Intra-Logistics with Integrated Automatic Deployment:
Safe and Scalable Fleets in Shared Spaces

H2020-ICT-2016-2017
Grant agreement no: 732737

DELIVERABLE 5.6

Distributed fleet coordination methods for industrial vehicles

Due date: month 36 (December 2019)
Deliverable type: R
Lead beneficiary: UNIPI

Dissemination Level: PUBLIC

Main author: Lucia Pallottino (UNIPI)

Contents

1 Introduction

Warehouse mobile robotics is nowadays entering the mass-production market, with companies building autonomous mobile robots with different sizes and purposes. The shared workspace among autonomous mobile robots and possibly human-piloted ones must be safely and efficiently managed. The high number of mobile robots in the warehouse has now to face new challenges since robots are heterogeneous in terms of dimensions, shapes, dynamic constraints but also in terms of tasks to be accomplished. Hence the fleet management system has to cope with both individual and fleet objectives to be achieved [1, 2]. In such cases coordination and control based on sensing and information sharing are fundamental to cope with physical/environmental constraints and limitations. Motion constraints, obstacles, unmodeled disturbances, communication constraints must be taken into account in motion planning, coordination and control of such systems. The coordination is performed through path computation to minimize one or multiple indexes representing the individual objective, such as travelled distance, time of arrival or energy spent. On the other hand collisions and stall situations must be avoided to guarantee a safe and continuous flow of the fleet. We refer to Cao et al. [3] and Parker [4] for an overview of the existing approaches and methodologies.

The problem of fleet coordination has been tackled by the scientific community with either centralized or distributed algorithms [5]. In a fully centralized framework a single decision entity collects all the available information on the fleet (position, task, objective etc. of all robots) and provides a decision or action to all the fleet members (the action to avoid an obstacle, another robot or the new task to be accomplished etc.). In a fully distributed approach robots interact with each other and based on their local information through network communications and/or local sensing decide the action to implement [6].

Although nowadays the computational power and parallel computation are improving enough to handle tens of agents with a centralized algorithm, the complexity of such algorithms will never be able to manage thousands of autonomous cars in a city, or to plan in real time the reaction to dynamical changes. From a practical point of view, management systems should also be able to handle robots of different manufacturing companies that have not been designed to interact with robots of other companies. Moreover, the communication required to send and update a global plan to every agent may not be reliable enough in a large environment and only local distributed communication may be available to be used to plan and coordinate [7]. Even though decentralized approaches cannot achieve global optimality, they manage to be robust to the lack of global information and to possible central unit faults. However, one of the main critical points in distributed algorithms is the prevention of deadlocks that may arise during the system evolution. Examples of negotiations to avoid deadlocks can be found in Jäger et al. [8], Koh et al. [9], and Manca et al. [10].

Since both approaches have benefits and drawbacks, in several applications ad hoc solutions must be found evaluating the right level of centralized/distributed approach. This depends on several aspects as the available hardware on the robots, their computational, sensorial and communication capabilities. Other important aspects are the particular responsibility that can be delegated to robots and those that must be kept in a central decision center. For example, collision avoidance is usually delegated to robots that based on the local information can compute alternative paths to avoid an unforeseen obstacle while the assignment of the tasks are decided centrally.

In this deliverable we focus on the industrial application scenario where multiple robots have to share a common environment. The environment can be narrow but structured and known. On the other hand, the environment can also be shared with other mobile platforms possibly driven by human operators. For this reason we have

approached the coordination problem in three different steps. First a particular case of local coordination of robots in narrow spaces is considered. In multi robot system, collision avoidance can be solved with a distributed approach based on interaction rules, such as keep right lane along roads for cars. In case such rules are not strictly defined coordination must ensure safety of the system. This becomes more difficult when entities involved in the conflicting situation are not a priori programmable with a common rule such as in case of human piloted robots. A deconfliction approach is hence proposed to solve robot-robot conflicts in case of a fully cooperative approach, i.e., each robot implement the same approach to avoid collisions. The approach is then tested in a human-robot conflict where the human does not follow any rule except that he/she is not is not ill-intentioned. We will show through simulations and experiments on small robotic platforms how the conflict is solved.

A second step is the implementation of a fully distributed approach where a mixed discrete-space/continuous-time distributed path planner is proposed for robot-robots collision avoidance while a regular flow of the fleet is maintained. The proposed approach is based on a discretized space on which the plan is performed while the speed is chosen to optimally use the shared resources (intersections) to avoid stop-and-go behaviours. Simulations and experimental results have been performed to sustain applicability and validity of the approach in providing collision free trajectories under some realistic assumptions.

Finally, the evaluation of centralized/distributed approach has been performed on the ILIAD coordinator proposed in D5.1. Several parts of the Iliad coordinator have been distributed to show how and until which point the responsability of the coordination can be delegated to robots. Simulation results show that the distributed approach can yield better performance of the fleet in terms of arrival times while preserving safety and liveness.

2 Deconfliction for mixed Human-Robot scenarios

2.1 Introduction

As a case scenario, picture two cars driving in opposite directions down a narrow corridor, as usually seen in parking lots and garages. When the road is clear, both drivers may tend to stay closer to the center of the roadway; consider the road to be wide enough to allow both cars to pass at the same time. When the two cars are facing *close enough*, they will start moving away from the center of the road in order to keep heading towards their respective goal; we will refer to this scenario as the *head-on scenario*.

The head-on scenario is illustrated in more detail in Figures 1 and 2, where two vehicles are operating in a right-hand rule driving environment. In the first example, Fig. 1a, the two agents are driving far enough from the center of the road, allowing both cars to pass; since no conflict is taking place, no further action needs to be taken. In Figs. 1b and 1c, the two cars are heading misaligned, therefore should avoid collision by steering to the right, in compliance with local traffic rules. Finally, in Fig. 1d, the two agents are, again, far enough from the center of the road, yet they are on the wrong side of the road, respectively. However, it is inefficient to force both cars to steer right, as road rules would require, since no conflicting motion is taking place.

Now, consider the scenario of Fig. 2, showing a hardware implementation with two small robots (GRITSbots), the first utilizing an autonomous driving algorithm (AV) and the second controlled remotely via a joystick by a human (HV): a further design goal is to make the AV *aware* of the HV driving direction bias, e.g., if HV steers to the left, so should AV, even if in contrasts to local traffic rules.

In this work we discuss a novel strategy of embedding traffic rules in navigation

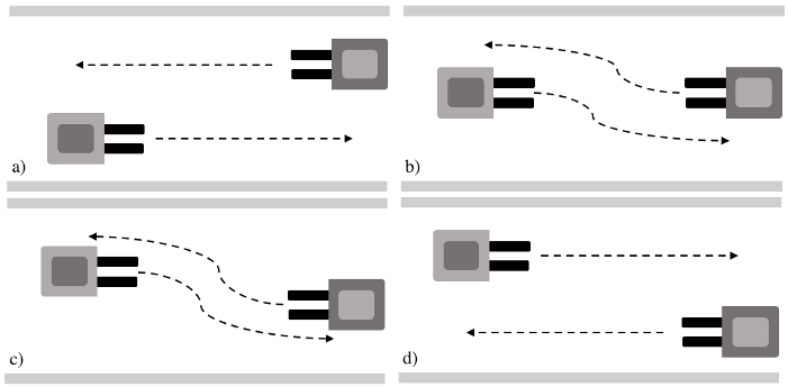

Figure 1: Head-on scenarios, defined as two agents driving in opposite directions down a narrow corridor. Four different head-on scenarios (right-hand driving). From a) to c) the cars overtake as the road rules impose; in d) the misalignment is strong enough for them to overtake regardless of the rules.

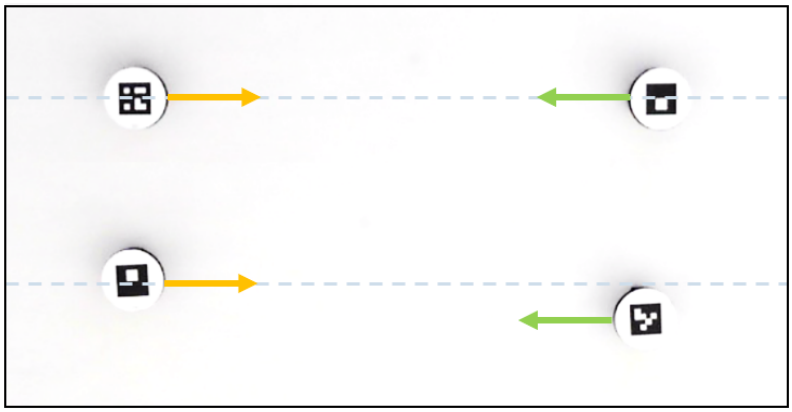

Figure 2: Snapshot of two different head-on alignments for hardware experiments.

algorithms by i) making use of Barrier Functions to ensure collision avoidance, expanding the work in Wang et al. [11], and ii) by relaxing the rules, allowing agents to break them, under defined circumstances, in order to improve the system's performance and human user's experience.

2.2 Background

2.2.1 Notation

Throughout, $\mathbb{R}$, $\mathbb{R}_0^+$ denote the set of real and non–negative real numbers, respectively. Int($\mathscr{C}$) and $\partial\mathscr{C}$ denote the interior and boundary of set $\mathscr{C}$, respectively. The open ball in $\mathbb{R}^n$ with radius $\epsilon \in \mathbb{R}^+$ and center at x_0 is denoted by $B_\epsilon(x_0) = \{x \in \mathbb{R}^n \mid \|x - x_0\| < \epsilon\}$. A continuous function $\alpha : [0, a) \to [0, \infty)$ for some $a > 0$ is said to be a class-$\mathscr{K}$ function if it is strictly increasing and $\alpha(0) = 0$. A continuous function $\beta : (-b, a) \to \mathbb{R}$ for some $a, b > 0$ is said to be an extended class-$\mathscr{K}$ function if it is strictly increasing and $\beta(0) = 0$. Given two vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$, we define the difference $\Delta\mathbf{x}_{ij}$ as $\Delta\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. Given

$f(x)$, $g(x)$ sufficiently smooth in a domain $D \subset \mathbb{R}^n$, we indicate the Lie Derivative as $L_f g(x) = \frac{\partial g}{\partial x} f(x)$.

2.2.2 Barrier Functions

A first step towards achieving the goal of securely deconflicting motion paths of vehicles, is to ensure that every action that the autonomous agents take is provably *safe* under appropriate assumptions, i.e., that agent-to-agent collisions will be avoided. To this end, we are going to use Safety Barrier Certificates (SBC) [11], based on Zeroing Control Barrier Functions (ZCBF) [12]. As the explicit purpose of this paper is to understand deconfliction in a formal manner, we are here focusing our attention on an idealized situation from an information-exchange vantage point. Throughout the text, we thus make the assumption that relevant information about nearby agents are made available, such as each agent's control input and goal position. Note that these types of information can be obtained through other means, e.g., through sensory data. For the sake of clarity, we omit this aspect and instead focus directly on the motion deconfliction problem.

Here, the fundamentals of ZCBF are briefly recalled: consider a dynamical system in control affine form,

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where the state $x \in \mathbb{R}^n$ and control $u \in U \subset \mathbb{R}^n$, f and g are locally Lipschitz continuous, and the system is forward complete, i.e., $x(t)$ is defined for all $t \geq 0$. By defining a set of conditions on the states of a system, for example imposing a minimal distance between agents, SBC ensure that the time evolution of the system is such that the states always satisfy the original conditions. In this application, the system's states will be the position and velocity of each agent. Therefore, a safe state will be a combination of position and velocity that will not result in a collision, given an acceleration command as input u.

Now, let $\mathscr{C} \subset \mathbb{R}^n$ be the safe set, i.e., the states from which it is possible to avoid collisions. In order to guarantee that a controller u is safe, we need to prove that such a controller renders x forward invariant, i.e., if $x(0) \in \mathscr{C}$, then $x(t) \in \mathscr{C}$ for all $t \geq 0$. We can encode $\mathscr{C}$ through the super-level set of a ZCBF candidate function $h : \mathscr{D} \to \mathbb{R}$, with $\mathscr{C} \subseteq \mathscr{D} \subset \mathbb{R}^n$

$$\mathscr{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}, \tag{2}$$

which means that $h(x)$, a function of the states in (1), is non-negative if the state x is safe, and negative otherwise.

By differentiating $h(x)$ with respect to time t (note that the state x is time-dependent, however, to simplify notation we denote $x(t)$ simply as x) and substituting it in (1), we gather

$$\frac{dh(x)}{dt} = L_f h(x) + L_g h(x) u. \tag{3}$$

The function $h(x)$ is said to be a ZCBF if there exists an extended class-$\mathscr{K}$ function κ such that

$$\sup_{u \in U} \{L_f h(x) + L_g h(x) u + \kappa(h(x))\} \geq 0 \tag{4}$$

for all $x \in \mathscr{D}$. Given a ZCBF, we can define the admissible control space, as a function of the states, as

$$S(x) = \{u \in U \mid L_f h(x) + L_g h(x) u + \kappa(h(x)) \geq 0\}, \tag{5}$$

with $x \in \mathscr{D}$. We now have all the necessary ingredients to state the following key results, e.g. found in Xu et al. [13].

Theorem 1 *Given a set $\mathscr{C} \subset \mathbb{R}^n$ defined by (2) and a ZCBF h defined on $\mathscr{D}$, with $\mathscr{C} \subseteq \mathscr{D} \subset \mathbb{R}^n$, any Lipschitz continuous controller $u : \mathscr{D} \to \mathbb{R}$ such that $u \in S(x)$ for the system in (1) renders the set $\mathscr{C}$ forward invariant. $\mathscr{C}$ is asymptotically stable in $\mathscr{D}$.*

In this work, consistent with Wang et al. [11], the particular choice of $\kappa(h(x)) = \gamma h^3(x)$, with $\gamma > 0$, will be adopted, which means that the controller needs to satisfy

$$L_f h(x) + L_g h(x) u + \gamma h^3(x) \geq 0 \tag{6}$$

to render the set $\mathscr{C}$ forward invariant. Controlling the system in (1) with a controller $u \in S(x)$ and defining conditions on what renders a state x safe, encoded in $\mathscr{C}$, we can ensure that, indeed, if $x(0) \in \mathscr{C}$, then $x(t) \in \mathscr{C}$, for all $t > 0$.

2.2.3 System Model

Now that we have a general formulation of the Barrier Certificates, we can formulate them in the particular context of autonomous driving. For this purpose, consider N mobile agents moving on the plane, where each agent is indexed by $\mathscr{N} = \{i \mid i = 1, 2, ..., N\}$. We model the agents' dynamics as double integrators, as acceleration limitations play a significant role when avoiding collisions

$$\begin{bmatrix} \dot{\mathbf{p}}_i \\ \dot{\mathbf{v}}_i \end{bmatrix} = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{v}_i \end{bmatrix} + \begin{bmatrix} 0_{2\times2} \\ I_{2\times2} \end{bmatrix} \mathbf{u}_i, \tag{7}$$

where $\mathbf{p}_i = (x_i, y_i) \in \mathbb{R}^2$, $\mathbf{v}_i \in \mathbb{R}^2$, and $\mathbf{u}_i \in \mathbb{R}^2$ represent the positions, velocities, and inputs (acceleration commands) for agent i. Velocity and acceleration of the agent are limited by $\|\mathbf{v}_i\|_\infty \leq \beta_i$ and $\|\mathbf{u}_i\|_\infty \leq \alpha_i$, with $\alpha_i, \beta_i \in \mathbb{R}^+$.

