A fully distributed motion coordination strategy for multi-robot systems with local information

Pian Yu and Dimos V. Dimarogonas

***Abstract*— This paper investigates the online motion coordination problem for a group of mobile robots moving in a shared workspace. Based on the realistic assumptions that each robot is subject to both velocity and input constraints and can have only local view and local information, a fully distributed multi-robot motion coordination strategy is proposed. Building on top of a cell decomposition, a conflict detection algorithm is presented first. Then, a rule is proposed to assign dynamically a planning order to each pair of neighboring robots, which is deadlock-free. Finally, a two-step motion planning process that combines fixed-path planning and trajectory planning is designed. The effectiveness of the resulting solution is verified by a simulation example.**

I. Introduction

One challenge for multi-robot systems (MRSs) is the design of coordination strategies between robots that enable them to perform operations safely and efficiently in a shared workspace while achieving individual/group motion objectives [1]. This problem was originated from 1980s and has been extensively investigated since. In recent years, the attention that has been put on this problem has grown significantly due to the emergence of new applications, such as smart transportation and service robotics. The existing literature can be divided into two categories: *path coordination* and *motion coordination*. The former category plans and coordinates the entire paths of all the robots in advance, while the latter category focuses on decentralized approaches that allow robots to resolve conflicts online as the situation occurs[1] [2]. This paper aims at developing a fully distributed strategy for multi-robot motion coordination (MRMC).

Depending on how the controller is synthesized for each robot, the literature concerning MRMC can further be classified into two types: the reactive approach and the planner-based approach. Typical methods that generate reactive controllers consist of potential-field approach [3], sliding mode control [4] and control barrier functions [5]. These reactive-style methods are fast and operate well in real-time. However, it is well-known that these methods are sensitive to deadlocks that are caused by local minima. Moreover, guidance for setting control parameters is not analyzed formally when explicit constraints on the system states and/or inputs are presented [2]. Apart from the above, other reactive methods include the generalized roundabout policy [6] and a family of biologically inspired methods [7].

An early example of the planner-based method is the work of Azarm and Schmidt [8], where a framework for online coordination of multiple mobile robots was proposed. Based on this framework, various applications and different motion planning algorithms are investigated. Roughly speaking, the motion planning algorithms used in planner-based approaches can be divided into two types: fixed-path planning [9], [10] and trajectory planning [8], [11], [12], while the former one differs from the latter one in that the motions for individual robots are fixed along specific paths. Guo and Parker [11] proposed a MRMC strategy based on the D* algorithm [12]. In this work, each robot has an independent goal position to reach and know all path information. In [13], a distributed bidding algorithm was designed to coordinate the movement of multiple robots, which focuses on area exploration. In the work of Liu [10], conflict resolution at intersections was considered for connected autonomous vehicles, where each vehicle is required to move along a pre-planned path. In general, the fixed-path planning method is more efficient. Its major disadvantage, however, lies in the fact that it fails more often. A literature review on MRMC can be found in [1].

In this paper, we investigate the MRMC problem on a realistic setup. Robots are assumed to have limited sensing capabilities and both velocity and input constraints are considered. Conflicts are assumed to be local and can occur at arbitrary locations in the workspace. To cope with this setup, a fully distributed MRMC strategy is proposed. The contributions of this paper can be summarized as follows. Building on top of a cell decomposition, a formal definition of spatial-temporal conflict is introduced first. This definition characterizes when replanning is required for each robot. Then, a simple rule is proposed for online planning order assignment, which is deadlock-free when only local information is available to each robot. Finally, a two-step motion planning process is proposed, i.e., the fixed-path planning is activated first while the trajectory planning is activated if and only if the fixed-path planning returns no feasible solution, which allows us to leverage both the benefits of fixed-path planning and trajectory planning.

The remainder of the paper is organized as follows. In Section II, notation and preliminaries on graph theory are introduced. Section III formalizes the considered problem. Section IV presents the proposed solution in detail, which is verified by simulations in Section V. Conclusions are given in Section VI.

This work was supported in part by the Swedish Research Council (VR), the Swedish Foundation for Strategic Research (SSF) and the Knut and Alice Wallenberg Foundation (KAW).

The authors are with School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 10044 Stockholm, Sweden. piany@kth.se, dimos@kth.se

[1]In some literatures, these two terms are also used interchangeably. In this paper, we try to distinguish between the two as explained above.

