mathsf{c}$, i.e., $\bar{\mathbf{p}} = \bar{\mathsf{v}}_1 \bar{\mathsf{v}}_2 \ldots$ with $\bar{\mathsf{v}}_k \in \bar{\mathcal{V}}_\mathsf{c}$, can be projected into a path in $\mathcal{G}_\mathsf{c}$, that is, $\mathbf{p} = \mathsf{v}_1 \mathsf{v}_2 \ldots$ with $\mathsf{v}_k \in \mathcal{V}_\mathsf{c}$. Furthermore, we have a sequence of finite regions $\mathbf{Z} := \{\mathbf{Z}_k = \mathbf{R}(\mathsf{v}_k) \in \mathbb{Z} : \mathsf{v}_k \in \mathbf{p}, k \in \mathbb{N}^+\}$.

Algorithm 2: Intersection-based Adjacency Matrix

Input: the partition $\mathbb{Z}$, the forbidden region $\mathbb{O} \subset \mathbb{X}$
Output: the matrix $A = [a_{ij}]$ and the set $\mathbb{I} = \{\mathbb{I}_{ij}\}$

1 **for** $i = 1 : 1 : M$ **do**
2 **if** $\bar{\mathbf{Z}}_i \cap \mathbb{O} \neq \bar{\mathbf{Z}}_i$ **then**
3 **for** $j = 1 : 1 : M$ **do**
4 **if** $\bar{\mathbf{Z}}_j \cap \mathbb{O} \neq \bar{\mathbf{Z}}_j$ **then**
5 $\Psi = \bar{\mathbf{Z}}_i \cap \bar{\mathbf{Z}}_j$
6 **if** $\Psi = \varnothing$ **then**
7 $a_{ij} = 0$
8 **else**
9 **if** $(\bar{\mathbf{Z}}_i \cup \bar{\mathbf{Z}}_j) \setminus (\Psi \cap \mathbb{O})$ *is connected* **then**
10 $a_{ij} = 1$ and $\mathbb{I}_{ij} = \Psi \setminus \mathbb{O}$
11 **else**
12 $a_{ij} = 0$
13 **else**
14 $a_{ij} = 0$
15 **else**
16 $a_{ij} = 0$ for $j \in \mathcal{M}$
17 **return** $A = [a_{ij}]$ and $\mathbb{I} = \{\mathbb{I}_{ij}\}$

Lemma 1: Consider the workspace $\mathbb{X} \subseteq \mathbb{R}^{n_1}$ and the specification φ_i of each robot. Let the graph $\bar{\mathcal{G}}_\mathsf{c}$ be constructed. φ_i can be realized if and only if

(1) there exists a path $\bar{\mathbf{p}}_i$ in $\bar{\mathcal{G}}_\mathsf{c}$ connecting δ_i and π_i;
(2) For the sequence $\mathbf{Z}_i$ from $\mathbf{p}_i$ and each $\mathbf{Z}_i^k \in \mathbf{Z}_i$, $k \in \mathbb{N}$, either of the following two conditions holds: (i) $\mathbf{Z}_i^k \setminus \mathbb{O}$ is connected; (ii) otherwise, there exists a connected subregion $\bar{\mathbf{Z}}_i^k \subset \mathbf{Z}_i^k \setminus \mathbb{O}$ such that $\bar{\mathbf{Z}}_i^k \cap (\mathbf{Z}_i^k \cap \mathbf{Z}_i^{k-1}) \cap (\mathbf{Z}_i^k \cap \mathbf{Z}_i^{k+1}) \neq \varnothing$, and $\bar{\mathbf{Z}}_i^k \cap \mathbf{R}(\pi_i) \neq \varnothing$ if $\mathbf{Z}_i^k \cap \mathbf{R}(\pi_i) \neq \varnothing$.