The aggregate state of all N agent's positions and velocities will be denoted as $(\mathbf{p}^T, \mathbf{v}^T)^T \in \mathbb{R}^{4N}$. Since the ZCBF $h(x)$ in (2) is a function of the aggregate states, we want to express it as a function of each agents' position and velocity for the system in (7), i.e., $h_{ij}(\mathbf{p}, \mathbf{v})$. Considering the interaction between two agents, i and j, we can define the pairwise set $\mathscr{C}_{ij}$ as

$$\mathscr{C}_{ij} = \{(\mathbf{p}_i, \mathbf{v}_i) \in \mathbb{R}^4 \mid h_{ij}(\Delta\mathbf{p}_{ij}, \Delta\mathbf{v}_{ij}) \geq 0\}, \quad \forall j \in \mathscr{N}_i. \tag{8}$$

As shown in [11], it is possible to express the safety barrier constraint for system (7) as a linear constraint in $\mathbf{u}_i$, which can be represented as

$$A_{ij}\mathbf{u}_i \leq b_{ij}. \tag{9}$$

In particular, given the safety distance $D_S \in \mathbb{R}^+$, the safety barrier constraint satisfying (6) is, as proven in [11],

$$\begin{aligned} -\Delta\mathbf{p}_{ij}^T \Delta\mathbf{u}_{ij} \leq \gamma h_{ij}^3 \left\|\Delta\mathbf{p}_{ij}\right\| - \frac{(\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij})^2}{\left\|\Delta\mathbf{p}_{ij}^2\right\|} + \\ + \left\|\Delta\mathbf{v}_{ij}\right\|^2 + \frac{(\alpha_i + \alpha_j)\Delta\mathbf{v}_{ij}^T \Delta\mathbf{p}_{ij}}{\sqrt{2(\alpha_i + \alpha_j)(\left\|\Delta\mathbf{p}_{ij}\right\| - D_s)}}, \end{aligned} \tag{10}$$

for all $i \neq j$. We can write $A_{ij}\mathbf{u}_i \leq b_{ij}$ as

$$A_{ij} = [0, ..., -\Delta\mathbf{p}_{ij}^T, ..., \Delta\mathbf{p}_{ij}^T, ..., 0] \tag{11}$$

and

$$b_{ij} = \gamma h_{ij}^3 \left\|\Delta \mathbf{p}_{ij}\right\| - \frac{(\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij})^2}{\left\|\Delta \mathbf{p}_{ij}^2\right\|} + \left\|\Delta \mathbf{v}_{ij}\right\|^2 + \\ + \frac{(\alpha_i + \alpha_j)\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij}}{\sqrt{2(\alpha_i + \alpha_j)(\left\|\Delta \mathbf{p}_{ij}\right\| - D_s)}}, \tag{12}$$

where $A_{ij} \in \mathbb{R}^{N-1\times 2}$, and $b_{ij} \in \mathbb{R}$. Therefore, for any $\mathbf{u}_i$ satisfying the inequality in (9), we ensure that the control input is safe, that is, the acceleration inputs will keep the state of the system in (1) such that the relative velocity and position of any two agents will not result in a collision.

In the following, we will refer to $\hat{\mathbf{u}}_i$ as the nominal controller, i.e. how we would like to control our system (or agent i). With $\mathbf{u}_i^*$ we will indicate an input to the system that is safe, i.e. that will not result in a collision. Given a nominal controller $\hat{\mathbf{u}}_i$ for the system in (1), if the condition in (9) holds for $\hat{\mathbf{u}}_i$, the nominal control input is safe, therefore making $\mathbf{u}_i^* = \hat{\mathbf{u}}_i$ will pose no harm to the system. In the following section we are going to provide a tool to keep the control $\mathbf{u}_i^*$ safe when the linear inequality in (9) is not satisfied by the nominal controller $\hat{\mathbf{u}}_i$.

2.2.4 Decentralized Safety Barrier Certificates

The time evolution of the system in (1) will be regulated by the nominal controller $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1^T, ..., \hat{\mathbf{u}}_N^T)^T$; as collisions approach, we wish for the actual control input $\mathbf{u}_i^*$ to be safe by respecting the inequality $A_{ij}\mathbf{u}_i^* \leq b_{ij}$ for all $j \neq i$, while staying as close as possible to $\hat{\mathbf{u}}_i$. Since we are able to express the safety constraint as a linear formulation of $\mathbf{u}_i$, we can add a quadratic cost that penalizes deviations from the nominal controller, while ensuring safety, resulting in a Quadratic Programming problem (QP). The decentralized version of the QP-based controller, as suggested in [11], is, for all $i \in \mathcal{N}$,

$$\begin{aligned} \mathbf{u}_i^* = \underset{\mathbf{u}_i \in \mathbb{R}^2}{\operatorname{argmin}} \quad & \|\mathbf{u}_i - \hat{\mathbf{u}}_i\|^2 \\ \text{subject to} \quad & \bar{A}_{ij}\mathbf{u}_i \leq \bar{b}_{ij}, \quad \forall j \in \mathcal{N}_i \\ & \|\mathbf{u}_i\|_\infty \leq \alpha_i \end{aligned} \tag{13}$$

where $\bar{A}_{ij} = -\Delta \mathbf{p}_{ij}^T$ and $\bar{b}_{ij} = \frac{\alpha_i}{\alpha_i + \alpha_j} b_{ij}$.

$\mathcal{N}_i$ is the neighboring set of agent i, defined as

$$\mathcal{N}_i = \{j \in \mathcal{N} \mid \left\|\Delta \mathbf{p}_{ij}\right\| \leq D_{\mathcal{N}}^i,\ j \neq i\}, \tag{14}$$

and where

$$D_{\mathcal{N}}^i = D_s + \frac{1}{2(\alpha_i + \alpha_{\min})}\left(\sqrt{\frac{2(\alpha_i + \alpha_{\max})}{\gamma}} + \beta_i + \beta_{\max}\right)^2 \tag{15}$$

is the size of the radius of the neighbors associated with $\mathcal{N}_i$, where

$$\alpha_{\min} = \min_{j \in \mathcal{N}}\{\alpha_j\} \tag{16}$$

$$\alpha_{\max} = \max_{j \in \mathcal{N}}\{\alpha_j\} \tag{17}$$

$$\beta_{\max} = \max_{j \in \mathcal{N}}\{\beta_j\} \tag{18}$$

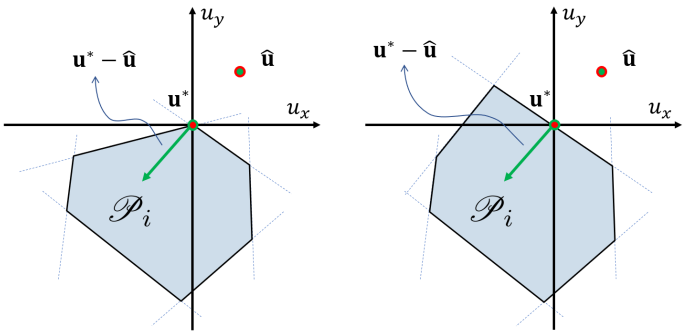

Figure 3: Graphical representations of the QP problem in (13), with a Type 1 deadlock (left) and Type 2 deadlock (right).

are the lower and upper bounds of all agents' acceleration limits and the upper bound of all agents' velocity limits, respectively. Using a controller $\mathbf{u}_i^*$ as the one defined in (13) and $\mathbf{u}^* = ({\mathbf{u}_1^*}^T, ..., {\mathbf{u}_N^*}^T)^T$, ensures safety of the system since collisions are always avoided.

2.2.5 Deadlock Detection

The QP problem in (13) ensures safety of the system in (1) regardless of the control input $\hat{\mathbf{u}}_i$ generated by the nominal high-level controller. Although safety is guaranteed, there are situations where the constraints imposed on the control input by (9) are such that the solution to (13) drives the acceleration and velocity of an agent to zero. This prevents the fulfillment of the original goal, if $\hat{\mathbf{u}}_i \neq \mathbf{0}$, and it depends on the geometry of the solution space and of the cost vector of (13).

Consider the admissible control space $\mathscr{P}_i$ for agent i

$$\mathscr{P}_i = \{\mathbf{u}_i \in \mathbb{R}^2 \mid \bar{A}_{ij}\mathbf{u}_i \leq \bar{b}_{ij}, \quad \forall j \in \mathscr{N}_i\}. \tag{19}$$

It is possible to evaluate the size of the feasible control space $\mathscr{P}_i$, termed *width of the feasible set* [14], with a Linear Program (LP)

$$\begin{aligned} \min_{\mathbf{u}_i \in \mathbb{R}^2, \delta_i \in \mathbb{R}} \quad & \begin{bmatrix}\mathbf{0}_{1\times 2} & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_i^T & \delta_i\end{bmatrix}^T \\ \text{subject to} \quad & \begin{bmatrix}\bar{A}_{ij} & -1\end{bmatrix}\begin{bmatrix}\mathbf{u}_i^T & \delta_i\end{bmatrix}^T \leq \bar{b}_{ij}, \quad \forall j \in \mathscr{N}_i \\ & \|\mathbf{u}_i\|_\infty \leq \alpha_i. \end{aligned} \tag{20}$$

The solution of the LP characterizes how much control margin is left for the strictest safety barrier constraint, i.e, if $\delta_i \leq 0$ then $\mathscr{P}_i$ is not empty. In other words, a negative δ_i indicates how much the b_{ij} of the strictest constraint can be translated before having $\mathscr{P}_i = \emptyset$.

Definition 2.1 *An agent i is said to be in deadlock if it does not move, that is, if the solution to (13) is $\boldsymbol{u}_i^* = 0$ and the speed is zero ($\boldsymbol{v}_i = 0$), while the nominal control command is $\|\hat{\boldsymbol{u}}_i\| \neq 0$.*

We identify three different deadlock scenarios, based on the geometry of the QP problem in (13) (see Fig. 3)

1. Type 1 deadlock: $\delta_i \leq 0$, $u_i \in \text{vertex}(\mathscr{P}_i)$;

2. Type 2 deadlock: $\delta_i \leq 0$, $u_i \in \text{edge}(\mathscr{P}_i)$;

3. Type 3 deadlock: $\delta_i > 0$.

In the following, we are going to build on the results in [11] by expanding the definition of deadlock, allowing agents to take deconflicting actions earlier in time, resulting in a faster responding disengagement. Moreover, given the particular case of interest seen in Fig. 1, we are going to provide formal and statistical results to show how this proposed method ensures collision avoidance in a pattern that matches the problem statement.

2.3 Robot–Robot Interaction

As a first step towards understanding the general motion deconfliction problem, we consider a two-agent system, $N = 2$, where the robots are forced to interact in a pattern relatable to Fig. 1. With this assumption of $N = 2$, $\mathscr{N}_i = \{j\}$, $i \neq j$ and $i, j = 1, 2$ if $\left\|\Delta \mathbf{p}_{ij}\right\| \leq D^i_{\mathscr{N}}$; $\mathscr{N}_i = \{\emptyset\}$, otherwise. This allows us to limit the deadlock types as seen in Definition 2.1: since we will never have more than one inequality constraint in the admissible control space in $\mathscr{N}_i$, for all $i \in \mathscr{N}$ the feasible set will never be empty. For this reason, for any solution, it holds that $\mathbf{u}^*_i \in \text{edge}(\mathscr{P}_i)$ and hence Type 1 and Type 3 deadlock cannot occur.

Definition 2.2 *Given $\underline{\alpha}$, $\underline{\beta} \in \mathbb{R}^+$ as the acceleration and velocity thresholds, respectively, and $\underline{\epsilon} \in \mathbb{R}^+$ as the control threshold, an agent i is said to be in a quasi-deadlock if $\|\boldsymbol{u}_i\| \leq \underline{\alpha}$, $\|\boldsymbol{v}_i\| \leq \underline{\beta}$, and the nominal control $\|\hat{\boldsymbol{u}}_i\| > \underline{\epsilon}$.*

By introducing a lower bound on the acceleration and velocity, we wish to identify those agents that are slowing down, while the difference between the nominal and the actual controller, $\underline{\epsilon}$, ensures that this slowing evolution is due to the safety constraints introduced in (9) and not by the actual control goal, and hence a risk of collision is approaching.

2.3.1 Quasi-deadlock Resolution

If an agent enters a deadlock it stops; [11] presents a deadlock resolution tool, but this requires the agent to have both speed and acceleration equal to zero. This results in a slow-reacting system as, before taking any deconfliciting motion, any agent must come to a complete stop. The quasi-deadlock aims at detecting when an agent is about to enter a deadlock, allowing the controller to take preventive actions without the need for either robot agent to come to a complete halt.

The conflict resolution tool proposed in the present paper as an improvement over [11] can be summarized as follows: given the nominal control $\hat{\mathbf{u}}_i$, if agent i finds itself to be in quasi-deadlock, perturb $\hat{\mathbf{u}}_i$ such that the new nominal control is $\Gamma_i \hat{\mathbf{u}}_i$ where $\Gamma_i = I + k_{\gamma i} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and $k_{\gamma i} \in \mathbb{R}$ for all $i \in \mathscr{N}$. This resolution tool is graphically explained in Fig. 4. $\Gamma \hat{\mathbf{u}}_i$ is a nominal perturbation to the left, or to the right, of $\hat{\mathbf{u}}_i$, depending on the sign of $k_{\gamma i}$.

This translates into $k_{\gamma i}$ encoding the concept of left-hand and right-hand driving, based on its sign. In fact, if $k_{\gamma i} > 0$ ($k_{\gamma i} < 0$) the control input is perturbed to the left (right): when an agent i slows down due to the constraint imposed by the presence of another agent j, agent i will reshape its control input, steering slightly to the left (or right), based on $\text{sign}\{k_{\gamma i}\}$. Moreover, according to the module of $k_{\gamma i}$, the perturbation will be more or less aggressive, that is,

$$k^{(1)}_{\gamma i} > k^{(2)}_{\gamma i} \rightarrow \left\|\hat{\mathbf{u}}_i - \Gamma^{(1)}_i \hat{\mathbf{u}}_i\right\| \geq \left\|\hat{\mathbf{u}}_i - \Gamma^{(2)}_i \hat{\mathbf{u}}_i\right\|. \tag{21}$$

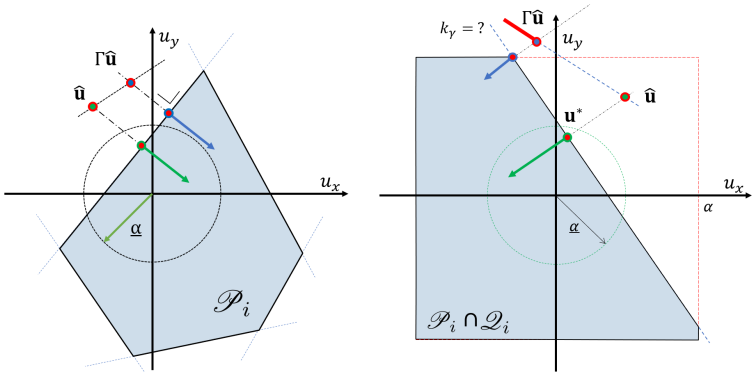

Figure 4: Graphical representation of $\mathscr{P}_i$, ($N_i = 5$) with quasi-deadlock resolution. On the left, $\hat{\mathbf{u}}$ is projected, resulting in $\Gamma\hat{\mathbf{u}}$; consequently the optimal solution $\mathbf{u}^*$ changes. On the right, k_γ can be computed precisely as long as the optimal solution to $\Gamma\hat{\mathbf{u}}$ is $\in$ edge($\mathscr{P}_i \cap \mathscr{Q}_i$).

For this reason, we will also refer to $k_{\gamma i}$ as agent i's *direction bias*. The quasi-deadlock resolution is summarized in Algorithm 1, where the subroutine *Decentralized_LP* computes the width of the feasible set as in (20) and *Decentralized_QP* computes the optimal safe controller as in (13).

Algorithm 1 Quasi-deadlock resolution

input $\mathbf{u}_i, \hat{\mathbf{u}}_i, \mathbf{v}_i$
$\delta \leftarrow$ Decentralized_LP
if $\|\mathbf{u}_i\| \leq \underline{\alpha}$ & $\|\mathbf{v}_i\| \leq \underline{\beta}$ & $\|\hat{\mathbf{u}}_i\| > \underline{\epsilon}$ & $\delta \leq 0$ **then**
  $\hat{\mathbf{u}}_i \leftarrow \Gamma_i \hat{\mathbf{u}}_i$
$\mathbf{u}_i^* \leftarrow$ Decentralized_QP($\hat{\mathbf{u}}_i$)
return $\mathbf{u}_i^*$

2.3.2 Direction Bias Estimation

Assume that every agent's goal is shared on the network, therefore $\hat{\mathbf{u}}_j$ is known by every agent i, while $k_{\gamma j}$ is not. We also assume that the final optimal control $\mathbf{u}_j^*$ is known by every agent. As a first approach, we are going to ignore limits on the acceleration input, i.e. $\alpha_i = +\infty$. The problem to be faced is, can agent i learn $k_{\gamma j}$ from observing j's behavior?