II. PRELIMINARIES

A. Notation

Let $\mathbb{R} := (-\infty, \infty)$, $\mathbb{R}_{\geq 0} := [0, \infty)$, and $\mathbb{Z}_{\geq 0} := \{0, 1, 2, \ldots\}$. Denote $\mathbb{R}^n$ as the n dimensional real vector space, $\mathbb{R}^{n\times m}$ as the $n \times m$ real matrix space. Let $|\lambda|$ be the absolute value of a real number λ, $\|x\|$ and $\|A\|$ be the Euclidean norm of vector x and matrix A, respectively. Given a set Ω, 2^Ω denotes its powerset and $|\Omega|$ denotes its cardinality. Given two sets Ω_1, Ω_2, the set $\mathcal{F}(\Omega_1, \Omega_2)$ denotes the set of all functions from Ω_1 to Ω_2. The operators $\cup$ and $\cap$ represent set union and set intersection, respectively. In addition, we use $\wedge$ to denote the logical operator AND and $\vee$ to denote the logical operator OR. The set difference $A \setminus B$ is defined by $A \setminus B := \{x : x \in A \wedge x \notin B\}$. Given a point $x \in \mathbb{R}^n$ and a constant $r \geq 0$, the notation $\mathcal{B}(x, r)$ represents a ball area centered at point x and with radius r.

B. Graph Theory

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a digraph with the set of nodes $\mathcal{V} = 1, 2, \ldots, N$, and $\mathcal{E} \subseteq \{(i, j) : i, j \in \mathcal{V}, j \neq i\}$ the set of edges. If $(i, j) \in \mathcal{E}$, then node j is called a neighbor of node i and node j can receive information from node i. The neighboring set of node i is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. A graph is called undirected if $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$, and a graph is called connected if for every pair of nodes (i, j), there exists a path which connects i and j, where a path is an ordered list of edges such that the head of each edge is equal to the tail of the following edge.

III. PROBLEM FORMULATION

Consider a group of N mobile robots, whose dynamics are given by:

$$\begin{aligned} \dot{x}_i(t) &= v_i(t)\cos(\theta_i(t)), \\ \dot{y}_i(t) &= v_i(t)\sin(\theta_i(t)), \\ \dot{\theta}_i(t) &= \omega_i(t), \\ \dot{v}_i(t) &= F_i(t), \\ \dot{\omega}_i(t) &= \tau_i(t), \quad i = 1, 2, \ldots, N. \end{aligned} \tag{1}$$

where $p_i := (x_i, y_i)$ is the Cartesian position, θ_i is the orientation, v_i, ω_i are respectively the tangential velocity and the angular velocity, F_i is the force input, and τ_i is the torque input of robot i. For convenience, we define $\xi_i := (x_i, y_i, \theta_i, v_i, \omega_i)$ and $u_i := (F_i, \tau_i)$. Then, (1) can be written as $\dot{\xi}_i = f(\xi_i, u_i)$, where $f(\xi_i, u_i) = (v_i\cos(\theta_i), v_i\sin(\theta_i), \omega_i, 0, 0) + (0, 0, 0, u_i)$. The velocity and input of each robot i are subject to the constraints

$$\begin{aligned} &|v_i(t)| \leq v_i^{\max}, |\omega_i(t)| \leq \omega_i^{\max}, \\ &|F_i(t)| \leq F_i^{\max}, |\tau_i(t)| \leq \tau_i^{\max}, \forall t \geq 0. \end{aligned} \tag{2}$$

Let $\mathbb{V}_i := \{v_i : |v_i| \leq v_i^{\max}\}, \mathbb{W}_i := \{\omega_i : |\omega_i| \leq \omega_i^{\max}\}$ and $\mathbb{U}_i := \{(F_i, \tau_i) : |F_i| \leq F_i^{\max}, |\tau_i| \leq \tau_i^{\max}\}$.

Given a vector ξ_i, define the projection operator $\texttt{proj}_{p_i}(\xi_i) : \mathbb{R}^5 \to \mathbb{R}^2$ as a mapping from ξ_i to its first 2 components p_i. A curve $\boldsymbol{\xi}_i : [0, T[\to \mathbb{R}^5$ is said to be a trajectory of robot i if there exists input $u_i(t) \in \mathbb{U}_i$ satisfying $\dot{\boldsymbol{\xi}}_i(t) = f(\boldsymbol{\xi}_i(t), u_i(t))$ for all $t \in [0, T[$. A curve $\mathbf{p}_i : [0, T[\to \mathbb{R}^2$ is a position trajectory of robot i if $\mathbf{p}_i(t) = \texttt{proj}_{p_i}(\boldsymbol{\xi}_i(t)), \forall t \in [0, T[$. Given a time interval $[t_1, t_2], t_1 < t_2$, the corresponding position trajectory is denoted by $\mathbf{p}_i([t_1, t_2])$.

Supposing that the sensing radius of each robot is the same, given by $R > 0$, then the communication graph formed by the group of robots is undirected. The neighboring set of robot i at time t is given by $\mathcal{N}_i(t) = \{j : \|x_i(t) - x_j(t)\| \leq R, j \in \mathcal{V}, j \neq i\}$, so that $j \in \mathcal{N}_i(t) \Leftrightarrow i \in \mathcal{N}_j(t), i \neq j, \forall t$. The group of robots are working in a common workspace $\mathbb{X} \subset \mathbb{R}^2$, which is populated with m closed sets O_i, corresponding to obstacles. Let $\mathbb{O} = \cup_i O_i$, then the free space $\mathbb{F}$ is defined as $\mathbb{F} := \mathbb{X} \setminus \mathbb{O}$.

