Toward distributed solutions for heterogeneous fleet coordination

Alessandro Palleschi[+], Anna Mannucci[#], Danilo Caporale[+], Federico Pecora[#], Lucia Pallottino[+]

***Abstract*—Warehouse mobile robotics is nowadays entering the mass-production market. Increasing the number of mobile robots up to decades raises new challenges: current industrial practice relies on centralized fleet management, which might hinder efficacy in the case of large fleets. This paper proposes and discusses a partially and a fully distributed extension of a centralized loosely coupled algorithm for multi-robot coordination. In particular, we aim at investigating: 1) how coordination can be distributed among robots, and 2) which is the minimum amount of local information required to enforce safety. Simulation results show that a partial distribution may improve performance in terms of arrival times while preserving safety and liveness.**

***Index Terms*—coordination, motion planning, multi-robot systems, mobile robots**

I. Introduction

The problem of fleet coordination has been tackled by the scientific community with either centralized, hierarchical, or distributed algorithms. Even though centralized methods deliver predictable fleet behavior, provable safety and liveness, and high performance [1], [2], they do not easily scale to hundreds of robots, are subject to single points of failure, and require the central agent to have complete knowledge of the environment and the robots in the fleets. Conversely, distributed or decentralized solutions are scalable by nature and relatively robust to communication failures [3], [4]. However, since they rely on local information, safety is highly dependent on their "degree" of distribution. In this paper, we aim to extend the centralized algorithm proposed in [1], [5] for coordinating heterogeneous fleets of vehicles subject to online non-cooperative relocation tasks. Elicited from practical applications, the approach is general with respect to the robotic platforms, robot controllers, and motion planners and poses a minimalist set of requirements to enforce safety. Aiming at large scale problems, we here investigate two possible modifications toward decentralization. To analyze their performance in terms of safety guarantees and complexity, we compare the proposed solutions with the original approach in several scenarios.

II. Notation and preliminaries

The approach in [1], [5] relies on a decoupled centralized coordinator whose main control loop, running at discrete time k, is listed in Algorithm 1. Safety is enforced by periodically revising precedence constraints that regulate access and progress through pairwise overlapping configurations along paths (namely, critical sections). Precedence orders are decided through heuristics, and conservative models of the robots' kinodynamics are used to filter out possibly unfeasible decisions. The key concepts are formally recalled in the following as a basis for our solutions (see [1], [5] for details).

Paths and spatial envelopes Consider a fleet of n (possibly heterogeneous) robots sharing an environment $\mathcal{W} \subset \mathbb{R}^3$. We use $(\cdot)_i$ to indicate that variable $(\cdot)$ refers to robot i. Let $\mathcal{Q}_i$ be the robot's configuration space, and $R_i(q) \subset \mathbb{R}^3$ its collision space when in configuration $q \in \mathcal{Q}_i$. Also, let $\mathcal{Q}_i^{\text{free}} \subseteq \mathcal{Q}_i$ be obstacle-free space. Then, $\boldsymbol{p}_i : [0,1] \to \mathcal{Q}_i^{\text{free}}$ is a path parametrized using the arc length $\sigma \in [0,1]$. For each $\boldsymbol{p}_i$, the *spatial envelope* $\mathcal{E}_i$ is defined as a set of constraints such that $\cup_{\sigma\in[0,1]} R_i(\boldsymbol{p}_i(\sigma)) \subseteq \mathcal{E}_i$ [6]. Henceforth, we assume that equality holds; also, let $\mathcal{E}_i^{\{\sigma',\sigma''\}} = \cup_{\sigma\in[\sigma',\sigma'']} R_i(\boldsymbol{p}_i(\sigma))$.

Critical sections Let $\mathcal{C}_{ij}$ be the decomposition of the set $\{q_i \in \mathcal{Q}_i, q_j \in \mathcal{Q}_j \mid R_i(q_i) \cap \mathcal{E}_j \neq \emptyset \vee R_j(q_j) \cap \mathcal{E}_i \neq \emptyset\}$ into its largest contiguous subsets, each of which is called a *critical section*. For each critical section $C \in \mathcal{C}_{ij}$, let $\ell_i^C \in [0,1]$ be the highest value of σ_i before robot i enters C; similarly, let $u_i^C \in [0,1]$ be the lowest value of σ_i after robot i exits C. We say that a critical section C is *active* whether none of the two robots has exited C. Let also $\mathcal{C}(\mathrm{k})$ be the set of active critical sections at each time k.

