Ref. Ares(2023)8909269 - 30/12/2023

This project is funded by the European Union
G.A No 101016906

Project Number: 101016906
Start Date of Project: 2021/01/01
Duration: 48 months

Type of document: D8.2 – V1.0

Multi-Robot Task Planning with Humans in the Shared Workspace

Dissemination Level	PUBLIC
Submission Date	2023-12-30
Work Package	WP8
Task	T8:2
Type	Report
Version	1.0
Authors	Xiao Tan, Shankar Deka, Xu Fang, Carlos Rodriguez de Cos
Approved by	UNIROMA3, UNICLAM

DISCLAIMER:
The sole responsibility for the content of this deliverable rests with the authors. It does not necessarily reflect the opinion of the European Union. Neither the Directorate-General for Communications Networks, Content and Technology, Artificial Intelligence and Digital Industry nor the European Commission are responsible for any use that may be made of the information contained therein.

This project is funded by the European Union G.A No 101016906

Executive Summary

This report summarises the research questions, developments, and results arising from Task 8.2 "Multi-robot Task Planning with Humans in the Shared Workspace". This task has the goal of designing online task planning solutions for agronomic robots that adapt their behaviors according to human agent behaviors in the workspace.

This report thus describes technical advancements of our proposed strategies that ensure safety of the human and multi-robot systems in a shared environment, while each robot accomplishes the specified agronomic task. We propose online re-planning and online minimal modification strategies to increase the flexibility of the agronomic robotic system. To expand a little, when the initial plans of individual agents are conflicting, the logistic robots need to re-plan online their paths in order to account for the uncertain human behaviors. The alternative approach is that, at the control level, the robots can modify their task-satisfying control signals in a minimal invasive manner by utilizing the distributed control barrier function techniques to ensure safety. Alongside the theoretical descriptions, respective hardware demonstration and validation in a vineyard environment at a local indoor facility are also included. A result on human agent physically helping manipulation together with robotic arms is also included, which is achieved by incorporating inferred human contributions in the control loop of the manipulator agent. In the end of this report, we briefly present the latest field experimental results with the developed methodologies, and more implementation details will be given in Deliverable D9.3 "Human-Multi-Robot Task Planning Field Validation".

Table of Contents

1 Introduction . . . 5
1.1 Purpose of the document . . . 5
1.2 Objectives of T8.2 . . . 5
1.3 Related works . . . 5
1.3.1 Spatial-temporal task planning for multi-robot systems . . . 5
1.3.2 Multi-robot safety problems . . . 6
1.4 Document overview . . . 7
2 Preliminaries . . . 8
2.1 Application scenarios . . . 8
2.2 Graph theory . . . 8
2.3 Linear temporal logic (LTL) . . . 9
2.4 Control barrier functions . . . 9
3 Multi-robot motion coordination under LTL with human in the workspace . . . 11
3.1 Problem formulation . . . 11
3.2 MRMC under local LTL specifications with online conflict resolution . . . 13
3.2.1 Offline pre-computation . . . 14
3.2.2 Initialization . . . 16
3.2.3 Online motion coordination . . . 17
3.2.4 Conflict detection . . . 17
3.2.5 Determine planning order . . . 18
3.2.6 Motion planning . . . 19
3.3 Numerical Validation . . . 23
3.4 MRMC under local LTL specifications in a dynamic environment . . . 28
3.4.1 Local communication case . . . 28
3.4.2 Communication-free case . . . 30
3.5 Laboratory experiments . . . 31
3.5.1 Local communication case . . . 32
3.5.2 Communication-free case with human as moving obstacle . . . 33
4 Safety assuring multi-robot coordination using distributed CBF . . . 35
4.1 Problem formulation . . . 35
4.2 Results and contribution . . . 35
4.2.1 Minimum function approximation . . . 36
4.2.2 Solving for the new coefficients . . . 37
4.3 Validation: formation control with a human walking around . . . 38
5 Human-robot co-manipulation . . . 44
5.1 Problem formulation . . . 44
5.1.1 Kinematics and dynamics . . . 44
5.2 Results and contributions . . . 47
5.3 Validation . . . 49
6 VR experiment and preliminary field experiment of Task T8.2 . . . 54
7 Conclusions . . . 57

This project is funded by the European Union
G.A No 101016906

Abbreviations and Acronyms

CO	Coordinator
EC	European Commission
WP	Work Package
LTL	Linear Temporal Logic
CBF	Control Barrier Functions
MRMC	Multi-Robot Motion Coordination

This project is funded by the European Union G.A No 101016906

1 Introduction

Human agents are an essential in Precision Agriculture and need to work together with robots in a safe and efficient way, e.g., avoiding potentially dangerous interactions. This requires the design of online task planning solutions that adapt the robot behavior according to humans moving through the workspace. However, the different sources of sensing/understanding from the robot's perspective are not easy to be considered in a unified and integrated manner. This is since the type of information they bring during real-time decision making, including both logical and temporal ingredients, is diverse and not directly applicable to traditional methodologies for task planning.

1.1 Purpose of the document

This report documents the theoretical and validation results of Task 8.2 "Multi-robot Task Planning with Humans in the Shared Workspace".

1.2 Objectives of T8.2

The main objective of this task is the automated synthesis of safe-assuring controllers for autonomous robotic systems that are given complex task and safety specifications with human agents in the shared workspace. Namely, given a plan consisting of desired sequences of robot spatial and temporal states (e.g. robots A-D must reach regions I-IV at times t1-t4), we convert the plan into a temporal logic specification and include the human presence as obstacles to be avoided in the multi-robot planning. On-line re-planning and safe-by-construction approaches will be required to account for humans moving through the workspace.

During the research period, we have developed strategies for fully distributed multi-robot motion coordination under the temporal logic tasks and conducted validation experiments at the KTH research facilities and in VR environment. When the robots are tasked with their own temporal logic tasks and share the same work space, a fully decentralized strategy is proposed to resolve any possible local conflicts and deadlocks; when the robots share collective tasks, a distributed coordination strategy is proposed with assured safety guarantee. Both strategies are tested in an indoor research facility in KTH. We have also integrated the temporal logic planner developed from the results as a ROS package and tested the functionalities in a VR environment developed by Paleblue, a result of Task 8.4. A brief up-to-date field experiment is included in the end.

1.3 Related works

1.3.1 Spatial-temporal task planning for multi-robot systems

During the past decade, there is a surge of using temporal logic formulas, such as linear temporal logic (LTL) [1], to concisely specify desired behaviors of robotic systems [2, 3, 4, 5, 6, 7, 8, 9, 10]. This is due to the expressiveness of LTL formulas in capturing many common robotic tasks, e.g., ordered reachability, collision avoidance, surveillance, and ordered supply delivery [11]. In addition, the availability of automated toolboxes [1, 12] make the use of LTL more appealing. The planning and control under LTL specifications have been extensively investigated for a single robot [2, 3, 4, 5] and multi-robot systems (MRSs) [6, 7, 8, 9, 10]. However, most of the existing literature assume that the environment is static and known a priori. When the environment is dynamic (e.g., there are moving obstacles like humans walking through), for which

This project is funded by the European Union G.A No 101016906

online replanning becomes a necessity, the problem of safe and efficient replanning under LTL specifications becomes challenging during the online execution of robots.

Adding on top of the capabilities provided by rigorous formal language specified robotic tasks, along the rapid advancement of automation technology, the past decade has witnessed a growing emphasis on human-robot collaboration. Indeed, many autonomous systems are performing their designated tasks while being supervised or collaborating with human operators [13]. On one hand, having a human-in-the-loop is useful for guiding the robot through challenging tasks. On the other hand, the possibly erroneous inputs from the human can be dangerous for the autonomous systems. Therefore, effectively managing real-time interactions between the autonomous system and humans is crucial for ensuring the safety and efficiency of the entire system. This calls for a proper consideration of human factors into the system design.

When a single robot is considered, authors in [14] propose a plan revising mechanism assuming the environment is static and partially known. Once a transition in the current plan becomes invalid due to, for example, the environment change or human intervention, the replanner finds the shortest path bridging up the two components. The work [15] develops an iterative repair strategy to resolve unknown obstacles by combining local patching with refined triangulation. However, this strategy cannot deal with moving obstacles. For the case of MRSs, most of research assumes that the MRS adheres to a global LTL specification, subsequently addressing an offline motion planning problem through a centralized approach [6, 7]. In [16], MRSs under local LTL specifications are studied and a distributed motion coordination algorithm is proposed to resolve conflicts. In this work, the environment is static and local communication between robots is required for conflict resolution. Furthermore, when humans are involved in the robot control, the concept of a mixed-initiative controller (MIC) is introduced in [17], which integrates external human inputs with the conventional navigation controller. This method is further investigated in [18], where the human initiative influences both the high level tasks and the low level continuous inputs.

1.3.2 Multi-robot safety problems

Autonomous systems now operate in open environments, engaging physically with humans and their surroundings. Ensuring safety is paramount yet challenging. In industrial automation, robotic manipulators were traditionally confined within cages for operator protection with pre-designed maneuvers; mobile robots were limited to well-defined environments like factories or warehouses, following predetermined routes for safety. Nowadays, robotic applications extend to less structured, dynamic environments with human operators going through the workspace. One core challenge towards smarter autonomy underscores the need for a robust framework ensuring correctness and safety in the face of constant change and nonlinear dynamics, especially with the presence of human operators.

In the literature, control barrier functions are specially useful and extensively investigated in recent years for the safety of multi-robot systems. In this direction, several different safety criteria have been explored, like inter-collision avoidance [19, 20, 21, 22], connectivity maintenance [21, 23], and temporal logic tasks [24]. These are all solved either in a centralized manner [20, 23], where one module has access to the states of all agents, or with a pre-allocation scheme [19, 24] that distributes the linear constraint among the agents involved in a feasible, but non-optimal way. This calls for a decentralized and optimal implementation scheme for the barrier function-based approach, and thus enable its application to larger robotic systems.

In this report, we are particularly interested in the applications of distributed control barrier function ap-

This project is funded by the European Union G.A No 101016906

proach to the problem of precision agriculture, through two scenarios related to grape collection in vineyards, where a network of agents must help harvesting and pruning while satisfying collision avoidance constraints between each agent and humans.

1.4 Document overview

The rest of this document is structured as follows: common preliminaries of this deliverable are detailed in Section 2, including the contextualization of the research lines presented here within the general scenario for WP8 and some common definitions.

Section 3 describes an online replanning algorithm on the planning level for multi-robot systems that resolves possible conflicts in their plans, accomplish its local tasks, and ensure the safety of the human operator and other moving robots.

Section 4 gives details on the a distributed control barrier function based approach to modify the low-level control of the mobile robots in a minimal invasive manner. This safety filter mechanism always guarantees safety of the human-robot systems and can act as a low-level fail-safe.

Section 5 investigates how human can help robotic manipulators to collectively move a heavy box. Algorithms are developed for the manipulators that can infer the force contributions of human workers and take that into the control loop.

We highlight that all the sections are accompanied with experimental demonstrations at a local facility in KTH, tested in pergola-like environment and/or a VR simulation environment. In the end of this deliverable, we will briefly recap the field validation experiment result up to now, with more details and more field experiments to be reported in Deliverable D9.3 “Human-Multi-Robot Task Planning Field Validation”.

It is worth noting that this deliverable is focused on the research advancements with Humans in the Shared Workspace where human agents are physically present. Human-in-the-loop task planning, where human agents are indirectly involved by, i.e., remote control or natural language commands, is considered in deliverable D8.3.

This project is funded by the European Union G.A No 101016906

2 Preliminaries

2.1 Application scenarios

For the grape harvesting and pruning scenario, a team of agronomic robots are expected to visit different locations in the grape vineyard at different times (e.g. robots A-D must reach regions I-IV at times t1-t4). This robotic tasks can be precisely specified by temporal logic specifications. In order to label the regions of interest, the field vineyard environment is partitioned into grids, and this can be readily mapped to from online SLAM module shown in Fig. 1. Details on SLAM architecture can be found in Deliverable 9.1.

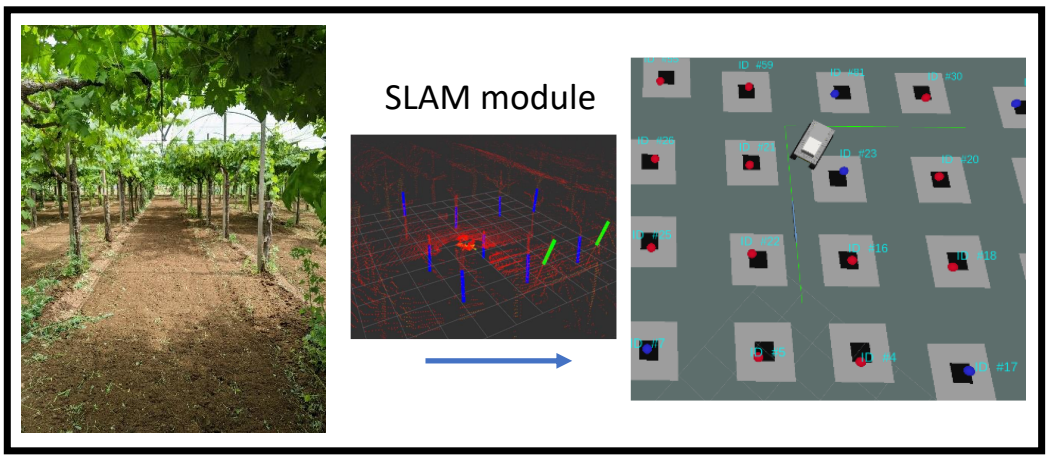


Figure 1: Grape vineyard environment and the mapped grid world from SLAM.

In this deliverable, we propose a safe-by-design motion plan with an on-line re-planning module that each robot achieves their respective temporal logic tasks while accounting for humans moving through the workspace.

2.2 Graph theory

We utilize a communication graph to represent the inter-agent communication topology. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with the set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$, and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, j \neq i\}$ being the set of edges. If $(i, j) \in \mathcal{E}$, then node j is called a neighbor of node i and node j can receive information from node i. The neighboring set of node i is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ and $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{i\}$.

A graph is called undirected if $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$. A graph is called directed if the condition $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$ does not hold in general. A graph is called connected if for every pair of nodes (i, j), there exists a path which connects i and j. A directed graph contains a directed spanning tree if there exists a node called the root such that there exist directed paths from this node to every other node.

The adjacency matrix of a graph is denoted by $\mathcal{A} = (a_{ij})_{N \times N}$ and is given by $a_{ij} = 1$, if $(i, j) \in \mathcal{E}$, otherwise $a_{ij} = 0$. Let $\mathcal{D} = (d_{ij})_{N \times N}$ represent the degree matrix which is a diagonal matrix with entries $d_{ii} = \sum_{j=1, j \neq i}^{N} a_{ij}$. Then the Laplacian matrix of the graph $\mathcal{G}$ is defined as $L = (l_{ij})_{N \times N} = \mathcal{D} - \mathcal{A}$.

This project is funded by the European Union G.A No 101016906

2.3 Linear temporal logic (LTL)

We use LTL to concisely specify the desired robot behaviour. LTL is built from a set of atomic propositions AP, the logic connectives of negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$), and the temporal operators next ($\bigcirc$), until (U), eventually ($\Diamond$), and always ($\Box$). An LTL formula is formed inductively according to the following syntax [1]:

$$\varphi ::= \top | a | \neg\varphi | \varphi_1 \wedge \varphi_2 | \bigcirc \varphi | \varphi_1 \mathsf{U} \varphi_2, \tag{1}$$

where $a \in AP, \varphi, \varphi_1, \varphi_2$ are LTL formulas. The logic connective $\vee$ and temporal operators $\Diamond$ and $\Box$ can be derived inductively. We omit the full LTL semantics due to space limitations and refer the reader to [1] for details. The satisfaction of an LTL formula φ over AP can be captured through a nondeterministic Büchi automaton (NBA) [25], defined as a tuple $\mathsf{B} = (S, S_0, 2^{AP}, \delta, F)$, where

- S is a finite set of states,
- $S_0 \subseteq S$ is the set of initial states,
- 2^{AP} is the input alphabet,
- $\delta : S \times 2^{AP} \to 2^S$ is the transition function, and
- $F \subseteq S$ is the set of accepting states.

An infinite *run* $\boldsymbol{s} = s_0 s_1 \dots$ is called *accepting* if $\mathtt{Inf}(s) \cap F \neq \emptyset$, where $\mathtt{Inf}(s)$ is the set of states that appear in $\boldsymbol{s}$ infinitely often. Several translation tools, e.g., LTL2BA [26], are available to obtain B given φ.

Within the workspace $\mathcal{W}$, there is a set of properties (atomic propositions) $AP = \{a_1, a_2, \cdots, a_M\}$, e.g. "this is a goal region", "this is a charging station". Let $\Phi := \{X_1, \dots, X_M\}$ be a partition of $\mathcal{W}$ such that $\mathcal{W} = \cup_{l=1}^{M} X_l$. Define $L : \mathcal{W} \to 2^{AP}$ as a labelling function. Given a point $p \in \mathcal{W}$, define the function $Q : \mathcal{W} \to \Phi$ as $Q(p) := \{X_l \in \Phi : p \in X_l\}$. It maps a state p into the region X_l that contains it. We say Φ is an *observation preserving* partition of $\mathcal{W}$ if it satisfies

$$L(p) = L(p'), \forall p, p' : Q(p) = Q(p').$$

Given an observation preserving partition of the workspace, the motion of one robot can be abstracted as a controlled transition system (CTS) $\mathcal{T}_i = (\Phi, \phi_{0,i}, \mathcal{U}_i, AP, \to_i, L)$, where $\phi_{0,i} \in \Phi$ is the initial state and $\to_i$: $\Phi \times \mathcal{U}_i \to \Phi$ is the transition relation.

2.4 Control barrier functions

Safety-critical control for dynamical systems has been researched and discussed for a long time. One way to achieve this is through control barrier functions (CBF), initially proposed by [27] and later developed in [28, 29, 30]. They are a convenient modular design tool based on the concept of set forward invariance, i.e., the system state should always remain in a safe set once it starts inside. This safety is represented as a linear constraint on the input, which is enforced using a computationally efficient implementation that leverages quadratic programs such that a pre-designed nominal controller is minimally modified to satisfy safety requirements. This approach has been widely investigated and applied with practical success.

Consider a general control-affine nonlinear system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}. \tag{2}$$

This project is funded by the European Union G.A No 101016906

where f, g are locally Lipschitz vector fields, $\boldsymbol{x}$ is the state and $\boldsymbol{u}$ is the control input. A safety set $\mathcal{C}$, shown in Fig. 2, represents a specific constraint of our design. It is defined as

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}, \tag{3}$$

where $h(\boldsymbol{x})$ is a differentiable function. We denote the set $\mathcal{D}$ as a superset of the safety set.

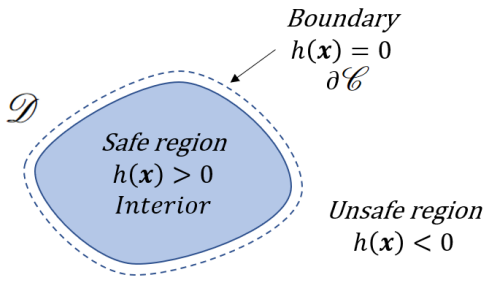


Figure 2: Geometric representation of the safe region $\mathcal{C}$.

Definition 1 (Extended class $\mathcal{K}$ function)**.** A continuous, increasing function $\alpha : (-b, a) \longrightarrow [0, \infty)$ with $\alpha(0) = 0, a, b > 0$, is called an extended class $\mathcal{K}$ function.

Definition 2 (CBF)**.** Let set $\mathcal{C}$ be defined by (3). $h(\boldsymbol{x})$ is a control barrier function (CBF) for the system (2) if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that:

$$\sup_{\boldsymbol{u} \in \mathbb{R}^m} [L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})\boldsymbol{u} + \alpha(h(\boldsymbol{x}))] \geq 0, \ \forall \boldsymbol{x} \in \mathcal{D} \supset \mathcal{C}, \tag{4}$$

where $L_f h = \nabla h^\top f(\boldsymbol{x}) \in \mathbb{R}$ and $L_g h = \nabla h^\top g(\boldsymbol{x}) \in \mathbb{R}^{1 \times m}$ and the operator $\nabla : C^1(\mathbb{R}^n) \to \mathbb{R}^n$ is defined as the gradient $\frac{\partial}{\partial x}$ of a scalar-valued differentiable function with respect to $\boldsymbol{x}$.

Notably, it has been shown [28, 29] that any locally Lipschitz control input $\boldsymbol{u}$ that satisfies the CBF constraint (4) renders the set $\mathcal{C}$ forward invariant and, if $\mathcal{C}$ is compact, it is also asymptotically stable.

This project is funded by the European Union G.A No 101016906

3 Multi-robot motion coordination under LTL with human in the workspace

This Section presents the online motion coordination solution for a group of mobile robots moving in a shared workspace, each of which is assigned a linear temporal logic specification. We propose a fully distributed multi-robot motion coordination strategy for fulfilling the LTL tasks. In contrast to most of existing literature, which presumes a static and known environment, our study focuses on dynamic environments that can have unknown moving obstacles like humans walking through.

In the end of this section, several experiments are reported that are conducted in a local indoor lab at KTH to demonstrate the effectiveness and the applicability of the proposed strategies.

3.1 Problem formulation

Robotic model and world model

Consider a group of robots moving in a bounded workspace $\mathbb{W} \subset \mathbb{R}^l$. The dynamics of robot i is given by

$$\dot{x}_i = F_i(x_i, u_i), i \in \mathcal{V}, \tag{5}$$

where

$$x_i := (p_i, \zeta_i) \in \mathbb{R}^n$$

represents the state of robot i, which contains its position state $p_i \in \mathbb{R}^l$ and non-position state $\zeta_i \in \mathbb{R}^{n-l}$ (*e.g.,* orientation and/or velocity), and $u_i \in \mathbb{R}^m$ represents the input of robot i. The function $F_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the state evolution of robot i. The output of robot i is the position state, *i.e.,* $y_i = \texttt{proj}_l(x_i) = p_i, \forall i$.