Each robot i is subject to its own task specification φ_i (in this work, we consider φ_i to be a reach-avoid type of task expressed as a linear temporal logic (Chapter 5 [15]) formula). Given a position trajectory $\mathbf{p}_i$, the satisfaction relation is denoted by $\mathbf{p}_i \models \varphi_i$. Given the position p_i of robot i, we refer to its *footprint* $\phi(p_i)$ as the set of points in $\mathbb{X}$ that are occupied by robot i in this position. The objective of the system is to ensure that the task specification φ_i of each robot is satisfied efficiently (in the sense that a pre-defined objective function, e.g., J_i, is minimized), while safety (no inter-robot collision) of the system is guaranteed.

Note that in this paper, it is assumed that each robot is not aware of the existence of other robots. Moreover, each robot has only local view and local information. Under these settings, the MRMC problem has to be broken into local distributed motion coordination problems and solved online for individual robots. Let $\mathbf{p}_j([t, t_j^*(t)]), j \in \mathcal{N}_i(t)$ be the local position trajectory of robot j that is available to robot i at time t, where $t_j^*(t)$ is determined by t. Then, the (online) motion coordination problem for robot i is formulated as

$$\min \quad J_i(\xi_i, u_i) \tag{3a}$$

subject to

$$(1), (2) \text{ and } \mathbf{p}_i \models \varphi_i, \tag{3b}$$

$$\phi(p_i(t')) \cap \phi(p_j(t')) = \emptyset, \forall j \in \mathcal{N}_i(t), \forall t' \in [t, t_j^*(t)], \tag{3c}$$

where constraint (3c) means that two robots can not arrive at the same cell at the same time for all $t' \in [t, t_j^*(t)]$, thus guarantees no inter-robot collision occurs.

IV. SOLUTION

The proposed solution to the motion coordination problem (3) consists of two layers: 1) an initialization layer and 2) an online coordination layer.

A. Structure of each robot

Before moving on, the structure of each robot is presented. Each robot i is equipped with four modules, the decision making module, the motion planning module, the control module and the communication module. The first three modules work sequentially while the communication module works in parallel with the first three.

Each robot i has two states: ACTIVE and PASSIVE (see Fig. 1). Robot i enters ACTIVE state when a task specification is active, and it switches to PASSIVE state if and only

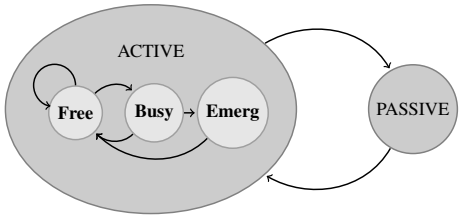

Fig. 1: Transitions of robot i.

if the task specification is completed. When robot i is in PASSIVE state, all the four modules are off and it will be viewed as a static obstacle. At the time instant that robot i enters ACTIVE state, the motion planning module is activated and an initial (optimal) plan is synthesized (explained later). During online implementation, robot i tries to satisfy its task specification safely by resolving conflicts with other robots. This is done by following some mode switching rules encoded into a Finite State Machine (FSM). Each FSM has the following three modes:

- **Free**: Robot moves as planned. This is the normal mode, in which there is no conflict detected.
- **Busy**: Robot enters this mode when conflicts are detected.
- **Emerg**: Robot starts an emergency stop process.

In Fig. 1, the transitions between different modes of the FSM are also depicted. Initially, robot i is in **Free** mode. Once conflict neighbors (will be defined later) are detected, robot i switches to **Busy** mode and the motion planning module is activated to solve the conflicts, otherwise, robot i stays in **Free** mode. When robot i is in **Busy** mode, it switches back to **Free** mode if the motion planning module returns a feasible solution, otherwise (e.g., no feasible plan is found), robot i switches to **Emerg** mode. Note that when robot i switches to **Emerg** mode, it will come to a stop but with power-on. This means that robot i will continue monitoring the environment and restart (switches back to **Free** mode) the task when it is possible.

B. Initialization

Denote by t_0^i the task activation time of robot i. Then, robot i enters ACTIVE state at t_0^i. Once robot i is ACTIVE, it first finds an optimal trajectory $\boldsymbol{\xi}_i$ (without the knowledge of other robots) such that the corresponding position trajectory $\mathbf{p}_i \models \varphi_i$. The trajectory planning problem for a single robot can be solved by many existing methods, such as search/sampling based method [16], [17], automata-based method [18] and optimization-based method [19], [20]. We note that the details of initial trajectory planning is not the focus of this paper. We refer to interested readers to corresponding literatures and the references therein.

C. Decision making

1) Conflict detection: Supposing that a cell decomposition is given over the workspace $\mathbb{X}$. The cell decomposition is a partition of $\mathbb{X}$ into finite disjoint convex regions $\Phi := \{X_1, \ldots, X_{M_1}\}$ with $\mathbb{X} = \cup_{l=1}^{M_1} X_l$. Given a set $S \subset \mathbb{R}^2$, define the map $Q : \mathbb{R}^2 \to 2^\Phi$ as

$$Q(S) := \{X_l \in \Phi : X_l \cap S \neq \emptyset\}, \tag{4}$$

which returns the set of cells in Φ that intersect with S. The cell decomposition can be computed exactly or approximately using existing approaches (Chapters 4-5 [14]). The choice depends on the particular models used for obstacles and other constraints.