Lemma 1 offers a way to check if the task of each robot can be realized. $\mathbf{Z}_i^0$ is the initial workspace $\mathbf{R}(\delta_i)$, and $\mathbf{Z}_i^{|\mathbf{p}_i|+1}$ is the region of interest $\mathbf{R}(\pi_i)$. The path $\mathbf{p}_i$ for each robot may not be unique. In this case, we choose the shortest path $\mathbf{p}_i^*$ for each robot and derive the region set $\mathcal{Z}_i$.

D. Offline Control for Each Robot

Since $\mathbf{p}_i^*$ is to realize the specification of each robot, we next only focus on the corresponding region set $\mathcal{Z}_i$ to design the controller for each robot. To this end, we first denote the intersection region set in $\mathcal{Z}_i$ as $\mathcal{I}_i := \{\mathcal{I}_i^{kl} = \mathcal{Z}_i^k \cap \mathcal{Z}_i^l : \forall \mathcal{Z}_i^k, \mathcal{Z}_i^l \in \mathcal{Z}_i\}$. In each $\mathcal{Z}_i^k$, the i-th robot aim to move from $\mathcal{I}_i^{(k-1)k}$ to $\mathcal{I}_i^{k(k+1)}$ while avoiding the obstacle $\mathbb{O}$. We denote the time-discretization of Σ_i as $\mathbf{T}_\tau(\Sigma_i)$, where τ is the sampling period. If $\mathcal{Z}_i^k \in \mathcal{Z}_i$ is the workspace, then the corresponding state space is $\mathbf{X}_i^k := \mathcal{Z}_i^k \times \Gamma_i$ and the system is denoted as $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i^k)$. Next, we consider $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i^k)$ and construct its symbolic abstraction as follows:

$$\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i^k) = (X_i, X_{i0}, \mathcal{U}_i, \Delta_i), \tag{7}$$

with (i) the state set $X_i = \mathcal{A}(\mathbf{X}_i^k, \lambda)$, which means the partition of $\mathbf{X}_i^k$ via a given parameter $\lambda > 0$; (ii) the set of initial states $X_{i0} = \mathcal{A}(\mathbf{X}_{i0}^k, \lambda)$ with $\mathbf{X}_{i0}^k \subseteq \mathbf{X}_i^k$; (iii) the input set $\mathcal{U}_i = \mathcal{A}(\mathbb{U}_i, \theta)$, which means the partition of $\mathbb{U}_i$ via a given parameter $\theta > 0$; (iv) the transition relation Δ_i defined

below: for $\mathsf{y}_1, \mathsf{y}_2 \in X_i$ and $v \in \mathcal{U}_i$, $\mathsf{y}_2 \in \Delta_i(\mathsf{y}_1, u)$ if and only if $\mathsf{y}_2 \in \{\bar{q} \in X_i : \|\mathbf{x}_i(\tau, \mathsf{y}_1, v) - \bar{q}\| \leq (0.5 + e^{L_i\tau})\varepsilon\}$, where $L_i > 0$ is the Lipschitz constant of the system (1).

In the above construction mechanism, the parameters $\lambda, \theta > 0$ are respectively to partition $\mathbf{X}_i^k$ and $\mathbb{U}_i$, and are constrained via the desired equivalence relation for $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i^k)$ and $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i^k)$. The details on the abstraction construction can be found in many existing works, e.g., [15], [18], [19]. Here, τ, λ, θ are set to be the same for all regions and robots. This setting is to guarantee that there exists a common abstract workspace for all robots; otherwise, each robot has a symbolic abstraction such that multi-layered symbolic abstractions are involved. For the union $\mathbf{X}_i := \cup \mathbf{X}_i^k$, the symbolic abstraction is denoted as $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$. The following lemma is from our previous work [15] and is to show how to ensure the feedback refinement relation; see [15], [18] for more details.