The state and input of robot i are constrained to the following compact sets

$$x_i(t) \in \mathbb{X}_i, u_i(t) \in \mathbb{U}_i, \forall t \geq 0. \tag{6}$$

Denote by $\boldsymbol{\xi}_i : [0, \infty) \to \mathbb{R}^n$ the trajectory of robot i with dynamics given by (5). Then, we further define $\boldsymbol{p}_i : [0, \infty) \to \mathbb{R}^l$ as the position trajectory of robot i, where $p_i(t) = \texttt{proj}_l(\xi_i(t)), \forall t \in \texttt{dom}(\boldsymbol{\xi}_i)$. Given a time interval $[t_1, t_2], t_1 < t_2$, the corresponding trajectory and position trajectory are denoted by $\boldsymbol{\xi}_i([t_1, t_2])$ and $\boldsymbol{p}_i([t_1, t_2])$, respectively. Denote by $\boldsymbol{\xi}_i([t, \infty))$ and $\boldsymbol{p}_i([t, \infty))$ the trajectory and the position trajectory of robot i from time t onwards, respectively.

Supposing that the sensing radius of each robot is the same, given by $R > 0$, then the communication graph formed by the group of robots is undirected. The neighboring set of robot i at time t is given by $\mathcal{N}_i(t) = \{j \in \mathcal{V} : \|x_i(t) - x_j(t)\| \leq R, j \neq i\}$, so that $j \in \mathcal{N}_i(t) \Leftrightarrow i \in \mathcal{N}_j(t), \forall i \neq j, \forall t$.

The group of robots are working in a common workspace $\mathbb{W}$, which is populated with a set of closed sets O_i, corresponding to obstacles. Let $\mathbb{O} = \cup_i O_i$, then the free space $\mathbb{F}$ is denoted by $\mathbb{F} := \mathbb{W} \setminus \mathbb{O}$.

Suppose that a cell decomposition is given over the workspace $\mathbb{W}$. The cell decomposition is a partition of $\mathbb{W}$ into finite disjoint convex regions $\Phi := \{X_1, \ldots, X_M\}$ with $\mathbb{W} = \cup_{l=1}^{M} X_l$. Given a point $p \in \mathbb{W}$, define the map $Q : \mathbb{W} \to \Phi$ as

$$Q(p) := \{X_l \in \Phi : p \in X_l\},$$

which maps a point p into a cell $X_l \in \Phi$ that contains it. Let AP_{φ_i} be the set of atomic propositions specified by φ_i. Define $AP = \cup_{i \in \mathcal{V}} AP_{\varphi_i}$.

Given a trajectory $\boldsymbol{\xi}_i$, the notation $\boldsymbol{\xi}_i \models \varphi_i$ means that the trajectory $\boldsymbol{\xi}_i$ satisfies the specification φ_i. Given the position p_i of robot i, we refer to its *footprint* $\phi_i(x_i)$ as the set of points in $\mathbb{W}$ that are occupied by robot i in this position. We note that the footprint of robot i can take into account not only the shape of robot i but also practical issues such as measurement errors.

Problem formulation

The objective is to find, for each robot i, a trajectory $\boldsymbol{\xi}_i$ such that $\boldsymbol{\xi}_i \models \varphi_i$ on the premise that safety (no collisions with static obstacles and no inter-robot collisions) is guaranteed. Let $t = 0$ be the task activation time of robot $i, \forall i$. Then, the centralized and offline version of the multi-robot motion coordination (MRMC) problem is formulated below:

$$\text{find} \quad \{\boldsymbol{\xi}_i([0,\infty))\}_{i \in \mathcal{V}} \tag{7a}$$

subject to

$$(5) \text{ and } (6), \forall i \in \mathcal{V}, \tag{7b}$$

$$\boldsymbol{\xi}_i([0,\infty)) \models \varphi_i, \forall i \in \mathcal{V}, \tag{7c}$$

$$\phi_i(\boldsymbol{p}_i([0,\infty)) \subset \mathbb{F}, \forall i \in \mathcal{V}, \tag{7d}$$

$$\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall i, j \in \mathcal{V}, i \neq j, \forall t. \tag{7e}$$

Constraint (7d) means that the footprint of each robot will not collide with the static obstacles at any time. Constraint (7e) means that the footprint of two different robots can not intersect at any time, thus guaranteeing no inter-robot collision occurs. Note that each robot has only local view and local information, *i.e.,* each robot considers only robots in its neighborhood $\mathcal{N}_i(t)$ at each time t and can have only local information of its neighbors.

Under these settings, the MRMC problem (7) is broken into local distributed motion coordination problems and solved online for individual robots. Let $\boldsymbol{p}_j([t, t_j^*(t)])$ be the local position trajectory of robot j that is available to robot i at time t, where $t_j^*(t) := \min_{t' > t}\{x_j(t') \notin \mathcal{B}(x_i(t), R)\}$. Then, the (online) motion coordination problem for robot i is formulated as

$$\text{find} \quad \boldsymbol{\xi}_i([t,\infty)) \tag{8a}$$

subject to

$$(5) \text{ and } (6), \tag{8b}$$

$$\boldsymbol{\xi}_i([0,t] \cup [t,\infty)) \models \varphi_i, \tag{8c}$$

$$\phi_i(\boldsymbol{p}_i([t,\infty))) \subset \mathbb{F}, \tag{8d}$$

$$\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t), \forall t' \in [t, t_j^*(t)], \tag{8e}$$

where $\boldsymbol{\xi}_i([0,t])$ is the history trajectory.

We assign an LTL specification φ_i to each robot i, which is specified over the set of atomic propositions AP. Nevertheless, the workspace is dynamic, for instance, there may be moving obstacles like humans walking through. In addition, it is assumed that each robot has only limited sensing capabilities, that is, the robot can

This project is funded by the European Union G.A No 101016906

detect other robots or the unknown moving obstacles only if they are within its sensing region $\mathcal{B}(p_i, R_i)$, where p_i is the position and R_i is the sensing radius of robot i. The initial trajectory planning (which is detailed in the next subsection) cannot take into account these unpredictable situations, and thus online replanning is necessary for each robot to guarantee safe operation and task satisfaction.

Given the abstract model (which shall be constructed later), i.e., the CTS $\mathcal{T}_i$, and the task specification φ_i of each robot i, the first objective is to design a planning algorithm for each robot i such that the LTL task φ_i is satisfied and safety is always guaranteed despite of the dynamic environments.

3.2 MRMC under local LTL specifications with online conflict resolution

The proposed solution to the motion coordination problem (8) consists of three layers: 1) an offline pre-computation layer, 2) an initialization layer, and 3) an online coordination layer. The offline computation layer computes the Workspace discretization and the braking distance and braking time in case of emergency; the initialization layer computes an initial trajectory for each robots that will be revised during the course of operation; the online coordination layer will detect possible conflicts between robots, determine the priorities of robots, and re-plan the trajectories online when necessary.

A high-level description of the structure of each robot is presented (see Fig. 3). Each robot i is equipped with five modules, the conflict detection, the planning order assignment, the trajectory planning, the control, and the communication module. The first four modules work sequentially while the communication module works in parallel with the first four. While this framework can account for complex dynamics, for CANOPIES robots, a simple dynamic model is adopted here for planning purpose and the trajectories from this section act as a reference to track by the onboard controllers. For CANOPIES robots, details on the low-level controllers are provided in Deliverables 7.1 and 9.1.

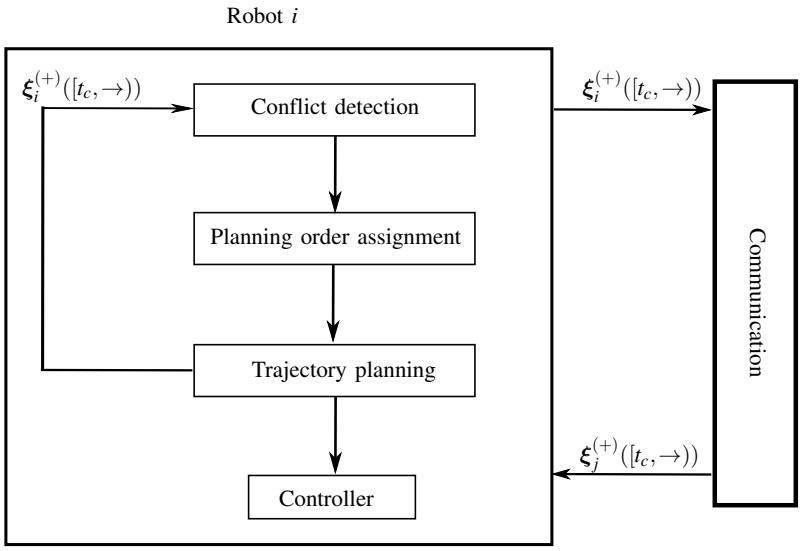


Figure 3: The structure of robot i.

This project is funded by the European Union G.A No 101016906

During online execution, robot i tries to satisfy its specification safely by resolving conflicts with other robots. This is done by following some mode switching rules encoded into a finite state machine (FSM), see Fig. 4. Each FSM has the following three modes:

- **Free**: Robot moves as planned. This is the normal mode.
- **Busy**: Robot enters this mode when conflicts are detected. In this mode, the planning order assignment module and the trajectory planning module are activated.
- **Emerg**: Robot starts an emergency stop process.

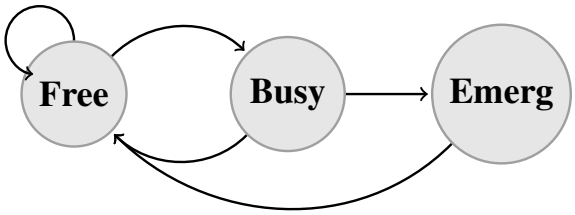


Figure 4: The three modes of robot i and the transitions among them.

In Fig. 4, the transitions between different modes of the FSM are depicted. Initially, robot i is in **Free** mode. The conflict detection module is activated when the online execution starts. Once conflict neighbors (will be defined later) are detected, robot i enters to **Busy** mode and the planning order assignment and the trajectory planning modules are activated to solve the conflicts, otherwise, robot i stays in **Free** mode. When robot i is in **Busy** mode, it switches back to **Free** mode if the trajectory planning module returns a feasible solution, and the solution will be broadcasted to the robot's neighboring area as well as sent to the controller for execution, otherwise (*e.g.,* no feasible plan is found), robot i switches to **Emerg** mode and a braking controller (defined later) is applied. Note that when robot i switches to **Emerg** mode, it will come to a stop but with power-on. This means that robot i will continue monitoring the environment and restart (switches back to **Free** mode) when it is possible.

3.2.1 Offline pre-computation

In this layer, each robot computes the braking distance and braking time in case of emergency. We also detail on workspace discretization. These are essential ingredients for online coordination.

Braking controller and braking distance

Given the initial state $x_i \in \mathbb{X}_i$ of robot i, define

$$t_i^*(x_i) := \min \quad T \tag{9a}$$
$$\text{subject to } x_i(0) = x_i, \tag{9b}$$
$$\dot{x}_i(t) = F_i(x_i(t), u_i(t)), \tag{9c}$$
$$x_i(t) \in \mathbb{X}_i, u_i(t) \in \mathbb{U}_i, t \in [0, T) \tag{9d}$$
$$\dot{p}_i(T) = 0_l, \tag{9e}$$

This project is funded by the European Union G.A No 101016906

as the minimal time needed to decelerate robot i to zero velocity (*i.e.*, $\dot{p}_i(t) = 0_l$). Let $\boldsymbol{u}_i^*(x_i)$ be the optimal solution of (9). Then, the *braking controller* $\boldsymbol{u}_i^{\mathsf{br}}(x_i)$ is designed as

$$u_i^{\mathsf{br}}(x_i)(t) = \begin{cases} u_i^*(x_i)(t), & \text{if } \dot{p}_i(t) \neq 0_l, \\ 0_m, & \text{if } \dot{p}_i(t) = 0_l. \end{cases} \tag{10}$$

Denote by $\boldsymbol{\xi}_i^{\mathsf{br}}(x_i)$ the *braking trajectory* of robot i starting at x_i. Then we have

$$\xi_i^{\mathsf{br}}(x_i)(t) = \xi_i(x_i, \boldsymbol{u}_i^{\mathsf{br}}(x_i), t), t \in [0, t_i^*(x_i)]. \tag{11}$$

The *braking position trajectory* of robot i, denoted by $\boldsymbol{p}_i^{\mathsf{br}}(x_i)$, is then given by

$$\boldsymbol{p}_i^{\mathsf{br}}(x_i) = \mathtt{proj}_l(\boldsymbol{\xi}_i^{\mathsf{br}}(x_i)). \tag{12}$$

Let $d_i^*(x_i) := \max_{p \in \boldsymbol{p}_i^{\mathsf{br}}(x_i)} \|p - \mathtt{proj}_l(x_i)\|$. Then, the *braking time* T_i^{br} and the *braking distance* D_i^{br} are defined as

$$T_i^{\mathsf{br}} := \max_{x_i \in \mathbb{X}_i} \{t_i^*(x_i)\}, \tag{13}$$

$$D_i^{\mathsf{br}} := \max_{x_i \in \mathbb{X}_i} \{d_i^*(x_i)\}. \tag{14}$$

Workspace discretization

Given the set of AP, there exists an observation preserving cell decomposition $\Phi = \{X_1, \ldots, X_M\}$ over the workspace $\mathbb{W}$. In general, $X_{l_1}, X_{l_2}, l_1 \neq l_2$ are of different shapes and sizes. For the sake of online coordination, we propose to further discretize each cell $X_l \in \Phi$ into smaller regions. Let $\Xi(X_l) := \{\hat{X}_l^1, \ldots, \hat{X}_l^{M_l}\}$ be a partition of X_l into M_l disjoint convex regions. Then, we define $\Xi := \cup_{X_l \in \Phi} \Xi(X_l)$. Ξ can be seen as a finer discretization over the workspace $\mathbb{W}$.

Given a set $S \subseteq \mathbb{W}$, let

$$\hat{Q}(S) := \{\hat{X} \in \Xi : \hat{X} \cap S \neq \emptyset\}, \tag{15}$$

which represents the set of regions in Ξ that intersect with S.

Given the position p_i of robot i, define

$$\mathbb{S}_i(p_i) := \{x_i \in \mathbb{X}_i : \mathtt{proj}_l(x_i) = p_i\}$$

as the subset of states in $\mathbb{X}_i$ that correspond to p_i. In addition, we define the *braking area* of robot i at position p_i as

$$\Phi(p_i) := \cup_{x_i \in \mathbb{S}_i(p_i)} \phi_i(\boldsymbol{p}_i^{\mathsf{br}}(x_i)), \tag{16}$$

where $\phi_i(\boldsymbol{p}_i^{\mathsf{br}}(x_i))$ is the footprint of the braking position trajectory $\boldsymbol{p}_i^{\mathsf{br}}(x_i)$. Let

$$\psi_i(p_i) := \mathcal{B}(\phi_i(p_i), D_i^{\mathsf{br}}). \tag{17}$$

Then, we have the following result.

This project is funded by the European Union G.A No 101016906

Proposition 1. The braking area of robot i at position p_i can be over-approximated by the set $\psi_i(p_i)$, that is,

$$\Phi(p_i) \subseteq \mathcal{B}(\phi_i(p_i), D_i^{\text{br}}), \forall p_i \in \mathbb{W}.$$

Due to the continuity of the dynamics, the state and the input spaces, the transition system that represents (5) is infinite for each robot i. To this end, a probabilistically complete sampling-based algorithm is proposed in [31] to approximate (5) by a finite transition system. Given a sampling interval τ_s, the finite transition system that represents (5) is denoted by $\mathcal{T}_i := (\hat{\mathbb{X}}_i, \hat{\mathbb{X}}_i^0, \mathbb{U}_i, \rightarrow_{c,i}, F_i, \mathbb{W}, h)$, where

- $\hat{\mathbb{X}}_i$ collects all sampling points in $\mathbb{X}_i$ that are safe (with respect to the static obstacles), *i.e.*, $\psi_i(\texttt{proj}_l(x_i)) \subseteq \mathbb{F}, \forall x_i \in \hat{\mathbb{X}}_i$),
- $\hat{\mathbb{X}}_i^0 \subseteq \hat{\mathbb{X}}_i$,
- $\rightarrow_{c,i} \subseteq \hat{\mathbb{X}}_i \times \mathbb{U}_i \times \hat{\mathbb{X}}_i$,

F_i is given in (5), $\mathbb{W}$ is the workspace as well as the observation space, and $h(\cdot) = \texttt{proj}_l(\cdot)$ is the observation map. Here, $\mathbb{X}_i, \mathbb{U}_i$ are the set of states and inputs, which are defined in (6). The transition relation $(x_i, u_i, x_i') \in \rightarrow_{c,i}$ if and only if $x_i' = \xi_i(x_i, u_i, \tau_s)$. Define

$$\texttt{Post}(x_i) := \{x_i' \in \hat{\mathbb{X}}_i : \exists u_i \in \mathbb{U}_i, x_i \xrightarrow{u_i} x_i'\}. \tag{18}$$

Once $\mathcal{T}_i$ is obtained, one can further construct the NBA $\text{B}_i := (S_i, S_i^0, 2^{AP_{\varphi_i}}, \delta_i, F_i)$ for the specification φ_i, the CTS $\mathcal{T}_{c,i} := (\hat{\mathbb{X}}_i, \hat{\mathbb{X}}_i^0, AP_{\varphi_i}, \rightarrow_{c,i}, L_{c,i})$, and then form the PBA $\mathcal{P}_i = \mathcal{T}_{c,i} \times \text{B}_i$. After that, the potential function for $\mathcal{P}_i$ can be computed.

3.2.2 Initialization

In this layer, each robot finds an initial feasible trajectory for their given LTL specifications, providing an initial plan that will be revised later.

Before proceeding, the following definition is required.

Definition 3. We call a trajectory $\boldsymbol{\xi}_i$ of (5) *safely satisfy* an LTL formula φ_i if i) $\boldsymbol{\xi}_i \models \varphi_i$ and ii) $\psi_i(p_i) \subseteq \mathbb{F}, \forall p_i \in \texttt{proj}_l(\boldsymbol{\xi}_i)$.

At the task activation time $t = 0$, robot i first finds a trajectory $\boldsymbol{\xi}_i^0$ that safely satisfy φ_i. The trajectory planning problem for a single robot can be solved by many existing methods, such as search- or sampling-based [32, 33], automata-based [34], and optimization-based methods [35, 36]. For details about this process, we refer to interested readers to corresponding literatures and the references therein.

The following assumption is needed to guarantee the feasibility of each task specification φ_i.

Assumption 1. *Initially, for each robot i, there exists a trajectory $\boldsymbol{\xi}_i^0$ that safely satisfy φ_i.*

This project is funded by the European Union G.A No 101016906

3.2.3 Online motion coordination

In this layer, each robot will start their voyage, detect the possible conflicts, and resolve them using local sensing and communication. This is essential for the success of motion coordination task since the initially planned trajectory of each robot does not consider the motion of other robots. Moreover, each robot has only local view and local information (about other robots). Therefore, motion coordination is required during online implementation. Based on the sensing information (about the workspace) and broadcasted information (from neighboring robots), each robot can detect conflicts within its neighborhood and then conduct motion replanning such that conflicts are avoided. In this report, we consider that the conflict detection is conducted at a sequence of sampling instants $\{t_k\}_{k\in\mathbb{N}}$ for each robot i, where $t_{k+1} - t_k \equiv \Delta$.

The details on each module in Fig. 3 is presented in the following sections.

3.2.4 Conflict detection

Before proceeding, the following notation is introduced. Given a position trajectory $\boldsymbol{p}_i([t_1, t_2])$ and a region $\hat{X}_l \in \Xi$, the function $\Gamma : \mathcal{F}(\mathbb{R}_{\geq 0}, \mathbb{R}^l) \times \Xi \to 2^{\mathbb{R}_{\geq 0}}$, defined as

$$\Gamma(\boldsymbol{p}_i([t_1, t_2]), \hat{X}_l) := \{t \in [t_1, t_2] : p_i(t) \in \hat{X}_l\}, \tag{19}$$

gives the time interval that the position trajectory $p_i([t_1, t_2])$ occupies the region $\hat{X}_l$.

Given the position $p_i(t_k)$, $\mathcal{B}(p_i(t_k), R)$ represents the sensing area of robot i at time t_k and $\hat{Q}(\mathcal{B}(p_i(t_k), R))$ represents the set of regions in Ξ that intersect with $\mathcal{B}(p_i(t_k), R)$. Let

$$t_i^{fl}(t_k) := \min_{t>t_k}\{p_i(t) \notin \mathcal{B}(p_i(t_k), R)\}$$

be the first time that robot i leaves its sensing area $\mathcal{B}(p_i(t_k), R)$. Then, define

$$S_i(t_k) := \hat{Q}\big(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)])\big) \tag{20}$$

as the set of regions traversed by robot i within $\mathcal{B}(p_i(t_k), R)$ until it leaves it at $t_i^{fl}(t_k)$. Moreover, for each $\hat{X}_l \in S_i(t_k)$, the braking area of robot i is contained in $\psi_i(\hat{X}_l)$. Then, we define

$$Res_i(\hat{X}_l) := M_i(\hat{X}_l) \tag{21}$$

as the set of *reserved regions* by robot i in order to safely brake when robot i is within $\hat{X}_l$. According to (19), the time interval that robot i occupies the region $\hat{X}_l \in S_i(t_k)$ is given by $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$. In addition, by the continuity of $\boldsymbol{p}_i([t_k, \infty))$, one can conclude that $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$ is given by one or several disjoint time interval(s) of the form $[a, b), a < b$. Supposing that

$$\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l) = \cup_{l=1}^{m}[a_l, b_l), \tag{22}$$

where m is the number of disjoint intervals in $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$. Denote by $\mathcal{T}_i(\hat{X}_l)$ the time interval that robot i reserves the area $Res_i(\hat{X}_l)$. Then, it can be over-approximated by

$$\mathcal{T}_i(\hat{X}_l) := \cup_{l=1}^{m}[a_l, b_l + T_i^{\text{br}}), \tag{23}$$

where T_i^{br} is defined in (13). Then, we have the following definition.