Proposition 1 *Consider the QP problem in (13), where $\alpha_i = +\infty$, $\forall\, j \in \mathscr{N}$, with the quasi-deadlock resolution defined in Algorithm 1. If the nominal controller $\hat{\mathbf{u}}_j$, and the solution to the QP problem $\mathbf{u}_j^*$ are known for all $j \in \mathscr{N}_i$, then agent i can compute $k_{\gamma j}$.*

Proof: Let us again consider the simple scenario of two agents driving toward each other (position swapping) as described in the previous sections. Since we are solving the problem in (13), a QP problem in $\mathbb{R}^2$, we know that the solution $\mathbf{u}_j^*$ will solve all inequality constraints and at least one of the inequalities will actually be solved as an equality [15]. Using the assumption that only two agents are present in the network, we can conclude

that the matrix A_{ji} will always have one row, and therefore the solution will always be along the line $\bar{A}_{ji}\mathbf{u}_j^* = \bar{b}_j$, where $\bar{A}_{ji} \in \mathbb{R}^{1\times 2}$ and $\bar{b}_j \in \mathbb{R}$. We can express the cost function in (13) as

$$\|\mathbf{u} - \Gamma\hat{\mathbf{u}}\|^2 = \mathbf{u}^T\mathbf{u} + \hat{\mathbf{u}}^T\Gamma^T\Gamma\hat{\mathbf{u}} - 2\mathbf{u}^T\Gamma\hat{\mathbf{u}}, \tag{22}$$

where, to ease up notation, we define $\hat{\mathbf{u}}_j = \hat{\mathbf{u}} = [\hat{u}_x, \hat{u}_y]^T$ and $\mathbf{u}_j = \mathbf{u} = [u_x, u_y]^T$. Using the Lagrangian multiplier $\lambda \in \mathbb{R}$, we can express the equality

$$H = \mathbf{u}^T\mathbf{u} + \hat{\mathbf{u}}^T\Gamma^T\Gamma\hat{\mathbf{u}} - 2\mathbf{u}^T\Gamma\hat{\mathbf{u}} + \lambda(A\mathbf{u} - b) \tag{23}$$

and, differentiating H along $\mathbf{u}$ we get

$$\frac{\partial H}{\partial \mathbf{u}} = 2\mathbf{u} - 2\Gamma\hat{\mathbf{u}} + \lambda A^T = 0. \tag{24}$$

Defining $\bar{A}_{ji} = A = [a_1, a_2]$ and $\bar{b}_j = b$, and recalling that $A\mathbf{u} = b$, we obtain $\mathbf{u} = \Gamma\hat{\mathbf{u}} - \frac{1}{2}\lambda A^T$ from the previous equation and we get

$$\lambda = 2(AA^T)^{-1}(A\Gamma\hat{\mathbf{u}} - b). \tag{25}$$

Finally, given that $\mathbf{u} = \Gamma\hat{\mathbf{u}} - \frac{1}{2}\lambda A^T$ we conclude that

$$\mathbf{u} = \Gamma\hat{\mathbf{u}} - (AA^T)^{-1}(A\Gamma\hat{\mathbf{u}} - b)A^T \tag{26}$$

which is linear with respect to the free parameter Γ. We define $(AA^T)^{-1} = \gamma$ and note that $\gamma \in \mathbb{R}$, $\forall A \in \mathbb{R}^{1\times N}$ and $AA^T \neq 0$, since $A \neq \mathbf{0}$ being it a distance (see (11)); moreover $\|\hat{\mathbf{u}}_i\| > 0$ from the definition of quasi-deadlock (Def. 2.2). Equation (26) is a two-equation system with one unknown parameter, k_γ; solving for $\mathbf{u}_x$, we obtain:

$$k_\gamma^{\text{est}} = \frac{\hat{u}_x - u_x - \gamma a_1(a_1\hat{u}_x + a_2\hat{u}_y - b)}{\hat{u}_y + \gamma a_1(a_2\hat{u}_x - a_1\hat{u}_y)} \tag{27}$$

What this means is that, for a two-agent system, if we know the objective controller $\hat{\mathbf{u}}$ and the optimal safe controller $\mathbf{u}^*$ for every agent, it is possible to obtain k_γ of the agents once they enter a quasi-deadlock scenario, i.e., $\Gamma \neq 1$, since if $\Gamma = 1 \rightarrow k_\gamma = 0 \rightarrow \|\mathbf{u} - \Gamma\hat{\mathbf{u}}\|^2 = \|\mathbf{u} - \hat{\mathbf{u}}\|^2$. □

In Proposition 1 we did not take into consideration an important constraint of the QP problem in (13): the acceleration limits $\left\|\mathbf{u}_j\right\|_\infty \leq \alpha_j$. These limits introduce a saturation in the system in the form of inequality constraints, limiting the admissible control space $\mathscr{P}_j \cap \mathscr{Q}_i$, where $\mathscr{Q}_i = \{\mathbf{u}_i \in \mathbb{R}^2 \mid \|\mathbf{u}_i\|_\infty \leq \alpha_i\}$. This limits the $k_{\gamma j}$ estimation tool we developed.

Proposition 2 *Consider the QP problem in (13), with the quasi-deadlock resolution defined in Algorithm 1. If the nominal controller $\hat{\mathbf{u}}_j$, and the solution to the QP problem $\mathbf{u}_j^*$ are known for all $j \in \mathscr{N}_i$, then agent i can compute a lower bound on $k_{\gamma j}$, where the solution to (27) is such that $\left\|k_{\gamma j}^{est}\right\| \leq \left\|k_{\gamma j}\right\|$.*

Proof: As described in Fig. 4b, if the projection $\Gamma\hat{\mathbf{u}}_j$ falls along the half-line on the right of $\Gamma\hat{\mathbf{u}}_j$, the new optimal solution $\mathbf{u}_j^*$ will fall on the vertex of $\mathscr{P}_j \cap \mathscr{Q}_i$, regardless of the actual $k_{\gamma j}$. However, the sign of $k_{\gamma j}$ will still be computed correctly and the resulting $|k_{\gamma j}|$, although wrong, will provide a lower bound on the actual value of $k_{\gamma j}$. The existence of $\mathbf{u}_j^*$ is guaranteed by the definition of Type 2 quasi-deadlock. □

In this section we propose a simple mathematical model to encode rule-based quasi-deadlock resolution, thanks to the direction bias k_γ: left vs right handed, sign$\{k_\gamma\}$, smoother vs rougher, abs$\{k_\gamma\}$. We then provide a way, for each agent i to compute the driving direction bias k_j for $j \in \mathscr{N}_i$: this allows each agent to gain knowledge of the driving behavior of the other nearby.

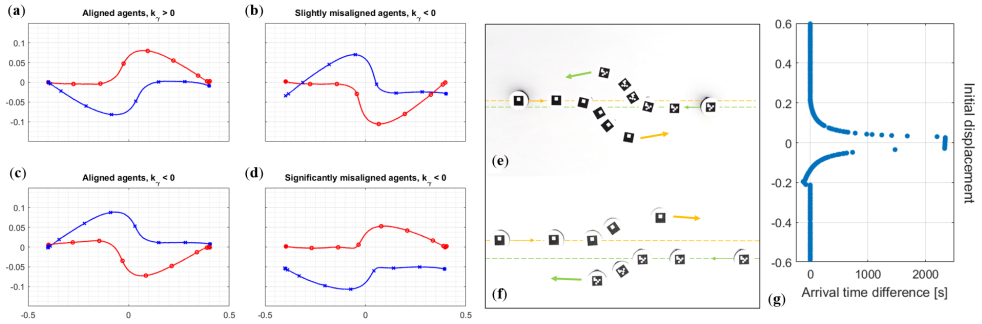

Figure 5: Experimental results for different head on resolutions. Figs. (c)–(e), and (d)–(f) show the same experiments, respectively, proposed as a series of consecutive snapshots from the Robotarium. In (g) the arrival time comparison (original formulation - quasi-deadlock resolution) of two agents heading towards each other, with a simulation time limit of 60 seconds is shown. The initial displacement between the two agents is randomly selected, for a total of 500 simulations.

2.3.3 Experimental Results

The Decentralized SBC with quasi-deadlock detection and resolution is here implemented and tested on the Robotarium at the Georgia Institute of Technology [16], a remotely accessible swarm-robotics testbed. In particular, two AVs are controlled with the goal of swapping positions on the plane; four series of experiments are proposed, in close relation to the problem statement of Fig. 1. The results are shown in Fig. 5.

In Figs. 5a and 5b the two agents are perfectly aligned, both implementing, respectively, right ($k_\gamma < 0$) and left ($k_\gamma > 0$) handed driving bias. In Fig. 5c, the two agents are slightly misaligned to their left on the vertical axis (both agents have $k_\gamma < 0$); however, since the misalignment is not significant, both agents steer to the right, as the rules of the road require. In the experiment of Fig. 5d, instead, the misalignment is significant, as in Fig. 1d, resulting in both agents holding their side of the road. Figs. 5e and 5f present these two experiments as a series of snapshots, to better visualize the agents' behaviour.

As a further performance test, simulations are performed in order to compare the behaviour of the Decentralized SBC with quasi-deadlock resolution (QD), as presented in this paper, with their original formulation, as presented in [17] (NQD). In Fig. 5g the results are shown for different misalignment conditions (ordinate); the simulations are performed with the same starting conditions with both formulations and the difference ($NQD_{\text{time}} - QD_{\text{time}}$) in the time arrival of the agents is shown (abscissa). Positive values of the time difference indicate a faster goal achievement for the QD resolution; a limit on the simulation time is fixed at 60 seconds. As the results suggest, the QD resolution outperforms the NQD around the point of interest (perfect alignment). We observe that there is an area where the NQD is faster; however, we argue that overall this new resolution is faster and it never encounters a new deadlock (in the case of QD, all goals are met before the time limit).

A video of the experiments is available online at https://goo.gl/ctzmM3.

2.4 Human–Robot Interaction

Consider now the case of a mixed two-agent network, where the first agent is autonomous and implements Decentralized SBC with quasi-deadlock resolution as described in Section 2.3, and the second agent is remotely controlled by a human driver. The human

driver controls the robot agent via a joystick or keyboard and is able to set the acceleration input at any instant; the human is asked to drive the robot *as straight as possible* to a specific goal point in the $x\,y$ plane, resulting in a head-on scenario like the one described in Fig. 1b. We make the assumption that the human driver is not ill-intentioned, i.e., he or she will not try to deliberately hurt the system, for example by avoiding to take any deconflicting action when a collision is imminent. Instead, the focus is on establishing driving biases as a way of deconlicting motions under relatively benign driving conditions.

We note that a human operator will not necessarily drive enforcing the notion of SBC as discussed in the previous sections of this paper. Safety of the systems in (13) can be guaranteed only when all agents respect the conditions of the problem. In the experiments presented in this section, we will assume that the human drives without the intent of harming the system, avoiding collision by steering (changing direction) when she or he considers it to be necessary. Even though we have no reason to assume that human drivers employ SBC, the autonomous vehicle will act as if the human was actually using SBC. As will be seen in the subsequent section, this assumption does indeed provide insight into a human driver's behavior when related to her or his driving direction bias.

Assume that the AV has perfect knowledge of the HV input controller and of the HV control goal, e.g., a position in the $x\,y$ plane. Let $\mathbf{u}_i$ be the control input for HV, i.e. the control signal imposed from the joystick. HV implements a modified version of SBC, i.e., it computes the inequality constraints for the problem in (13) as if in the autonomous case, without however applying the computed optimal control, since it is directly imposed by the outside user. The AV can observe the behavior of HV and has all the necessary tools to mimic its QP problem, as seen in Section 2.3 for the fully autonomous case.

Let $\mathbf{u}_{0i}(t)$ be the control input expected from agent i at any instant t and, dropping the notion of time for ease of notation, let $\mathbf{u}_{0i}^{\perp}$ and $\mathbf{u}_i^{\perp}$ be the projection of $\mathbf{u}_{0i}$ and $\mathbf{u}_i$, respectively, on the equality constraint $A_{ij}\mathbf{u}_i = b_i$, where $A_{ij} \in \mathbb{R}^{1\times 2}$ and $b \in \mathbb{R}$. Let $B_\epsilon(\mathbf{u}_{0i}^{\perp})$ be the ball of radius ϵ centered at $\mathbf{u}_{0i}^{\perp}$ and let $\mathscr{L}_i = \{\mathbf{u}_i \mid \mathbf{u}_i^{\perp} \in B_\epsilon(\mathbf{u}_{0i}^{\perp})\}$. This is exlained graphically in Fig. 6.

Definition 2.3 *A user's control input $\boldsymbol{u}_i$ is said to be goal compatible with the control goal $\boldsymbol{u}_{0i}$ if $\boldsymbol{u}_i \in \mathscr{L}_i$.*

If the autonomous agent finds that HV is driving with goal compatible inputs, no further actions will be taken, as AV considers that the human driver is trying to achieve her or his original goal, i.e., it is accelerating or decelerating in the direction of the goal. On the other hand, if $\mathbf{u}_i \notin \mathscr{L}_i$, AV considers HV to be perturbing its control away from the original goal, e.g., to avoid a collision, as seen in Fig. 6.

With this new tool we can construct the human–robot interaction problem as the robot–robot problem of Section 2.3, where $\mathbf{u}_0^{\perp} = \mathbf{u}^*$ and $\mathbf{u}^{\perp} = \mathbf{u}_Q^*$, where $\mathbf{u}_Q^*$ is the optimal solution associated to $\Gamma\hat{\mathbf{u}}$ of Algorithm 1.

2.4.1 Experimental Results

A limited number of experiments is performed (more than 10) to validate the illustrated results. In particular, the same human subject is presented with various head on scenarios, each time with a different initial value of the autonomous agent k_γ; the human subject was asked to deconflict the agent according to personal preference and the reaction of the autonomous agent and the $k_{\gamma H}$ estimation was analyzed. No significant difference in the behaviour of the autonomous agent was observed.

Experimental results for one of such experiments are shown here in Fig. 7 and 8. In particular, the AV agent is programmed with $k_{\gamma A} > 0$ (left-hand driving) and the HV is allowed to steer to the left or to the right, as controlled by a human driver via a joystick. The autonomous agent ignores whether the upcoming agent is autonomous (and using

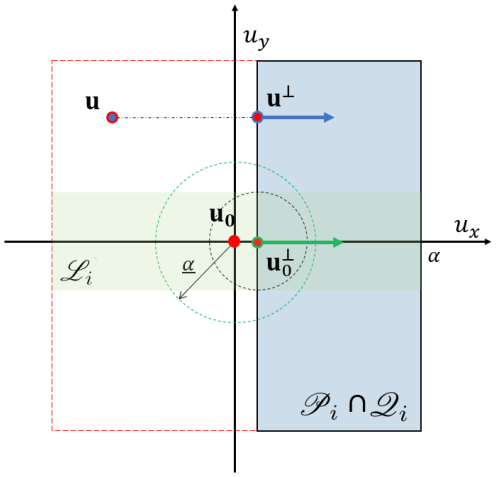

Figure 6: Graphical representation of $\mathscr{P}_i$ of HV as seen from AV in a simple two agents network. In this example, $\mathbf{u} \notin \mathscr{L}_i$, therefore the control $\mathbf{u}$ is considered to be not goal compatible, revealing a right-hand biased for HV.

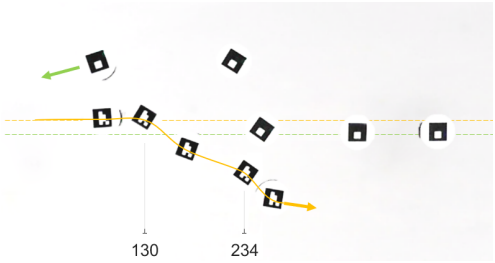

Figure 7: Head-on scenario with mixed human-robot interaction. The AV starts with a left driving bias, but updates it based on HV's behavior. Snapshot evolution of head-on interaction between HV (left) and AV (right).

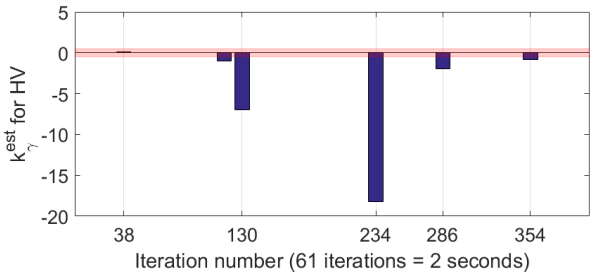

Figure 8: Head-on scenario with mixed human-robot interaction. The AV starts with a left driving bias, but updates it based on HV's behavior. k_γ of HV, estimated by AV. The shaded area highlights the actions that are considered *goal compatible* and, therefore, ignored.

SCB) or is controlled by a human. Despite the lack of this information, given the generality of the approach, the AV is able to estimate the driving direction bias and can update its internal value of $k_{\gamma H}$ accordingly.