Lemma 2: Consider the system $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i)$ and its abstraction $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$ with the parameters $\tau, \lambda, \theta > 0$. Given a precision $\varepsilon > 0$, if the map $\mathcal{F} : \mathbf{X}_i \to X_i$ is given by $\mathcal{F}(\mathsf{x}) = \{\mathsf{y} \in X_i : \|\mathsf{x} - \mathsf{y}\| \leq \varepsilon\}$, then $\mathcal{F}$ is a feedback refinement relation from $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i)$ to $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$, that is, $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i) \preceq_{\mathcal{F}} \mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$.

From Lemma 2 and [15], [18], if $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i) \preceq_{\mathcal{F}} \mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$ and there exists an abstract controller $\mathcal{C}_{\mathsf{a}}^i : X_i \to 2^{\mathcal{U}_i}$ such that the specification φ_i is satisfied for $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$, then the abstract controller $\mathcal{C}_{\mathsf{a}}^i$ can be refined as the controller $\mathcal{C}_i(\mathsf{x}) := \mathcal{C}_{\mathsf{a}}^i(\mathcal{F}(\mathsf{x}))$ for any $\mathsf{x} \in \mathbf{X}_i$, which further guarantees that φ_i is satisfied for $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i)$. In this way, we only need to check the existence of the abstract controller, which can be established via many existing software toolboxes and the techniques from the fields of computer science; see [20] for more details. Since we aim to minimize the cost function in (3), the abstract controller design is formulated into the following optimization problem.

$$\min \quad \mathcal{J}_i(\mathbf{y}_i, u_i) \tag{8a}$$
$$\text{s.t.} \quad \mathbf{y}_i(k+1) = \Delta_{i2}(\mathbf{y}_i(k), u_i(k)), \tag{8b}$$
$$\mathsf{q}_i \models \varphi_i, \quad \mathsf{q}_i(k) \cap \mathbf{E}_\varepsilon(\mathbb{O}) = \varnothing, \quad \forall k \in \mathbb{N}, \tag{8c}$$
$$\mathbf{y}_i(k) \in \mathcal{A}(\mathcal{Z}_i, \lambda), \quad u_i(k) \in \mathcal{A}(\mathbb{U}_i, \theta), \tag{8d}$$

where $\mathbf{y}_i(k) \in \mathbb{R}^n$ is the abstract state at the time step $k \in \mathbb{N}$, and $\mathsf{q}_i(k) \in \mathbb{R}^{n_1}$ is the abstract position in $\mathbf{y}_i(k)$.

In the problem (8), the constraint (2) is embedded in the symbolic abstraction $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$. Due to the feedback refinement relation in Lemma 2, $\mathbf{E}_\varepsilon(\mathbb{O})$ is introduced and the difference $\mathcal{J}_i(\mathbf{x}_i, u_i) - \mathcal{J}_i(\mathbf{y}_i, u_i)$ is a function of the precision $\varepsilon > 0$. That is, the solution to (8) results in an approximate solution to (3). In addition, the smaller the precision $\varepsilon > 0$ is, the smaller this difference is while the higher the computational complexity is.

Since the refinement of the abstract controller depends on the feedback refinement relation, we can update the transitions in $\mathbf{T}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$ by trimming all transitions that cannot guarantee the feedback refinement relation. The updated symbolic abstraction is denoted as $\bar{\mathbf{T}}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$, which will be used in the following online control strategy.

Algorithm 3: Potential Collision Detection: Detect()

Input: $\mathbf{p}_i, \mathsf{q}_i(k,h), \mathsf{q}_j(k,h), k \in \mathbb{N}, i, j \in \mathcal{N}, h \in \{0, 1, \ldots, \mathsf{N}\}$
Output: $\bar{\mathcal{N}}_i(k)$
Initialize: $\bar{\mathcal{N}}_i(k) = \varnothing$

```
Compare all paths p_i to derive all joint vertices
for each zonotope that corresponds to a joint vertex do
    for j ∈ N_i(k) do
        if (10) holds for some h ∈ {0, 1, ..., N} then
            N̄_i(k) = N̄_i(k) ∪ {j}
if N̄_i(k) ≠ ∅ then
    The i-th robot switches to DET mode
```

IV. Online Coordination and Control

Since each robot is considered individually in Section III, the collision among different robots is inevitable when all designed controllers are implemented such that all robots move simultaneously. Hence, it is necessary to have an online coordination and control strategy to ensure the satisfaction of the constraints (3c)-(3d), which is investigated in this section.