Definition 4. We say that there is a *spatial-temporal conflict* between robot i and j at time t_k if $\exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k)$ such that $Res_i(\hat{X}_l) \cap Res_j(\hat{X}_{l'}) \neq \emptyset$ and $\mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_j(\hat{X}_{l'}) \neq \emptyset$.

Based on Definition 4, we define the set of conflict neighbors of robot i at time t_k, denoted by $\tilde{\mathcal{N}}_i(t_k)$, as

$$\begin{aligned}\tilde{\mathcal{N}}_i(t_k) := \{j \in \mathcal{N}_i(t_k) : \exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k) \text{ s.t.} \\ Res_i(\hat{X}_l) \cap Res_j(\hat{X}_{l'}) \neq \emptyset \wedge \mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_j(\hat{X}_{l'}) \neq \emptyset\}.\end{aligned} \tag{24}$$

Then, we have the following Proposition.

Proposition 2. For robot i, if $\tilde{\mathcal{N}}_i(t_k) = \emptyset$, then one has

i) $\psi_i(p_i(t)) \cap \psi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t_k), \forall t \in [t_k, t_i^{fl}(t_k)]$;

ii) $\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t_k), \forall t \in [t_k, t_i^{fl}(t_k)]$.

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

Algorithm 1 *conflictDetection*

Require: $S_j(t_k), \Gamma(\boldsymbol{p}_j([t_k, t_j^{fl}(t_k)]), \hat{X}_l), \forall \hat{X}_l \in S_j(t_k)$ for each $j \in \mathcal{N}_i^+(t_k)$.
Ensure: Set of conflict neighbors $\tilde{\mathcal{N}}_i(t_k)$.
1: Initialize $\tilde{\mathcal{N}}_i(t_k) = \emptyset$.
2: Compute $Res_j(\hat{X}_l), \mathcal{T}_j(\hat{X}_l)$ for each $j \in \mathcal{N}_i^+(t_k), \hat{X}_l \in S_j(t_k)$,
3: **for** $j \in \mathcal{N}_i(t_k)$ **do**
4: **if** $\exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k)$ s.t. $Res_i(\hat{X}_l) \cap Res_j(\hat{X}_{l'}) \neq \emptyset \wedge \mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_i(\hat{X}_{l'}) \neq \emptyset$ **then**
5: $\tilde{\mathcal{N}}_i(t_k) = \tilde{\mathcal{N}}_i(t_k) \cup \{j\}$,
6: **end if**
7: **end for**

3.2.5 Determine planning order

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

The number of neighbors and conflict neighbors of robot i at time t_k are given by $|\mathcal{N}_i(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)|$, respectively. Then, we have the following definition.

Definition 5. We say that robot i has *advantage* over robot j at time t_k if $\tilde{\mathcal{N}}_j(t_k) \neq \emptyset$ and

1) $|\mathcal{N}_i(t_k)| > |\mathcal{N}_j(t_k)|$; OR

2) $|\mathcal{N}_i(t_k)| = |\mathcal{N}_j(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)| > |\tilde{\mathcal{N}}_j(t_k)|$.

This project is funded by the European Union G.A No 101016906

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

For each neighbor $j \in \mathcal{N}_i(t_k)$, if j is in **Emerg** mode (and thus will be viewed as a static obstacle) or $\tilde{\mathcal{N}}_j(t_k) = \emptyset$, then robot j has higher priority (lines 3-5). Otherwise, robot j has higher priority in motion planning if robot j has advantage over robot i (lines 6-8). However, for the special case, *i.e.,* $|\mathcal{N}_i(t_k)| = |\mathcal{N}_j(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)| = |\tilde{\mathcal{N}}_j(t_k)|$, neither robot i nor j has advantage over the other. In this case, the priority is determined by the initially uniquely assigned priority score for each robot i (*i.e.,* $P_i^0 \neq P_j^0, \forall i, j$). Denote by P_i^0 the priority score of robot i. We say that robot i has *priority* over j if $P_i^0 > P_j^0$ (lines 9-11).

Algorithm 2 *planningOrderAssignment*

Require: $\mathcal{N}_j(t_k), \tilde{\mathcal{N}}_j(t_k), P_j^0, j \in \mathcal{N}_i(t_k) \cup \{i\}$.
Ensure: Set of higher priority neighbors $\mathcal{Y}_i(t_k)$.
1: Initialize $\mathcal{Y}_i(t_k) = \emptyset$.
2: **for** $j \in \mathcal{N}_i(t_k)$ **do**,
3: **if** j is in **Emerg** mode or $\tilde{\mathcal{N}}_j(t_k) = \emptyset$ **then**,
4: $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
5: **else**
6: **if** j has advantage over i **then**,
7: $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
8: **else**
9: **if** neither robot i nor j has advantage over the other and $P_j^0 > P_i^0$ **then**,
10: $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
11: **end if**
12: **end if**
13: **end if**
14: **end for**

Proposition 3 (Deadlock-free in planning order assignment)**.** The planning order assignment rule given in Algorithm 2 will result in no cycles, *i.e.,* $\nexists \{q_m\}_1^{\hat{K}}, \hat{K} \geq 2$ such that $q_{\hat{K}} \in \mathcal{Y}_{q_1}(t_k)$ and $q_{m-1} \in \mathcal{Y}_{q_m}(t_k), \forall m = 2, \ldots, \hat{K}$.

3.2.6 Motion planning

Before starting to plan, robot i needs to wait for the updated plan from the set of neighbors that have higher priority than robot i (*i.e.,* $j \in \mathcal{Y}_i(t_k)$) and consider them as moving obstacles. For those neighbors $j \in \mathcal{Y}_i(t_k)$, denote by $\boldsymbol{\xi}_j^+([t_k, \infty))$ (correspondingly $\boldsymbol{p}_j^+([t_k, \infty))$) the updated trajectory (position trajectory) of robot j at time t_k and let $t_j^{fl+}(t_k)$ be the first time that robot j leaves its sensing area $\mathcal{B}(p_j(t_k), R)$ according to the updated position trajectory $\boldsymbol{p}_j^+([t_k, \infty))$. Then, similar to (20), one can define

$$S_j^+(t_k) := \hat{Q}(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]))$$

as the updated set of regions traversed by robot j within $\mathcal{B}(p_j(t_k), R)$. For each $\hat{X}_l \in S_j^+(t_k)$, the time interval that robot j occupies the region $\hat{X}_l$ is given by $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l)$. Supposing that $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l) =$

This project is funded by the European Union G.A No 101016906

$\cup_{l=1}^{\hat{m}}[\hat{a}_l, \hat{b}_l)$, where $\hat{m}$ is the number of disjoint intervals in $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l)$. Denote by $\mathcal{T}_j^+(\hat{X}_l)$ the time interval that robot j reserves the area $Res_j(\hat{X}_l)$ according to the updated position trajectory $\boldsymbol{p}_j^+([t_k, \infty))$. Similarly to (23), it can be over-approximated by $\mathcal{T}_j^+(\hat{X}_l) := \cup_{l=1}^{\hat{m}}[\hat{a}_l, \hat{b}_l + T_i^{\text{br}}]$.

Then, the trajectory planning problem (TPP) can be formulated as follows:

$$\text{find} \quad \boldsymbol{\xi}_i([t_k, \infty)), \tag{25a}$$

$$\text{subject to } (5) \text{ and } (6), \tag{25b}$$

$$\boldsymbol{\xi}_i([0, t_k] \cup [t_k, \infty)) \models \varphi_i, \tag{25c}$$

$$\psi_i(\texttt{proj}_l(\boldsymbol{\xi}_i([t_k, \infty)))) \subset \mathbb{F}, \tag{25d}$$

$$\psi_i(\texttt{proj}_l(\xi_i(t))) \cap Res_j(\hat{X}_l) = \emptyset,\ t \in \mathcal{T}_j^+(\hat{X}_l), \quad \forall j \in \mathcal{Y}_i(t_k), \forall \hat{X}_l \in S_j^+(t_k). \tag{25e}$$

Constraints (25d) and (25e) guarantee respectively that there will be no robot-obstacle collisions and inter-robot collisions for robot i.

When relatively simple specifications, *e.g.,* reach-avoid type of tasks, are considered, various existing optimization toolboxes, *e.g.,* IPOPT [37], ICLOCS2 [38], and algorithms, *e.g.,* the configuration space-time search [39], the Hamilton-Jacobian reachability-based trajectory planning [40], RRT^X [41], and the fast robot motion planner [42] can be utilized to solve (25). However, if the specifications are complex LTL formulas, the constraint (25c) is not easy to be verified online using the methods mentioned above (the PBA $\mathcal{P}_i$ can be used to verify (25c), however, it can not deal with the spatial-temporal collision avoidance constraint (25e) at the same time). Recently, an online RRT-based algorithm is proposed in [43] to generate local paths that guarantee the satisfaction of the global specification. Motivated by this work, an online motion replanning structure is proposed in this paper, which contains a local and a global trajectory generation algorithms.

Before proceeding, the following notations are required. Denote by

$$NI_i(t_k) := \{Res_j(\hat{X}_l), \mathcal{T}_j^+(\hat{X}_l), \forall \hat{X}_l \in S_j^+(t_k), \forall j \in \mathcal{Y}_i(t_k)\}$$

the set with respect to robot i which contains all local trajectory information of higher priority neighbors at time t_k. Given a state $s_i \in \hat{\mathbb{X}}_i$, define the function $B_i : \hat{\mathbb{X}}_i \to 2^{S_i}$ as a map from a state $s_i \in \hat{\mathbb{X}}_i$ to a subset of valid Büchi states which correspond to s_i (*i.e.,* $B_i(s_i)) \subseteq \beta_{\mathcal{P}_i}(s_i)$). Function B_i is used to capture the fact that given a partial trajectory, not all Büchi states in $\beta_{\mathcal{P}_i}$ are valid. At the task activation time 0, one has $B_i(\xi_i(0)) = \beta_{\mathcal{P}_i}(\xi_i(0))$. During the online implementation, $B_i(\xi_i(t_k)), t_k > 0$ is updated by

$$B_i(\xi_i(t_k)) = \beta_{\mathcal{P}_i}(\xi_i(t_k)) \cap \{B_i(\xi_i(t_{k-1})) \cup \texttt{Post}(B_i(\xi_i(t_{k-1})))\} \tag{26}$$

where

$$\texttt{Post}(B_i(\xi_i(t_{k-1}))) = \cup_{s_i \in B_i(\xi_i(t_{k-1}))} \texttt{Post}(s_i). \tag{27}$$

Firstly, the local trajectory generation process is outlined in Algorithm 3. At each time instant t_k, Algorithm 3 takes the state $\xi_i(t_k)$ of robot i, $B_i(\xi_i(t_k))$, $\texttt{Post}(B_i(\xi_i(t_k)))$, the offline computed PBA $\mathcal{P}_i$, potential function $V_{\mathcal{P}_i}$, the set of static obstacles $\mathbb{O}$, and the local trajectory information of higher priority neighbors, *i.e.,* $NI_i(t_k)$

This project is funded by the European Union G.A No 101016906

Algorithm 3 *localTrajectoryGeneration*

Require: $\xi_i(t_k), B_i(\xi_i(t_k)), \texttt{Post}(B_i(\xi_i(t_k))), \mathcal{P}_i, V_{\mathcal{P}_i}, \mathbb{O}$, and $NI_i(t_k)$.
Ensure: A local transition system $\mathcal{T}_{c,i}^L$ and a leaf node ξ_i^f.

1: Initialize $\mathcal{T}_{c,i}^L = (S_i^L, S_{i,0}^L, AP_{\varphi_i}, \rightarrow_{c,i}^L, L_{c,i})$ and $\xi_i^f = \emptyset$, where $S_i^L = S_{i,0}^L = \xi_i(t_k)$ and $\rightarrow_{c,i}^L = \emptyset$, $\texttt{St}(\xi_i(t_k)) = 0$.
2: **for** $k = 1, \ldots, N_i^{\max}$ **do**,
3: $\quad \xi_s \leftarrow$ *generateSample*$(SA_i(t_k))$,
4: $\quad \xi_n \leftarrow$ *nearest*(S_i^L, ξ_s),
5: $\quad$ Solve the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$, which returns (ξ_r, u_i^*),
6: $\quad B_i(\xi_r) \leftarrow \beta_{\mathcal{P}_i}(\xi_r) \cap \{B_i(\xi_n) \cup \texttt{Post}(B_i(\xi_n))\}$,
7: $\quad$ **if** $B_i(\xi_r) \neq \emptyset \wedge V_{\mathcal{T}_{c,i}}(\xi_r, B_i(\xi_r)) < \infty$ **then**,
8: $\quad\quad \mathbb{O}_i(t_k) \leftarrow$ *updateObstacle*$(\mathbb{O}, NI_i(t_k), [\texttt{St}(\xi_n)\tau_s, (\texttt{St}(\xi_n)+1)\tau_s])$,
9: $\quad\quad$ **if** $\texttt{dist}(\texttt{proj}_l([\xi_n, \xi_r]), \mathbb{O}_i(t_k)) \geq D_i^{\text{br}}$ **then**,
10: $\quad\quad\quad S_i^L \leftarrow S_i^L \cup \{\xi_r\}; \rightarrow_{c,i}^L = \rightarrow_{c,i}^L \cup \{\xi_n \xrightarrow{u_i^*} \xi_r\}$,
11: $\quad\quad\quad \texttt{St}(\xi_s) \leftarrow \texttt{St}(\xi_n) + 1$,
12: $\quad\quad$ **end if**
13: $\quad$ **end if**
14: $\quad$ **if** $\texttt{proj}_l(\xi_r) \notin \mathcal{B}(\texttt{proj}_l(\xi_i(t_k)), R)$, **then**
15: $\quad\quad k = N_i^{\max} + 1$,
16: $\quad\quad \xi_i^f \leftarrow \xi_r$,
17: $\quad$ **end if**
18: **end for**

as input. The output is a local CTS $\mathcal{T}_{c,i}^L := (S_i^L, S_{i,0}^L, AP_{\varphi_i}, \rightarrow_{c,i}^L, L_{c,i})$ that is constructed incrementally and a leaf node ξ_i^f.

In line 5, the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$ is given by

$$\min_{u_i \in \mathbb{U}_i} \quad \|\xi_r - \xi_s\|, \tag{28a}$$

$$\text{subject to } \xi_i(0) = \xi_n, \tag{28b}$$

$$\xi_n + \int_0^t F_i(\xi_i(s), u_i) ds \in \mathbb{X}_i, \forall t \in [0, \tau_s], \tag{28c}$$

$$\xi_r = \xi_n + \int_0^{\tau_s} F_i(\xi_i(s), u_i) ds, \tag{28d}$$

and the optimal solution is u_i^*.

The root state of $\mathcal{T}_{c,i}^L$ is robot i's state $\xi_i(t_k)$. The function $\texttt{st} : S_i^L \rightarrow \mathbb{N}$ maps a state $x \in S_i^L$ to the number of time steps needed to reach the root state $\xi_i(t_k)$. Initially, $\mathcal{T}_{c,i}^L$ contains one state $\xi_i(t_k)$ and 0 transitions, *i.e.,* $\texttt{st}(\xi_i(t_k)) = 0$, and the leaf node $\xi_i^f = \emptyset$ (line 1). In each iteration (lines 2-18), a new state ξ_s is generated randomly from the set $SA_i(t_k)$ using the *generateSample* procedure (line 3), where $SA_i(t_k)$ is the sampling

This project is funded by the European Union G.A No 101016906

area around robot i, given by

$$SA_i(t_k) := \{(p, \zeta) \in \mathbb{X}_i : p \in \mathcal{B}(\mathtt{proj}_l(\xi_i(t_k)), R + \eta)\},$$

where $\eta > 0$ is an offline chosen constant, which guarantees that there exists $s \in SA_i(t_k)$ such that $\mathtt{proj}_l(s) \notin \mathcal{B}(\mathtt{proj}_l(\xi_i(t_k)), R)$. This condition is essential for checking the terminal condition (line 15). The *nearest* function (line 4) is a standard RRT primitive [33] which returns the nearest state in S_i^L to the new sample ξ_s. Then, one further finds, within the set of states that are reachable from ξ_n at time τ_s, the closest one to the new sample ξ_s, *i.e.,* ξ_r, and the corresponding input u_i^* (line 5). Here, τ_s is the sampling interval (the same one used for constructing the transition system $\mathcal{T}_i$ in Section 3.2.1). Once ξ_r is obtained, we further compute the subset of valid Büchi states which correspond to ξ_r, *i.e.,* $B_i(\xi_r)$, according to (26) (line 6). After that, if both conditions $B_i(\xi_r) \neq \emptyset$ and $V_{\mathcal{T}_{c,i}}(\xi_r, B_i(\xi_r)) < \infty$ are satisfied (which guarantees that there exists a path, starting from ξ_r, that reaches a self-reachable accepting state of $\mathcal{P}_i$), obstacles that appear during the time interval $[\mathtt{st}(\xi_n)\tau_s, (\mathtt{st}(\xi_n) + 1)\tau_s]$ are added into the workspace using the function *updateObstacles* (Algorithm 4). Finally, the state ξ_r is added into S_i^L and the transition relation $\xi_n \xrightarrow{u_i^*} \xi_r$ is added into $\rightarrow_{c,i}^L$ if the distance between the line segment $\mathtt{proj}_l([\xi_n, \xi_r])$ and the obstacles is no less than the braking distance of robot i (lines 9-10), and then the time step needed for ξ_r to reach the root state $\xi_i(t_k)$ is recorded (line 11). The algorithm is terminated when the local sampling tree reaches the outside of the sensing area of robot i, and the leaf node ξ_i^f is then given by the corresponding state ξ_r (line 14-17).

Algorithm 4 *updateObstacle*

Require: $\mathbb{O}$, $NI_i(t_k)$ and a time interval $[t_1, t_2]$.
Ensure: $\mathbb{O}_i(t_k)$.
1: $\mathbb{O}_i(t_k) \leftarrow \mathbb{O}$,
2: **for** $j \in \mathcal{Y}_i(t_k)$, **do**
3: **for** $\hat{X}_l \in S_j^+(t_k)$, **do**
4: **if** $\mathcal{T}_j^+(\hat{X}_l) \cap [t_k + t_1, t_k + t_2] \neq \emptyset$, **then**
5: $\mathbb{O}_i(t_k) \leftarrow \mathbb{O}_i(t_k) \cup (Res_j(\hat{X}_l) \cap SA_i(t_k))$,
6: **end if**
7: **end for**
8: **end for**

After the local CTS $\mathcal{T}_{c,i}^L$ is obtained, we further need to find a path, starting from the leaf node ξ_i^f, that reaches one of the maximal self-reachable accepting states $F_{p,i}^*$ of $\mathcal{P}_i$. Define

$$P_i(\xi_i^f) := \cup_{s_i \in B_i(\mathcal{T}_{c,i}^L)}(\xi_i^f, s_i)$$

as the set of states in the PBA $\mathcal{P}_i$ that correspond to ξ_i^f. Then, the global trajectory generation process is outlined in Algorithm 5. Algorithm 5 takes the set $P_i(\xi_i^f)$ and the the PBA $\mathcal{P}_i$ as input. It first finds the state p_i^* in $P_i(\xi_i^f)$ that has the minimum potential (line 1). Then, if $p_i^* \in F_{p,i}^*$, the function *DijksCycle*($\mathcal{P}_i$, *source*) (defined in [10]) is used to compute a shortest cycle from the *source* state back to itself (line 3); otherwise, the function *DijksTargets*($\mathcal{P}_i$, *source*, *targets*) (defined in [10]) is used to compute a shortest path in $\mathcal{P}_i$ from

This project is funded by the European Union G.A No 101016906

Algorithm 5 *globalTrajectoryGenration*

Require: $P_i(\xi_i^f)$ and $\mathcal{P}_i$.
Ensure: a path $\boldsymbol{\rho}_i$.
1: $p_i^* = \min_{p_i \in P_i(\xi_i^f)}\{V_{\mathcal{P}_i}(p_i)\}$,
2: **if** $p_i^* \in F_{p,i}^*$ **then**
3: $\quad \boldsymbol{p}_i \leftarrow$ *DijksCycle*$(\mathcal{P}_i, p_i^*)$,
4: **else**
5: $\quad \boldsymbol{p}_i \leftarrow$ *DijksTargets*$(\mathcal{P}_i, p_i^*, F_{p,i}^*)$,
6: **end if**
7: $\boldsymbol{\rho}_i = \mathtt{pj}_{\hat{\mathbb{X}}_i}(\boldsymbol{p}_i)$,

"*source*" state to one of the state belonging to the set "*targets*" (line 5). The required path is then the projection of $\boldsymbol{p}_i$ on the state space of $\mathcal{T}_{c,i}$ (line 7).

It is possible that after the maximum number of iterations (*i.e.,* $N_i^{\max}$), there exists no local path that reaches the outside of the sensing area of robot i. In this case, the TPP (25) is considered infeasible, robot i switches to **Emerg** mode and the braking controller (10) is applied. When robot i is in **Emerg** mode, it will continue monitoring the environment (by updating $\mathcal{T}_{c,i}^L$). Once a feasible local path is found, it will switch back to **Free** mode. The whole motion coordination process is summarized in Algorithm 6.

The following result shows hat safety is guaranteed under all circumstances.