In the experiment of Fig.and 8, the autonomous agent knows that the HV needs to reach a point in the plane along its current trajectory of motion; any acceleration input along that direction will not influence the computation of $k_{\gamma H}$. In Fig. 7 the evolution of one particular experiment performed on a pair of GRITSbots is shown and in Fig. 8 the $k_{\gamma H}$ estimation performed by the AV is plotted against time. At about 1 second from the start of this experiment (iteration 38) a *goal compatible* acceleration input is given: the user commands the agent to accelerate forward, towards the goal. Later in the experiment, just after the 4-second mark (iteration 125) the user starts steering the robot agent significantly, resulting in a well-defined sign of $k_{\gamma H}$, suggesting that the HV is, indeed, steering to the negative values of k_{γ}, i.e. to the right of the goal.

2.5 Conclusions

A tool to solve conflicting motion paths for networks of multiagent robots was presented. Focusing on the interaction of two robot vehicles heading towards a collision, we provided a tool to deconflict the agents' motion, while ensuring safety with the notion of traffic rules. Based on the agents' relative position and velocity, the agents will act according to road rules, or decide to break them if particular conditions are present. We introduced the notion of driving direction bias, as the direction (left or right) that will preferably be used by the autonomous agent to deconflict its motion, together with a tool to estimate other agents' direction bias. Finally, we investigated a strategy to adapt the model, presented for the autonomous vehicles, to a mixed human and robot interaction, where the robot vehicle morphs its driving direction bias based on the observed human behavior.

Even though there is no reason to believe that human drivers employ barrier certificates, the construction in this paper suggests that this assumption still allows for an effective deconfliction strategy between human and autonomous drivers. Moreover, we believe that these results can be further generalized, removing the need to share the agents' state on the network and exploiting tools already present in the literature to estimate the local agents' states from sensor data [18]. Indeed, the performed experiments in the Human-Robot interaction scenario are limited to just one human subject. Further work is required to present a statistically significant set of experiments that should aim at benchmarking the behaviour of this resolution tool with different human subjects and, therefore, driving behaviours.

3 Distributed coordination for industrial vehicles

3.1 Introduction

Efficiently managing groups of robots in known environments involves path computation to minimize one or multiple indexes, such as travelled distance, time of arrival or energy spent. Mutual collisions and stall situations have to be avoided to guarantee a safe and continuous flow of the fleet. Fast and reliable negotiation protocols in critical regions, such as crosses and one-way roads, are hence required in case of a large number of robots.

The problem has been tackled in the Multi–Robot Systems path planning literature with either centralized or distributed algorithms, applied on a representation of the environment either discrete or continuous. Scientific literature on the topic is very rich and an interesting survey can be found in [5].

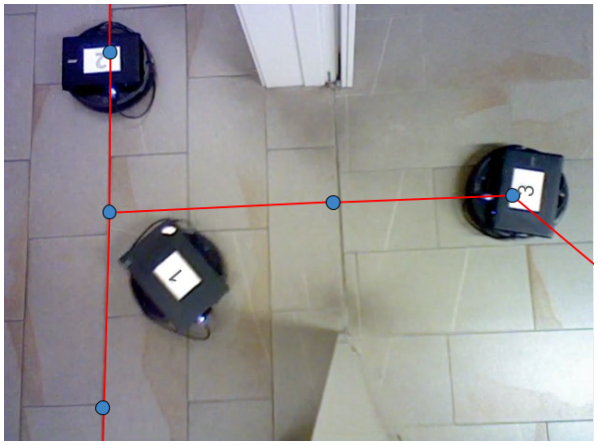

Figure 9: A screenshot of three robots negotiating an intersection

One of the most common environment representation is a discrete graph that ensures obstacle avoidance by construction, i.e. nodes and edges are considered only on the obstacle free part of the map. Examples of the generation of such graphs can be found in [19] and [20]. Within such graphs, the problem of finding the optimal path for each robot has been proved to be NP-hard in [21] and different algorithms based on heuristics have been proposed, see e.g. [22, 23, 24, 25].

Different centralized approaches use a continuous space modelization and a sample based motion planner (in [26] collisions are avoided by using special bounding boxes and RRT* [27] is used as a planner while PRM* is used in [28]).

One of the main criticality in distributed algorithms is the prevention of deadlocks that may arise during the system evolution. Examples of negotiations to avoid deadlocks can be found in [8, 29, 30, 31, 9, 10].

A preliminary distributed protocol for multi-robot coordination has been proposed in [32], in which both space and time were discrete and the collision free trajectories were planned on a Time Expanded Network. In this work a new mixed discrete-space/continuous-time distributed path planner is proposed based on a similar approach. The developed algorithm is based on a random sampling procedure in a hybrid configuration space to exploit the ability of robots in tuning the speed (in a continuous interval) for collision avoidance while a regular flow is maintained. To the authors best knowledge, this is the first online distributed planning algorithm in space-time dimension that uses an hybrid discrete–continuous configuration state to handle both robot paths and speed. Taking inspiration mainly from FTM* [33] and RRT* variants [34, 35, 36], a sample based planner is first used by each robot to compute a trajectory in the hybrid configuration space. Second, collision free trajectories are computed based on information exchanged by neighboring robots. The proposed approach is proven to provide collision free trajectories under some realistic assumptions. Finally, simulations and experimental results are reported to sustain applicability and validity of the approach.

3.2 Problem Description

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a topological graph (roadmap) representing the n dimensional workspace, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of arcs (e.g., $e_{ij} = (v_i, v_j)$ is the arc from node v_i to v_j).

The node position is defined as $q(v_i)$ i.e. for $n = 2$, $q(v_i) = (x_i, y_i)$, and each arc e_{ij}

is a path from $q(v_i)$ to $q(v_j)$. A weight ω_{ij} that represents the length of the path in $\mathbb{R}^n$ from $q(v_i)$ to $q(v_j)$ is associated to each arc. Consider a set $\mathscr{R} = \{r_1, \ldots, r_M\}$ of M mobile robots r_i, ordered by priority. Each robot r_i navigates the environment along the roadmap with maximum admissible speed $v_{\max}$. We will denote with r_i the i-th robot (in terms of priority) and with $r_i(t)$ its coordinates in $\mathbb{R}^n$. The goal of this paper is to determine a collision free trajectory on the roadmap for each robot through communication among close-by robots.

The mixed discrete-continuous space is characterized by configurations $s \in \mathscr{R}_n \times \mathbb{R}_+$ where $\mathscr{R}_n$ is the discrete subset of $\mathbb{R}^n$ that contains the discrete poses of the robots. Hence, first component is the pose of the vehicle in the environment and the second is time, i.e. for $n = 2$, $s = (x, y, t)$. In the following we denote with $q(s)$ and $\tau(s)$ the discrete coordinates in $\mathbb{R}^n$ and the continuous time t associated to s respectively. In this state-time space, configurations s are associated to the roadmap, i.e. $q(s) \in \mathscr{G}$. In the following, we will refer to this state-time space as *time-expanded roadmap* denoted with $\mathscr{G}_t$. Let D_s be the minimum required distance (*safety distance*) to be kept between robots to avoid collisions. This distance may take into account robots' physical dimensions and motion constraints.

Definition 3.1 (Robot Collision) *A* collision *between robots r_i and r_j at time t occurs if* $\|r_i(t) - r_j(t)\| \leq D_s$.

To avoid collisions between robots that move along the graph, some assumptions on the roadmap topology and robots desired positions are needed.

Assumption 1 (Roadmap Topology) *Arcs are paths in the free environment such that the distance between any two points along two different arcs is larger than D_s whenever the points are at distance larger than a given D from any node. Moreover, any two nodes of the graph must have distance larger than $2D$.*

The node is then characterized not only by its coordinates in the environment but more precisely by the disc centered in the node with radius D. Hence, we can define

Definition 3.2 (Node and Time Interval Occupancy) *A node $v \in \mathscr{V}$ of the roadmap is* occupied *by robot r in the* occupancy time interval $T = [t_1, t_2]$ *if* $\forall\, t \in T$, $\|r(t) - q(v)\| < D$.

Obviously, the occupancy time interval T depends on the path followed by the robot (whose length is the cost associated to the arc) and its speed. However, based on Assumption 1, a robot can not occupy two nodes simultaneously. In case of roadmaps in which arcs are segments between nodes, the value of D can be directly computed as follows. Let α be the minimum angle formed by arcs entering in a node. When two robots, moving along two arcs entering in a node, are at distance larger than $D = \frac{D_s}{2\sin\frac{\alpha}{2}}$ from the node, no collision may occur. Notice that Assumption 1 generalizes such condition to non rectilinear arcs. The main idea is to have a roadmap in which collisions may occur only in the node occupancy disc or along the same arc.

Assumption 2 (Roadmap always connected) *The considered roadmap is a strongly connected graph. Moreover, robots never receive a task whose position in nodes or along arcs lead to a graph connectivity loss.*

Such assumptions are not restrictive and usually verified by city maps or warehouse plans, where agents do not end their travel in the middle of a street/aisle blocking other robots.

Since collisions between robots may occur only along a single arc or at distance smaller than D from the nodes, robots will be coordinated so that any pair of robots may occupy the same node only in disjoint occupancy time intervals.

Let us now introduce the concepts of path and trajectory. First, the set of adjacent nodes of z in $\mathcal{G}$ is $\text{adj}(z) = \{v \in \mathcal{V} | (z, v) \in \mathcal{E}\}$.

Definition 3.3 (Path) *A* path $P(v_s, v_g) = \{v_s, \ldots, v_g\}$ *is a sequence of consecutive adjacent nodes of* $\mathcal{G}$ *from* v_s *to* v_g.

Definition 3.4 (Trajectory) *A* trajectory $\sigma \in \mathcal{G}_t$ *is a sequence* $\sigma = \{s_1, \ldots, s_k\}$ *of configurations* s_i *associated to nodes, i.e. such that* $q(s_i) \in \mathcal{V}$. *The sequence of nodes* $\{q(s_1), \ldots, q(s_k)\}$ *is a path from* $q(s_1)$ *to* $q(s_k)$ *and the sequence of* node traversing time *instants* $\{\tau(s_1), \ldots, \tau(s_k)\}$ *is positive and monotone.*

To each trajectory it is possible to associate a *trajectory cost* based on the problem specifications, e.g., minimum travelled distance or minimum travel time. In this paper we propose a trajectory cost that weights both travelled distance and time of travel. Given a trajectory $\sigma = \{s_1, \ldots, s_k\}$, its cost is

$$\mathcal{C}(\sigma) = h_t \sum_{i=1}^{k} \tau(s_i) + h_c \sum_{i=1}^{k-1} \omega_{i\,i+1}, \tag{28}$$

where $\omega_{i\,i+1}$ is the length of the arc $(q(s_i), q(s_{i+1}))$, h_t and h_c are weighting factors. It is worth noting that, with the proposed cost function, in case of two trajectories $\sigma_1 = \{s_1, \ldots, s_k\}$ and $\sigma_2 = \{z_1, \ldots, z_h\}$ with same travel time ($\tau(s_k) = \tau(z_h)$) and same total length, the one with smaller node traversing time is preferred. This choice is motivated by the fact that the future necessity of coordination to avoid collisions is not predictable by robots and hence it is preferable to move as fast as possible, i.e. small values for the initial node traversing time instants.

In this work, a roadmap respecting Assumption 1 has been obtained based on automatic Voronoi generator from [20]. The generated roadmap includes cycles to allow for alternative routes, and encodes traversing costs for each edge, see for example the map in Fig. 10.

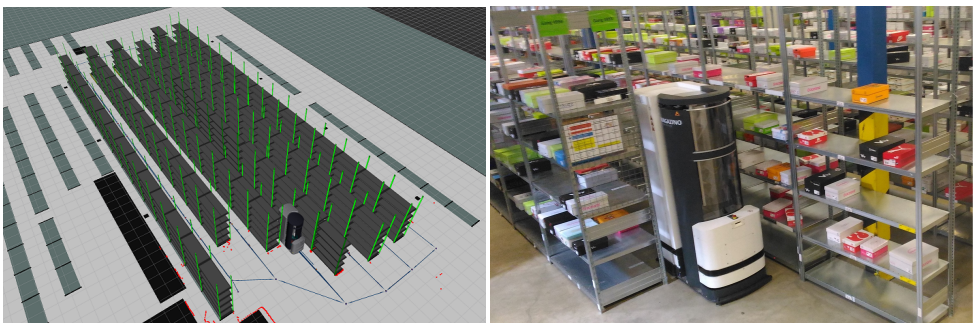

Figure 10: An example of a graph built over a planar map of a customer of Magazino GmbH and an image of the real environment

3.3 The Multi–Robot Path Planning Approach

The purpose of this paper is to provide a distributed algorithm, the *Multi-robot Fast Expanded Roadmap Tree* (in short Mr.FERT), that, taking M initial and target robots configurations, provides collision free trajectories controlling both robots speed and paths to

reduce energy consumed for speed changes. Robots communicate only with neighboring robots, i.e. robots at distance smaller than a communication range $R_c > D_s$ as described in Section 3.3.2.

Definition 3.5 (Neighboring robots) *Let* $N(p,t) = \{i | 1 \leq i \leq M, \|r_i(t) - r_p(t)\| \leq R_c\}$ *be the set of indexes of* neighbors of robots r_p, *i.e. the indexes set of robots at a distance closer than* R_c *to* r_b.

For the sake of simplicity, the time dependency of set $N(p,t)$ will be omitted in the following.

Our approach consists of alternating phases of trajectories computations and negotiations. In the *trajectory computation* phase, each robot plans its own trajectory over the time-expanded roadmap $\mathcal{G}_t$, taking into account the static obstacles. These obstacles are generated in the *negotiation* phase considering potential collisions in space-time with higher priority neighbors. Indeed, each robot r_p receives, only from neighboring robots in $N(p)$, their latest planned trajectory σ. Given the trajectory $\sigma_l = \{s_1, \ldots, s_k\}$ received from robot r_l with $l < p$, the occupancy time interval for each node associated to the received trajectories is computed as follows. For any arc $(q(s_{i-1}), q(s_i))$ in σ_l, the average speed on the arc $e_{i-1\,i}$ is $\bar{v}_{i-1\,i} = \frac{\omega_{i-1\,i}}{\tau(s_i) - \tau(s_{i-1})}$. Given average speeds along arcs, the time interval during which node $q(s_i)$ is occupied by r_l is $\left[\tau(s_i) - \frac{2D}{\bar{v}_{i-1i}}, \tau(s_i) + \frac{2D}{\bar{v}_{ii+1}}\right]$. With this approach the occupancy time interval is computed taking into account the average speeds of the robot along the arcs entering and exiting from the node $q(s_i)$. During such time interval the node $q(s_i)$ is occupied by robot r_l that has higher priority than r_p and hence it represents a static obstacle for robot r_p in $\mathcal{G}_t$. The computed time interval corresponds to a line segment in $\mathcal{G}_t$ (represented as a cylinder in Fig. 11) along the time dimension associated to node $q(s_i)$ and represents a static obstacle for robot r_p generated by σ_l.

Remark 1 *It is worth noting that, with the proposed concept of node occupancy time interval, it is also possible to handle unforeseen obstacles in the environment and detected by robots. Indeed, in this case, if the obstacle is recognized as static, an infinite occupancy time interval is associated to the node. A path can be found if the roadmap remains strongly connected also if the node is neglected.*

From the Definition 3.2 of occupied node and occupancy time interval, it follows straightforwardly the following

Proposition 3 *Let* $\Sigma_p = \cup_{i \in N(p), i<p} \sigma_i$ *be the set of trajectories of neighboring robots with higher priority than* r_p, *and let* $\sigma_p = (s_1, \ldots, s_k)$ *be the trajectory computed by* r_p *at a certain time instant. For any node* $v \in \{q(s_i) | i = 1, \ldots, k\}$ *let* T_v *be the intersection of occupancy time intervals for each* $i \in N(p)$ *and* $i \leq p$. *If* T_v *is empty for any node* v *in* σ_p, *the robot* r_p *will have no collision with robots* r_i *with* $i < p$ *when it is at a distance smaller than* D *from nodes* v *along trajectory* σ_p.

Fig. 11 shows an example of a trajectory computed to avoid collisions within space-time intervals.

3.3.1 Trajectory computation

In the trajectory computation phase, collision free trajectories for each robot along arcs of the roadmap $\mathcal{G}$ will be computed under the conditions of Proposition 3 to avoid collisions at nodes. Collisions along arcs will be managed by the negotiation phase.