To deal with the potential collision, each robot is assumed to have the following two modes. The first mode is REG mode, which is the regular mode for the collision-free case and where the offline controller is implemented. The second mode is DET mode, where the potential collisions are detected and how to plan and control the robot needs to be reconsidered. Each robot is initialized in REG mode and switches among these two modes.

A. Potential Collision Detection

We start with the potential collision among all robots. From the graph $\mathcal{G}$ in Section III-D, the paths $\mathbf{p}_i$ for all robots are derived and can be compared here to check the joint vertices among different paths. Since each vertex means a zonotope, the existence of a joint vertex implies the corresponding zonotope is share by at least two robots. If the path of a robot does not intersect with other paths, then this robot just implements the offline controller and there is no need to consider the potential collision. Hence, we compare all derived paths and only consider all zonotopes that correspond to the joint vertices. This step reduces the computational complexity of the local replanning and control.

In the zonotope corresponding to a joint vertex, a necessary condition for the robot collision is that the neighbor set $\mathcal{N}_i(k)$ at the current time step $k \in \mathbb{N}$ is not empty. If a robot senses other robots, then it needs to detect the potential collision. For this purpose, we implement the MPC-based techniques and assume that the prediction step of all robots are the same and are set as $\mathsf{N} \in \mathbb{N}$. For each $h \in \{1, \ldots, \mathsf{N}\}$, the predicted abstract position of each robot is denoted as $\mathsf{q}_i(k,h) \in \mathbb{R}^{n_1}$, and all N-step predicted abstract positions are $\mathsf{q}_i[k, \mathsf{N}] \in \mathbb{R}^{n_1}$. From Lemma 2, we define the cell $\mathsf{C}(\mathsf{q}_i(k)) = \{x \in \mathbb{R}^{n_1} : \|x - \mathsf{q}_i(k)\| \leq \varepsilon\}$.

For any two robots, if $\mathsf{C}(\mathsf{q}_i[k, \mathsf{N}]) \cap \mathsf{C}(\mathsf{q}_j[k, \mathsf{N}]) = \varnothing$ with $i, j \in \mathcal{N}$, then the collision between the i-th and j-th robots

Algorithm 4: Online Motion Controller Design

Input: the offline controller from (8) for each robot
Output: Real-time planning and control strategy
Initialize: Robot i is in REG mode and moves via the offline controller

while φ_i *is not satisfied* **do**
  $\bar{\mathcal{N}}_i(t_k) \leftarrow \mathsf{Detect}()$
  if $\bar{\mathcal{N}}_i(t_k) = \varnothing$ **then**
    The robot switches to REG mode
  else
    The robot switches to DET mode
    Solve the problem (11)
    if *the solution exists* **then**
      Update the offline controller
      The robot switches to REG mode
    else
      The robot stops as a static obstacle
      Repeat Step 7

does not exist. If there exists an $h \in \{0, 1, \ldots, \mathsf{N}\}$ such that

$$\mathsf{C}(\mathsf{q}_i[k+h, 2]) \cap \mathsf{C}(\mathsf{q}_j[k+h, 2]) \neq \varnothing, \tag{9}$$

then the collision between the i-th and j-th robots exists. The condition (9) shows that any two successive predicted cells for each robot do not intersect with those for other robots. The condition (9) is motivated by the continuity of the robot motion from one cell to another. In particular, the condition (9) includes the special case where only each time step is considered. That is, $\mathsf{C}(\mathsf{q}_i(k,h)) = \mathsf{C}(\mathsf{q}_j(k,h))$, which means that two robot will reach the same abstract position while cannot show the continuous moving from one cell to another. Hence, for each robot at time step k, we define the following set of all its collision neighbors:

$$\begin{aligned}\bar{\mathcal{N}}_i(k) := \{j \in \mathcal{N}_i(k) : \exists h \in \{0, 1, \ldots, \mathsf{N}\} \\ \text{such that (9) holds}\}.\end{aligned} \tag{10}$$