Precedence constraints and critical points Given a critical section $C \in \mathcal{C}(\mathrm{k})$, the precedence constraint $\langle m_i, u_j^C\rangle$, $m_i, u_j^C \in [0,1]$ is a constraint stating that robot i is not allowed to navigate beyond arc length m_i along its path until robot j has passed the arc length u_j^C - formally, $\sigma_j(t) < u_j^C \Rightarrow \sigma_i(t) < m_i$, where

$$m_i(t) = \begin{cases} \max\left\{\ell_i^C, r_{ij}(t)\right\} & \text{if } \sigma_j(t) \leq u_j^C \\ 1 & \text{otherwise} \end{cases}, \text{ with } r_{ij}(t)$$

$$\text{being } \sup_{\sigma} \left\{\sigma \in [\sigma_i(t), u_i^C] : \mathcal{E}_i^{\{\sigma_i(t),\sigma\}} \cap \mathcal{E}_j^{\{\sigma_j(t),u_j^C\}} = \emptyset\right\}.$$

Let $\mathcal{T}(\mathrm{k})$ be the set of precedence constraints such that $C \in \mathcal{C}(\mathrm{k})$ implies either $\langle m_i, u_j^C\rangle$ or $\langle m_j, u_i^C\rangle \in \mathcal{T}(\mathrm{k})$. Safety is ensured since $\mathcal{T}(\mathrm{k})$ imposes a total order of robots' motion through $\mathcal{C}(\mathrm{k})$. In particular, we define the *critical point* $\bar{\sigma}_i(\mathrm{k})$ of robot i at time k as the value of σ corresponding to the last reachable configuration along $\boldsymbol{p}_i$ which adheres to the set of constraints $\mathcal{T}(\mathrm{k})$. Thus, the *coordination problem* consists in computing and updating periodically the set of critical points $\bar{\Sigma}(\mathrm{k}) = \{\bar{\sigma}_i(\mathrm{k})\}_{i=1}^n$ such that collisions do not occur.

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 732737 (ILIAD), and in part by the Italian Ministry of Education and Research in the framework of the CrossLab project (Departments of Excellence).

[+] Dipartimento di Ingegneria dell'Informazione and Research Center "E. Piaggio", University of Pisa, Italy

[#] Center for Applied Autonomous Sensor Systems, Örebro University, Sweden

Algorithm 1: Coordination at time k.

```
1 get last state messages $\{s_i\}_{i=1}^n$;
2 if new goals have been posted then
3     update the set of paths $\mathcal{P}(\mathrm{k})$ (using appropriate
      planners);
4     update the set $\mathcal{C}(\mathrm{k})$ of critical sections;
5 revise the set $\mathcal{T}(\mathrm{k})$ of precedence constraints;
6 compute the set of critical points $\bar{\Sigma}(\mathrm{k})$;
7 communicate changed critical points;
8 sleep until control period $T_c$ has elapsed;
```

Communication The approach assumes a pear-to-pear communication between robots and the coordinator through a wireless communication channel. Robots are required to asynchronously send an updated state message s_i (each robot control period T_i) which contains the vector $(q_i, \dot{q}_i, \ddot{q}_i)$ (see [5] for further details). This information is used in line 5 to revise precedence.

The approach is proved to be safe [5] and to have linear complexity with the number of critical sections $|\mathcal{C}|$. Note also that $|\mathcal{T}| = |\mathcal{C}|$ by definition. In the next section we aim at reducing this complexity for large scale problems by exploiting different levels of decentralization.

III. Toward Distributed Coordination

We focused on two different levels of decentralization, described in the following.