The main idea of the proposed approach is that each robot r_p uses a sample based planning algorithm to determine a collision free trajectory from the currently occupied

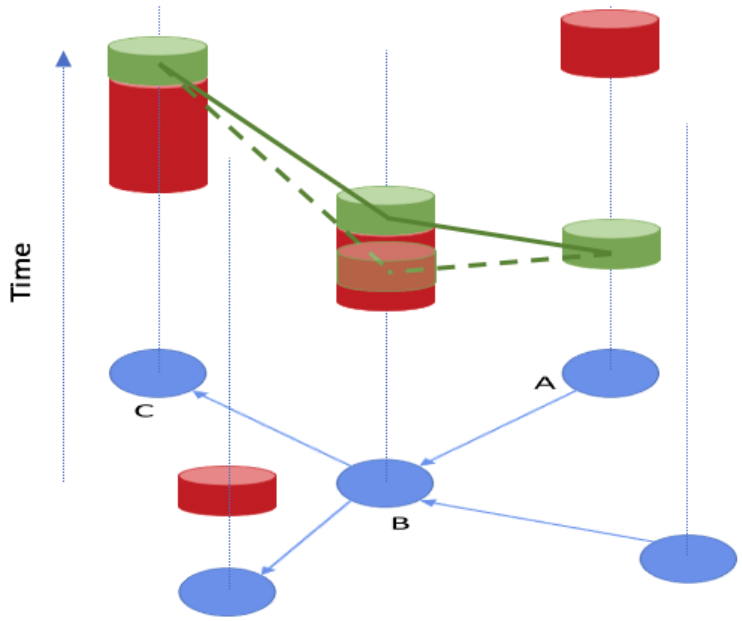

Figure 11: A trajectory from node A to node C built over a space-time with 5 nodes. The red cylinders are obstacles induced by other robots moving in the map, the green ones are the occupied nodes of the current plan. The green dashed trajectory would intersect an obstacle in B, while the continuous green line represents a collision free trajectory.

node v_s to a desired one v_g. The robots extract samples $s \in \mathcal{G}_t$ on the time-expanded roadmap with $q(s) \in \mathcal{V}$ and $\tau(s) \in [0, \infty)$. Hence, nodes of the roadmap $\mathcal{G}$ are sampled with an associated random node traversing time. Samples are then rejected or connected in order to grow a tree $\mathcal{T} \subset \mathcal{G}_t$ consisting of unoccupied space-time sampled nodes. We illustrate the procedure reported in Algorithm 2 used to obtain in short time a non optimal collision free trajectory. Once one has been found, an optimization procedure is run as reported in Algorithm 3.

Starting with the tree growth Algorithm 2, a sample can be connected to a node of the tree if the associated nodes in the roadmap $\mathcal{G}$ are connected by an arc in $\mathcal{E}$. Any node $v \in \mathcal{G}$, associated to a sample $s \in \mathcal{T}$, i.e., with $q(s) = v$, is marked as visited and inserted in the set $V \subset \mathcal{V}$ of visited nodes. An additional set of samples $Q \subset \mathcal{G}_t$, from which the tree grows, is initialized with the initial configuration Start $= (v_s, t_s)$ where t_s is the current time. Inspired by FMT* [33], the sampling procedure picks a node w from the growing set Q (Line 7), then it selects one of the adjacent nodes v of $q(w) \in \mathcal{V}$ in the roadmap $\mathcal{G}$ that are not marked as visited (Line 9), i.e. $(q(w), v) \in \mathcal{E}$ such that $v \notin V$, and thus samples a number $\bar{t} \geq \frac{\omega_{(q(w),v)}}{v_{max}}$ to obtain a node traversing time $\bar{t} + \tau(w)$. The node traversing time must be greater than the minimum time taken by the robot to reach configuration v from $q(w)$ while respecting its dynamic constraints plus the node traversing time $\tau(w)$. Indeed, smaller values of the node traversing time would correspond to unfeasible trajectories. The new node sample $s = (v, \bar{t} + \tau(w))$ is feasible if the occupancy time interval does not intersect static obstacles in $\mathcal{G}_t$. In this case the occupancy time interval is computed taking into account the average speed along the arc entering the node since the one on the exiting arc will be computed in later iterations. If the new sample $s = (v, \bar{t} + \tau(w))$ is feasible, it is added to the growing set Q (Line 13) and to the tree $\mathcal{T}$ with the arc $(q(w), v)$ (Line 14), while v is marked as visited and inserted in V (Line 12). If a collision occurs,

the sample is discarded and a new one is generated (Line 17). Samples whose associated nodes have all visited neighbors, i.e. adjacent nodes are all in V, are removed from Q (Line 19). In case of empty Q the algorithm returns an unreachable goal message. On the other hand, in case the goal node v_g is reached, the algorithm provides the final node traversing time t_g and a trajectory σ from (v_s, t_s) to (v_g, t_g) on the tree $\mathcal{T}$ (Line 21). With the proposed procedure a feasible trajectory can be found in short time but no optimality criterion is used in the trajectory construction.

Algorithm 2 Trajectory computation growing phase

1: **Data:** Graph $\mathcal{G}$, Nodes v_s, $v_g \in \mathcal{V}$, Time $t_s \in \mathbb{R}_+$
2: **Result:** Non-optimized solution to single query
3: Goal$\leftarrow (v_g, \infty)$, Start$\leftarrow (v_s, t_s)$ % Initialization
4: $s \leftarrow$ Start, $\mathcal{T} \leftarrow \{\}$
5: $Q \leftarrow \{s\}$, $V \leftarrow \{q(s)\}$
6: **while** $q(s) \neq q(\text{Goal})$ **do**
7: **if** $Q = \emptyset$ **then**
8: return goal unreachable
9: $w \leftarrow$ sample node from Q
10: **if** adj$(q(w)) \not\subset V$ **then**
11: $v \leftarrow$ sample a node in adj$(q(w)) \setminus V$
12: $t \leftarrow$ sample time greater than $\tilde{t}$
13: $s \leftarrow (v, \tau(w) + t)$
14: **if** s is feasible **then**
15: add $q(s)$ to V
16: add s to Q
17: add node s and arc $(q(w), s)$ to $\mathcal{T}$
18: **else**
19: continue
20: **else**
21: remove w from Q
22: $t_g \leftarrow \tau(s)$
23: Trajectory $\sigma \leftarrow$ Navigate $\mathcal{T}$ from (v_s, t_s) to (v_g, t_g) **return** σ

Once a trajectory σ is obtained with Algorithm 2, an informed sampling approach, similar to the one proposed in [35], is used to compute a trajectory of smaller cost. The cost $\mathcal{C}(\sigma)$ can be used as an upper bound for the informed sampling of nodes in $\mathcal{V}$. Indeed, given a node $v \in \mathcal{V}$ the cost of any trajectory from Start to (v, t) is larger than $\mathcal{C}(\text{Start}, v) = h_c \|q(\text{Start}) - v\| + h_t \frac{\|q(\text{Start}) - v\|}{v_{max}}$ for any t (cost corresponding to a straight line path covered at maximum speed). Similarly the cost of any trajectory from (v, t) to Goal is larger than $\mathcal{C}(v, \text{Goal}) = h_c \|v - q(\text{Goal})\| + h_t \frac{\|v - q(\text{Goal})\|}{v_{max}}$ for any t. Hence, any trajectory from $start$ to $goal$ through (v, t) has cost larger than $\mathcal{C}(\text{Start}, v) + \mathcal{C}(v, \text{Goal})$. If such sum is larger than $\mathcal{C}(\sigma)$, node v can be neglected since it will not be part of the optimal trajectory. Hence, v is removed from the set $\mathcal{W}$ of candidate nodes initialized with $\mathcal{V}$. This procedure is repeated for all the candidate nodes in $\mathcal{W}$ anytime a trajectory of lower cost is found. In that case the upper bound is updated and the candidate nodes checked and possibly removed from $\mathcal{W}$ as described above. Such procedure, named InformedPruning is reported in Line 7 of Algorithm 3. Once non optimal nodes are removed from $\mathcal{W}$ a candidate node in $\mathcal{W}$ is sampled (Line 8) together with a sample time t (Line 9) to create a node $w = (v, t)$ in the time-expanded roadmap $\mathcal{G}_t$. If node $w \in \mathcal{G}_t$ and does not overlap any obstacle, a BestNeighbor subroutine is performed, similarly as in RRT*. The BestNeighbor subroutine (Line12) checks all the neighboring samples in

Algorithm 3 Trajectory computation informed phase

Data: Trajectory σ from Algorithm 2, Graph $\mathscr{G}$
Result: Optimized solution to single query
$Q \leftarrow$ {Start}
BestCost $\leftarrow \infty$
$\mathscr{W} \leftarrow \mathscr{V}$
while stop not called **do**
  if $\mathscr{C}(\sigma) <$ BestCost **then**
    BestCost $\leftarrow \mathscr{C}(\sigma)$
    $\mathscr{W} \leftarrow$ InformedPruning(BestCost)
  $v \leftarrow$ sample from $\mathscr{W}$
  $t \leftarrow$ sample from $[0, \infty)$
  $w \leftarrow (v, t)$
  if w is feasible **then**
    $s \leftarrow$ BestNeighbor(w)
    if s is not empty **then**
      add w to Q
      Rewire Q if necessary
      $\sigma \leftarrow$ UpdateSolution
return σ

Q for the one, if any, that can be connected to w with minimum cost with a trajectory that does not violate speed limits. If a sample $s \in E$ is returned w is added in Q with an associated trajectory from s to w covered at constant speed in time $\tau(w) - \tau(s)$ (Line 14). Finally, a rewiring procedure, similar to the one used in RRT*, provides, in probability, the optimal path. Moreover, the motion feasibility inside BestNeighbor procedure uses the agent's maximum speed to reject invalid motions in space-time whenever the required speed is larger than the maximum allowable or is negative (this would correspond to a backward moving in time that is not allowed). As soon as the algorithm is stopped, by e.g. a time thresholds, it provides the minimum cost trajectory over the tree built starting from the trajectory obtained in Algorithm 2.

3.3.2 Negotiation Protocol

As the robot navigates through the environment, it exchanges information useful to coordinate with other robots in its communication range. The planning problem hence changes continuously and it is important that the robot can generate a new solution to account for the latest information. Such re-planning must be performed in a very timely manner, so that solutions are not out-of-date when they are finally generated, which would lead to possible collisions.

The communication range must hence be determined in order to ensure that each robot is able to prevent possible collisions. Let T_A be the time dedicated to a trajectory computation, through Algorithms 2 and 3. In the worst case scenario all the robots in the communication range are heading toward the same node, and the lowest priority robot travels at maximum speed. Moreover, such robot detects a collision after the second last robot has found a collision free trajectory with all other robots with higher priority (in a cascading effect), i.e. after a time equal to $T_A(M-1)$. The maximum distance travelled before a possible collision is hence $D_{\max} = T_A(M-1)v_{\max}$. Concluding, to ensure collision avoidance robots must communicate with all other robots at a smaller distance than $R_c = 2D_{\max}$. The ability of improving the paths when more planning time T_A is available allows to dynamically tune T_A based on the distance from the collision. This is a behavior

similar to the human driver, which in front of an intersection does not change its speed if other cars are far away, slows down if they are close, and brakes in case of emergency.

Each robot sends to neighboring robots its own priority and trajectory from its current configuration to the goal one. Lower priority robots check if their trajectories in $\mathcal{G}_t$ are going to intersect static objects given by the trajectories computed by robots with higher priorities as described at the beginning of Section 3.3. In case of a detected collision, a new trajectory computation phase is initialized using the computed static obstacles. It may occur that, after a fruitful negotiation, two robots would follow the same arc of the roadmap at different speeds, with the faster robot behind the slower one leading to a collision that must be avoided. More formally:

Definition 3.6 (Overtaking) *An overtaking on an arc* $(v_i, v_j) \in \mathcal{E}$ *between* r_l *and* r_k *occurs when* $\exists i, j, t_0 < t_1 < t_2 < t_3$ *such that* $(v_i, t_0) \in \sigma_l$ *and* $(v_j, t_3) \in \sigma_l$ *while* $(v_i, t_1) \in \sigma_k$ *and* $(v_j, t_2) \in \sigma_k$, *where* σ_l *and* σ_k *are the trajectories of* r_l *and* r_k *in* $\mathcal{G}_t$ *respectively.*

To avoid overtaking and consequent collisions along the arcs, at each step of the negotiation phase, nodes (v_j, t_2) and (v_j, t_3) are swapped between σ_l and σ_k. For multiple agents on the same arc, the same approach is extended and provides a proper reordering so that the robots occupy nodes v_i and v_j with the same temporal order. Hence, based on that and on the assigned speed, robots on the same arc in $\mathcal{G}$, with non intersecting trajectories in $\mathcal{G}_t$, verify the safety distance D_s.

Whenever a robot fails to find a path through Algorithm 2, an *emergency situation* occurs (usually in case of traffic jams). The agent hence sets its speed to zero and sends to all its neighbors an emergency message containing the currently or the last occupied node. The occupancy time interval associated to the node in the message is an obstacle for any other neighboring robot i independently from its priority. Moreover, the interval width is set very large to induce each robot i to slow down before colliding with the robot in emergency state. Indeed, once an emergency message is received, robot i has to recompute a path through Algorithm 2 if one of the nodes in its path is the one received in the message. In case of a computed trajectory that requires a velocity smaller than a given threshold, the speed is set to zero (as usually done with real mobile robots) and the robot i enters in the emergency situation and propagates a new emergency message.

3.4 Algorithm Properties

We can now prove that the proposed Mr.FERT protocol provides both collision and deadlock free trajectories under given assumptions.

Theorem 2 *Based on Assumption 1 and in case of a communication radius* $R_c = 2D_{\max}$, *the proposed coordination protocol ensures a collision free robots evolution.*

Proof 1 *With the assumed communication radius, we allow robots to complete the negotiation phase even in the worst case scenario. In case of an emergency situation, collisions are avoided with the protocol described above. If no emergency occurs, any robot has been able to compute a trajectory whose occupancy time intervals do not overlap the obstacles generated by other robots. From Proposition 3 collisions close to nodes are hence avoided. Overtakings, if any, on the same arc are managed by the negotiation phase as mentioned in previous section and hence collisions are avoided along the arc and in the nodes; hence the thesis.*

To ensure that stall situations are also avoided, we recall the deadlock given in Coffman et al. [37], considering the robots as processes and the graph nodes as resources.

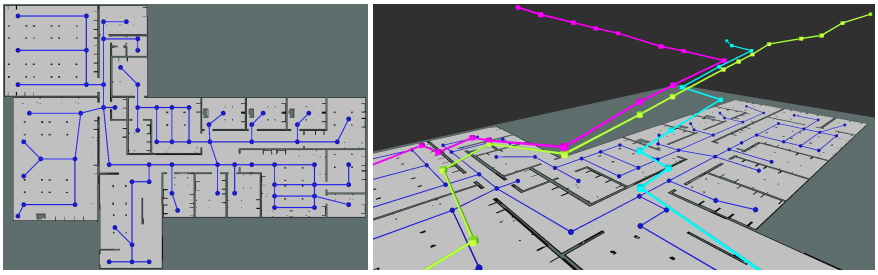

Figure 12: An example of a graph built over a planar map and trajectory of three robots in the related Time-Expanded Roadmap.

Theorem 3 *Under Assumptions 1 and 2, in case of non occurrence of emergency situations the Mr.FERT policy is deadlock free.*

Proof 2 *To verify that the evolution of the multi-robot system is deadlock free it is sufficient to deny one of the four hypothesis of the definition of deadlock in [37]. In particular, in our scenario, a circular chain of robots can not exist since the time-expanded roadmap is an acyclic graph thanks to the increasing evolution of the time. In other words, the arcs of the time-expanded roadmap only connect nodes with strictly increasing time and hence cycles are not possible.*

Under Assumption 2, a node is indefinitely occupied by the same robot only in case the robot is in the emergency state. The case of all robots in emergency state occurs only under high traffic levels in the roadmap. It is possible to prove that in case there exists on the roadmap an arc free from robots for at least a segment of length D, at least one of the robots on the arc is able to compute a collision free trajectory through Algorithm 2 and hence it exits from the emergency situation solving the temporarily stall. Formal proof of this statement is omitted for space limitations and because of limited interest since in case of traffic jams speed management is not a priority and the jam can be solved differently, e.g., changing agent priorities.

3.5 Experiments and Results

The Mr.FERT algorithm has been tested both in simulation and in real world experiments also in case of a non strongly connected roadmap to test applicability of the proposed solution (an exemplary scenario is reported in Fig. 12).

First, the trajectory computation has been compared to the version of RRT* taken from OMPL library, accordingly modified to work in $\mathbb{R}^3$ (i.e. x, y, t) with the addition of a constrained maximum robot speed. The map used is a square large 250×250 with a roadmap of 760 nodes.

Results of such simulations are reported in Fig. 13. The cost computed by our single robot trajectory planner is smaller than the RRT* implementation, while the success rate is always near 100% even for very small planning times. This is well suited to our multi robot setup, where re-plans have to be done in real time. Both algorithms can further improve the solution cost with more available time, however, given the considered application. However, for replanning purposes, we are not interested in studying the behavior for planning times larger than few seconds.