The above collision detection mechanism is summarized in Alg. 3. Once the potential robot collision is detected, the robot switches to DET mode immediately.

B. Local Planning and Control

If the potential collision is detected, then the next is to replan the robots and to re-design the controller to resolve the collision. We aim to apply the MPC-based techniques to re-plan the motion of each robot. Each robot communicates with its neighbour robots and searches the follow-up N steps to derive a tradeoff between the motions of itself and its collision neighbors. Hence, the local planning and control are formulated into the following optimization problem:

$$\begin{aligned}
\min \quad & \mathcal{J}_i(\mathbf{y}_i, u_i) && \text{(11a)}\\
\text{s.t.} \quad & \mathbf{y}_i(k+1) = \Delta_{i2}(\mathbf{y}_i(k), u_i(k)), && \text{(11b)}\\
& \mathsf{q}_i \models \varphi_i, \quad \mathsf{q}_i(k) \cap \mathbf{E}_\varepsilon(\mathbb{O}) = \varnothing, && \text{(11c)}\\
& \mathbf{y}_i(k) \in \mathcal{A}(\mathbf{X}_i^k, \lambda), \quad u_i(k) \in \mathcal{A}(\mathbb{U}_i, \theta), && \text{(11d)}\\
& \mathsf{C}(\mathsf{q}_i(k)) \cap \mathsf{C}(\mathsf{q}_j(k)) = \varnothing, \quad \forall j \in \bar{\mathcal{N}}_i(k). && \text{(11e)}
\end{aligned}$$

Comparing with (8), the constraint (11e) is embedded in (11) to ensure the collision avoidance. Note that (8) is solved offline via the abstraction-based control technique, which is a backward searching method to recursively search the predecessor states from the target region, whereas (11) needs to be solved online via certain forward searching method.

To solve (11), we can focus on the symbolic abstraction $\bar{\mathbf{T}}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$ and the next N steps for each robot. At the current time step, if the potential collision is detected, then the local goal is to resolve the robot collision that may occur in the next N steps. In this case, all robots involved in the potential collision have their initial positions $\mathsf{q}_i(k)$ and local target position $\mathsf{q}_i(k, \mathsf{N})$, $i \in \mathcal{N}$ and $k \in \mathbb{N}$. Since $\bar{\mathbf{T}}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$ is discrete-time and discrete-state, the local goal can be solved by the graph-searching techniques to find a local position trajectory such that (9) does not hold for any $h \in \{1, \ldots, \mathsf{N}\}$. Since all control inputs from $\mathcal{U}_i$ needs to be checked for N steps, we estimate that the computational complexity of solving (11) is $O(N \sum_{i=1}^{N} |\mathcal{U}_i|^{\mathsf{N}})$.

The solvability of the optimization problem (11) depends on both the abstraction construction and the abstract position of each robot. On the one hand, the finer the partitions of the state and input spaces are, the more the abstract positions and transitions will be. In this way, each robot can search more transitions to find a feasible local trajectory, whereas the finer partition will inevitably result in higher computational complexity. On the other hand, if one robot is surrounded by other robots, that is, all neighbour abstract positions of this robot are taken, then the problem (11) cannot be solved, and this robot stops immediately as a static obstacle and wait for the motion of other robots. In this case, the robot is still power-on to switch back to REG mode if it is possible. The online motion control strategy is stated in Alg. 4. Since the feedback refinement relation is ensured between $\mathbf{T}_\tau(\Sigma_i, \mathbf{X}_i)$ and $\bar{\mathbf{T}}_{\tau,\lambda,\theta}(\Sigma_i, \mathbf{X}_i)$, the motion control strategy from Alg. 4 is available for the system (1) to solve the problem (3).