The notation $\mathbf{p}_i([t, \to))$ represents the position trajectory of robot i from time t onwards. Given a position trajectory $\mathbf{p}_i([t_1, t_2])$ and a (set of) cell(s) $\Phi_l \subset \Phi$, the function $\Gamma : \mathcal{F}(\mathbb{R}_{\geq 0}, \mathbb{R}^2) \times 2^\Phi \to 2^{\mathbb{R}_{\geq 0}}$, defined as

$$\Gamma(\mathbf{p}_i([t_1, t_2]), \Phi_l) := \{t \in [t_1, t_2] : \phi(\mathbf{p}_i(t)) \cap \Phi_l \neq \emptyset\}, \tag{5}$$

gives the time interval that robot i occupies Φ_l. Let t_c be the current time and $t_i^{fl} := \min_{t > t_c}\{\mathbf{p}_i(t) \notin \mathcal{B}(p_i(t_c), R)\}$ be the first time that robot i leaves its sensing area $\mathcal{B}(p_i(t_c), R)$. Then, denote by $S_i(t_c) := \cup_{t \in [t_c, t_i^{fl}]} Q\big(\phi(\mathbf{p}_i(t))\big)$ the set of cells traversed by robot i within the time interval $[t_c, t_i^{fl}]$. Similarly, for each $j \in \mathcal{N}_i(t_c)$, let t_j^{fl} be the first time that robot j leaves its sensing area $\mathcal{B}(p_j(t_c), R)$ and $S_j(t_c)$ the set of cells traversed by robot j within $[t_c, t_j^{fl}]$. Then, the *conflict region* between robot i and j at time t_c is defined as

$$C_{i,j}(t_c) = S_i(t_c) \cap S_j(t_c). \tag{6}$$

According to (5), the time interval that robot $i(j)$ occupies the conflict cell $X_l \in C_{i,j}(t_c)$ is given by $\Gamma(\mathbf{p}_i([t_c, t_i^{fl}]), X_l)\big(\Gamma(\mathbf{p}_j([t_c, t_j^{fl}]), X_l)\big)$. Then, we have the following definition.

Definition 1: We say that there is a *spatial-temporal conflict* between robot i and j at time t if $\exists X_l \in C_{i,j}(t_c)$ such that $\Gamma(\mathbf{p}_i([t_c, t_i^{fl}]), X_l) \cap \Gamma(\mathbf{p}_j([t_c, t_j^{fl}]), X_l) \neq \emptyset$.

Based on Definition 1, define the set of conflict neighbors of robot i at time t_c, denoted by $\tilde{\mathcal{N}}_i(t_c)$, as

$$\tilde{\mathcal{N}}_i(t_c) := \{j \in \mathcal{N}_i(t_c) : \exists X_l \in C_{i,j}(t_c) \text{ s.t. } \Gamma(\mathbf{p}_i([t_c, t_i^{fl}]), X_l) \cap \Gamma(\mathbf{p}_j([t_c, t_j^{fl}]), X_l) \neq \emptyset\}.$$

Robot i switches to **Busy** mode if and only if the set of conflict neighbors is non-empty (i.e., $\tilde{\mathcal{N}}_i(t_c) \neq \emptyset$). The conflict detection process is outlined in Algorithm 1.

2) Determine planning order: Based on the neighboring relation, the graph $\mathcal{G}(t_c) = \{\mathcal{V}, \mathcal{E}(t_c)\}$ formed by the group of robots is naturally divided into one or multiple connected subgraphs, and the motion planning is conducted in parallel within each connected subgraph in a sequential manner. In order to do that, a planning order needs to be decided within each connected subgraph. In this work, we propose a simple rule to assign priorities between each pair of neighbors.

The number of neighbors of robot i at time t_c is given by $|\mathcal{N}_i(t_c)|$. Denote by $C_{i,\mathcal{N}_i(t_c)}(t_c) := \cup_{j \in \mathcal{N}_i(t_c)}\{C_{i,j}(t_c)\}$ the entire conflict region of robot i at time t_c. Let

$$\underline{\mathcal{T}}_i(t_c) = \min_{X_l \in C_{i,\mathcal{N}_i(t_c)}(t_c)} \{\min\{\mathcal{T}_i(X_l)\}\} \tag{7}$$

be the earliest time that robot i enters a conflict cell. Then, we have the following definition.