The very fast growth and the informed sampling approach of our trajectory computation, coupled together, allow for a first solution in less than 0.05 s, while a sub-optimal solution (with cost less than 130% of the optimum obtained with an exhaustive approach) can be found in less than 0.2 s.

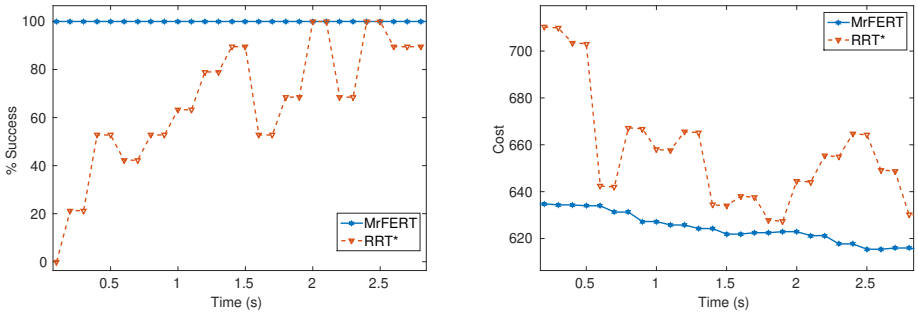

Figure 13: Comparison between RRT* and Mr.FERT for a single robot, Left: Percentage of successful plan, Right: Solution Cost

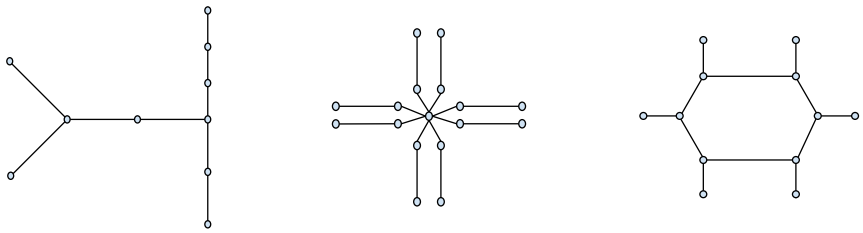

Figure 14: A team of robots tests the Mr.FERT policy on those three different roadmaps, in simulation and in real world experiment

The informed sampling performs proportionally better when the distance between start and goal is smaller, providing even 105% of the optimum in less than 0.1 s when the distance is less than few meters (in Fig. 13 the distance between start and goal was 35 m).

The Mr.FERT algorithm has then been tested in multi-robot environments with custom designed roadmaps and robot goals, in order to stress the algorithm capabilities and requirements. An example of collision free trajectories in the time expanded roadmap for three robots can be found in Fig. 12.

The three scenarios reported in Fig. 14 have been used as a simulation or experiment benchmarks. In particular, the left roadmap of Fig. 14 is related to an experiment with three iRobotCreate platforms crossing a door into a corridor in office space, see Fig. 9.

In our tests, the communication range is set to three times the maximum speed (2 m/s) multiplied by the maximum allowed planning time, 0.5 s, added to an additional margin of tolerance in node occupation to avoid delays of non-perfect robots (during real experiments). Such a configuration allows to negotiate trajectories usually two–three nodes before any possible collision, giving enough time to the fastest robot to re-plan during the negotiation phase. Indeed, no collision or deadlock occurred during the extensive conducted simulations.

The algorithms were implemented in C++ using the OMPL framework for planning, and ROS for controlling the robots, both in simulations using Gazebo and with real iRobot Create2 using our own public ROS interface.[1] Videos of simulations and experiments together with the code and the instruction to run some simulations of this work can be found at https://distributedplanning.bitbucket.io.

[1]The iRobot Create2 ROS interface can be found at https://github.com/CentroEPiaggio/irobotcreate2ros.

3.6 Conclusions

A new approach for distributed multi-robot coordination using a hybrid space-time modelization has been proposed. The approach allows to automatically tune robots speed along their path, and was proven to be deadlock and collision free. A sampling based planner and a negotiation protocol have been developed to allow on-line distributed robot coordination where robots plan their path while moving. A set of simulations and real experiments, have been performed to validate the approach and to test robustness and applicability.

Proposed algorithm can be extended to coordinate multiple aerial vehicles, as a roadmap can be generated for a 3-dimensional space, and the time can be added as the fourth dimension. Moreover, it is possible to handle different robots with different motion abilities in the same framework. Finally, proofs of deadlock and collision avoidance do not depend on the cardinality of the roadmap, as they only use a safety radius that can be easily used for generating a safety sphere around nodes in 3D space.

4 Distributing the ILIAD fleet coordinator

4.1 Introduction

Deploying fleets of autonomous robots in real-world applications requires addressing three problems: motion planning, coordination, and control. These three problems are intrinsically dependent: robot motions must be physically realizable by controls computed by robot controllers, and must also be coordinated in order to avoid collisions and deadlocks. Although methods for addressing these problems jointly have been studied, real-world applications often pose further requirements that narrow down the range of methods that can be used. Different motion planning strategies are applicable in different environments and for different types of robots. Fleets may include different types of robots with fundamentally different control schemes, and robot controllers are often certified black boxes that ship with the robot platform and cannot be modified. In practice, there are many reasons for considering motion planning, coordination and control separately. The work presented in Pecora et al. [38] proposes a lightweight coordination method that implements a high-level controller for a fleet of potentially heterogeneous robots. Very few assumptions are made on robot controllers, which are required only to be able to accept set point updates and to report their current state. The approach can be used with any motion planning method for computing kinematically-feasible paths. Coordination uses heuristics to update priorities while robots are in motion, and a simple model of robot dynamics to guarantee dynamic feasibility. The approach avoids a priori discretization of the environment or of robot paths, allowing robots to “follow each other” through critical sections.

Even though centralized methods deliver predictable fleet behavior, provable safety and liveness, and high performance [39, 40, 41, 38, 42], they are not easily scalable, suffer from single-point of failures and require for the central agent to have a complete knowledge of the environment and the robots of the fleets. Therefore, it is of great interest investigating strategies that try reducing centralization, in favor of more distributed approaches. Indeed, distributed or decentralized solutions are by nature scalable and relatively robust to communication failures [43, 10, 44, 45, 46, 47]. However, since they rely on local information, the guarantees on their safety is highly dependent on their “degree” of distribution.

Having said that, we investigated the possibility of distributing the coordination algorithm presented in Pecora et al. [38], studying different strategies and levels of distribution

and comparing them to understand which approach is better suited for the fleet coordination problem.

4.2 Notation and preliminaries

We first introduce key concepts and a high-level description of the algorithm [38] as a basis for our approach.

Paths and spatial envelopes. Consider a fleet of n (possibly heterogeneous) robots sharing an environment $\mathscr{W} \subset \mathbb{R}^3$. We use $(\cdot)_i$ to indicate that variable $(\cdot)$ refers to robot i. Let $\mathscr{Q}_i$ be the robot's configuration space, and $R_i(q) \subset \mathbb{R}^3$ its collision space when in configuration $q \in \mathscr{Q}_i$. Consider a set of obstacles $\mathscr{O} \subset \mathscr{W}$, so that $\mathscr{Q}_i^{\text{free}} = \{q \in \mathscr{Q}_i : R_i(q) \cap \mathscr{O} = \emptyset\}$ is the set of feasible (i.e., collision free) configurations. Let $\boldsymbol{p}_i : [0,1] \to \mathscr{Q}_i$ be a path in the configuration space parametrized using the arc length $\sigma \in [0,1]$. Then, *path planning* is the problem of finding a (possible executable) path $\boldsymbol{p}_i(\sigma) \in \mathscr{Q}_i^{\text{free}}$ from one feasible starting configuration q^{start} to a final one $q^{\text{goal}} \in \mathscr{Q}_i^{\text{free}}$, such that $q^{\text{start}} = \boldsymbol{p}_i(0)$ and $q^{\text{goal}} = \boldsymbol{p}_i(1)$, typically subject to a set of kinematic constrains $f_i(q, \dot{q}) \leq 0$ (Fig. 15.a). Furthermore, for each $\boldsymbol{p}_i$, the *spatial envelope* $\mathscr{E}_i$ is defined as a set of constraints such that $\cup_{\sigma \in [0,1]} R_i(\boldsymbol{p}_i(\sigma)) \subseteq \mathscr{E}_i$. If the equality holds (which we assume from now on), a spatial envelope is the sweep of the robot's footprint along its path (Fig. 15.b). Henceforth, let $\mathscr{E}_i^{\{\sigma', \sigma''\}} = \cup_{\sigma \in [\sigma', \sigma'']} R_i(\boldsymbol{p}_i(\sigma))$.

Note that $\mathscr{E}_i \cap \mathscr{O} = \emptyset \; \forall i \in \{1, \ldots, n\}$ by construction, that is, collisions between robots and the set of obstacles $\mathscr{O}$ are avoided via path planning. Also, we assume that robots are provided with a low-level safety system for detecting and avoiding obstacles that are not other robots and are not included in $\mathscr{O}$. The focus of the fleet controller proposed in this paper is therefore to avoid inter-robot collisions, not other unforeseen obstacles.

Critical sections. Given a pair of paths $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, collisions may happen only in the set $\{q_i \in \mathscr{Q}_i, q_j \in \mathscr{Q}_j \mid R_i(q_i) \cap \mathscr{E}_j \neq \emptyset \vee R_j(q_j) \cap \mathscr{E}_i \neq \emptyset\}$. In particular, let $\mathscr{C}_{ij}$ be the decomposition of this set into its largest contiguous subsets, each of which is called a *critical section* (Fig. 15.c and 15.d). For each critical section $C \in \mathscr{C}_{ij}$, 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. Considering two temporal profiles $\sigma_i(t)$ and $\sigma_j(t)$, if there exists a time t' such that $R_i(\boldsymbol{p}_i(\sigma_i(t'))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t'))) \neq \emptyset$ (i.e., the robots collide while laying in their envelopes), then $\ell_i^C < \sigma_i(t') < u_i^C$ and $\ell_j^C < \sigma_j(t') < u_j^C$. Hence, given a set of paths $\mathscr{P}$, the *coordination problem* is the problem of synthesizing, for each pair $(i, j \neq i)$ such that $\mathscr{E}_i \cap \mathscr{E}_j \neq \emptyset$, a constraint on temporal profiles $\sigma_i(t)$ and $\sigma_j(t)$ such that $R_i(\boldsymbol{p}_i(\sigma_i(t'))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t'))) = \emptyset$ for all t'. We assume that, when idle, a robot i is placed in a parking position defined by a path $\boldsymbol{p}_i$ of length one. This entails that idle robots are considered in the computation of critical sections.

Precedence constraints and critical points. Precedence constraints are relations among the temporal profiles of two robots. A precedence constraint is a pair $\langle m_i, m_j \rangle$, with $m_i, m_j \in [0,1]$, stating that robot i is not allowed to navigate beyond arc length m_i along its path until robot j has reached arc length m_j along its path — formally, $\sigma_j(t) < m_j \Rightarrow \sigma_i(t) < m_i$. As explained in [38], m_i changes over time to reflect updated precedences and to allow for robots to "follow each other" through critical sections. In general, collisions are avoided if, for each $C \in \mathscr{C}_{ij}$ and for each t, $\sigma_i(t)$ and $\sigma_j(t)$ adhere to the constraint $\langle m_i(t), u_j^C \rangle$, that is, robot i yields for robot j at an appropriately computed arc length $m_i(t)$ along its reference path; this arc length depends on whether robot j has exited critical section C (that is, reached arc length u_j^C) and on its current progress through the

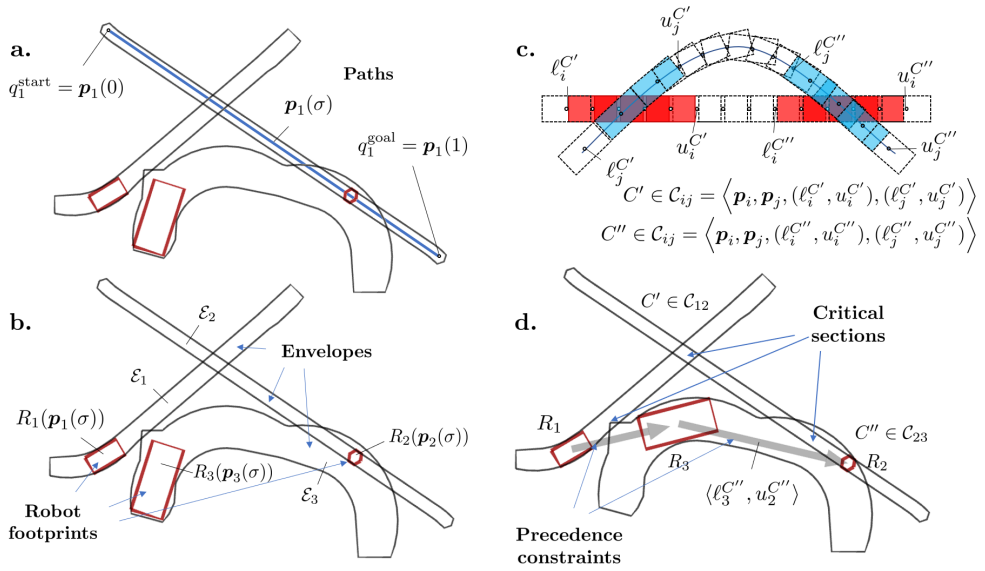

Figure 15: Preliminary concepts.

critical section:

$$m_i(t) = \begin{cases} \max\left\{\ell_i^C, r_{ij}(t)\right\} & \text{if } \sigma_j(t) \le u_j^C \\ 1 & \text{otherwise} \end{cases} \tag{29}$$

where $r_{ij}(t)$ is defined as

$$\sup_\sigma \left\{ \sigma \in [\sigma_i(t), u_i^C] : \mathscr{E}_i^{[\sigma_i(t),\sigma]} \cap \mathscr{E}_j^{[\sigma_j(t),u_j^C]} = \emptyset \right\}. \tag{30}$$

Let $\mathscr{T}$ be the set of precedence constraints regulating the motion of the robots in the fleet. A constraint $\left\langle m_i, u_j^C \right\rangle \in \mathscr{T}$ defines unambiguously which robot should yield, where it should yield, and until when yielding is necessary for critical section C. We use $(i <_C j) \in \mathscr{T}$ to indicate that robot j has precedence over robot i at a critical section C. A key feature of the approach is that $\mathscr{T}$ can be updated while robots are in motion. In particular, any heuristic function can be used to determine the precedence constraints in $\mathscr{T}$, as long as a conservative model of each robot's dynamics is employed to filter out ordering decisions that may not be physically realizable (as detailed in [38]).

Let $T_i(t) = \{m_i \mid \exists j : \left\langle m_i, u_j^C \right\rangle \in \mathscr{T}(t)\}$ be the set of all the arc lengths at which robot i may be required to yield. We define the *critical point* $\bar{\sigma}_i(t)$ of robot i at time t as the value of σ corresponding to the last reachable configuration along $\boldsymbol{p}_i$ which adheres to the set of constraints $\mathscr{T}(t)$, i.e.,

$$\bar{\sigma}_i(t) = \begin{cases} \operatorname{argmin}_{m_i \in T_i(t)} m_i & \text{if } T_i(t) \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases} \tag{31}$$

Then, *coordination* is the problem of computing and updating periodically the set of critical points $\bar{\Sigma} = \{\bar{\sigma}_1, \ldots, \bar{\sigma}_n\}$, such that collisions do not occur. Algorithm 4 shows the main body of the supervisory control loop proposed in [38].

[2] Assuming an ideal communication network, current robot states are available via message passing without delays, message loss or disorder.

Algorithm 4 Coordination at time t.

sample states[2]

1: **if** new goals have been posted **then**
2: update the set of paths $\mathscr{P}$ (using appropriate planners)
3: update the set $\mathscr{C}$ of critical sections
4: revise the set $\mathscr{T}(t)$ of precedence constraints
5: compute the set of critical points $\bar{\Sigma}(t)$
6: communicate changed critical points
7: sleep until control period T_c has elapsed

Under the assumption of a perfect communication (messages are not delayed or lost), and conservative models of the robots' dynamics, the algorithm ensures that collisions never happen (see [38] for a formal proof).

Now that the main elements of the coordination framework have been introduced, in the next section we will analyze possible solutions for its distribution. We focused on three different level of distributions, where each of them propose a modified version of the original coordination algorithm 4.