Theorem 1 (**Safety).** *If the sensing radius of the robots satisfies* $R > 2\max_{i\in\mathcal{V}}\{D_i^{br} + \Delta \max_{p_i\in\mathbb{W}}\{\|\dot{p}_i\|\}\}$*, where* Δ *is the conflict detection interval, then the resulting real-time plan of Algorithm 6 guarantees that there will be no obstacle-robot and inter-robot collisions for robot* i*.*

Theorem 2 (**Correctness).** *Let* $\boldsymbol{\xi}_i^{rt}([0,\infty))$ *be the real-time moving trajectory of robot* i*. Then,* $\boldsymbol{\xi}_i^{rt}([0,\infty)) \models \varphi_i$ *if robot* i *never enters the* ***Emerg*** *mode or stays in* ***Emerg*** *mode for a finite time.*

The detailed analysis and proof can be found in [16] and neglected here.

3.3 Numerical Validation

Consider a MRS consisting of $N = 4$ robots with unicycle dynamics, and the velocity and input constraints. For robots $i \in \{1, 2\}$, one has that $v_{i,\max} = 1\mathrm{m/s}$, $\omega_{i,\max} = 0.5\mathrm{rad/s}$, and $a_{i,\max} = 2\mathrm{m/s^2}$. For robots $i \in \{3, 4\}$, one has that $v_{i,\max} = 1\mathrm{m/s}$, $\omega_{i,\max} = 0.5\mathrm{rad/s}$, and $a_{i,\max} = 1.5\mathrm{m/s^2}$. Then, using the braking controller, one can get that $T_i^{\mathsf{br}} = 0.5\mathrm{s}, D_i^{\mathsf{br}} = 0.25\mathrm{m}, i = 1, 2$ and $T_i^{\mathsf{br}} = 0.5\mathrm{s}, D_i^{\mathsf{br}} = 0.33\mathrm{m}, i = 3, 4$. The sensing radius of each robot is $R = 3.5\mathrm{m}$ and the conflict detection period is $\Delta = 0.1\mathrm{s}$. Then one has that $R > 2\max_{i\in\mathcal{V}}\{D_i^{\mathsf{br}} + \Delta v_{i,\max}\}$.

As shown in Fig. 5, the workspace $\mathbb{W}$ we consider is a $20 \times 20\mathrm{m}^2$ square, where the gray areas, marked as O_1, O_2, O_3, represent three static obstacles, the colored small circles represent the initial position of the robots, and the light blue areas, marked as $T_1, T_2, \ldots, T_5$, represent a set of target regions in the workspace. Initially, $\theta_i = 0, v_i(0) = 0, \forall i$.

Each robot is assigned to persistently survey two of the target regions in the workspace. This relates to

This project is funded by the European Union G.A No 101016906

Algorithm 6 *motionCoordination*

Require: (offline) $M_i(\hat{X}_l), \forall \hat{X}_l \in \Xi, \mathcal{T}_{c,i}, \mathrm{B}_i, \mathcal{P}_i$, and $V_{\mathcal{P}_i}$.
Ensure: Real-time plan $\boldsymbol{\xi}_i^+([t_k, \infty)), t_k \geq 0$.

Initialize: $\boldsymbol{\xi}_i^-([0,\infty)) \leftarrow \boldsymbol{\xi}_i^0$, $B_i(\xi_i(0)) \leftarrow \beta_{\mathcal{P}_i}(\xi_i(0))$, $\mathtt{Post}(B_i(\xi_i(0))) \leftarrow \cup_{s_i \in B_i(\xi_i(0))} \mathtt{Post}(s_i)$, and Robot i is in **Free** mode.
while $t_k > 0$ and φ_i is not completed **do**,
  Compute $B_i(\xi_i(t_k))$ and $\mathtt{Post}(B_i(\xi_i(t_k)))$ according to (26) and (27),
  $\tilde{\mathcal{N}}_i(t_k) \leftarrow$ *conflictDetection*(),
  if $\tilde{\mathcal{N}}_i(t_k) \neq \emptyset$ **then**
    Robot i switches to **Busy** mode,
    $\mathcal{Y}_i(t) \leftarrow$ *planningOrderAssignment*(),
    $(\mathcal{T}_{c,i}^L, \xi_i^f) \leftarrow$ *localTrajectoryGeneration*(),
    if $\xi_i^f \neq \emptyset$, **then**
      $\boldsymbol{\rho}_i \leftarrow$ *globalTrajectoryGeneration*(),
      $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow$ *DijksTargets*$(\mathcal{T}_{c,i}^L, \xi_i(t_k), \xi_i^f) \uplus \boldsymbol{\rho}_i$,
      Robot i switches to **Free** mode,
    else
      Robot i switches to **Emerg** mode,
      $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i^{\mathrm{br}}(\xi_i(t_k))$,
      $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow \boldsymbol{\xi}_i(\xi_i(t_k), \boldsymbol{u}_i, [0, \infty))$,
    end if
  else
    $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow \boldsymbol{\xi}_i^-([t_k, \infty))$,
  end if
end while

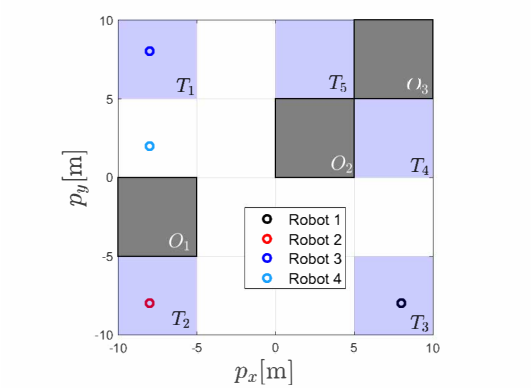


Figure 5: The workspace for the group of robots in Example 1.

agronomic tasks that require persistent monitoring, for example, to check and record the quality of grapes in the field. The LTL specification for each robot is given by

This project is funded by the European Union G.A No 101016906

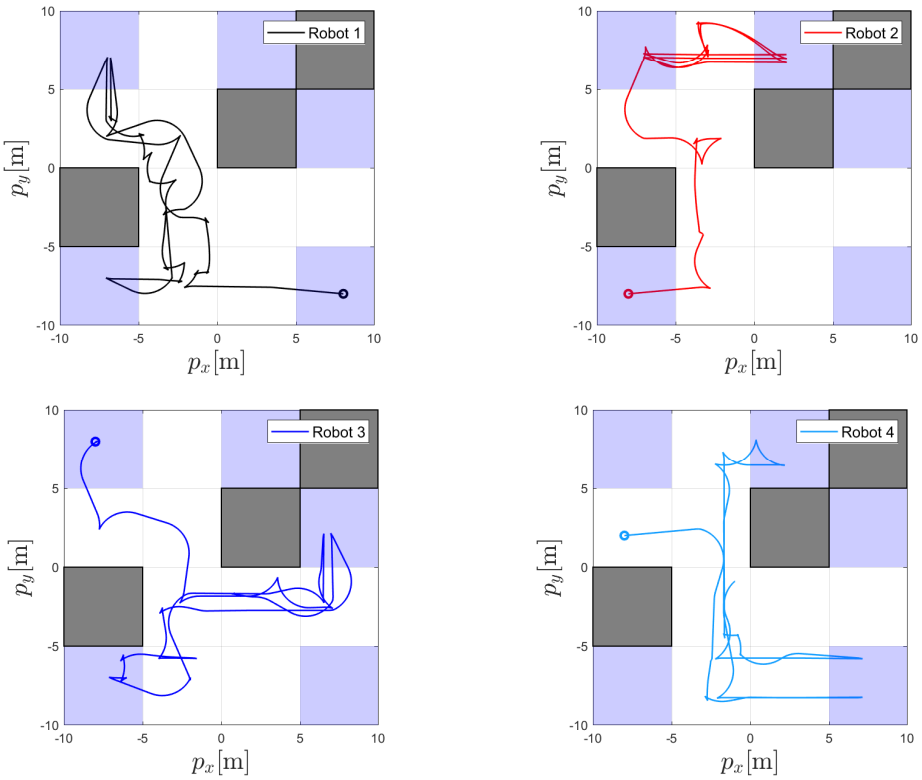


Figure 6: The real-time position trajectory of each robot, where the small circle represents the initial position of each robot.

- $\varphi_1 = \square(\mathbb{W} \wedge \neg\mathbb{O}) \wedge \square\Diamond T_1 \wedge \square\Diamond T_2$,
- $\varphi_2 = \square(\mathbb{W} \wedge \neg\mathbb{O}) \wedge \square\Diamond T_1 \wedge \square\Diamond T_5$,
- $\varphi_3 = \square(\mathbb{W} \wedge \neg\mathbb{O}) \wedge \square\Diamond T_2 \wedge \square\Diamond T_4$,
- $\varphi_4 = \square(\mathbb{W} \wedge \neg\mathbb{O}) \wedge \square\Diamond T_3 \wedge \square\Diamond T_5$,

where $\mathbb{O} = O_1 \cup O_2 \cup O_3$. Firstly, a rough cell decomposition is given over the workspace $\mathbb{W}$. Then, for each cell $X_l \subset \mathbb{W} \backslash \mathbb{O}$, a grid representation with grid size 0.5m is implemented as a finer workspace discretization. The NBA B_i associated with $\varphi_i, \forall i$ has 3 states and 8 edges by [26]. The CTS $\mathcal{T}_i$ and the PBA $\mathcal{P}_i$ for each robot are constructed using LTLCon toolbox [44]. Finally, the map M_i and potential function for each $\mathcal{P}_i$ are computed.

Initially, each robot i finds a safely satisfying trajectory $\boldsymbol{\xi}_i^0$ (in the form of prefix-suffix). Then, the motion coordination algorithm (Algorithm 6) is implemented for each robot during online execution. The real-time position trajectory of each robot is depicted in Fig. 6 (a)-(d), respectively. It can be seen that during the simulation time $[0, 120]\text{s}$, the monitoring task of each robot is satisfied at least once. Fig. 7 shows the evolution of the position trajectories of all robots with respect to time. One can see that the real-time

This project is funded by the European Union G.A No 101016906

position trajectories are collision-free. During the simulation time, conflicts are detected in total 11 times. Whenever conflicts are detected, the planning order assignment module and the trajectory planning module are activated to resolve the conflicts. In Fig. 8 (a)-(d), 4 cases of local motion replanning are depicted, respectively. In each of the subfigures, the circle, marked as dotted line, represents the sensing area of the robot that replans, the dashed lines represent the planned local position trajectories of the robots, the colored small squares represent its (over-approximated) braking areas, the yellow squares represent the conflict region, and the solid line represents the replanned local position trajectory. A video recording of the real-time implementation can be found here.

The real-time evolution of velocity v_i and inputs (a_i, ω_i) of each robot i are plotted in Fig. 9 and Fig. 10, respectively. One can see that the velocity constraints and the input constraints are satisfied by all robots at any time.

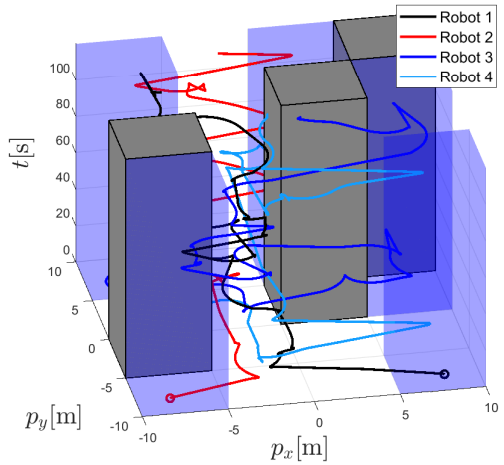


Figure 7: The evolution of the position trajectories with respect to time.

Finally, we report the computation times of this example. All simulations are carried out in Matlab 2018b on a Dell laptop with Windows 10, Intel i7-6600U CPU2.80GHz and 16.0 GB RAM. For the 11 local replannings, the average computation time is 1.51s and the maximum computation time is 3.30s. The computation of the global trajectory (Algorithm 5) is quite fast (less than 0.1s). Note that after the local and global trajectories are obtained, the toolbox ICLOCS2 [38] is further employed for trajectory smoothness. In addition, it is worth to point out that we have tried 10 different sets of initial states and specifications for the group of robots, and we find that the initial states and the specifications have little effect on the average time and maximum time of local replanning. There is no change of order of magnitude in the computation time for all trials.

This project is funded by the European Union G.A No 101016906

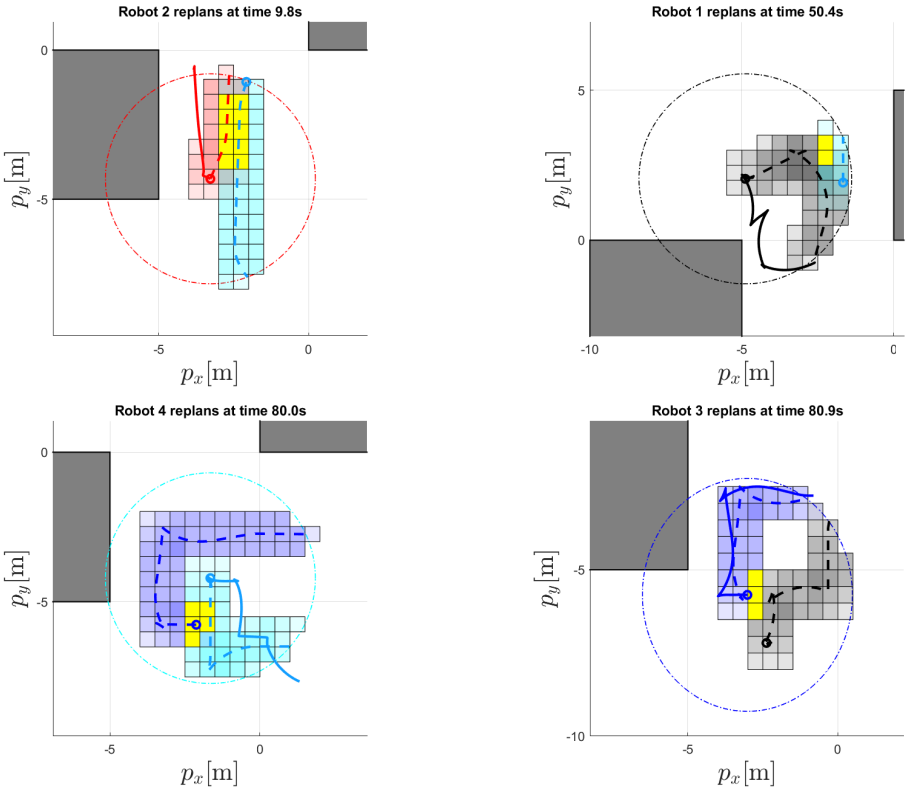


Figure 8: The local motion replanning between different conflict robots. The circle, marked with a dotted line, represents the sensing area of the robot that replans, the dashed lines represent the planned local position trajectories of the robots, the colored small squares represent its (over-approximated) braking areas, the yellow squares represent the conflict region, and the solid line represents the replanned local position trajectory.

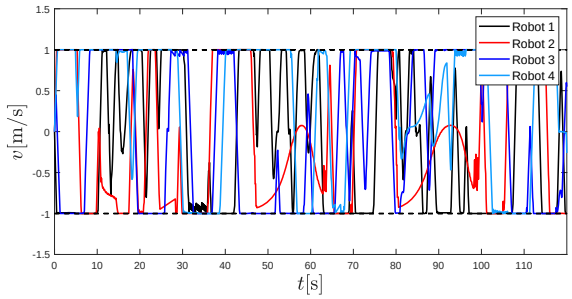


Figure 9: The real-time evolution of velocity v_i for each robot, where $v_{i,\max} = 1\mathrm{m/s}, i \in \{1, 2, 3, 4\}$.

This project is funded by the European Union G.A No 101016906

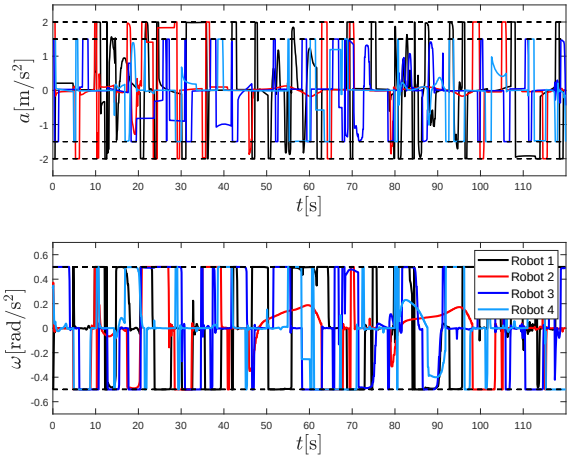


Figure 10: The real-time evolution of inputs (a_i, ω_i) for each robot, where $a_{i,\max} = 2\mathrm{m/s}^2, \omega_{i,\max} = 0.5\mathrm{rad/s}, i \in \{1,2\}$ and $a_{i,\max} = 1.5\mathrm{m/s}^2, \omega_{i,\max} = 0.5\mathrm{rad/s}, i \in \{3,4\}$.

3.4 MRMC under local LTL specifications in a dynamic environment

In previous sections, MRSs under local LTL specifications are studied and a distributed motion coordination algorithm is proposed to resolve conflicts. Yet the environment is static and local communication between robots is required for conflict resolution. In agricultural scenarios as CANOPIES project, the environment is constantly changing and farmers and farm vehicles are moving around and may not be able to communicate with robots. Now we develop reactive planning and control algorithms for MRSs under LTL specifications by addressing dynamic environments, which present a non-trivial challenge.

Note that the initially planned trajectory for each robot i does not account for the motion of other robots or changing of the environment. Thus, online replanning is necessary during the implementation. Two distinct cases are considered for online replanning, one is based on local communication between robots and the other assumes no communication between robots. In the former case, each robot has access to information (i.e., broadcast data from other robots) within its sensing region while the latter case does not.

3.4.1 Local communication case

In this section, we consider that each robot can identify conflicts within its sensing region using the information broadcast by other robots. Before proceeding, the definition of a conflict is needed.

Let $p_i(t) = [x_i(t), y_i(t)]^T$ be the position of robot i at time t. Denote by $p_i([t, t+\Delta_i])$ the local trajectory of robot i, where $\Delta_i = \min_{t' \geq t}\{p_i(t') \notin \mathcal{B}_i(p_i(t), R_i)\}$. We say there is a *conflict* between robot i and j at time t if there exists a region $X_k \in \Phi$ such that $p_i([t, t+\Delta_i]) \cap X_k \neq \emptyset$ and $p_j([t, t+\Delta_j]) \cap X_k \neq \emptyset$. This means that the local trajectories of robots i and j pass through the same region X_k.

Using the local trajectory information $p_j([t, t+\Delta_j])$ broadcast by the neighboring robots $j \in \mathcal{N}_i(t) := \{j : \|p_i(t) - p_j(t)\| \leq R_i\}$, each robot i can detect conflicts. Once conflicts are detected, online replanning is conducted to ensure that conflicts are avoided and the LTL task is satisfied. The online replanning procedure

This project is funded by the European Union G.A No 101016906

consists of a local and a global trajectory generation algorithms.

The local trajectory generation algorithm is detailed in Algorithm 7. Whenever conflicts are detected among robots, a priority hierarchy is first created to coordinate the planning order. Subsequently, a sampling-based algorithm is employed to create a local collision-free trajectory.

Algorithm 7 localTrajectoryGeneration

Require: $\xi_i, \mathcal{B}_i, \mathcal{P}_i, V_{\mathcal{P}_i}, \mathbb{O}, \mathbb{O}_{conf}$

Ensure: A local transition system $\mathcal{T}_i^L$ and a leaf node ξ_i^f.

1: Initialize $\mathcal{T}_i^L = (S_i^L, S_{i,0}^L, \mathcal{U}_i, AP, \rightarrow_i^L, L)$ and $\xi_i^f = \emptyset$, where $S_i^L = S_{i,0}^L = \xi_i$ and $\rightarrow_i^L = \emptyset$.
2: **for** $k = 1, \ldots, N_i^{\max}$ **do**,
3: $\quad \xi_s \leftarrow$ *generateSample*(SA_i),
4: $\quad \xi_n \leftarrow$ *nearest*(S_i^L, ξ_s),
5: $\quad$ Solve the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$, which returns (ξ_r, u_i^*),
6: $\quad B_i(\xi_r) \leftarrow$ *trackBuchiStates*$(\mathcal{B}_i)$,
7: $\quad$ **if** $B_i(\xi_r) \neq \emptyset \wedge V_{\mathcal{P}_i}(\xi_r, B_i(\xi_r)) < \infty$ **then**,
8: $\quad\quad$ **if** $\mathrm{proj}_2([\xi_n, \xi_r])\, isObstaclesFree(\mathbb{O}, \mathbb{O}_{conf}) \wedge safeMotion(\xi_r, R_{safe})$ **then**,
9: $\quad\quad\quad S_i^L \leftarrow S_i^L \cup \{\xi_r\}; \rightarrow_i^L = \rightarrow_i^L \cup \{\xi_n \xrightarrow{u_i^*} \xi_r\}$,
10: $\quad\quad$ **end if**
11: $\quad$ **end if**
12: $\quad$ **if** $\mathtt{proj}_2(\xi_r) \notin \mathcal{B}(p_i, R_i)$, **then**
13: $\quad\quad k = N_i^{\max} + 1$,
14: $\quad\quad \xi_i^f \leftarrow \xi_r$,
15: $\quad$ **end if**
16: **end for**

Algorithm 7 starts with randomly sampling the expanded sensing region of robot i. Then by constructing a local transition system, a collision-free trajectory is synthesized that simultaneously guarantees the fulfillment of the LTL task φ_i. It takes as input the current state $\xi_i = (x_i, y_i, \theta_i)$ of robot i, the offline constructed NBA B_i and PBA $\mathcal{P}_i$, the offline computed potential function $V_{\mathcal{P}_i}$, the set of known static obstacles $\mathbb{O}$, and the set of conflict regions $\mathbb{O}_{conf}^t$ as input. Firstly, a local transition system $\mathcal{T}_i^L$ is initialized (line 1). At each iteration, a new state ξ_s is taken randomly from the sampling area $SA_i := \{(p, \theta) : p \in \mathcal{B}(p_i, R_i + \eta)\}$ (line 3), where $\eta > 0$ is an offline constant which ensures that one can sample outside of the sensing region $\mathcal{B}(p_i, R_i)$, where p_i is the current position of robot i. Then through the function *nearest* (S_i^L, ξ_s) an RRT primitive is applied, which returns the nearest state to ξ_s in S_i^L (line 4). At this stage an optimization problem $\mathcal{P}(\xi_n, \xi_s, \tau_s)$ is solved, so as to find the closest reachable state from the new sample ξ_s:

$$\min_{u_i \in \mathbb{U}_i} \|\xi_r - \xi_s\|$$

$$\text{subject to: } \xi_i(0) = \xi_n,$$

$$\xi_r = \xi_n + \int_0^{\tau_s} F_i(\xi_i(s), u_i)\, ds,$$

$$u_i \in \mathcal{U}_i,$$

This project is funded by the European Union G.A No 101016906

where τ_s represents the sampling time, while $F_i\left(\xi_i(s), u_i\right)$ describes the robot's dynamics (line 5). Once ξ_r is obtained, the corresponding subset of valid Büchi states $B_i\left(\xi_r\right)$ is computed using algorithm *trackBuchiState* (given in [31], Algorithm 1) (line 6). If both conditions $B_i\left(\xi_r\right) \neq \emptyset$ and $V_{\mathcal{P}_i}\left(\xi_r, B_i\left(\xi_r\right)\right) < \infty$ are satisfied (which ensures the existence of a path originating from ξ_r that leads to a self-reachable accepting state of $\mathcal{P}_i$), then a potential new state is considered. Such state ξ_r is added into S_i^L and the corresponding transition relation $\xi_n \overset{u_i^*}{\rightarrow} \xi_r$ is added into $\rightarrow_i^L$, if the path connecting ξ_n with ξ_r is obstacles free and the motion is considered safe. To verify these requirements, two new algorithms are designed in this work. The collision free requirement is checked through the algorithm *isObstaclesFree*, which computes the distance between the line segment $\mathrm{proj}_2\left([\xi_n, \xi_r]\right)$ and the set of obstacles $\mathbb{O} \cup \mathbb{O}_{conf}$. If the line segment $\mathrm{proj}_2\left([\xi_n, \xi_r]\right)$ is collision-free, we further check the safety of the motion using algorithm $safeMotion(\xi_r, R_{safe})$ (considering the base footprint of the robot), where R_{safe} is the safe distance that we specified a priori (lines 7-11). The algorithm halts when the local sampling tree extends beyond the sensing area, and the leaf node ξ_i^f is subsequently returned (as determined by the corresponding state ξ_r) (lines 12-15).