Algorithm 1 *Conflict Detection*

Input: $S_j(t_c), \Gamma(\mathbf{p}_j([t_c, t_j^{fl}]), X_l), \forall X_l \in S_j(t_c), \forall j \in \mathcal{N}_i(t_c) \cup \{i\}$.
Output: $\tilde{\mathcal{N}}_i(t_c)$.
1: Initialize $\tilde{\mathcal{N}}_i(t_c) = \emptyset$.
2: **for** $j \in \mathcal{N}_i(t_c)$ **do**
3: **if** $\exists X_l \in S_i(t_c) \cap S_j(t_c)$ s.t. $\Gamma(\mathbf{p}_i([t_c, t_i^{fl}]), X_l) \cap \Gamma(\mathbf{p}_j([t_c, t_j^{fl}]), X_l) \neq \emptyset$, **then**
4: $\tilde{\mathcal{N}}_i(t_c) = \tilde{\mathcal{N}}_i(t_c) \cup \{j\}$,
5: **end if**
6: **end for**
7: **if** $\tilde{\mathcal{N}}_i(t_c) \neq \emptyset$ **then**
8: Robot i switches to **Busy** mode.
9: **end if**

Definition 2: We say that robot i has *advantage* over robot j at time t_c if
1) $|\mathcal{N}_i(t_c)| > |\mathcal{N}_j(t_c)|$; OR
2) $|\mathcal{N}_i(t_c)| = |\mathcal{N}_j(t_c)|$ and $\underline{T}_i(t_c) < \underline{T}_j(t_c)$.

Let $\mathcal{Y}_i(t_c)$ be the set of neighbors that have higher priority than robot i at time t_c. The planning order determination process is outlined in Algorithm 2.

Algorithm 2 *Determine planning order*

Input: $\mathcal{N}_i(t_c), P_i^0$ and $\underline{T}_j(t_c), P_j^0, j \in \mathcal{N}_i(t_c)$.
Output: $\mathcal{Y}_i(t_c)$.
1: Initialize $\mathcal{Y}_i(t_c) = \emptyset$.
2: Compute $\underline{T}_i(t_c)$ according to (7),
3: **for** $j \in \mathcal{N}_i(t_c)$ **do**,
4: **if** j is in PASSIVE state or **Emerg** mode **then**,
5: $\mathcal{Y}_i(t_c) = \mathcal{Y}_i(t_c) \cup j$,
6: **else**
7: **if** j has advantage over i **then**,
8: $\mathcal{Y}_i(t_c) = \mathcal{Y}_i(t_c) \cup j$,
9: **else**
10: **if** neither robot i nor j has advantage over the other and $P_j^0 > P_i^0$ **then**,
11: $\mathcal{Y}_i(t_c) = \mathcal{Y}_i(t_c) \cup j$,
12: **end if**
13: **end if**
14: **end if**
15: **end for**

Proposition 1 (Deadlock-free): The planning order assignment rule given in Algorithm 2 will result in no cycles, i.e., $\nexists \{q_m\}_1^{\hat{k}}, \hat{k} \geq 2$ such that $q_{\hat{k}} \in \mathcal{Y}_{q_1}$ and $q_{m-1} \in \mathcal{Y}_{q_m}, \forall m = 2, \ldots, \hat{k}$.

Remark 1: The rationale behind our rule can be explained as follows. The total time required to complete the motion planning is given by $K\mathcal{O}(dt)$, where $\mathcal{O}(dt)$ represents the time complexity of one round of motion planning and K represents the number of rounds (if multiple robots conduct motion planning in parallel, it is counted as one round), which is determined by the priority assignment rule being used (e.g., if fixed priority is used, the number of rounds is $K = N$). In our rule, we assign the robot with more neighbors the higher priority, and in this way, the minimal number of rounds can be achieved. Furthermore, if two conflict robots have the same number of neighbors, then the one that arrives earlier at the conflict region should have higher priority.

D. Motion planning

The motion planning module consists of two submodules, i.e., fixed-path planning and trajectory planning. The fixed-path planning is activated first while the trajectory planning is activated if and only if the fixed-path planning returns no feasible solution.

1) Fixed-path planning: To define the fixed-path planning problem formally, the following notations are required. Denote by $\mathbf{g}_i \subset \mathbb{R}^2$ the *path* of robot i, which is a one dimensional manifold (curves in position space) that can be parameterized using the distance s_i along the path, then $\mathbf{g}_i(s_i) \in \mathbb{R}^2$. In this case, the path starts at $\mathbf{g}_i(0)$, the tangent vector $\dot{\mathbf{g}}_i(s_i) = \partial \mathbf{g}_i / \partial s_i$ has unit length. The *velocity profile* of robot i is denoted by $s_i(t)$, which is a mapping from time to distance along the path. Before starting to plan, robot i needs to wait for the updated plan from the set of neighbors that have higher priority than robot i (i.e., $j \in \mathcal{Y}_i(t_c)$) and consider them as moving obstacles. Denoted by $\mathbf{p}_j^+([t_c, \rightarrow))$ the updated position trajectory of robot j and let t_j^{fl+} be the first time that robot j leaves its sensing area $\mathcal{B}(p_j(t_c), R)$ according to $\mathbf{p}_j^+([t_c, \rightarrow))$. Then, one can define $S_j^+(t_c)$ as the set of cells traversed by robot j within $[t_c, t_j^{fl+}]$.