4.3 Coordination distribution - Critical points computation

The first attempt at distribution is based on modifying the way critical points are computed (line 5 of algorithm 4) after the set of precedence constraints has been defined. Recalling that the computation of the precedence constraints $\mathscr{T}$ defines the set $T_i(t)$ of the arc lengths at which robot i may be required to yield, our first solution is to modify the definition of this set. We propose to filter the set $T_i(t)$ so as to include only the elements that are close to the worst-case stopping arc length at time t, $s_i(t)$. This quantity is defined as the lowest arc length at which the robot could stop assuming it is driving at full speed. The filtered set is formally defined as $\tilde{T}_i(t) = \{m_i \mid \exists j : \left\langle m_i, u_j^C \right\rangle \in \mathscr{T}(t) \wedge m_i - s_i(t) < \epsilon\}$, the critical points can be computed similarly to (31) as follows:

$$\tilde{\sigma}_i(t) = \begin{cases} \underset{m_i \in \tilde{T}_i(t)}{\operatorname{argmin}} m_i & \text{if } \tilde{T}_i(t) \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases} \tag{32}$$

The idea is that a robot should modify its behavior only when it becomes "critical", i.e., in case it would be impossible for the robot to stop in time if no action is promptly performed. However, while allowing for maintaining the safety of the algorithm, avoiding collisions if a conservative model of the robot is used to define $s_i(t)$ and the proximity threshold ϵ, this solution does not offer advantages from the point of view of the scalability. Indeed, the centralized system is still required to compute all the precedence constraints. This can hence be seen as a first attempt to define control laws for a safe coordination based on local information and not taking into account all the available information to the central unit.

This approach can not be directly used as a full distribution of the coordination protocol where control laws depend only on nearby robots. Indeed, there exist critical sections that are big enough so that different robots requiring access to the critical section may be far apart, e.g., long corridors. For this reason, the proposed coordination is based on the closest point where the robot can stop to leave access of the associated critical section required also by a possible far apart robot. By computing critical points only for pairs of nearby robots would lead to unsolvable conflicts along corridors where access should be managed by far apart robots.

4.4 Coordination distribution - Precedence constraints computation

A different approach, that aims at reducing the computations required to the centralized system, is to modify the way the set $\mathscr{T}(t)$ of precedence constraints is revised (line 4 of algorithm 4). In the current ILIAD coordinator implementation, precedence constraints are computed for all the critical sections $C \in \mathscr{C}_{ij}$ involving the robots. However, a possible solution might be to filter out the critical sections that are still "far" and should not influence the current state of the two robots. To implement this criterion, we rely on the concept of worst-case stopping arc-length at time t previously defined and the information on whether a robot is inside the critical section. Thus, a precedence constraint for a critical section $C \in \mathscr{C}_{ij}$ should be computed if the maximum distance of the robots between the start of the critical section and their worst-case stopping arc-length is lower than safety threshold, i.e., $\max(l_i^C - s_i(t), l_j^C - s_j(t)) < \epsilon$.

Using this approach, the number of critical sections for which a precedence constraint has to be computed is reduced, and some of the conflicts might be resolved naturally without the real need of centralized coordination. As for the previous case, collisions can be avoided if a conservative model of the robot is used to define $s_i(t)$ and select the safety threshold ϵ.

4.5 Coordination distribution - Complete Distribution

Algorithm 5 Coordination at time t - Distributed approach.

sample state;
1: communicate state
2: listen for other robot states
3: update the set $\mathscr{C}$ of critical sections
4: revise the set $\mathscr{T}(t)$ of precedence constraints
5: compute the set of critical points $\tilde{\Sigma}(t)$
6: sleep until control period T_c has elapsed

While the other two proposed approaches still rely on a centralized system that has information on all the robots (or possibly distant robot) of the fleets and communicates with all of them, a completely distributed approach would require to distribute the responsibility of coordination, and each agent should define its set of constraints on the temporal profile based on local sensing and communication.

In this completely distributed approach, the centralized algorithm 4 is substituted by the distributed coordination algorithm 5. At each control period, each robot of the fleets samples its state, communicates it to the robots within its communication radius, and receives their states. Based on the received information, the critical sections, the precedence constraints and the critical points are updated. In the completely decentralized approach the coordination is performed only for the robots that are able to communicate with each other, and the precedence constraints are resolved locally by each agent. While this approach reduce the computational cost of the required computations there are cases in which the presence of large critical sections may jeopardise the overall safety as shown in the simulation results.

4.6 Simulations

In this section we aim at showing the behaviours of the different approaches in different scenarios and highlight their points of weakness and strength. We compare the ILIAD

approach described in D5.1, named *Centralized* in the following, and the three proposed approaches: the critical points computation approach based on the worst-case stopping length (Section 4.3), named *Filtered Critical Points*, the precedence constraints computation approach based on closest critical points (Section 4.4), named *Filtered Precedence Constraints*, and the fully distributed approach (Section 4.5), named *Distributed*. Different approaches are tested and compared in different scenarios where their different behaviours can be evaluated. First, in Section 4.6.1, a two robot scenario is considered where a robot have multiple access to the same critical section. Second a long critical section, as a warehouse aisle, is considered for two robots in case of a platooning and a head-to-head scenarios (Section 4.6.2). Scenarios of three robots are then proposed in a generic scenario (Section 4.6.3) and in a classical one point of intersection scenario where deadlocks are possible (Section 4.6.4). Finally a comparison of the approaches is performed in a more generic scenario with four robots and four critical sections (Section 4.6.5).

In the figures reported below each robot is numbered and higher priority robot has a lower associated number. Moreover, when a dependency between robots is taken into account, a grey arrow from the lower priority robot to the higher priority one is reported. Recalling the different methods compared in this section, in the *Centralized* case dependency are always detected if critical sections between pair of robots occurs. In the *Filtered Critical Points* approach dependency is considered whenever the lower priority robot is sufficiently close to the closest critical point of critical sections. In the *Filtered Precedence Constraints* approach the dependency is considered whenever both robots interested in the closest critical section are sufficiently close to the beginning of the critical section. Finally, in the *Distributed* approach dependency are considered only if robots interested in a critical section are sufficiently closed.

4.6.1 Coordination of two robots

Referring to Fig. 16, consider a scenario with two robots with a single critical section that the highest priority robot has to traverse twice along its path. In the current ILIAD implementation the lower priority robot has to wait until the highest priority one traverses it twice even if it could traverse it in between the two crossings. In this scenario, the *Filtered Precedence Constraints* and the *Filtered Critical Points* approaches do not show differences with respect to the *Centralized* one except for when the dependency is detected. On the other hand, with the *Decentralized* approach, after once the highest priority robot has traversed the critical section the robots update the critical section and the lower priority robot may move toward its final destination, since no other robot is in the communication radius and hence no section has to be shared with other robots. Indeed, as it can be seen in figure 16 the robot at lower priority can move toward its goal while the one at highest priority has still to traverse the initial critical section for the second time. Such behaviour explains Table 1 where the time to reach the final destination is reported for each approach and each robot. As expected the first three approaches produce equivalent time for both robots while in the *Decentralized* approach the lowest priority robot has an improvement of around 30%.

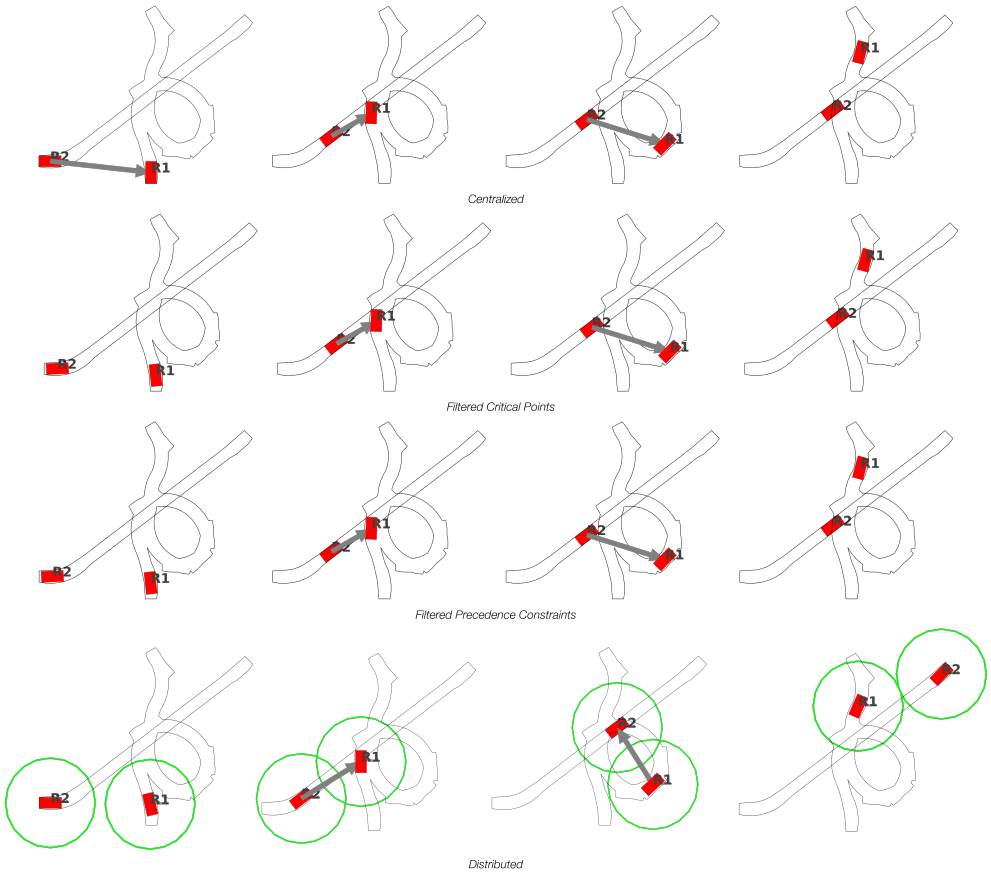

Figure 16: Coordination for two robots and a single critical section

Method	t_{R1}	t_{R2}	Number of Collisions
Centralized	23.04	27.30	0
Filtered Critical Points	23.07	27.26	0
Filtered Precedence Constraints	23.01	27.14	0
Distributed	23.06	18.41	0

Table 1: Traveling times and number of collisions for the case of two robots and a single critical section

4.6.2 Coordination in case of long critical section

Long critical sections are not unusual in logistics applications due to the presence of aisle in warehouses. In such cases it is common to have two robots that need to traverse the aisle along the same direction and in opposite directions. We have performed simulations in both platooning (see Fig. 17) and head-to-head (see Fig. 18) situations to evaluate the different approaches. Also in this case there is one single critical section and the critical points are close to the robots in both the platooning and the head-to-head scenarios. Hence, the *Centralized*, the *Filtered Critical Points* and the *Filtered Precedence Constraints* approaches have the same behavior while differences lay in the time at which dependency occurs.

It is worth noting that in the platooning scenario, the *Distributed* approach allows for a correct coordination and a behavior similar to the other cases is obtained even though from Table 2 shorter time is required by the lowest priority robot to reach its final destination. Indeed, in this case, based on the value of the communication radius the robots are able to cover the aisle closer to each other (see Fig. 17). On the other hand, in the head-to-head scenario limitations of a fully distributed approach is shown for long critical sections. In fact, due to the dimension of the communication radius, both the robots enter the critical section without coordination (no dependency is detected) leading to a collision.

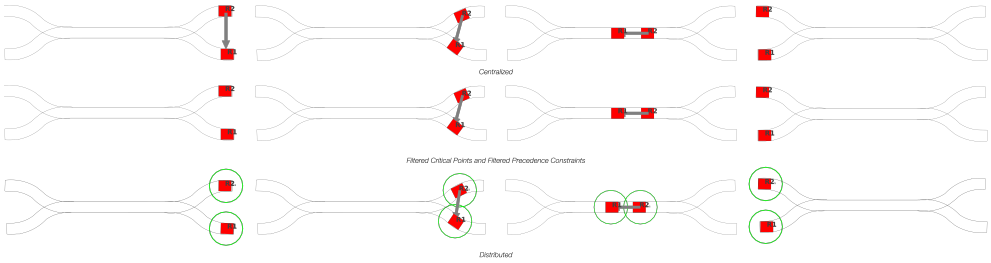

Figure 17: Coordination in case of long critical section - Same Direction

Method	t_{R1}	t_{R2}	Number of Collisions
Centralized	11.46	13.41	0
Filtered Critical Points	11.43	13.28	0
Filtered Precedence Constraints	11.43	13.23	0
Distributed	11.49	13.10	0

Table 2: Traveling times and number of collisions for the case of two robots and long critical section - Same Direction

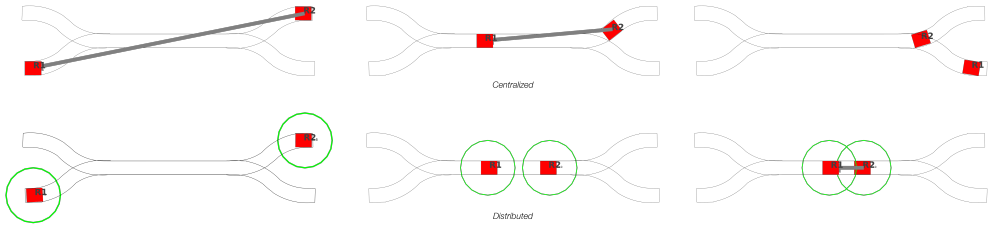

Figure 18: Coordination in case of long critical section - Opposite Direction

Method	t_{R1}	t_{R2}	Number of Collisions
Centralized	11.43	18.24	0
Filtered Critical Points	11.41	18.41	0
Filtered Precedence Constraints	11.44	18.34	0
Distributed	-	-	1

Table 3: Traveling times and number of collisions for the case of two robots and long critical section - Opposite Direction

4.6.3 Coordination of three robots

Let us now consider a scenario with three robots and two different critical sections. Referring to Fig. 19 there are three robots where $R2$'s path first intersects $R1$'s path and then intersects $R3$'s one. In the *Centralized* approach dependency are computed immediately also if $R2$ and $R3$ are far from each other and $R2$ is far from the critical sections, this lead $R3$ to wait until $R2$ passes. The *Filtered Critical Points*, as also in other cases, changes only in the time of dependency detection. Indeed, dependency of $R3$ from $R2$ is detected at the beginning since $R3$ is close to its critical point while $R2$ dependency form $R1$ is detected only when $R2$ comes closer to the first critical section. In both approaches $R3$ has to wait for $R2$ to pass. In the *Filtered Precedence Constraints* approach no dependency is detected at the beginning and the dependency of $R3$ from $R2$ never occurs and hence $R3$ does not have to wait for the higher priority robot to pass and the time to reach the destination is drastically reduced (almost 50%, see Table 4). In this particular scenario the *Distributed* approaches have a similar behaviour with respect to the *Filtered Precedence Constraints* one.