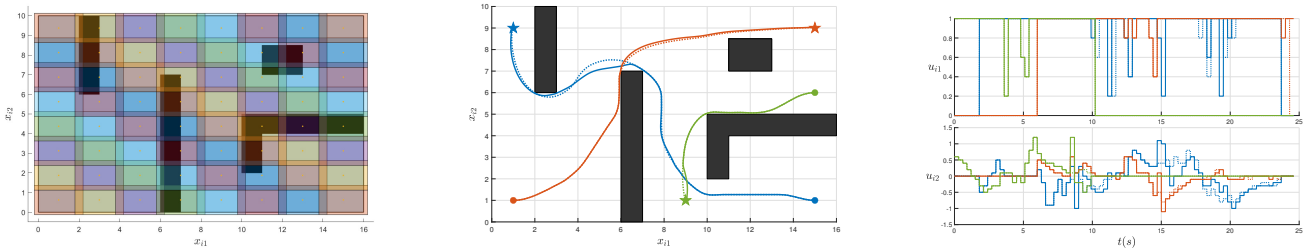
V. Numerical Example

Consider the following 3 autonomous vehicles

$$\begin{cases} \dot{x}_{i1}(t) = u_{i1}(t)\cos(x_{i3}(t)), \\ \dot{x}_{i2}(t) = u_{i1}(t)\sin(x_{i3}(t)), \\ \dot{x}_{i3}(t) = u_{i2}(t), \end{cases}$$

where $i \in \{1, 2, 3\}$, $\mathsf{p}_i := (x_{i1}, x_{i2}) \in \mathbb{R}^2$ is the vehicles position, $\eta_i := x_{i3} \in \mathbb{R}$ is orientation of the vehicle, $u_{i1} \in \mathbb{R}$ is the linear velocity and $u_{i2} \in \mathbb{R}$ is the angular velocity. The control input is assumed to be in $\mathbb{U}_i = [0, 1] \times [-1.5, 1.5]$.

The three vehicles move in the same state space $\mathbb{X} \times \Gamma$ with $\mathbb{X} = [0, 16] \times [0, 10]$ and $\Gamma = [-\pi, \pi]$. Initially, all vehicles are placed statically (see the points in Fig. 1(b)). All vehicles are to move to the target position (see the stars in Fig. 1(b)) while avoiding the obstacles $\mathbb{O} = \{\mathbb{O}_1, \mathbb{O}_2, \mathbb{O}_3, \mathbb{O}_4\}$ (i.e., the black regions in Fig. 1(a)). All vehicles do not need to move simultaneously from the initial time. Vehicle 1 (the green one in Fig. 1(b)) moves from the initial time, vehicle 2 (the blue one in Fig. 1(b)) starts to move from 2.1s, while vehicle 3 (the red one in Fig. 1(b)) starts to move from 6.3s.



(a) (b) (c)

Fig. 1. Simulation of the motion coordination problem among 3 vehicles. (a) Illustration of the workspace covered by the generated zonotopes. (b) The position trajectories of all vehicles, where the points are the initial positions and the stars are the target positions. The dotted curves are the position trajectories based on the offline controller, while the solid curves are the position trajectories based on the online controller. (c) The evolution of the control inputs. The dotted lines are the offline control inputs, while the solid curves are the online control inputs.