The global trajectory generation is similar to the initial trajectory generation, which uses the leaf node ξ_i^f (which is obtained by Algorithm 7) as input of the LTL core & planner, and replans the sequence of actions in order to accomplish the LTL task φ_i.

3.4.2 Communication-free case

A communication-free scenario is also considered in pursuit of extending the work to other real-world scenarios where reliable communications between robots may not be available. A reactive collision avoidance algorithm is designed in combination with the local planner.

Before proceeding, the concept of "trap state" is needed. *Trap states* are PBA states from which the Büchi acceptance condition cannot be fulfilled, i.e., states that cannot reach accepting states that appear infinitely often. For the PBA $\mathcal{P}_i$ of robot i, the set of all trap states is denoted $\mathbb{G}_i$. An algorithm for computing trap states is implemented in [45].

In this case, the online replanning is activated for robot i whenever obstacles (can be other robots and unknown static/moving obstacles) are detected within its sensing region $\mathcal{B}_i(p_i, R_i)$. Let $\mathbb{O}_{obs}^t$ be the set of moving obstacles detected at time t. In order to avoid trap states $\mathbb{G}_i$ (thus guaranteeing task feasibility), the set of known static obstacles $\mathbb{O}$, and the set of moving obstacles $\mathbb{O}_{obs}^t$ (thus guaranteeing safety), we

This project is funded by the European Union G.A No 101016906

consider a local model predictive controller:

$$\begin{aligned}\min_{u_i} \quad & \int_{t=0}^{T} (\xi_i(t) - X_{des})^T Q (\xi_i(t) - X_{des}) + u_i^T R u_i \\ & + (\xi(T) - X_{des})^T Q_N (\xi(T) - X_{des}) \\ & + \int_{t=0}^{T} w_{\mathbb{O}_i} \frac{1}{dist(\xi(t), \mathbb{O}_{obs}^t \cup \mathbb{O})} + w_{\mathbb{G}_i} \frac{1}{dist(\xi(t), \mathbb{G}_i)} \\ \text{subject to: } & \xi_i(t) = \xi_i(0) + \int_0^t F_i\left(\xi_i(s), u_i\right) ds, \\ & u_i \in \mathcal{U}_i, \\ & \xi_i(T) \in X_{des} \\ & \xi_i(0) = \xi_i,\end{aligned} \tag{29}$$

where T is the horizon and ξ_i is the current state. The first 2 terms of the cost function constitute a quadratic cost of both state and input with the matrices Q, R, Q_N being positive definite and $X_{des} \in \Phi$ denotes the goal state that one can choose based on the current satisfying trajectory (recall that the satisfying trajectory is discrete and have a prefix-suffix structure). The last term of the cost function considers both the distance between robot i and the obstacles $\mathbb{O}_{obs}^t \cup \mathbb{O}$ (which is updated at each time step) and the distance between robot i and the trap states $\mathbb{G}_i$, where the distance function is defined as $dist(x, A) := \inf_{y \in A}\{||x - y||\}$ and $w_{\mathbb{O}_i}, w_{\mathbb{G}_i}$ are the corresponding weight parameters. For instance if one prioritizes the satisfaction of the given LTL task rather than not collide with obstacles, the weight $w_{\mathbb{G}_i}$ can be set higher than $w_{\mathbb{O}_i}$.

3.5 Laboratory experiments

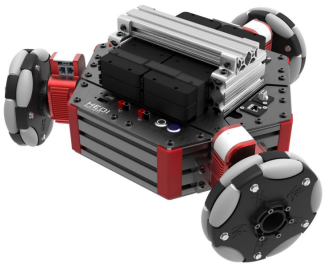
In this section, we present experimental studies to validate our results. The proposed strategies hold potential for application in precision agriculture, enabling farmworkers to collaborate effectively with robot teams in carrying out agronomic tasks, such as harvesting or pruning in table-grape vineyards.

We use the Rosie HEBI mobile base (see Fig. 11 left), which is an omnidirectional mobile platform with three omnidirectional wheels. Note that by the time this research result was being developed, the mobile bases of CANOPIES project were not ready for testing navigation tasks. We chose in-house robots at KTH to demonstrate the developed planning and coordination schemes. The motion coordination algorithms were later tested in CANOPIES VR environment and on CANOPIES hardware platforms. As far as the laboratory experiments are concerned, through the use of the Qualisys motion capture system, the motion of the involved rigid bodies is tracked in real time. Each HEBI Rosie mobile base includes an on-board computer equipped with a proper ROS version, and autonomous control is accomplished from the user PC using the ROS API in conjunction with a wireless platform connection. The experiment workspace is an $5m \times 6m$ region as shown in Fig. 11 right, which is abstracted into 30 $1m \times 1m$ squares. We consider a group of 2 or 3 HEBI Rosie robots (Rosies 0, 1, and 2). In addition, each Rosie is subject to the inputs constraints $|v_i| \leq 0.35m/s$ and $|w_i| \leq 0.35rad/s$. The sensing radius of each robot is $R_i = 0.8m, \forall i$.

Consider $N = 2$ HEBI Rosie robots. The LTL specification for each robot is given by

- $\varphi_0 = ((\square\lozenge R8) \wedge (\square\lozenge R20))$,

This project is funded by the European Union G.A No 101016906



R25	R19	R13	R7	R1
R26	R20	R14	R8	R2
R27	R21	R15	R9	R3
R28	R22	R16	R10	R4
R29	R23	R17	R11	R5
R30	R24	R18	R12	R6

Figure 11: Rosie HEBI mobile base (left) and workspace discretization (right).

- $\varphi_1 = (\square\Diamond R17) \wedge (\square\Diamond R21)$.

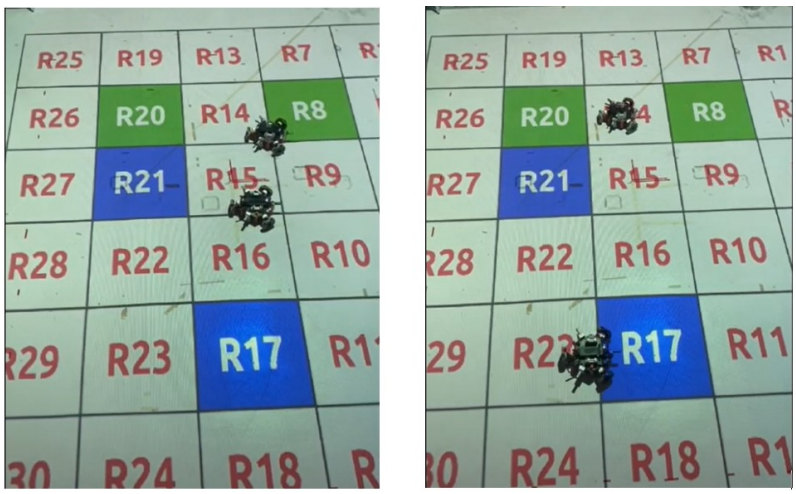


Figure 12: Snapshots of the robots replanning online to avoid collisions

3.5.1 Local communication case

Initially each robot synthesizes a satisfying trajectory using the LTL core & planner [45]. During the online implementation, possible collisions are detected, and then the local trajectory generation algorithm (Algorithm 1) is executed taking into consideration the future trajectories of neighboring robots (which is acquired through local communication). The real-time position trajectories of the 2 Rosies are depicted in Fig. 13. During the experiment time horizon $140s$, both robots successfully complete the monitoring tasks. Conflicts are detected 4 times in total, and the local trajectory generation algorithm is activated to resolve them every time. Note also that the input constraints are satisfied at all time. Two snapshots are shown in Fig. 12 where the online replanning module is activated. A video demonstration of this experiment can be found here.

This project is funded by the European Union G.A No 101016906

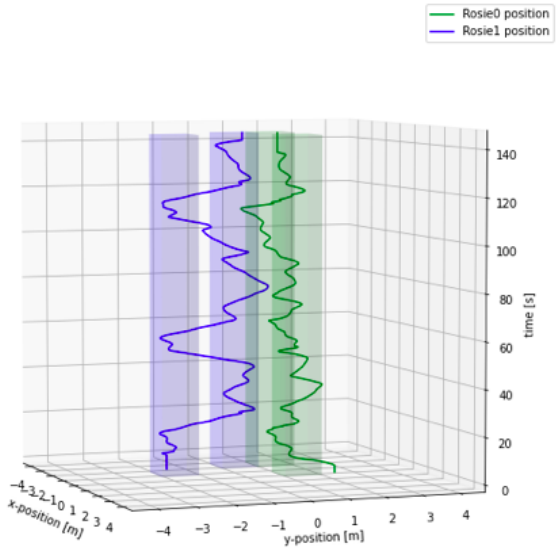


Figure 13: Local communication case: the real-time position trajectories of Rosies 0 (green line) and 1 (purple line). The light green and purple regions represent the target regions of Rosies 0 and 1, respectively.

3.5.2 Communication-free case with human as moving obstacle

In this case, we additionally consider a human walking in the workspace $\mathbb{W}$ to evaluate the effectiveness of the model predictive controller.

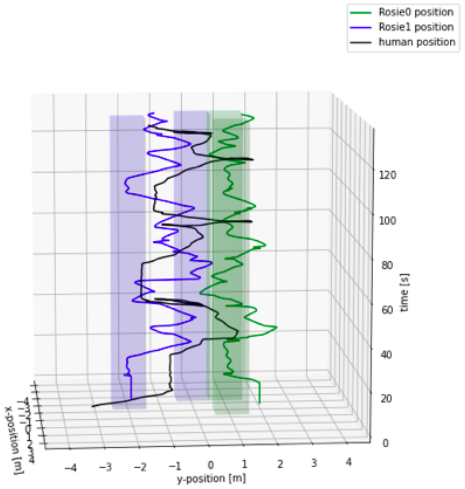


Figure 14: Communication-free case: the real-time position trajectories of Rosies 0 (green line), 1 (purple line), and human (black line).

Fig. 14 depicts the real-time position trajectories of the 2 Rosies and the human, where the proposed model

This project is funded by the European Union G.A No 101016906

predictive controller (29) is applied for both Rosies. Differently from the local communication case where replanning is activated only if conflicts are detected, in the communication-free case, the online replanning is activated whenever a robot detects obstacles in its sensing region. As a result, the motion of each robot is affected not only by other robots, but also by the human randomly walking through the workspace.

Replanning is conducted 14 times in total over the experiment time horizon $120s$, and one can see that the model predictive controller (29) guarantees the task satisfaction and collision avoidance (with other robots and the moving human) for each robot. When compared to the local communication case, the number of replanning is higher since the replanning process is activated whenever obstacles are detected and the model predictive controller does not take into account future trajectories of other robots. A snapshot of the experiment with human walking through is shown in Fig. 15 below. A video demonstration of this experiment can be found here.



Figure 15: A snapshot of human walking through a shared workspace with robots achieving their LTL tasks

This project is funded by the European Union G.A No 101016906

4 Safety assuring multi-robot coordination using distributed CBF

In this section, we present a safety-critical control scheme for multi-robot systems in the presence of multiple constraints using control barrier functions. The proposed scheme expands previous work where only one single constraint can be handled. Here we show how to transform multiple constraints to a collective one using a smoothly approximated minimum function.

Applications of the proposed algorithm to the problem of precision agriculture are shown in two scenarios associated with grape collection in vineyards. We simulate the case where a network of agents must help harvesting and pruning while satisfying collision avoidance constraints between each agent and humans walking through the workspace.

4.1 Problem formulation

When accounting for multiple safety constraints, a general control barrier function based controller that would always satisfy the constraints is given by the following Quadratic Problem (QP)

$$\begin{aligned} &\min_{\boldsymbol{u}} \|\boldsymbol{u}-\boldsymbol{u}_{nom}\|^2 \\ &s.t. \sum_{i\in\mathcal{I}} \boldsymbol{a}_i^{m\top}(\boldsymbol{x}_{loc,i})\boldsymbol{u}_i + \sum_{i\in\mathcal{I}} b_i^m(\boldsymbol{x}_{loc,i}) \leq 0, \quad \text{for } m=1,2,...,M \end{aligned} \tag{30}$$

where M is the total number of all the safety constraints. This formulation is reasonable considering the common constraints including inter-agent collision avoidance, connectivity maintenance, etc. However, this is not a distributed formulation and can be difficult to implement since it requires each robot knows the state of all the other robots. In this section, we report a novel way to distribute this QP-based controller while still retaining the safety assurance property inherited from CBF approaches.

4.2 Results and contribution

The aforementioned controller in (30) is based on a centralized approach and therefore needs more than just the locally available state. When dealing with only one constraint, i.e., $M=1$ in (30), [46] proposed that instead of solving one centralized local QP, the following equivalent QP is to be solved by each individual agent $i\in\mathcal{I}$

$$\begin{aligned} &\min_{\boldsymbol{u}_i\in\mathbb{R}^{m_i}} \|\boldsymbol{u}_i-\boldsymbol{u}_{nom,i}\|^2 \\ &s.t.\ \boldsymbol{a}_i^{\top}(\boldsymbol{x}_{loc,i})\boldsymbol{u}_i + \sum_{j\in N_i}(y_i-y_j) + b_i(\boldsymbol{x}_{loc,i}) \leq 0, \end{aligned} \tag{31}$$

where the CBF condition only uses the locally available information $\boldsymbol{x}_{loc,i}$ and $\boldsymbol{y}=(y_1,y_2,...,y_N)\in\mathbb{R}^N$ is an auxiliary variable. If all the local QPs are feasible, the CBF condition (4) is fulfilled for any $\boldsymbol{y}$. Define the following local variable

$$c_i = \frac{1}{\boldsymbol{a}_i^{\top}\boldsymbol{a}_i}(\boldsymbol{l}_i\boldsymbol{y} + \boldsymbol{a}_i^{\top}\boldsymbol{u}_{nom,i} + b_i), \tag{32}$$

where $\boldsymbol{l}_i$ is the i-th row of L and $\boldsymbol{y}$ is updated according to the discontinuous adaptative law (33) with positive gain k_0.

$$\dot{\boldsymbol{y}}_i = -k_0 \operatorname{sign}(\sum_{j \in N_i} (\boldsymbol{c}_i - \boldsymbol{c}_j)). \tag{33}$$

Lemma 1. *Assume that $\boldsymbol{a}_i, \boldsymbol{u}_{nom,i}$ and $\boldsymbol{b}_i$ are slowly time-varying and the gain k_0 is large enough. Then $\boldsymbol{c}$ reaches consensus within a finite time and the optimal $\boldsymbol{y}$ has been found [46].*

Additionally, instead of solving the local QP from (31), the following analytical solution can be used

$$\boldsymbol{u}_i = \boldsymbol{u}_{nom,i} - \max(0, c_i)\boldsymbol{a}_i, \ \forall i \in \mathcal{I}. \tag{34}$$

There are two types of functions identified in [46] whose CBF condition can be written in the form of (31), namely, $h(\boldsymbol{x}) = \sum_{k \in \mathcal{I}} b_k(\boldsymbol{x}_i)$ and $h(\boldsymbol{x}) = \sum_{e \in E} b_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We note that $b_k(\boldsymbol{x}_i), b_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are not constraints. In this paper, we assume there are two types of constraints, given below

1. Individual constraints $h_k(\boldsymbol{x}_i)$ that only depend on the state of one agent i.
2. Dual constraints $h_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which depend on the state of two neighboring agents.

In the following we will show how these two types of constraints can be transformed into the desired form in [46]. We will show the derivations for the second case since the derivation for the first case follows the same line and is thus omitted. In the experiment both types of constraints are demonstrated.

Here we will extend the solution shown in [46] to more constraints through the use of the minimum function.

4.2.1 Minimum function approximation

For the system to stay safe, all the functions $h_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ have to be non-negative. Here, to group them all together into one equation, the minimum function can be used, that is, we require $\min_{e \in E} h_e(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$. However, this does not follow the format from [46]. Instead, observe that

$$\min_{e \in E} h_e(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq -\frac{1}{p} \log \left(\sum_{e \in E} e^{-p h_e(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right), \tag{35}$$

for $(i, j) \in e$, where p is a positive constant relating to the error of the approximation, with higher values meaning less error, but more computationally expensive to calculate due to the exponential. Denoting $h_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as h_e for simplicity, one obtains

$$-\frac{1}{p} \log \left(\sum_{e \in E} e^{-p h_e} \right) \geq 0 \Leftrightarrow \log \left(\sum_{e \in E} e^{-p h_e} \right) \leq 0, \tag{36}$$

using the exponential, the condition in (36) is equivalent to

$$\sum_{e \in E} e^{-p h_e} \leq 1 \Leftrightarrow 1 - \sum_{e \in E} e^{-p h_e} \geq 0, \tag{37}$$

This project is funded by the European Union G.A No 101016906

and finally, incorporating everything inside the summation sign, we define

$$h(\boldsymbol{x}) = \sum_{e \in E} \left(\frac{1}{|E|} - e^{-ph_e} \right) =: \sum_{e \in E} \tilde{h}_e(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \tag{38}$$

where $|E|$ denotes the cardinality of the set E. Now this satisfies the desired form from [46]. Additionally, the gradient of the CBF needed for the CBF condition (4) is calculated as

$$\nabla h(\boldsymbol{x}) = \sum_{e \in E} p e^{-ph_e} \nabla h_e =: \sum_{e \in E} \nabla \tilde{h}_e(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{39}$$

This result works for both dual and individual constraints. We note that it is more conservative than just using the minimum function due to the approximation in (35).

Assumption 2. *For both types of constraints, $\alpha(h(\boldsymbol{x}))$ is linear, that is*

$$\alpha(h(\boldsymbol{x})) = a \cdot h(\boldsymbol{x}), \tag{40}$$

where a is a constant positive parameter.

Implementing this solution into (4) gives us

$$\nabla h^T(\boldsymbol{x})(f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}) + \alpha(h(\boldsymbol{x})) \geq 0, \tag{41}$$

which can further be written as

$$\sum_{e \in E} \nabla \tilde{h}_e^\top(\boldsymbol{x}_i, \boldsymbol{x}_j) g \boldsymbol{u} + \sum_{e \in E} \left(\nabla \tilde{h}_e^\top(\boldsymbol{x}_i, \boldsymbol{x}_j) f + \alpha(\tilde{h}_e(\boldsymbol{x}_i, \boldsymbol{x}_j)) \right) \geq 0. \tag{42}$$

The final step is to derive a formula for each $\boldsymbol{a}_i$ and b_i in (30) for each agent $i \in \mathcal{I}$.

4.2.2 Solving for the new coefficients

Denote

$$\boldsymbol{a}_{i,dual} = -\sum_{e \in E} I_e(i) \nabla_{\boldsymbol{x}_i} \tilde{h}_e(\boldsymbol{x}_i, \boldsymbol{x}_j) g_i^\top, \tag{43}$$

$$b_{i,dual} = -\sum_{e \in E} I_e(i) \left(\nabla_{\boldsymbol{x}_i} \tilde{h}_e(\boldsymbol{x}_i, \boldsymbol{x}_j)^\top f_i + \frac{a}{2} \left(\frac{1}{\beta} - e^{-ph_e(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right) \right), \tag{44}$$

where in this case $\beta = |E|$ when all functions $\tilde{h}_e$ are of the same type, for example collision avoidance between neighbours in the network. The indicator function $I_e(i)$ is

$$I_e(i) = \begin{cases} 1, & \text{if edge } e \text{ contains } i \\ 0, & \text{if edge } e \text{ does not contain } i \end{cases} \tag{45}$$

This project is funded by the European Union G.A No 101016906

Thus, $\boldsymbol{a}_{i,dual}$ and $b_{i,dual}$ only use local information of agent i and its neighbours $j \in N_i$.