Definition 3: We say there is a *spatial conflict* between robot i and $j, j \in \mathcal{Y}_i(t_c)$ before (after) the fixed-path planning if $S_i(t_c) \cap S_j^+(t_c) \neq \emptyset$ ($S_i^+(t_c) \cap S_j^+(t_c) \neq \emptyset$).

Due to the fixed-path property, one can conclude that $S_i(t_c) \cap S_j^+(t_c) = \emptyset \Rightarrow S_i^+(t_c) \cap S_j^+(t_c) = \emptyset$. Therefore, in fixed-path planning, only the higher priority neighbors that have spatial conflict with robot i before the fixed-path planning need to be considered. Based on this observation, we define $C_{i,j}^+(t_c) := S_i(t_c) \cap S_j^+(t_c)$ as the (updated) set of conflict cells between robot i and j at time t_c and let $\tilde{\mathcal{Y}}_i(t_c) := \{j \in \mathcal{Y}_i(t_c) : C_{i,j}^+(t_c) \neq \emptyset\}$ be the set of higher priority robots that have spatial conflict with robot i. Denote by $\mathbf{g}_i^{t_c} := \mathbf{p}_i([t_c, \rightarrow))$ the path of robot i at time t_c. Then, the fixed-path planning problem (FPPP) is formulated as follows:

$$\min \quad J_i(\xi_i, u_i) \tag{8a}$$

subject to

$$\dot{\mathbf{g}}_i^{t_c}(s_i)\dot{s}_i \in \Pi(\mathbf{g}_i^{t_c}), \tag{8b}$$

$$\mathbf{g}_i^{t_c}(s_i(t)) \notin X_l,\ t \in \Gamma(\mathbf{p}_j^+([t_c, t_j^{fl+}]), X_l), \quad \forall j \in \tilde{\mathcal{Y}}_i(t_c), \forall X_l \in C_{i,j}^+(t_c), \tag{8c}$$

where s_i is the speed profile that needs to be optimized and $\Pi(\mathbf{g}_i^{t_c})$ is defined as

$$\Pi(\mathbf{g}_i^{t_c}) := \{\dot{\mathbf{g}}_i^{t_c}(s_i)\dot{s}_i : \exists v_i \in \mathbb{V}_i, \omega_i \in \mathbb{W}_i, u_i \in \mathbb{U}_i, s.t., \dot{\mathbf{g}}_i^{t_c}(s_i)\dot{s}_i = (\cos(\theta_i), \sin(\theta_i))v_i, \dot{\theta}_i = \omega_i, (\dot{v}_i, \dot{\omega}_i) = u_i\}.$$

2) Trajectory planning: If fixed-path planning returns no feasible solution, then it is necessary to replan the trajectory (path and velocity profile). The trajectory planning problem (TPP) can be formulated as follows:

$$\min \quad J_i(\xi_i, u_i), \tag{9a}$$

subject to

$$(3b), \tag{9b}$$

$$\texttt{proj}_{p_i}(\xi_i(t)) \notin X_l,\ t \in \Gamma(\mathbf{p}_j^+([t_c, t_j^{fl+}]), X_l), \quad \forall j \in \mathcal{Y}_i(t_c), \forall X_l \in S_j^+(t_c) \cap Q(\mathcal{B}(p_i(t_c), R)). \tag{9c}$$

If both FPPP (8) and TPP (9) return no feasible solution, robot i switches to **Emerg** mode. In this mode, robot i will continue monitoring the environment, and once the TPP (9) becomes feasible, it will switch back to **Free** mode. The motion planning process is outlined in Algorithm 3.

Remark 2: Various existing optimization toolboxes, e.g., IPOPT [22], ICLOCS2 [23], and algorithms, e.g., the configuration space-time search [24] and the Hamilton-Jacobian reachability-based motion planning [25] can be utilized to solve (8) and (9). We note that, in general (no matter which method is used), the computational complexity of TPP (9) is much higher than that of FPPP (8).

Algorithm 3 *Motion Planning*

Input: $S_j^+(t_c), \Gamma(\mathbf{p}_j^+([t_c, t_j^{fl+}]), X_l), \forall j \in \mathcal{Y}_i(t_c) \cup \{i\}, \forall X_l \in S_j^+(t_c)$.
Output: $\mathbf{p}_i^+([t_c, \to))$.

```
Compute Ỹ_i(t_c) and solve the FPPP (8),
if Solution obtained (denoted by s_i^*), then
    p_i^+([t_c, →)) = g_i^{t_c}(s_i^*),
    Robot i switches to Free mode,
else
    Solve the TPP (9),
    if Solution obtained (denoted by ξ_i^*) then,
        p_i^+([t_c, →)) = proj_{p_i}(ξ_i^*),
        Robot i switches to Free mode,
    else
        Robot i switches to Emerg mode,
        if The TPP (9) is feasible, then
            Robot i switches to Free mode,
        end if
    end if
end if
```

Remark 3: Due to the distributed fashion of the solution and the locally available information, the proposed MRMC strategy is totally scalable in the sense that the computational complexity of the solution is not increasing with the number of robots. In addition, it is straightforward to extend the work to MRSs scenarios where moving obstacles are presented.