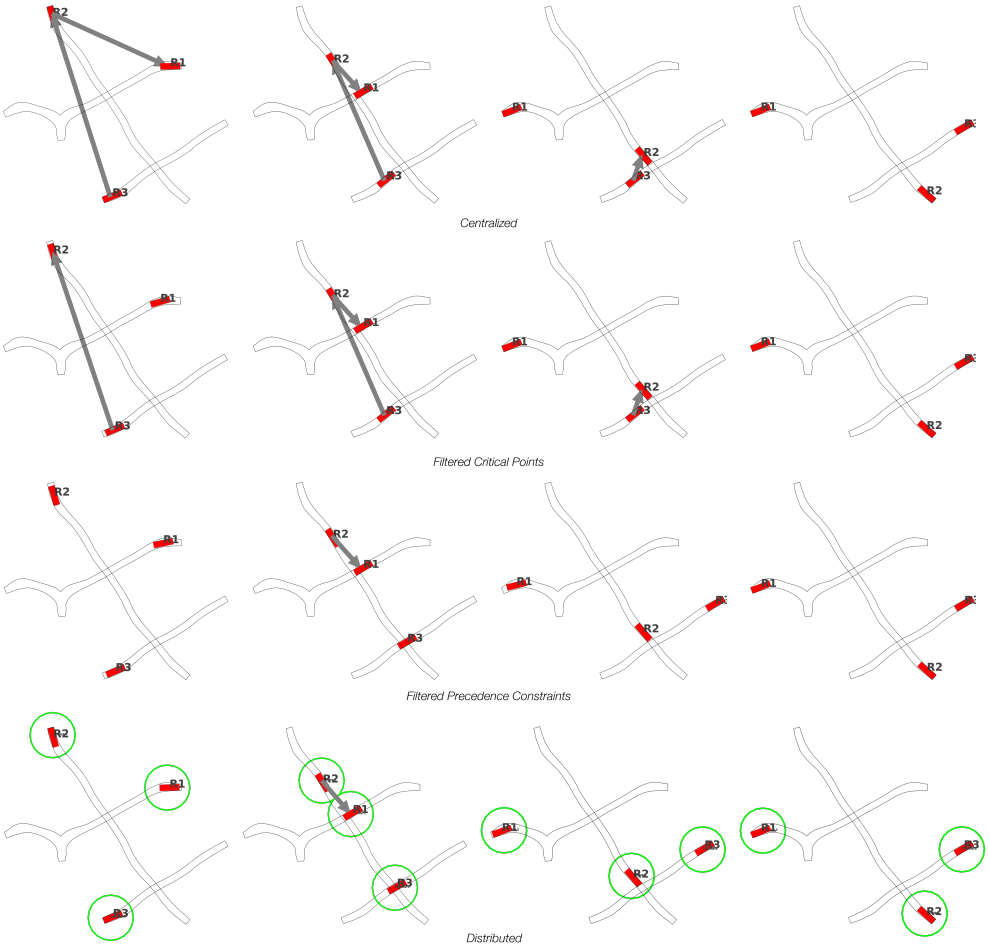

Figure 19: Coordination for Three Robots

Method	t_{R1}	t_{R2}	t_{R3}	Number of Collisions
Centralized	13.46	15.04	19.03	0
Filtered Critical Points	13.46	15.03	19.02	0
Filtered Precedence Constraints	13.44	15.01	10.01	0
Distributed	13.46	15.00	10.01	0

Table 4: Traveling times and number of collisions for the case of three robots navigating

4.6.4 Deadlocks

A different scenario with three robots and one critical section is now considered to evaluate deadlocks. Referring to Fig. 20 there are three robots that want to reach a point along a straight path generating a critical section common to all three agents. In case precedences are not correctly defined a deadlock scenario may occur. Indeed, suppose robot $R1$ has lower priority than $R3$ (and as above $R3$ lower than $R2$ and $R2$ lower that $R1$) no coordination is possible with the *Centralized* and the *Filtered Critical Points* approaches since robot get stacked before the critical section. On the contrary, even small asymmetries in the scenario allow the *Filtered Precedence Constraints* and the *Distributed* approaches to avoid the deadlock and coordinate the robot toward their final destination.

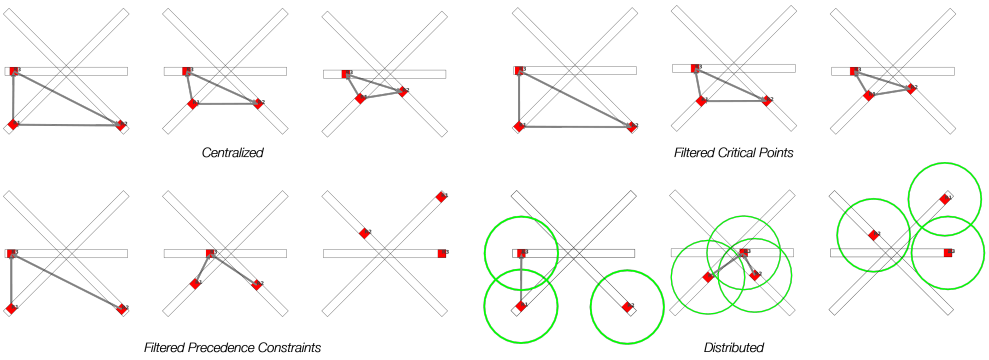

Figure 20: Scenario inducing possible deadlocks

Method	t_{R1}	t_{R2}	t_{R3}	Number of Collisions
Centralized	-	-	-	-
Filtered Critical Points	-	-	-	-
Filtered Precedence Constraints	11.12	11.22	10.71	0
Distributed	11.12	11.24	10.69	0

Table 5: Traveling times and number of collisions for the case of three robots coordinating with deadlock-inducing heuristics

4.6.5 Coordination of four robots

Finally a scenario with four robots and four critical sections is considered in Fig. 21. Also in this scenario it is possible to appreciate how the *Filtered Precedence Constraints* and the *Distributed* approaches allow a more convenient coordination for the lower priority

robot that saves almost 50% of the time (see Table 6) with respect to the *Centralized* and the *Filtered Critical Points* approaches.

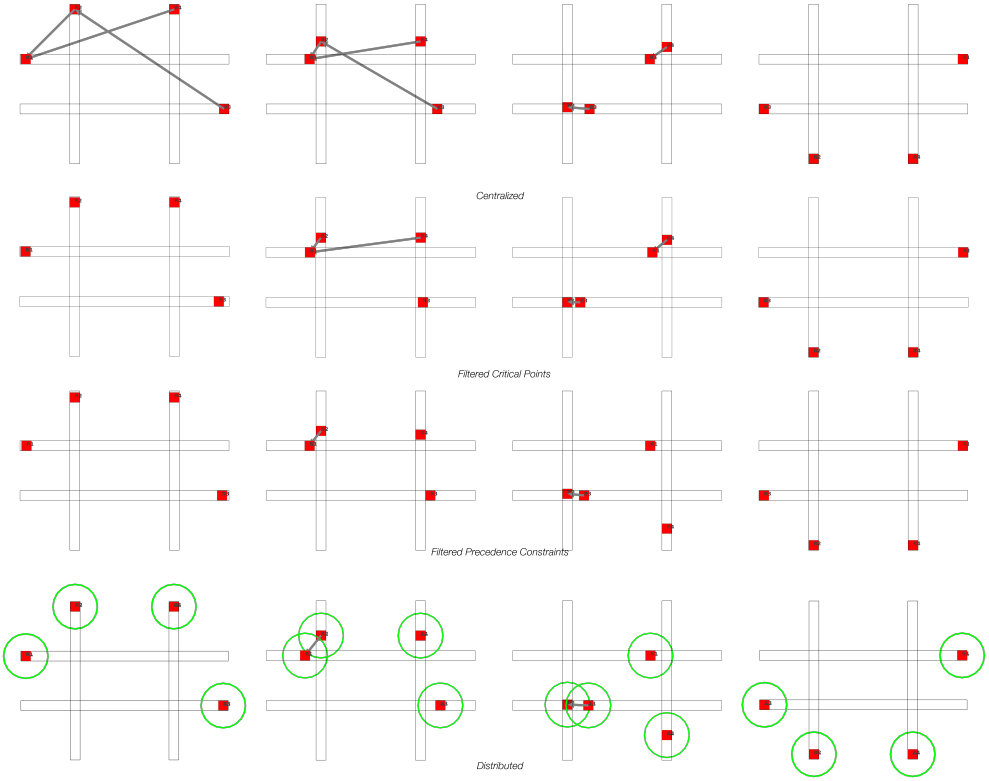

Figure 21: Coordination for Four Robots

Method	t_{R1}	t_{R2}	t_{R3}	t_{R4}	Number of Collisions
Centralized	14.39	13.18	15.37	20.13	0
Filtered Critical Points	14.34	13.35	15.43	20.04	0
Filtered Precedence Constraints	14.48	13.55	15.40	11.65	0
Distributed	14.38	13.78	15.49	11.52	0

Table 6: Traveling times and number of collisions for the case of four robots.

4.7 Comments

We have proposed different possible approaches that are characterised by modifications of increasing level of distribution, compared to the centralized ILIAD coordinator. Approaches have been compared in different scenarios that show their potentiality and limits. First an approach based on the worst-case stopping length for critical point computation has been proposed (*Filtered Critical Points*) that does not offer advantages from the point of view of scalability since the centralized system is still required to compute all the precedence constraints. This can hence be seen as a first attempt to define control laws for a safe coordination based on local information and not taking into account all the available information to the central unit. Then the second approach computes precedence constraints based only closest (hence local) critical points *Filtered Precedence*

Constraints, and finally the fully distributed approach (*Distributed*) has been proposed that coordinates only robots close to each other.

From the comparison we can deduce that the *Filtered Precedence Constraints* performs better with respect to the other approaches since it improves the time of travel of low priority robots (as also the *Distributed* one) and solves possible deadlocks. The approach is based on the possibility that robots at large distances but interested in the same resource can exchange information, and hence coordinate, as in the case of the head-to-head aisle crossing. However, this requires a communication system (or local coordinator) mounted along the resource that allows the information exchange between agents requiring access to the resource that may not communicate with a direct vechile-to-vehicle communication system. With such local coordinators associated to the aisles, the *Filtered Precedence Constraints* is able to coordinate robots in such head-to-head conflicts that the *Distributed* approach cannot handle.

Concluding, a fully distributed approach may not be able to solve conflicts in all possible scenarios while a hierarchical approach such as the one required by the *Filtered Precedence Constraints* is a good balance between centralized and distributed levels of the ILIAD coordinator; both in terms of performance, and conflicts and deadlocks resolution.

References

[1] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "A Survey and Analysis of Multi-Robot Coordination". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399. DOI: 10.5772/57313.

[2] A. Bicchi, A. Fagiolini, and L. Pallottino. "Towards a Society of Robots". In: *IEEE Robotics Automation Magazine* 17.4 (2010), pp. 26–36.

[3] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. "An overview of recent progress in the study of distributed multi-agent coordination". In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 427–438. ISSN: 15513203. DOI: 10.1109/TII.2012.2219061. arXiv: 1207.3231.

[4] Lynne E Parker. "Path planning and motion coordination in multiple mobile robot teams". In: *Encyclopedia of complexity and system science* (2009), pp. 5783–5800.

[5] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "A survey and analysis of multi-robot coordination". In: *International Journal of Advanced Robotic Systems* 10 (2013).

[6] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

[7] Stefan Schemmer, Edgar Nett, and Michael Mock. "Reliable real-time cooperation of mobile autonomous systems". In: *Reliable Distributed Systems, 2001. Proceedings. 20th IEEE Symposium on*. IEEE. 2001, pp. 238–246.

[8] Markus Jäger and Bernhard Nebel. "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2001, pp. 1213–1219.

[9] Wee Lit Koh and Suiping Zhou. "An extensible collision avoidance model for realistic self-driven autonomous agents". In: *Distributed Simulation and Real-Time Applications, 2007. DS-RT 2007. 11th IEEE International Symposium*. IEEE. 2007, pp. 7–14.

[10] S. Manca, A. Fagiolini, and L. Pallottino. "Decentralized coordination system for multiple agvs in a structured environment". In: *IFAC Proc. Volumes* 44.1 (2011), pp. 6005–6010.

[11] Li Wang, Aaron D. Ames, and Magnus Egerstedt. "Safety barrier certificates for collisions-free multirobot systems". In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674. ISSN: 15523098. DOI: 10.1109/TRO.2017.2659727.

[12] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3861–3876. ISSN: 00189286. DOI: 10.1109/TAC.2016.2638961. arXiv: 1609.06408.

[13] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. "Robustness of Control Barrier Functions for Safety Critical Control". In: *IFAC-PapersOnLine* 48.27 (2015), pp. 54–61. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.11.152. arXiv: 1612.01554.

[14] Benjamin Morris, Matthew J. Powell, and Aaron D. Ames. "Sufficient conditions for the lipschitz continuity of QP-based multi-objective control of humanoid robots". In: *Proceedings of the IEEE Conference on Decision and Control* (2013), pp. 2920–2926. ISSN: 01912216. DOI: 10.1109/CDC.2013.6760327.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 9780521833783. DOI: 10.1037/a0025001. arXiv: 1310.1707.

[16] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. "The Robotarium: A remotely accessible swarm robotics research testbed". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2017), pp. 1699–1706. ISSN: 10504729. DOI: 10.1109/ICRA.2017.7989200. arXiv: 1609.04730.

[17] Li Wang, Aaron Ames, and Magnus Egerstedt. "Safety barrier certificates for heterogeneous multi-robot systems". In: *Proceedings of the American Control Conference*. Vol. 2016-July. 2016, pp. 5213–5218. ISBN: 9781467386821. DOI: 10.1109/ACC.2016.7526486.

[18] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. "A framework for worst-case and stochastic safety verification using barrier certificates". In: *IEEE Transactions on Automatic Control* 52.8 (2007), pp. 1415–1428. ISSN: 00189286. DOI: 10.1109/TAC.2007.902736. arXiv: 1309.7825 [astro-ph.CO].

[19] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. "ACS-PRM: Adaptive cross sampling based probabilistic roadmap for multi-robot motion planning". In: *Intelligent Autonomous Systems 12*. Springer, 2013, pp. 843–851.

[20] Patrick Beeson, Nicholas K Jong, and Benjamin Kuipers. "Towards autonomous topological place detection using the extended voronoi graph". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 4373–4379.

[21] Jingjin Yu and Steven M LaValle. "Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs." In: *AAAI*. 2013.

[22] Pavel Surynek. "SOLVING ABSTRACT COOPERATIVE PATH-FINDING IN DENSELY POPULATED ENVIRONMENTS". In: *Computational Intelligence* 30.2 (2014), pp. 402–450.

[23] Liron Cohen, Tansel Uras, and Sven Koenig. "Feasibility study: using highways for bounded-suboptimal multi-agent path finding". In: *Eighth Annual Symposium on Combinatorial Search*. 2015.

[24] Jingjin Yu and Steven M LaValle. "Planning optimal paths for multiple robots on graphs". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3612–3617.

[25] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. “Conflict-based search for optimal multi-agent pathfinding”. In: *Artificial Intelligence* 219 (2015), pp. 40–66.

[26] Ulrich Schwesinger, Roland Siegwart, and Paul Furgale. “Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 63–68.

[27] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[28] Glenn Wagner, Minsu Kang, and Howie Choset. “Probabilistic path planning for multiple robots with subdimensional expansion”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2886–2892.

[29] Ryan Luna and Kostas E Bekris. “Network-guided multi-robot path planning in discrete representations”. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 4596–4602.

[30] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart. “Reciprocal collision avoidance for multiple car-like robots”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 360–366.

[31] Yusuke Ikemoto, Yasuhisa Hasegawa, Toshio Fukuda, and Kazuhiko Matsuda. “Zipping, weaving: control of vehicle group behavior in non-signalized intersection”. In: *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*. Vol. 5. IEEE. 2004, pp. 4387–4391.

[32] Mirko Ferrati and Lucia Pallottino. “A time expanded network based algorithm for safe and efficient distributed multi-agent coordination”. In: *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE. 2013, pp. 2805–2810.

[33] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions”. In: *The International Journal of Robotics Research* (2015), p. 0278364915577958.

[34] Farzana Islam, Jauwairia Nasir, Usman Malik, Yasar Ayaz, and Osman Hasan. “Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution”. In: *Mechatronics and Automation (ICMA), 2012 International Conference on*. IEEE. 2012, pp. 1651–1656.

[35] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. “Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic”. In: *arXiv preprint arXiv:1404.2334* (2014).

[36] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. “Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 3067–3074.

[37] Edward G Coffman, Melanie Elphick, and Arie Shoshani. “System deadlocks”. In: *ACM Computing Surveys (CSUR)* 3.2 (1971), pp. 67–78.

[38] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov. “A Loosely-Coupled Approach for Multi-Robot Coordination, Motion Planning and Control”. In: *Proc. 28th Int. Conf. Autom. Planning & Scheduling*. 2018.

[39] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic. “Time windows based dynamic routing in multi-AGV systems”. In: *IEEE Trans. Autom. Sci. Eng.* 7.1 (2010), pp. 151–155.

[40] J. Peng and S. Akella. “Coordinating multiple robots with kinodynamic constraints along specified paths”. In: *Int. J. Robot. Research* 24.4 (2005), pp. 295–310.

[41] F. Pecora, M. Cirillo, and D. Dimitrov. “On mission-dependent coordination of multiple vehicles under spatial and temporal constraints”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.* 2012, pp. 5262–5269.

[42] M. Čáp, J. Gregoire, and E. Frazzoli. “Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.* 2016, pp. 5113–5118.

[43] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich. “A decentralized control strategy for the coordination of AGV systems”. In: *Control Eng. Practice* 70 (2018), pp. 86–97.

[44] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan. “Decentralized control of multi-AGV systems in autonomous warehousing applications”. In: *IEEE Trans. Autom. Sci. Eng.* 13.4 (2016), pp. 1433–1447.

[45] K. E Bekris, D. K Grady, M. Moll, and L. E Kavraki. “Safe distributed motion coordination for second-order systems with different planning cycles”. In: *Int. J. Robot. Research* 31.2 (2012), pp. 129–150.

[46] D. Bareiss and J. Van den Berg. “Generalized reciprocal collision avoidance”. In: *Int. J. Robot. Research* 34.12 (2015), pp. 1501–1514.

[47] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto. “Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.* 2017, pp. 236–243.