Following the same derivation, we can obtain the coefficients for individual constraints of the same type. Denote

$$\boldsymbol{a}_{i,ind} = -\sum_{k \in K} \nabla \tilde{h}_k(\boldsymbol{x}_i) g_i^\top, \tag{46}$$

$$b_{i,ind} = -\sum_{k \in K} \left(\nabla \tilde{h}_k(\boldsymbol{x}_i)^\top f_i + a \left(\frac{1}{\beta} - e^{-p h_k(\boldsymbol{x}_i)} \right) \right), \tag{47}$$

where $\beta = |\mathcal{I}||K|$ and K is the number of individual constraints of the same type on every agent $i \in \mathcal{I}$. For example, in a 2D problem where a certain polygon is to be avoided, each of the sides of the polygon would be an individual constraint of the same type.

Each type of function h_e and h_k generates a constraint and therefore an $\boldsymbol{a}_i$ and b_i for each agent following (43), (44), (46) and (47). The $\boldsymbol{a}_i$ terms can be directly summed up to get the final formulation, which can be easily seen from the derivation for (38) using a bigger M, representing all the safety constraints in the system, instead of the edge set E

$$\boldsymbol{a}_i = \sum_{l_e \in L_E} \boldsymbol{a}_{i,dual}^{l_e} + \sum_{l_k \in L_K} \boldsymbol{a}_{i,ind}^{l_k}, \tag{48}$$

while for the b_i terms a minor modification is needed since $\beta = M = |L_E||E| + |L_K||\mathcal{I}||K|$ and

$$\boldsymbol{b}_i = \sum_{l_e \in L_E} \boldsymbol{b}_{i,dual}^{l_e} + \sum_{l_k \in L_K} \boldsymbol{b}_{i,ind}^{l_k}, \tag{49}$$

where $\mathcal{I}$ is the set of nodes in the network, $l_e \in L_E$, $l_k \in L_K$, and L_E and L_K are the different type of dual and individual constraints, respectively. For example, if two types of dual constraints are considered, like collision avoidance (CA) and connectivity maintenance (CM), $L_E = \{CA, CM\}$, and if three types of individual constraints are used, like three polygons to be avoided by the robots, $L_K = \{pol_1, pol_2, pol_3\}$.

Now we summarize our main results in the following theorem.

Theorem 3. *Consider the multi-agent system with multiple safety constraints. Assume that Assumption 2 holds, together with the assumptions of Lemma 1, and $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are computed using* (48) *and* (49)*, respectively. Then, as long as the local QPs given in* (31) *are feasible, i.e., $\sum_{j \in N_i}(y_i - y_j) + b_i(\boldsymbol{x}_{loc,i}) \leq 0$ whenever $\boldsymbol{a}_i = 0, \forall i \in \mathcal{I}$, the multi-agent system will satisfy all the constraints $h_e \geq 0$ and $h_k \geq 0$ in* (35) *for all time.*

Additionally, the distributed nature of the algorithm allows it to scale better than the centralized alternative that considers the whole network.

4.3 Validation: formation control with a human walking around

Two testing scenarios have been created in a lab setting to simulate an artificial vineyard, as shown in Fig. 16, where our network of robots must work in a collaborative manner while satisfying some basic safety constraints. These are, remaining inside the vineyard/arena, avoiding inner obstacles, inter-collision

This project is funded by the European Union G.A No 101016906

avoidance and connectivity maintenance between robots, and avoidance with a human worker in the same environment.



Figure 16: Artificial vineyard lab setting.

The robots used for the experiments are the Nexus 4WD holonomic robots, shown in Fig. 17, using either a NVIDIA Jetson TX2 or Intel NUC as onboard computer running a standard Ubuntu 18.04 distribution with ROS Melodic.

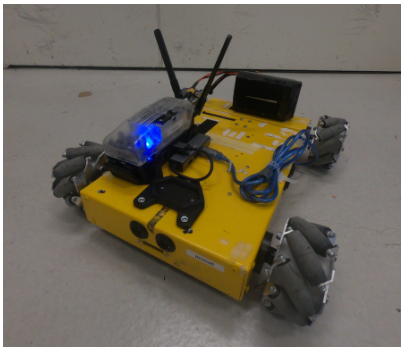


Figure 17: Nexus 4WD robot with a NVIDIA Jetson TX2.

The robots receive their position (state) through the use of markers and the Qualisys motion capture system. A diagram of this process is shown in Fig. 18.

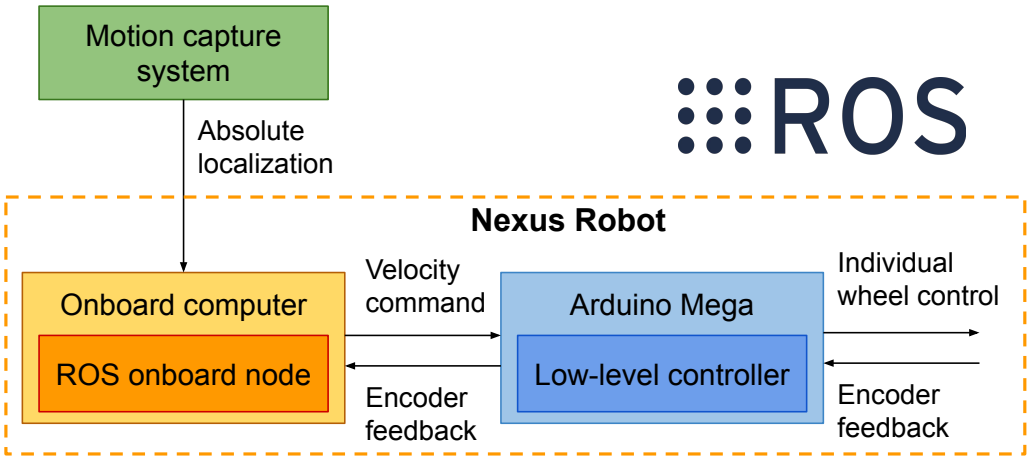


Figure 18: Diagram of the communications for each Nexus.

Five Nexus have been used for the experiments, following the communication topology in Fig. 19. The systems dynamics are modelled as a simple integrator $\dot{\boldsymbol{x}} = \boldsymbol{u}$, which is equivalent to (2) with $f(\boldsymbol{x}) = 0_{10}$ and $g(\boldsymbol{x})$ being the identity matrix $\mathbb{I}_{10}$.

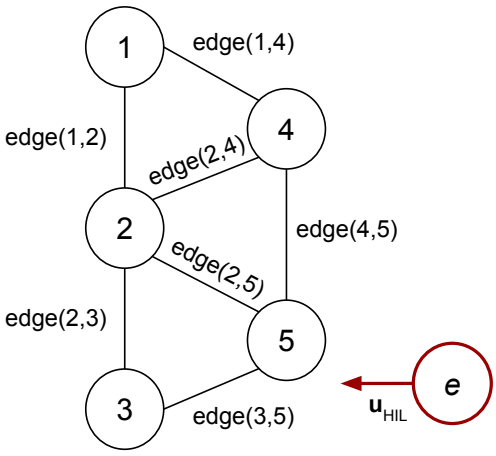
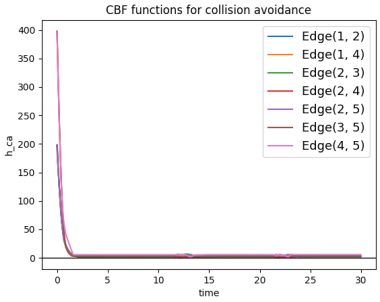


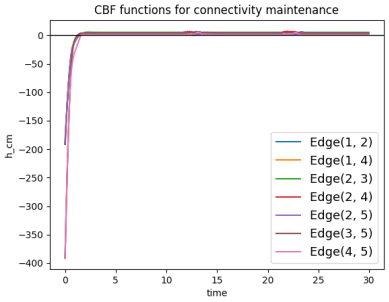
Figure 19: Network created by 5 agents and extra human e.

In this case, the nominal controller is designed such that the robots stay together in the specific formation shown in Fig. 19, while the safety constraints explained at the beginning of this section are always maintained. In particular, the aforementioned distributed formation controller is used for each agent as the nominal controller. This could, for example, represent a swarm of logistic robots that enter the vineyard in

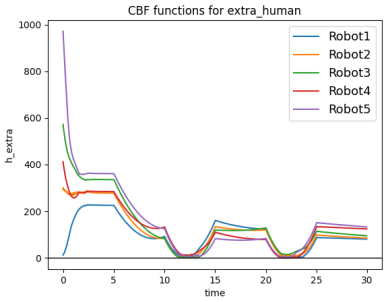
a close formation to collaboratively carry an empty basket. Thereafter, they get loaded with grapes and take them back to a storage area while avoiding human workers fulfilling other tasks.



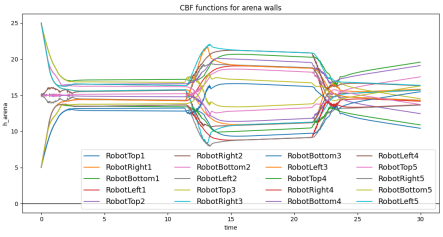
(a) CBFs h_{ca} for collision avoidance.



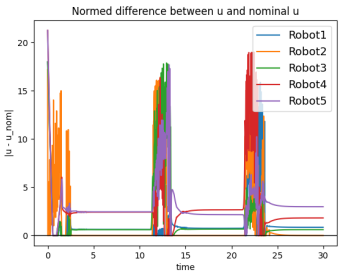
(b) CBFs h_{cm} for connectivity maintenance.



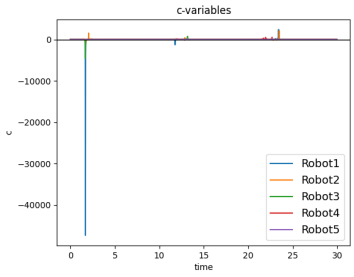
(c) CBFs h_{extra} for avoidance with human.



(d) CBFs h_{arena} for each side of the rectangular arena.



(e) Normed difference $\|\boldsymbol{u}(t) - \boldsymbol{u}_{nom}\|$.



(f) Auxiliary variable c (8).

Figure 20: Evolution of CBFs for collision avoidance (a), connectivity maintenance (b), arena (d) and avoidance with human (c), as well as the normed difference of the control input (e) and the consensus of c (f) for the first experiment scenario.

To create the single combined constraint, we first analyze each type separately. For **dual constraints**, taking the case of connectivity maintenance, each edge creates a different constraint, which implies the CBF function is the following

$$h_e(\boldsymbol{x}_i, \boldsymbol{x}_j) = r^2 - ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 \geq 0, \tag{50}$$

where r is the maximum allowed safety distance, $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 = (x_i - x_j)^2 - (y_i - y_j)^2$ and $\forall (i,j) \in E$.

We approximate the seven dual constraints as in (38) with $|E| = 7$. The gradient shown in (39) is used, with the specific gradient of the connectivity maintenance condition being

$$\nabla h_e(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{bmatrix} \frac{\partial h_e}{\partial x_i} \\ \frac{\partial h_e}{\partial y_i} \\ \frac{\partial h_e}{\partial x_j} \\ \frac{\partial h_e}{\partial y_j} \end{bmatrix} = \begin{bmatrix} -2(x_i - x_j) \\ -2(y_i - y_j) \\ 2(x_i - x_j) \\ 2(y_i - y_j) \end{bmatrix}, \tag{51}$$

for each dual constraint pair. Using (43) and (44) we can compute the $\boldsymbol{a}_{i,dual}$ and $b_{i,dual}$ terms for the QP. Similar terms can be developed for inter-collision avoidance constraints.

This project is funded by the European Union G.A No 101016906



Figure 21: Snapshot of a human walking through the workspace while robots trying to stay a formation

For **individual constraints**, we consider the rectangular arena limits for each agent, that is, there is one constraint per side ($|K| = 4$ and $|I| = 5$) and the corresponding functions become $h_k^{x_{\max}} = x_{\max} - x_i$, $h_k^{x_{\min}} = x_i - x_{\min}$, $h_k^{y_{\max}} = y_{\max} - y_i$ and $h_k^{y_{\max}} = y_i - y_{\min}$; with the gradient

$$\nabla h_i = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{52}$$

Computing the $\boldsymbol{a}_{i,ind}$ and $b_{i,ind}$ terms using (46) and (47), allows to obtain the final single constraint variables $\boldsymbol{a}_i$ and b_i from (48) and (49).

Collision avoidance with an extra human is similar to the one between robots, but adapted for individual type constraints and therefore, with a modification to account for the speed and trajectory of the human. Since the gradient computed only takes into account the robot state $\boldsymbol{x}_i$ and not the one from the human $\boldsymbol{x}_e$, we need to add this extra term to b_i, such that

$$\boldsymbol{a}_{i,extra} = -\nabla_{\boldsymbol{x}_i} \tilde{h}(\boldsymbol{x}_i, \boldsymbol{x}_e) g_i^\top, \tag{53}$$

$$b_{i,extra} = -\left(\nabla_{\boldsymbol{x}_i} \tilde{h}(\boldsymbol{x}_i, \boldsymbol{x}_e)^\top f_i + a \left(\frac{1}{|I|} - e^{-ph_e(\boldsymbol{x}_i, \boldsymbol{x}_e)} \right) \right) - \nabla_{\boldsymbol{x}_e} \tilde{h}(\boldsymbol{x}_e, \boldsymbol{x}_i) g_i^\top. \tag{54}$$

which can then be added to (48) and (49) as a standard individual type constraint $l_k = (extra)$.

The results for a random execution of this first case can be seen in Fig. 20 and a video of it can be found here. A snapshot of this experiment is shown in Fig. 21. In this scenario, four different types of CBFs are shown, *(a)* for the collision avoidance between robots, *(b)* for the connectivity maintenance between robots, *(c)* for the collision avoidance between each robot and the human and *(d)* for bounding the robots inside the four walls of the arena (Top, Right, Bottom and Left). All of them are positive for all time and therefore

This project is funded by the European Union G.A No 101016906

the conditions are fulfilled, even when initialized from a non-satisfying point, that is when $h_e < 0$ and/or $h_k < 0$, since the agents quickly reposition to be safe. Additionally, *(e)* shows the difference between the nominal controller and the output of the QP and *(f)* shows the evolution of the c variable from (32).

This project is funded by the European Union G.A No 101016906

5 Human-robot co-manipulation

In this section, we propose a control strategy for human-robot cooperative manipulation under the *ambiguous* collaboration of a human agent. By ambiguity, we mean that the intention of the human operator and the actual effect (in terms of force and torque) are uncertain. This is usually the case when a farmer is interacting with mobile manipulators harvesting and moving grapes. When the human has the same intention as the manipulators, we refer that human is providing assistance; otherwise, human is disturbing the manipulation. To cope with this uncertainty, an adaptive update law inferring the human contribution to the system dynamics from basic perception feedback through the human arm stiffness is used. Furthermore, the robustness and accuracy of the approach is enhanced by redundantly tracking the shared load references and its associated end-effector position references. To validate the control strategy, both theoretical Lyapunov stability analysis and experimental results –employing two robot manipulators with 6 degrees of freedom under external disturbances– are provided.

5.1 Problem formulation

Let us consider the system depicted in Fig. 22, composed by: a team of $n \geq 2$ robots (R) –each one equipped with a N_i-DoF RM, $N_i \geq 3, i = 1, ..., n$–; a human operator arm (H) –assumed equivalent to a 7-DoF RM, as in [47, 48] but enriched with flexible modes[1]–; and a rigid body (L) –whose load is shared by (R) and (H)–. Each of these agents is characterised by:

(R) joint-space configuration $\boldsymbol{\gamma}^\top := [\boldsymbol{\gamma}_1^\top, \cdots, \boldsymbol{\gamma}_n^\top] \in \mathbb{R}^N$, with $\boldsymbol{\gamma}_i \in \mathbb{R}^{N_i}$ the joint-space of the ith agent and $N = \sum_{i=1}^{n} N_i$ the total number of DoF; and Cartesian EE positions $\mathbf{p}^\top := [\mathbf{p}_1^\top, \cdots, \mathbf{p}_n^\top] \in \mathbb{R}^{3n}$.

(M) joint-space configuration, including the angle of the links $\boldsymbol{\eta} \in \mathbb{R}^7$ and of the motors $\boldsymbol{\mu} \in \mathbb{R}^7$, where $\boldsymbol{\delta} := \boldsymbol{\eta} - \boldsymbol{\mu}$ are the flexible deflections (also denoted as flexible modes); and Cartesian EE position $\mathbf{q} \in \mathbb{R}^3$.

(H) CoG position $\mathbf{s} \in \mathbb{R}^3$ and attitude with respect to the base frame $\boldsymbol{\theta} := [\phi, \theta, \psi]^\top \in \mathbb{R}^3$ in RPY, and the fixed grasping points of both (R) and (H), namely $\mathbf{p}, \mathbf{q}$.

The design task for this system is to *simultaneously* track: i) the CoG position and attitude references of (L), $\mathbf{s}_r$ and $\boldsymbol{\theta}_r$, and ii) the EE position references of (R), $\mathbf{p}_r$; with the *ambiguous* (i.e. not *a priori* positive to the performance of the whole system) collaboration of (H). Accordingly, the task-space for each agent is defined as: $\mathbf{x}_L^\top := [\mathbf{s}^\top, \boldsymbol{\theta}^\top]$ for (L), $\mathbf{p}$ for (R), and $\mathbf{q}$ for (H); where the latter is not controllable, as (H) is an external ambiguous agent.

5.1.1 Kinematics and dynamics

The kinematics of the task-space of the system is

$$\dot{\mathbf{x}}_L := \mathbb{T}_L^{-1} \mathbf{v}_L, \tag{55a}$$

$$\mathbf{v}_R := J_R \dot{\boldsymbol{\gamma}}, \tag{55b}$$

$$\mathbf{v}_H := J_H \dot{\boldsymbol{\eta}}, \tag{55c}$$

[1] Flexible modes describe the additional dynamic complexity induced by the extra flexible DoF, generally in the form of vibrations/oscillations.

This project is funded by the European Union G.A No 101016906

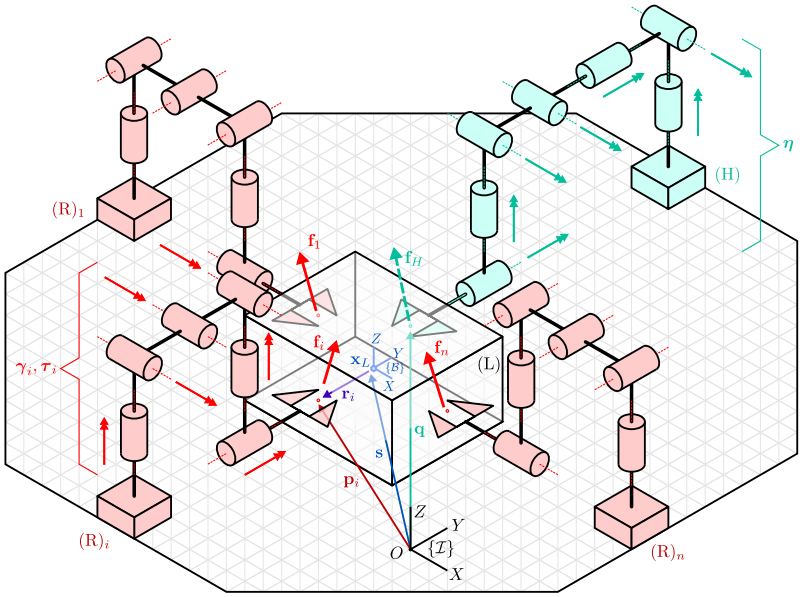


Figure 22: Scheme of the system including the grasping setting –with (R) in red and (H) in aquamarine–, and $\{\mathcal{B}\}$ and $\{\mathcal{I}\}$ reference frames.

where $\mathbf{v}_R^\top := [\dot{\mathbf{p}}^\top, \boldsymbol{\omega}_R^\top]$ is the (R) operational-space velocity, with $\boldsymbol{\omega}_R$ the (R) angular velocity, and J_R is the geometric Jacobian of (R); J_H is the geometric Jacobian of (H), $\mathbf{v}_H^\top := [\dot{\mathbf{q}}^\top, \boldsymbol{\omega}_H^\top]$ is the (H) operational-space velocity, with $\boldsymbol{\omega}_H$ the (H) angular velocity; and $\mathbf{v}_L^\top := [\dot{\mathbf{s}}^\top, \boldsymbol{\omega}_L^\top]$ the corresponding one for (L), with $\boldsymbol{\omega}_L := T_L \dot{\boldsymbol{\theta}}$ the (L) angular velocity, and

$$T_L := \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \sin\psi\cos\theta & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{pmatrix}, \quad \mathbb{T}_L := \begin{pmatrix} I_3 & 0_3 \\ 0_3 & T_L \end{pmatrix}.$$

As the task only includes (R) and (H) Cartesian positions, let us also define their position Jacobians for simplicity –i.e. only using their associated rows in J_R, J_H–, namely

$$\dot{\mathbf{p}} := J_R^p \dot{\boldsymbol{\gamma}}, \quad \dot{\mathbf{q}} := J_H^p \dot{\boldsymbol{\eta}}.$$