Partially Distributed The first strategy is based on updating precedence constraints belonging only to a subset $\mathcal{C}_\epsilon \subseteq \mathcal{C}$: precedence constraints related to critical sections that are further than a minimum safety distance ϵ from the two robots are omitted in $\mathcal{T}(\mathrm{k})$, since they do not affect the robot behaviours at the current time. Formally, either $\langle m_i, u_j^C \rangle$ or $\langle m_j, u_i^C \rangle \in \mathcal{T}(\mathrm{k}) \iff C \in \mathcal{C}(\mathrm{k})$ and $\max(l_i^C - \xi_i(\mathrm{k}), l_j^C - \xi_j(\mathrm{k})) < \epsilon$, with $\xi_i(\mathrm{k})$ being the worst-case stopping arc-length at each time k[1], and ϵ being a user-defined integer threshold. This does not change complexity (which is still linear in $|\mathcal{C}(\mathrm{k})|$), but speeds up computation in lines 5–7 whenever $|\mathcal{C}_\epsilon(\mathrm{k})|$ (and hence $|\mathcal{T}(\mathrm{k})|$) is significantly lower than $|\mathcal{C}(\mathrm{k})|$. Note also that safety is preserved if ϵ is computed considering message delays in the worst case (due to asynchronous communication — see [5] for details) and conservative kinodynamic models.

Fully Distributed Aiming to distribute the overall complexity, we propose a fully distributed version of Algorithm 1, which is based on local communication between neighboring robots. We assume robots can communicate with all the ones that are inside their communication radius. Each robot i runs a local version of the coordination algorithm (each at time k_i), which consists in:

1) updating its path $\boldsymbol{p}_i$ when a new goal is received and broadcasting periodically the envelope $\mathcal{E}_i(\mathrm{k}_i)$ and the status $s_i(\mathrm{k}_i)$;
2) listening to incoming messages (envelopes and status) by neighboring robots $\mathcal{N}_i(\mathrm{k}_i)$;
3) updating a local set of critical sections $\mathcal{C}_i(\mathrm{k}_i) = \bigcup_{j \in \mathcal{N}_i} \mathcal{C}_{ij}(\mathrm{k}_i)$ according to incoming messages;
4) updating a precedence constraint for each $C \in \mathcal{C}_i(\mathrm{k}_i)$ – we use fixed priorities based on robot IDs to decide precedence orders while filtering out unfeasible constraints as in the original version of the algorithm;
5) computing its own critical point;
6) sleeping until the control period T_i has elapsed.

This allows to cut down complexity from $|\mathcal{C}(\mathrm{k})|$ to $|\mathcal{C}_i(\mathrm{k}_i)|$. However, the approach cannot ensure the absence of collisions whenever critical sections end beyond the communication radius, as shown in the simulation results.

[1]This quantity corresponds to the nearest arc length at which the robot can stop (if required to) when driving with maximum velocity.

IV. Simulations

We here compare the performance of an implementation of Algorithm 1 (named *Centralized* in the following) with the two proposed strategies: the *Partially Distributed* and the fully distributed (named *Distributed* in the following). We leverage the low fidelity simulator presented in [1], [5]. Robot priorities are defined so that the lower the robot ID, the higher the priority. Grey arrows in the following figures indicate precedence constraints (Ri $\rightarrow$ Rj whenever robot i yields for robot j at a given C). All tests ran on an Intel Core i7 Processor (6x2.20 GHz) and 16 GB DDR4 RAM Laptop.

Coordination of two robots We consider the scenario shown in Fig. 1 involving two robots and two critical sections. This is a particular case in which the envelope of one robot ($\mathcal{E}_1$ in the figure) intersects the other robot's envelope in the same configurations along $\boldsymbol{p}_1$. We expect the *Partially Distributed* to have the same behavior as the *Centralized* one (even though the precedence constraints related to the two critical sections are computed at a different time k). Conversely, we expect a different behavior with the *Distributed* strategy and small communication radius: since the two critical sections are further than the communication radius, the robot with the lowest priority (R2) may move toward its goal while the one with the highest priority (R1) has still to traverse the second critical section along its path. This result is confirmed by simulations, as shown qualitatively in Fig. 1 and quantitatively in Table I — we list the time the two robots reach the destination with each strategy in the latter. Note that the first two strategies have equivalent times for both robots, while this time is 30% lower for robot R2 in the Distributed case.

Coordination in case of long critical section Long critical sections are not unusual in logistics applications due to the presence of aisle in warehouses. It is also common to have two robots that need to traverse the aisle in the same direction and opposite directions. We have performed simulations in both platooning (see Fig. 2) and head-to-head (see Fig. 3) situations

TABLE I
Traveling times and collisions for two robots and a single critical section

Method	t_{R1}	t_{R2}	Collisions
Centralized	23.04	27.30	0
Partially Distributed	23.01	27.14	0
Distributed	23.06	18.41	0