Remark 4: We assume that the deceleration (i.e., negative force input) that each robot can take when switching to **Emerg** mode is unbounded. This guarantees that no inter-robot collision will occur during the emergency stop process since in the worst case, the robot can stop immediately. The problem of safety guarantees under bounded deceleration in **Emerg** mode will be studied in future work.

V. Simulation

We illustrate the results of the paper on a MRS consisting of $N = 7$ robots. The velocity and input constraints for each robot are given by $|v_i| \leq 2m/s, |\omega_i| \leq 115rad/s, |F_i| \leq 2m/s^2, |\tau_i| \leq 115rad/s^2$ and the sensing radius is $R = 5m$. The common workspace for the group of robots is depicted in Fig. 2 (xy axis), where the gray areas represent the obstacles and a grid representation with grid size $0.5m$ is implemented as cell decomposition. For each robot, the task specification is given by $\varphi_i := \square\neg\mathbb{O} \wedge \Diamond\square X_i^f, \forall i$, where X_i^f (marked as colored region in Fig. 2) represents the target set for robot i, while $\neg$, $\square$ and $\Diamond$ are respectively "negation", "always" and "eventually" operators in a linear temporal logic formula.

The initial optimal trajectories for each robot are depicted in Fig. 2, where the colored arrows show the moving direction of each robot. It can be seen that conflicts (i.e., inter-robot collisions) occur between robot pairs $(1,3), (1,7), (4,5)$ and $(3,6)$. During online implementation, conflicts are detected by each robot and replanning is conducted by robots 1, 4 and 6 at time instants $19.7s, 7.5s$ and $8.4s$, respectively. In the motion planning module, the ICLOCS2 [23] toolbox is implemented to solve the FPPP (8) and the TPP (9). The real-time moving trajectory for each robot is shown in Fig. 3, where all the conflicts are resolved.

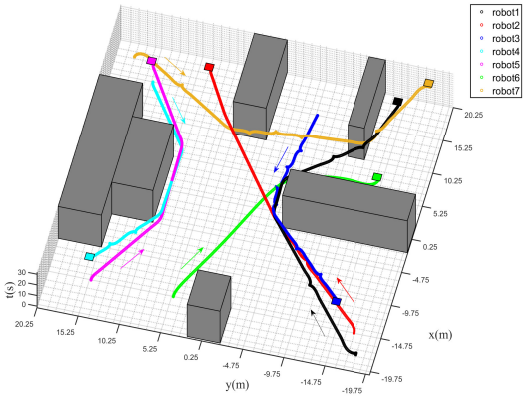

Fig. 2: The initial optimal trajectories of each robot.

The real-time evolution of velocities (v_i, ω_i) and inputs (F_i, τ_i) for each robot are given in Fig. 4 and Fig. 5, respectively. One can see that the tangential and angular velocity constraints and the force and torque input constraints are satisfied by all robots at all times. All the simulations were run in Matlab 2018b on a DELL laptop of 2.6GHz using Intel Core i7.

VI. Conclusion

In this paper, the online MRMC problem is considered. Under the assumptions that each robot has only local view and local information, and subject to both velocity and input

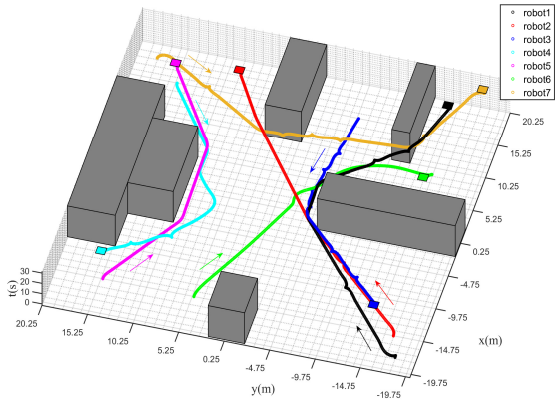

Fig. 3: The real-time trajectories of each robot.

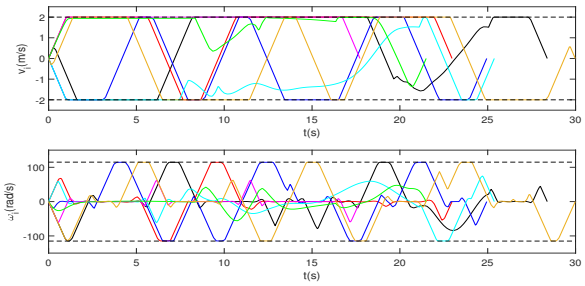

Fig. 4: The real-time evolution of v_i and ω_i.