On the one hand, using the Euler-Lagrange and Newton-Euler formalisms to respectively obtain the dynamic equations of (R) and (L) results in

$$M\dot{\mathbf{v}}_L + \boldsymbol{\omega}_L^\times I_L \boldsymbol{\omega}_L + \mathbf{g}_L = G\mathbf{f} + G_H \mathbf{f}_H, \tag{56a}$$

$$B_R\ddot{\boldsymbol{\gamma}} + C_R\dot{\boldsymbol{\gamma}} + \mathbf{g}_R = \boldsymbol{\tau} - J_R^\top \mathbf{f}, \tag{56b}$$

with $M \in \mathbb{R}^{6\times6}$ the inertial matrix of (L), with $I_L \in \mathbb{R}^{3\times3}$ its rotation inertia in $\{\mathcal{B}\}$, and $\mathbf{g}_L \in \mathbb{R}^6$ the gravitational terms for (L); $B_R, C_R \in \mathbb{R}^{N\times N}$ the inertial and Coriolis matrices for (R), and $\mathbf{g}_R$ the gravitational terms for (R). In turn, the control input of the system (56) are the (R) joint torques $\boldsymbol{\tau} \in \mathbb{R}^N$, which is applied

on to (L) via the generalised contact forces $\mathbf{f} \in \mathbb{R}^{6n}$ in (56a) (see Fig. 23), where $G \in \mathbb{R}^{6\times 6n}$ denotes the (R) grasp matrix, given for $\mathbf{f}$ in $\{\mathcal{I}\}$ [49] by

$$G := \left(\begin{array}{cc|c|cc} I_3 & 0_3 & \cdots & I_3 & 0_3 \\ \mathbf{r}_1^\times & I_3 & & \mathbf{r}_n^\times & I_3 \end{array}\right), \quad \mathbf{r}_i := \mathbf{s} - \mathbf{p}_i.$$

On the other hand, the well-known flexible joint RM model in [50] is employed to represent the intrinsic complexity of (H) (see Assumption A4 below), namely

$$B_H^l \ddot{\boldsymbol{\eta}} + C_H \dot{\boldsymbol{\eta}} + K\boldsymbol{\delta} + \mathbf{g}_H = -J_H^\top \mathbf{f}_H, \tag{57a}$$

$$B_H^m \ddot{\boldsymbol{\mu}} - K\boldsymbol{\delta} = \boldsymbol{\tau}_H, \tag{57b}$$

where the friction terms are assumed to be negligible [50]; $B_H^l, B_H^m, C_H^l, C_H^m \in \mathbb{R}^{7\times 7}$ are the inertial and Coriolis matrices for (H) –with super-index l and m referring to the link and motor DoF–, and $\mathbf{g}_H$ denotes the gravitational terms for (H); K the *unknown* stiffness of the flexible DoF of (H); and $\mathbf{f}_H$ the forces and torques exerted by (H) to (L) through the grasp matrix $G_H \in \mathbb{R}^{6\times 6}$, given for $\mathbf{f}_H$ in $\{\mathcal{I}\}$ [49] by

$$G_H := \begin{pmatrix} I_3 & 0_3 \\ \mathbf{r}_H^\times & I_3 \end{pmatrix}, \quad \mathbf{r}_H := \mathbf{s} - \mathbf{q}.$$

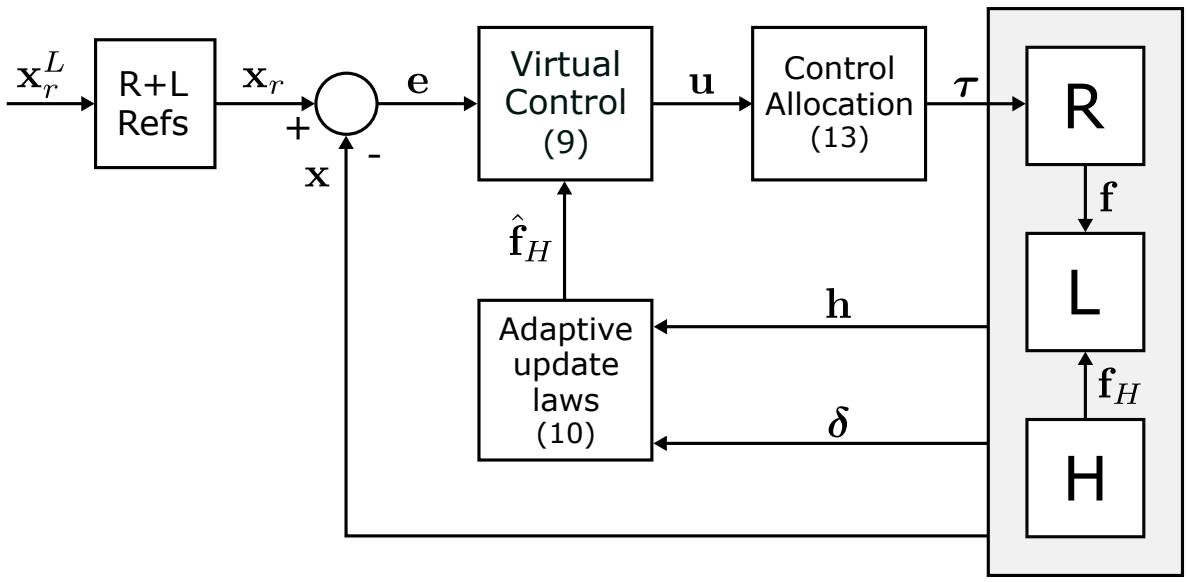


Figure 23: Control scheme of the proposed strategy (with equation numbers), including the virtual controller and the control allocation.

Assumptions

A series of common assumptions are used to formulate the proposed controller, namely

A1. The references $\mathbf{x}_{L_r}$ and $\mathbf{p}_r$ are feasible.

This project is funded by the European Union G.A No 101016906

A2. The Jacobians J_R and J_H are full-ranked.

A3. T_L is full-ranked, i.e. $\theta \neq \pm\pi/2$.

A4. Similarly to variable impedance approaches like [51, 52, 53] –but in the joint-space instead of the operational space–, (H) is assumed to be equivalent to a RM with flexible-actuated compound joints (57), in which:

A4.1. The joint velocities of (H) are small and, thus, the equilibrium configuration coincides with the initial configuration, i.e. $\boldsymbol{\mu}(0) = \boldsymbol{\eta}(0)$, and does not move beyond a small area, i.e. $\boldsymbol{\mu}(t) \approx \boldsymbol{\mu}(0),\ \forall t \geq 0$.

A4.2. The (H) actuation $\boldsymbol{\tau}_H$ in (57b) cancels the inertial, gravitational and Coriolis terms, i.e. $\boldsymbol{\tau}_H = -K\boldsymbol{\delta} - B_H^m\ddot{\boldsymbol{\delta}} - B_H^m(B_H^l)^{-1}(C_H\dot{\boldsymbol{\eta}} + \mathbf{g}_H)$, while the external forces are compensated by the flexible modes, i.e. $K\boldsymbol{\delta} = -J_H^{\top}\mathbf{f}_H$.

While (R) is the actuated agent used to control (L), (H) can be seen as an *ambiguous* external disturbance to the system. Accordingly, we propose estimating the forces and torques applied by (H) from the deflection of their flexible modes and coping with the induced uncertainty via an adaptive controller on K. For this purpose, a simplified –and inherently pseudo-static– model of the human arm response is obtained under the assumptions in A4:

$$K\boldsymbol{\delta} = \mathbf{d}_H - J_H^{\top}\mathbf{f}_H, \tag{58}$$

where $\mathbf{d}_H \in \mathbb{R}^7$ is an *unknown* slow time-varying disturbance modelling the small deviations in (57) from assumption A4.2, which are also coped via an adaptive update law. It is worth noting that as (H) has 7 DoF, (58) lacks –in general– a solution for $\mathbf{f}_H$. Nevertheless, from a physical point of view, (58) describes how the flexibility-related generalised forces are distributed throughout these flexible joints. Accordingly, a real distribution of joint-space torques corresponds to a real value of $\mathbf{f}_H$ and, therefore, the lack of a solution is produced by disturbances. Consequently, the least-squares solution given by

$$\hat{\mathbf{f}}_H = (J_H^{\top})^{\dagger}\left[\hat{\mathbf{d}}_H - \hat{K}\boldsymbol{\delta}\right] \tag{59}$$

is considered a suitable estimate to these generalised forces for three reasons: i) due to its formulation, it partially filters the aforementioned noise, ii) it is linear with respect to $\hat{K}$ and $\hat{\mathbf{d}}_H$, and thus suitable for an adaptive approach to cope with the (H) uncertainties, and iii) apart from this update law, the estimation only requires to measure the joint-space configuration of (H) to be implemented.

5.2 Results and contributions

Let us define a velocity fitting with these spaces as $\mathbf{v}^{\top} := [\mathbf{v}_L^{\top}, \dot{\boldsymbol{\gamma}}^{\top}]$, with $\dot{\mathbf{x}} = W\mathbf{v}$ and

$$W = \begin{pmatrix} \mathbb{T}_L^{-1} & 0 \\ 0 & J_R^p \end{pmatrix}.$$

This project is funded by the European Union G.A No 101016906

Then, under the assumptions A2 and A3, the kinematics (55a)-(55b) and dynamics (56a)-(56b) on the combined (R)+(L) system can be rewritten in error terms as

$$\dot{\mathbf{e}}_I = \mathbf{e}_P, \tag{60a}$$
$$\dot{\mathbf{e}}_P = \mathbf{e}_D, \tag{60b}$$
$$\dot{\mathbf{e}}_D = \ddot{\mathbf{x}}_r + \dot{W}W^+(\mathbf{e}_D - \dot{\mathbf{x}}_r) - WB^{-1}(\mathbf{u} + \mathbf{u}_H - \mathbf{b}), \tag{60c}$$

where $\mathbf{e}_P := \mathbf{x}_r - \mathbf{x}$ is the tracking error and $\mathbf{e}_I$ its implicit integral counterpart;

$$B = \begin{pmatrix} M & 0 \\ 0 & B_R \end{pmatrix}$$

the inertia matrix of the whole system, and

$$\mathbf{u} := \begin{bmatrix} \mathbf{0}_6 \\ \boldsymbol{\tau} \end{bmatrix} + \begin{pmatrix} G \\ -J_R^\top \end{pmatrix} \mathbf{f}, \tag{61}$$
$$\mathbf{u}_H := \begin{bmatrix} G_H \mathbf{f}_H \\ \mathbf{0}_N \end{bmatrix}, \tag{62}$$

the virtual control input and its associated representation of the effects of (H) on the (R)+(L) system via the grasping of (L); and $\mathbf{b}^\top = [(\boldsymbol{\omega}_L^\times I_L \boldsymbol{\omega}_L)^\top + \mathbf{g}_L^\top, (C_R \dot{\boldsymbol{\gamma}})^\top + \mathbf{g}_R^\top]$.

Among the different approaches fitting the virtual problem in (60), an adaptive integral backstepping approach is chosen: i) adaptive to cope with the (H) uncertainties, and ii) integral to handle zero steady-state error. This controller is given by

$$\mathbf{u} = BW^+ \left(\Gamma \mathbf{e} + \ddot{\mathbf{x}}_r - \dot{W}W^+ \dot{\mathbf{x}}_r \right) + \mathbf{b} - \hat{\mathbf{u}}_H, \tag{63}$$

with $\mathbf{e} := [\mathbf{e}_I^\top, \mathbf{e}_P^\top, \mathbf{e}_D^\top]^\top$, $\hat{\mathbf{u}}_H^\top := [(G_H \hat{\mathbf{f}}_H)^\top, \mathbf{0}_N^\top]$ and $\Gamma := (\Gamma_I, \Gamma_P, \Gamma_D + \dot{W}W^+)$,

$$\begin{aligned} \Gamma_I &:= \Gamma_1 + \Gamma_3\Gamma_2\Gamma_1 + \Gamma_3, \\ \Gamma_P &:= \Gamma_2\Gamma_1 + \Gamma_3\Gamma_1 + \Gamma_3\Gamma_2 + 2I_S, \\ \Gamma_D &:= \Gamma_1 + \Gamma_2 + \Gamma_3, \end{aligned}$$

where $\Gamma_1, \Gamma_2, \Gamma_3 \in \mathbb{R}^{S \times S}$ are positive definite gain matrices; along with the adaptive update laws for the estimate of (62) –with $\hat{\mathbf{f}}_H$ defined in (59)– given by

$$\dot{\hat{\mathbf{d}}}_H = -\Gamma_d \mathbf{h}, \tag{64a}$$
$$\dot{\hat{K}} = \Gamma_K \boldsymbol{\delta} \mathbf{h}^\top, \tag{64b}$$

with $\Gamma_d, \Gamma_K \in \mathbb{R}^{7\times7}$ positive definite, $\mathbf{h} := \Lambda_h \Gamma_h \mathbf{e}$ a vector encapsulating the connection between (L) and the human-related disturbances, $\Lambda_h := (J_H^\dagger G_H^T M^{-1} \mathbb{T}_L^{-T}, 0_{7\times6})$, and $\Gamma_h := (\Gamma_2\Gamma_1 + I_S, \Gamma_2 + \Gamma_1, I_S)$.

Proposition 4. Consider the system (60) and the estimation of $\mathbf{f}_H$ in (59) under the assumptions A1-A4. Then, under the control law (63) and its associated adaptive update laws (64), the closed-loop system is asymptotically stabilised to zero, i.e. $\mathbf{e} \to \mathbf{0}$ as $t \to \infty$.

he (R) forces on the load can be then determined by pre-multiplying (61) by $(I_6, 0_N)$, namely

$$\mathbf{f} = G^{+} \begin{pmatrix} I_6 & 0_N \end{pmatrix} \mathbf{u}, \tag{65}$$

where the right-hand pseudoinverse of the *full-ranked* grasp matrix G produces the minimal actuation needed to meet the virtual requirements. It is worth noting that adding terms in the nullspace of G would have no interest for fixed grasping as these would only result in internal forces in (L) [49]. Pre-multiplying (61) by $(0_6, I_N)$ and plugging (65), the control input that replicates the effects of (61) on the system reads

$$\boldsymbol{\tau} = \begin{pmatrix} J_R^{\top} G^{+} & I_N \end{pmatrix} \mathbf{u}. \tag{66}$$

Combining the virtual controller (63) and the control allocation (66), the final real control law for the system is

$$\boldsymbol{\tau} = \Lambda \left(\Gamma \mathbf{e} + \ddot{\mathbf{x}}_r - \dot{W} W^{+} \dot{\mathbf{x}}_r \right) + J_R^{\top} G^{+} (\mathbf{b}_L - G_H \hat{\mathbf{f}}_H) + \mathbf{b}_R, \tag{67}$$

with

$$\Lambda := \begin{pmatrix} J_R^{\top} G^{+} M \mathbb{T}_L & B_R (J_R^{p})^{+} \end{pmatrix}.$$

5.3 Validation

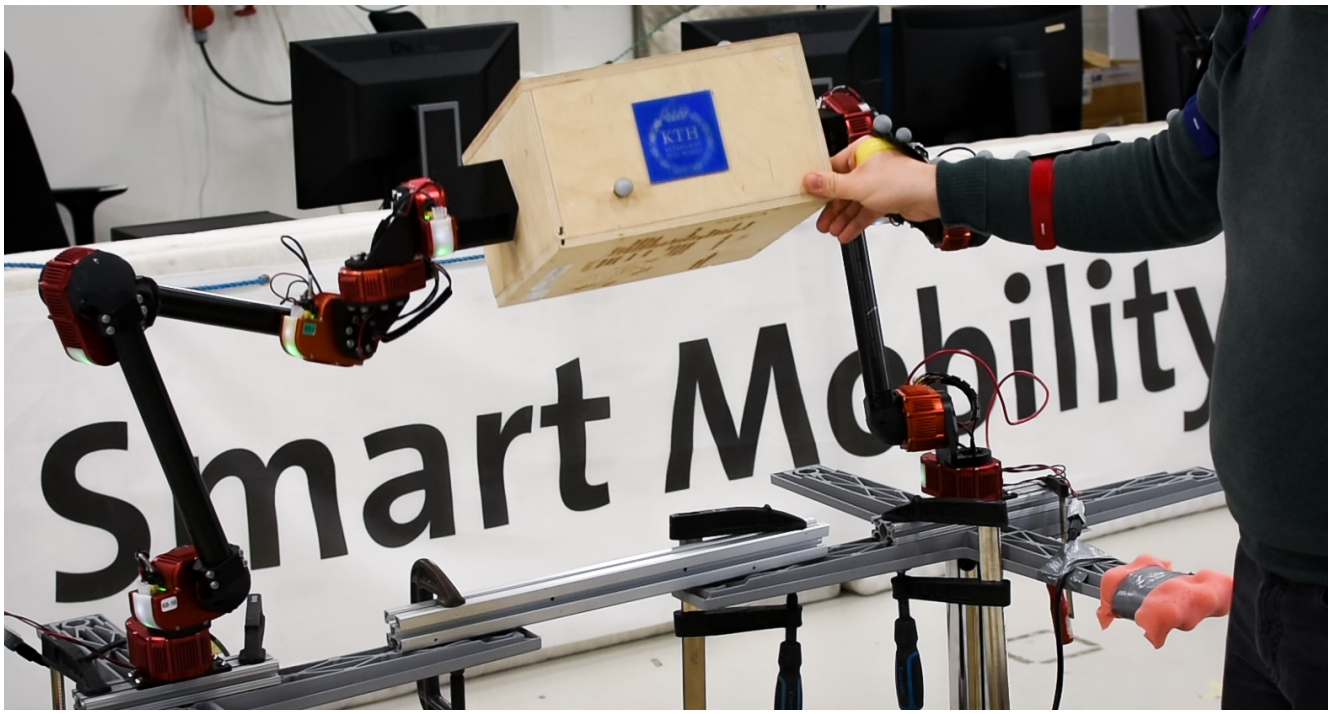


Figure 24: Experimental validation of the control strategy. These were carried out by C. R. de Cos at the SML facility in KTH.

To validate the proposed control strategy, we test the robustness of the solution against implementation disturbances and the *ambiguous* human interaction (Fig. 24).

This project is funded by the European Union G.A No 101016906

Table 1: Control gains

Agent	Component	Γ_1	Γ_2	Γ_3
(L)	All positions	0.5	30.0	6.0
	Roll and pitch	1.5	120.0	6.0
	Yaw	0.6	160.0	0.8
(R)	Horizontal position	0.4	9.0	4.0
	Vertical position	0.6	18.0	4.0

Agent	Component	Γ_K	Γ_d	σ
(H)	All	0.15	0.15	200

The validation experiments are carried out with and without[2] updating the adaptive law in (64) in three scenarios: i) (H) trying to assist (R), ii) (H) having no interaction with (R), and iii) (H) actively disturbing (R). In CANOPIES setting, we could view these three scenarios as simplified cases where a farmer is helping/disturbing mobile manipulators for moving a box of grapes.

While the second scenario is chosen as a control case showing the basic capabilities of the strategy, i) is designed to evaluate if the solution fits a nominal HRC task and analyse the impact human help; and iii) is specifically aimed at validating the disturbance rejection capabilities obtained with both contribution C1 and C2. Furthermore, these cases are studied with and without updating the adaptive laws to identify which advances correspond mainly to C2 (without) and which to C1 (with).

In the three scenarios the same (L) references are given for consistency. These are chosen to pose a challenge to the manipulation system even without disturbances: relatively fast displacements for manipulation tasks requiring high accuracy ($\sim$0.1 m/s) going through varied regions of the manipulability ellipsoid for the position tracking, and slower but wide attitude references demanding significant reconfigurations of the manipulators themselves.

Furthermore, the experiments including interactions are performed in a strict sequence so that the operator forces and motion are as consistent as possible, i.e. following a choreography for the experiment without adaptive laws and, immediately afterwards, repeating it for the one with the adaptive module. Although this approach has limitations, the operator achieved an acceptable degree of repeatability, as evinced in Fig. 25 by the repetition of certain error peaks at similar times.

The hardware consists of 2 6-DoF HEBI Robotics A-2085-06 RMs. Those were used due to lack of access to the relevant CANOPIES platforms at the time. Note that the results are platform independent and thus applicable to other cooperative manipulation HW. The robots' bases are the origin of $\{\mathcal{I}\}$, for $(\mathrm{R})_1$, and displaced $0.987\mathrm{m}$ in direction Y, for $(\mathrm{R})_2$ (see Fig. 22); a load of mass $2.117\mathrm{kg}$ (above the sum of the maximum payloads of both RMs) and moments of inertia in $\{\mathcal{B}\}$ $I_{L_x} = 0.0297\mathrm{kg\,m^2}$, $I_{L_y} = 0.0316\mathrm{kg\,m^2}$ and $I_{L_z} = 0.0440\mathrm{kg\,m^2}$, where the grasping points in $\{\mathcal{B}\}$ are $\mathbf{r}_1 = [0, -0.145, 0]^\top\mathrm{m}$, $\mathbf{r}_2 = [0, 0.145, 0]^\top\mathrm{m}$ and

[2] Setting an empirical trade-off value for the adaptive parameters.