To achieve the task of all vehicles, the workspace $\mathbb{X}$ is covered by generating zonotopes as in Alg. 1. 64 zonotopes are generated in Fig. 1(a). Following Alg. 2, we construct a graph and further derive the offline motion planning for each vehicle. Let $\varepsilon = 0.2, \tau = 0.3, \lambda = 0.05$ and $\theta = 0.1$ to construct the local symbolic models. Using the SCOTS toolbox [20] and by solving the optimization problem (8) via the constructed local symbolic models, the offline abstract controller for each vehicle is derived; see the dotted lines in Fig. 1(c). With the offline abstract controllers, the position trajectories of all vehicles are depicted in Fig. 1(b).

To deal with the potential collisions, we apply the proposed collision detection mechanism in Section IV-A. Let $\mathsf{R} = 1$ and $\mathsf{N} = 3$. The collisions are detected by vehicles 1 and 2 at 8.4s and 15s, respectively. By implementing Alg. 4, the real-time position trajectories are shown via the solid curves in Fig. 1(b). Fig. 1(b), all the collisions are resolved and the tasks of all vehicles are accomplished. The real-time control inputs are depicted in Fig. 1(c). All computations are performed on a workstation with dual Intel Xeon Gold 6230R processors (2.10GHz) and 224GB RAM.

VI. Conclusion

In this paper, we addressed the motion coordination control problem of multi-robot systems with reach-avoid tasks. An abstraction-based control strategy was proposed. Using the workspace covering and abstraction techniques, we developed the abstract motion planning and controller for each robot. To resolve all inter-robot collisions, we proposed a prediction-based detection mechanism and implemented the forward-searching techniques to design local abstract controllers. Future work will be devoted to the extension to deal with complex tasks for multi-robot systems.

References

[1] J. K. Verma and V. Ranga, "Multi-robot coordination analysis, taxonomy, challenges and future scope," *J. Intell. Robot. Syst.*, vol. 102, no. 1, pp. 1–36, 2021.

[2] J.-C. Latombe, *Robot Motion Planning*. Springer Science & Business Media, 2012, vol. 124.

[3] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions," *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 617–632, 2015.

[4] S. G. Nersesov, P. Ghorbanian, and A. G. Aghdam, "Stabilization of sets with application to multi-vehicle coordinated motion," *Automatica*, vol. 46, no. 9, pp. 1419–1427, 2010.

[5] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2018, pp. 3723–3729.

[6] J. Wang, M. Q.-H. Meng, and O. Khatib, "EB-RRT: Optimal motion planning for mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 2063–2073, 2020.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[8] A. Hakobyan and I. Yang, "Distributionally robust risk map for learning-based motion planning and control: A semidefinite programming approach," *IEEE Trans. Robot.*, 2022.

[9] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[10] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 61–70, 2007.

[11] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?" *Annu. Rev. Control*, vol. 50, pp. 233–252, 2020.

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

[13] M. Farrokhsiar, G. Pavlik, and H. Najjaran, "An integrated robust probing motion planning and control scheme: A tube-based MPC approach," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1379–1391, 2013.

[14] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multirobot systems under LTL specifications," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1047–1062, 2021.

[15] W. Ren, J. Calbert, and R. Jungers, "Zonotope-based controller synthesis for LTL specifications," in *Proc. IEEE Conf. Decis. Control*. IEEE, 2021, pp. 580–585.

[16] W. Ren and R. Jungers, "Reachability-based control synthesis under signal temporal logic specifications," in *Proc. Am. Control Conf.* IEEE, 2022, pp. 2078–2083.

[17] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT press, 2008.

[18] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Trans. Autom. Control*, vol. 62, no. 4, pp. 1781–1796, 2017.

[19] W. Ren and D. V. Dimarogonas, "Dynamic quantization based symbolic abstractions for nonlinear control systems," in *Proc. IEEE Conf. Decis. Control*. IEEE, 2019, pp. 4343–4348.

[20] M. Rungger and M. Zamani, "SCOTS: A tool for the synthesis of symbolic controllers," in *International Conference on Hybrid Systems: Computation and Control*. ACM, 2016, pp. 99–104.