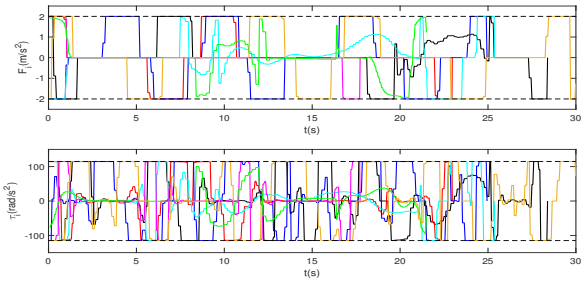

Fig. 5: The real-time evolution of F_i and τ_i.

constraints, a fully distributed motion coordination strategy was proposed for steering individual robots in a common workspace, where each robot is assigned independent task specifications. It was shown that the proposed strategy can guarantee collision-free motion of each robot. A next step is to perform real-world experiments.

ACKNOWLEDGEMENT

The authors would like to thank Yulong Gao for valuable discussions.

REFERENCES

[1] Z. Yan, N. Jouandeau, and A. A. Cherif, A survey and analysis of multi-robot coordination, *International Journal of Advanced Robotic Systems*, 10(12): 399, 2013.

[2] L. E. Parker, Path planning and motion coordination in multiple mobile robot teams, *Encyclopedia of complexity and system science*, 2009: 5783-5800.

[3] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *The International Journal of Robotics Research*, 5(1): 90-98, 1986.

[4] L. Gracia, F. Garelli, and A. Sala, Reactive siding-mode algorithm for collision avoidance in robotic systems, *IEEE Transactions on Control Systems Technology*, 21(6): 2391-2399, 2013.

[5] L. Wang, D. A. Ames, and M. Egerstedt, Safety barrier certificates for collisions-free multirobot systems, *IEEE Transactions on Robotics*, 33(3): 661-674, 2017.

[6] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Fazzoli, Decentralized cooperative policy for conflict resolution in multivehicle systems, *IEEE Transactions on Robotics*, 23(6): 1170-1183, 2007.

[7] G. A. Bekey, Autonomous Robots: From Biological Inspiration to Implementation and Control. MIT press, 2005.

[8] K. Azarm, G. Schmidt, Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation, *Proceedings of International Conference on Robotics and Automation*, 4: 3526-3533, 1997.

[9] T. Siméon, S. Thierry, and J. P. Lauumond, Path coordination for multiple mobile robots: A resolution-complete algorithm, *IEEE Transactions on Robotics and Automation*, 18(1): 42-49, 2002.

[10] C. Liu, W. Zhan, and M. Tomizuka, Speed profile planning in dynamic environments via temporal optimization, *2017 IEEE Intelligent Vehicles Symposium*, 2017: 154-159.

[11] Y. Guo, and L. E. Parker, A distributed and optimal motion planning approach for multiple mobile robots, *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, 3: 2612-2619, 2002.

[12] A. Stentz, Optimal and efficient path planning for partially known environments, *Intelligent Unmanned Ground Vehicles*, Springer, Boston, MA, 1997: 203-220.

[13] W. Sheng, Q. Yang, J. Tan, and N Xi, Distributed multi-robot coordination in area exploration, *Robotics and Autonomous Systems*, 54(12): 945-955, 2006.

[14] J. C., Latombe, Robot motion planning, Springer Science & Business Media. Vol. 124, 2012.

[15] C. Baier and J. P. Katoen, Principles of model checking. MIT press, 2008.

[16] B. Subhrajit, Search-based path planning with homotopy class constraints, *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010: 1230-1237.

[17] S. M. LaValle and J. J. Kuffner Jr, Rapidly-exploring random trees: Progress and prospects, *Algorithmic and Computational Robotics: New Directions*, 2000: 293-308.

[18] M. M. Quottrup, T. Bak, and R. I. Zamanabadi, Multi-robot planning: A timed automata approach, *IEEE International Conference on Robotics and Automation (ICRA)*, 5: 4417-4422, 2004.

[19] T. M. Howard, C. J. Green, and A. Kelly, Receding horizon model-predictive control for mobile robot navigation of intricate paths, *Field and Service Robotics*, 2010: 69-78.

[20] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization, *Proceedings of the Robotics: Science and Systems Conference*, 9(1): 1-10, 2013.

[21] J. P. Van Den Berg, and M. H. Overmars, Prioritized motion planning for multiple robots, *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005: 430-435.

[22] A. Wächter and L. Biegler, IPOPT-an interior point OPTimizer, 2009.

[23] Y. Nie, O. Faqir, and E. C. Kerrigan. ICLOCS2: Solve your optimal control problems with less pain, *in Proc. 6th IFAC Conference on Nonlinear Model Predictive Control*, 2018.

[24] D. Parsons, J. Canny, A motion planner for multiple mobile robots, *IEEE International Conference on Robotics and Automation*, 1990: 8-13.

[25] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, Robust Sequential Trajectory Planning Under Disturbances and Adversarial Intruder, *IEEE Transactions on Control Systems Technology*, 27(4): 1566-1582, 2018.

[26] M. Bennewitz, W. Burgard, and S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobiel robots, *Robotics and Autonomous Systems*, 41(2):89-99, 2002.