This project is funded by the European Union G.A No 101016906

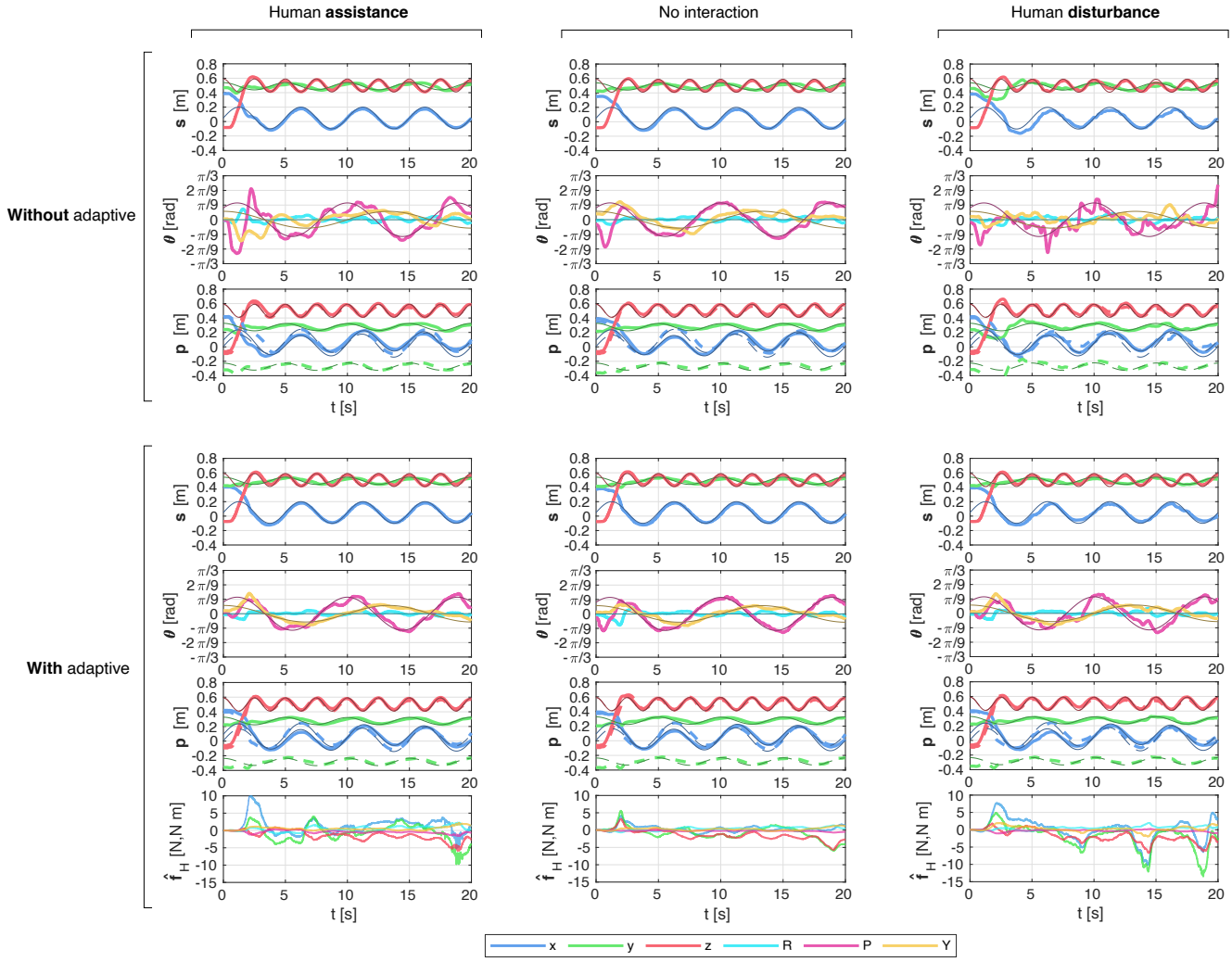


Figure 25: Experimental results with and without the adaptive update law (64) under three scenarios: human assistance, no interaction, and human disturbance. The components of each variable are displayed using the colour scheme in the bottom legend (where R, P, Y stand for roll, pitch and yaw) –with their associated references being thinner and darker–, and $\mathbf{p}_1$ and $\mathbf{p}_2$ are shown using solid and dashed lines, respectively.

$\mathbf{r}_H = [0.153, 0.145, -0.085]^\top \mathrm{m}$; and a computer equipped with a Intel Core i7-1085-H CPU at 2.70 GHz and 32 GB of RAM (DDR4). The feedback is obtained both from the sensors mounted on (R), and from a Qualisys Motion Capture System [54] with 12 Ocus-400 cameras for (H). Regarding software, the controller runs in real-time at $100\mathrm{Hz}$ using the Robotic Systems Toolbox [55] in Simulink, with the control gains in Table 1. This includes a σ-modification of the adaptive update laws to cope with noise-induced drift [56]. For the connection between the computer and (R), the HEBI MATLAB API [57] is employed. In turn, the Simulink ROS Toolbox is used to subscribe to the motion capture data for (L) and (H).

This project is funded by the European Union G.A No 101016906

Table 2: Statistical analysis, RMSE $[\mathrm{m}, \mathrm{rad}]$

		$\mathbf{x}_L$					
		x	y	z	ϕ	θ	ψ
Transient ($t < 5\mathrm{s}$)	Without adaptive	0.118	0.058	0.272	0.066	0.454	0.196
	With adaptive	0.124	0.050	0.278	0.079	0.308	0.156
	Variation (%)	*4.76*	*-13.01*	*2.44*	*19.82*	*-32.10*	*-20.09*
Steady-state ($t \geq 5\mathrm{s}$)	Without adaptive	0.025	0.023	0.017	0.049	0.158	0.123
	With adaptive	0.027	0.019	0.016	0.041	0.108	0.059
	Variation (%)	*9.37*	*-17.23*	*-6.65*	*-15.75*	*-31.84*	*-51.58*

Moving to the validation, we here analyse the experimental results.[3] As shown in Fig. 25, the controller is capable of tracking all the references without the adaptive update law when there is no interaction. It is specially significant that the response of base strategy in attitude is acceptably smooth and shows no noticeable steady state error. However, the lack of compensation without updating the parameters –and specially under disturbances– clearly results in a poorer tracking performance. Not only can be this seen in the attitude tracking response, which is considerable less robust and shows substantial oscillations, but also in the deterioration of the transient response in positions and the –small but significant– presence of steady-state error. This situation, nonetheless, does not reach unstable behaviours, and the base of the strategy (i.e. the *simultaneous* tracking of the (L) and (R) references) is shown to provide a robust –but improvable– foundation upon which the adaptive update law is added to cope with this downgrade during interaction.

In contrast, the complete strategy maintains the nominal tracking performance during the HRI thanks to the compensation coming from $\hat{\mathbf{f}}_H$ (last row, Fig. 25). For instance, while without it the disturbances produced noticeable oscillations in the (L) attitude, these are here comparable to the human-assisted adaptive case, significantly smoother than the scenario without adaptive updates, and just slightly wider in amplitude than the control case. In turn, a comparison between both cases in the x component of the (L) position tracking shows that the controller with the adaptive law is smoother and faster during the initial transient than the base one, and it does not display significant steady-state error.

Moving to the $\hat{\mathbf{f}}_H$ terms themselves, these evolve under excitation to make the response smoother and, under significant disturbances, (such as at 14 s or 17 s for the human disturbance case) to mitigate these external actions. Moreover, a pattern can be detected if we compare the response with the adaptive laws for both interaction cases: the generalised force profiles are similar for both, with just a more aggressive response of the update laws under disturbances, as expected. It is also worth noting that –apart from a peak in x during the initial transient– these profiles show forces whose main components are in y and z, corresponding to the main directions in which the forces in the interaction cases were applied. It is also worth highlighting that the complete strategy also improves the already solid response of the base controller without interactions (central column in Fig. 25), specially for smoothening the attitude response.

[3] Video available here.

This project is funded by the European Union G.A No 101016906

Table 3: Statistical analysis, RMSE $[\mathrm{m}, \mathrm{rad}]$

		$\mathbf{p}_1$			$\mathbf{p}_2$		
		x	y	z	x	y	z
Transient ($t < 5\mathrm{s}$)	Without adaptive	0.107	0.055	0.259	0.160	0.064	0.276
	With adaptive	0.114	0.046	0.264	0.140	0.053	0.285
	Variation (%)	*6.73*	*-17.53*	*2.04*	*-12.02*	*-17.98*	*3.25*
Steady-state ($t \geq 5\mathrm{s}$)	Without adaptive	0.041	0.022	0.018	0.041	0.022	0.023
	With adaptive	0.032	0.019	0.017	0.032	0.022	0.020
	Variation (%)	*-21.86*	*-11.54*	*-1.65*	*-22.79*	*-0.53*	*-14.33*

Finally, we include a broader statistical analysis to quantify the solidity of the base strategy and the improvements due to the adaptive modification. For that purpose, we analyse the transient and steady-state regimes[4] of the strategy with and without the adaptive update law (Table 2 and Table 3). These two cases are evaluated in terms of root-mean-square error (RMSE) for 16 separate experiments which prioritise the disturbance rejection: 4 human-assisted, 4 without interaction, and 8 human-disturbed; half of the experiments per scenario with and the other half without the adaptive update law.

This analysis shows that the performance of both the base and the complete solution are similar during transients, with comparable position errors and a marginally superior attitude response for the complete strategy. In both cases, nonetheless, the transient performance is sufficient for the fast and nontrivial tracking scenario proposed. For instance, the (L) position norm RMSE in both is $\sim$30 cm for a reference change of $\sim$85 cm, which is comparable to a first-order response with a time constant (63%) of 1.2 s.

In contrast, the advances in the steady-state regime due to the adaptive estimation of the human-induced wrench are substantial. The complete strategy clearly outperforms the base solution in all variables included in the task-space, except for the x position of the load –which grows marginally–, and z position of $(\mathrm{R})_1$ and y position of $(\mathrm{R})_2$ –whose decrease is insignificant–. This improvement is especially important in the (L) attitude, where the yaw RMSE is halved and the overall orientation error (in norm) is reduced by a 37% –from 0.21 rad (12°) to 0.13 rad (7°)–, as already detected in the analysis of the results in Fig. 25.

[4]The transition between both has been empirically set to $t = 5\mathrm{s}$.

This project is funded by the European Union G.A No 101016906

6 VR experiment and preliminary field experiment of Task T8.2

Before we implement our algorithms on the CANOPIES mobile bases in the field, we tested our LTL planner in the VR environment provided by PaleBlue. The VR environment has a physical engine with a high-fidelity collision models. This environment can be easily integrated with Gazebo. It also provides a ROS interface, which is the same as the real robot platform, making it an ideal place for testing.

We have tested our online re-planning algorithms in VR simulation environment. One screenshot for such an experiment is shown in Fig. 26 where the two robots are on monitoring tasks of several different regions in the vineyard, and the human avatar is walking around. The robotic systems can successfully navigate and accomplish their tasks in this obstacle-cluttered environment and account for human's movement by local coordination.

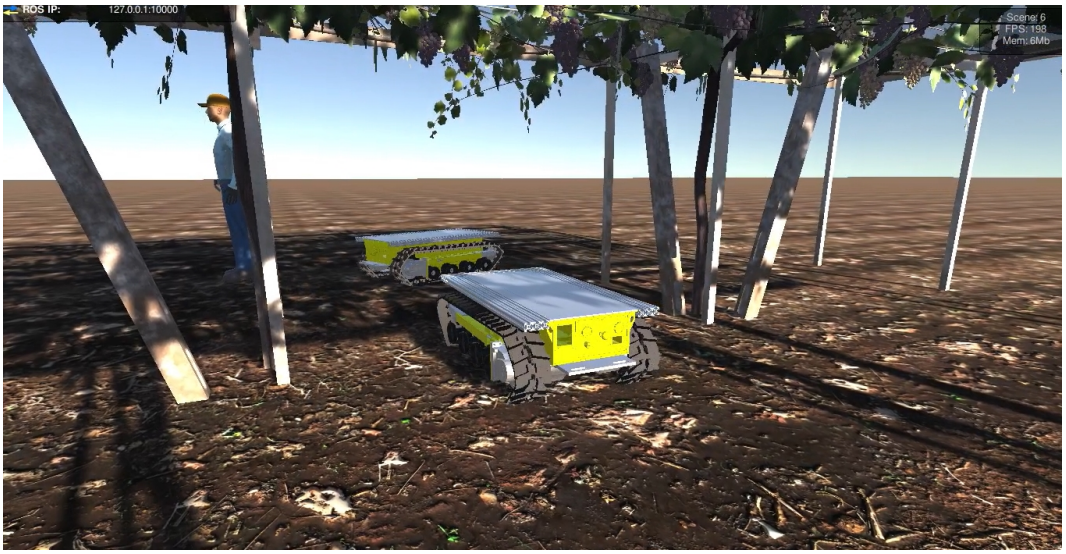


Figure 26: VR vineyard environment with two CANOPIES mobile bases and one human avatar. Here each robot is running an LTL planner and accomplishing their tasks, while accounting for the movement of the other robot and the human avatar that resembles an end-user in the field.

We have also conducted preliminary field experiments with our LTL planner to demonstrate online task replanning capabilities when there are humans in the shared physical environment. We use two mobile bases to demonstrate multi-robot task planning with one human in the shared workspace. In alignment with the objectives of WP8.2, the human interact with the robots in two different ways in our experimental setup: firstly human is present in the field as a dynamic obstacle, secondly human operators directly control one robot through the Alitrack controller device and this robot is acting as a moving obstacle. In both of these setups, humans create obstruction to the motion of the autonomous agent(s). As the autonomous robot detects a conflict and temporarily deviates from its original planned path due to avoid collisions, an online re-planning is triggered, and the robot follows this new plan once it is safely able to do so.

This project is funded by the European Union G.A No 101016906



Figure 27: Robotic and computer hardware for field validation. Each mobile base has a planner NUC that communicated with a Laptop. The human user can send real-time commands and monitor the robot navigation and planning through the laptops for field testing.

While choosing the agronomic tasks for our demonstrations, we have appropriately adapted to the availability of the robotic hardware (see Fig. 27) on field. We tested the scenarios where human is moving around the field (Fig. 28) and where one of the robots is fully remotely controlled by a human operator (Fig. 29) and the other robot needs to coordinate locally to ensure safety and task fulfillment. The preliminary field demonstrations have successfully validated the applicability of the LTL planner and our approaches in a real operational context.

We note that the preliminary demonstrations related to WP8.2 do not have any manipulation tasks or Box exchange tasks. Agronomic tasks such as “picking grapes” or “loading/unloading box” are instead performed by humans while the mobile bases move appropriately within the field to cooperate with the humans in carrying out these tasks. These functionalities are expected to be integrated toward the final demonstration. More details will be found in the Deliverable 9.3 report.

This project is funded by the European Union G.A No 101016906



Figure 28: Physical interaction such as obstructing robots. The humans in the shared workspace can act as dynamic obstacles and activate safety controllers in the robots, forcing them to perform collision avoidance maneuvers.



Figure 29: One of the mobile bases is tele-operated using the Alitrack handheld controller. This robot acts as a dynamic obstacle for the other robot, which is expected to avoid collisions and replan its tasks accordingly.

This project is funded by the European Union G.A No 101016906

7 Conclusions

This report documents the research developments of automated synthesis for safety-assuring motion coordination strategies for autonomous robotic systems that are given complex task and safety specifications with human agents in the shared workspace. We develop on-line re-planning and safe-by-construction approaches that are essential to account for humans moving through the workspace.

In this report, we present our research advancements from the following three aspects:

- proposing an online conflict resolution algorithm on the planning level for multi-robot systems. The proposed strategy guarantees to resolve possible conflicts among neighboring agents, accomplish local LTL tasks, and ensure the safety of the human operator and other moving robots (Section 3);
- developing a distributed CBF-based safety filter for arbitrary control signals in a minimal invasive manner (Section 4);
- designing an algorithm for robotic manipulators to infer the force contributions of human workers and take that into account in the control loop (Section 5).

Besides all the theoretical guarantees detailed in each section, we highlight that all the results are also accompanied with experimental demonstrations at a local facility in KTH and in a virtual VR environment. Up-to-date field validation experiments are also briefly summarized, with more details and more field experiments to be reported in Deliverable D9.3.

This project is funded by the European Union G.A No 101016906

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

[2] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[3] Calin Belta, Antonio Bicchi, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George J. Pappas. Symbolic planning and control of robot motion [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):61–70, 2007.

[4] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[5] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[6] Alphan Ulusoy, Stephen L Smith, Xu Chu Ding, Calin Belta, and Daniela Rus. Optimality and robustness in multi-robot path planning with temporal logic constraints. *The International Journal of Robotics Research*, 32(8):889–911, 2013.

[7] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Multirobot coordination with counting temporal logics. *IEEE Transactions on Robotics*, 2019.

[8] Yiannis Kantaros and Michael M Zavlanos. STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *The International Journal of Robotics Research*, 39(7):812–836, 2020.

[9] Javier Alonso-Mora, Jonathan A DeCastro, Vasumathi Raman, Daniela Rus, and Hadas Kress-Gazit. Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles. *Autonomous Robots*, 42(4):801–824, 2018.

[10] Meng Guo and Dimos V Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[11] Meng Guo and Michael M Zavlanos. Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Transactions on Automatic Control*, 63(12):4051–4066, 2018.

[12] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

[13] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003.

[14] Meng Guo, Karl H Johansson, and Dimos V Dimarogonas. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *2013 IEEE international conference on robotics and automation*, pages 5025–5032. IEEE, 2013.

This project is funded by the European Union G.A No 101016906

[15] Morteza Lahijanian, Matthew R Maly, Dror Fried, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics*, 32(3):583–599, 2016.

[16] Pian Yu and Dimos V Dimarogonas. Distributed motion coordination for multirobot systems under LTL specifications. *IEEE Transactions on Robotics*, 38(2):1047–1062, 2021.

[17] Savvas G Loizou and Vijay Kumar. Mixed initiative control of autonomous vehicles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1431–1436. IEEE, 2007.

[18] Meng Guo, Sofie Andersson, and Dimos V Dimarogonas. Human-in-the-loop mixed-initiative control under temporal tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6395–6400. IEEE, 2018.

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

[20] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters*, 1(2):310–315, 2017.

[21] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Distributed coordination control for multi-robot networks using lyapunov-like barrier functions. *IEEE Transactions on Automatic Control*, 61(3):617–632, 2015.

[22] Mrdjan Jankovic, Mario Santillo, and Yan Wang. Multi-agent systems with cbf-based controllers – collision avoidance and liveness from instability. *arXiv*, 07 2022.

[23] Beatrice Capelli and Lorenzo Sabattini. Connectivity maintenance: Global and optimized approach through control barrier functions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5590–5596. IEEE, 2020.

[24] Lars Lindemann and Dimos V Dimarogonas. Barrier function based collaborative control of multiple robots under signal temporal logic tasks. *IEEE Transactions on Control of Network Systems*, 7(4):1916–1928, 2020.

[25] J Richard Büchi. On a decision method in restricted second order arithmetic. In *The Collected Works of J. Richard Büchi*, pages 425–435. Springer, 1990.

[26] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *International Conference on Computer Aided Verification*, pages 53–65. Springer, 2001.

[27] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings*, 40(12):462–467, 2007.

[28] Xiangru Xu, Paulo Tabuada, Jessy W. Grizzle, and Aaron D. Ames. Robustness of control barrier functions for safety critical control. In *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, volume 48, pages 54–61, 2015.

[29] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

This project is funded by the European Union G.A No 101016906

[30] Xiao Tan, Wenceslao Shaw Cortez, and Dimos V. Dimarogonas. High-order barrier functions: robustness, safety and performance-critical control. *IEEE Transactions on Automatic Control*, Early access, 2021.

[31] Cristian Ioan Vasile and Calin Belta. Sampling-based temporal logic path planning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4817–4822, 2013.

[32] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *AAAI*, pages 1230–1237, 2010.

[33] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, (5):293–308, 2001.

[34] Michael Melholt Quottrup, Thomas Bak, and RI Zamanabadi. Multi-robot planning: A timed automata approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4417–4422, 2004.

[35] Thomas M Howard, Colin J Green, and Alonzo Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Field and Service Robotics*, pages 69–78. Springer, 2010.

[36] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.

[37] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[38] Yuanbo Nie, Omar Faqir, and Eric C Kerrigan. ICLOCS2: Solve your optimal control problems with less pain. *IFAC-PapersOnLine*, 2018.

[39] David Parsons and John Canny. A motion planner for multiple mobile robots. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 8–13, 1990.

[40] Mo Chen, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Robust sequential trajectory planning under disturbances and adversarial intruder. *IEEE Transactions on Control Systems Technology*, 27(4):1566–1582, 2018.

[41] Michael Otte and Emilio Frazzoli. RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7):797–822, 2016.

[42] Hsien-Chung Lin, Changliu Liu, and Masayoshi Tomizuka. Fast robot motion planning with collision avoidance and temporal optimization. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 29–35, 2018.

[43] Cristian Ioan Vasile and Calin Belta. Reactive sampling-based temporal logic path planning. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4310–4315, 2014.

[44] M Kloetzer and C Belta. LTLCon: software for control of linear systems from LTL formulas over linear predicates. *Boston Univ., Boston, MA*, 2005.

This project is funded by the European Union G.A No 101016906

[45] Robin Baran, Xiao Tan, Peter Varnai, Pian Yu, Sofie Ahlberg, Meng Guo, Wenceslao Shaw Cortez, and Dimos V Dimarogonas. A ros package for human-in-the-loop planning and control under linear temporal logic tasks. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2182–2187. IEEE, 2021.

[46] Xiao Tan and Dimos V. Dimarogonas. Distributed implementation of control barrier functions for multi-agent systems. *IEEE Control Systems Letters*, 6:1879–1884, 2022.

[47] Giuseppe Averta, Cosimo Della Santina, Edoardo Battaglia, Federica Felici, Matteo Bianchi, and Antonio Bicchi. Unvealing the principal modes of human upper limb movements through functional analysis. *Frontiers in Robotics and AI*, 4:37, 2017.

[48] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics. Modelling, planning and control*. Springer-Verlag (UK), 2009.

[49] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. springer, 2016.

[50] Alessandro De Luca, Alin Albu-Schaffer, Sami Haddadin, and Gerd Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630. IEEE, 2006.

[51] T. Tsumugiwa, R. Yokogawa, and K. Hara. Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task. In *2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 644–650 vol.1, 2002.

[52] Christos N. Mavridis, Konstantinos Alevizos, Charalampos P. Bechlioulis, and Kostas J. Kyriakopoulos. Human-robot collaboration based on robust motion intention estimation with prescribed performance. In *2018 European Control Conference (ECC)*, pages 249–254, 2018.

[53] Matthew Dyck and Mahdi Tavakoli. Measuring the dynamic impedance of the human arm without a force sensor. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–8, 2013.

[54] Qualisys AB. Qualisys Motion Capture System. `https://www.qualisys.com/`. Online; accessed 12 September 2021.

[55] MathWorks. Robotics System Toolbox. `https://uk.mathworks.com/products/robotics.html`. Online; accessed 12 September 2021.

[56] P.A. Ioannou and J. Sun. *Robust Adaptive Control*. Dover Books on Electrical Engineering Series. Dover Publications, Incorporated, 2012.

[57] HEBI Robotics. HEBI Robotics MATLAB API. `https://docs.hebi.us/tools.html#matlab-api`. Online; accessed 12 September 2021.