Reactive and human-in-the-loop planning and control of multi-robot systems under LTL specifications in dynamic environments*

Pian Yu[1], Gianmarco Fedeli[2], and Dimos V. Dimarogonas[3]

***Abstract*— This paper investigates the planning and control problems for multi-robot systems under linear temporal logic (LTL) specifications. 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. Depending on whether local communication is allowed between robots, we consider two different online re-planning approaches. When local communication is allowed, we propose a local trajectory generation algorithm for each robot to resolve conflicts that are detected on-line. In the other case, i.e., no communication is allowed, we develop a model predictive controller to reactively avoid potential collisions. In both cases, task satisfaction is guaranteed whenever it is feasible. In addition, we consider the human-in-the-loop scenario where humans may additionally take control of one or multiple robots. We design a mixed initiative controller for each robot to prevent unsafe human behaviors while guarantee the LTL satisfaction. Using our previous developed ROS software package, several experiments are conducted to demonstrate the effectiveness and the applicability of the proposed strategies.**

I. INTRODUCTION

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]–[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]–[5] and multi-robot systems (MRSs) [6]–[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 online replanning becomes a necessity, the problem of safe and efficient replanning under LTL specifications becomes challenging during the online execution of robots.

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.

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, it 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.

This work aims to develop reactive planning and control algorithms for MRSs under LTL specifications. First, we build upon our prior research [16] by addressing dynamic environments, which present a non-trivial challenge. Additionally, when humans participate in controlling the robots, we expand the MIC proposed in our previous work [18] to accommodate MRSs. The contributions are threefold: i) Depending on whether local communication is allowed between robots, we propose two different online re-planning approaches for MRSs operating in dynamic environments. In the first scenario, where local communication among robots is allowed, we propose a local trajectory generation algorithm to resolve conflicts. In the second scenario, where there is no communication between the robots, we develop a model predictive controller to reactively handle potential collisions. In both cases task satisfaction is guaranteed whenever it

*This work was partially supported by the Swedish Research Council (VR), the Knut and Alice Wallenberg Foundation (KAW), the EIC Horizon Europe SymAware, the EU project CANOPIES, the ERC COG LEAFHOUND (Grant agreement ID: 864720), and the ERC ADG FUN2MODEL (Grant agreement ID: 834115).

[1]Pian Yu is with the Department of Computer Science, Oxford University, Oxford, United Kingdom pian.yu@cs.ox.ac.uk

[2]Fedeli Gianmarco is with Bosch, Braga, Portugal gianmarco.fedeli1@gmail.com

[3]Dimos Dimarogonas is with the Division of Decison and Control Systems, KTH, Stockholm, Sweden dimos@kth.se

is feasible. ii) When one or multiple robots are controlled by human operators, a MIC is designed for the MRSs to guarantee safety and LTL task satisfaction. iii) Several experiments are carried out to demonstrate the effectiveness of the proposed strategies.

II. Preliminaries and Problem Formulation

A. 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 ($:$), conjunction ($\wedge$), and disjunction ($\vee$), and the temporal operators next ( ), until (U), eventually ( ), and always ( ). An LTL formula is formed inductively according to the following syntax [1]:

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

where $a \in AP$; φ; φ_1; φ_2 are LTL formulas. The logic connective $\vee$ and temporal operators and 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) [19], defined as a tuple $\mathcal{B} = (S, S_0, 2^{AP}, \delta, F)$, 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* $s = s_0 s_1 \cdots$ is called *accepting* if $\mathtt{Inf}(s) \cap F \neq \emptyset$, where $\mathtt{Inf}(s)$ is the set of states that appear in s infinitely often. Several translation tools, e.g., LTL2BA [20], are available to obtain $\mathcal{B}$ given φ.

B. Robot dynamics and motion abstraction

Consider a set of N robots navigating within a bounded workspace W. The dynamics of robot i is given by:

$$\begin{aligned} \dot{x}_i(t) &= v_i(t)\cos(\theta_i(t)), \\ \dot{y}_i(t) &= v_i(t)\sin(\theta_i(t)), \\ \dot{\theta}_i(t) &= w_i(t), \end{aligned} \tag{2}$$

where $p_i = [x_i, y_i]^T$ is the Cartesian position, θ_i is the orientation, and $u_i = [v_i, w_i]^T$ is the input vector of robot i. The input of robot i is constrained to the compact set U_i, i.e., $u_i(t) \in U_i, \forall t \geq 0$.

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

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

Given an observation preserving partition of the workspace, the dynamics of robot i (2) can be abstracted as a controlled transition system (CTS) $\mathcal{T}_i = (\Pi, \Pi_{0,i}, U_i, AP, \rightarrow_i, L)$, where $\Pi_{0,i} \in \Pi$ is the initial state and $\rightarrow_i \subseteq \Pi \times U_i \times \Pi$ is the transition relation.

C. Objectives

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 detect other robots or the unknown moving obstacles only if they are within its sensing region $B(p_i, R_i)$, 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 section) 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, i.e., the CTS $\mathcal{T}_i$, and the task specification φ_i of each robot i, the first objective (O1) 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. In addition, we further consider the human-in-the-loop context, where human operators can take over the control of the robots from the on-board autonomous controller. The second objective (O2) is to design control algorithm for each robot i, which can react to (possibly dangerous) human inputs while preserving safety and task satisfaction.

III. Online Replanning in Dynamic Environments

Before implementation, an initial satisfying plan needs to be generated for each robot i. This procedure is implemented in the package LTL core & planner [21]. It is based on constructing a product Büchi automaton (PBA) $\mathcal{P}_i$ between the CTS $\mathcal{T}_i$ and the NBA $\mathcal{B}_i$ (which is translated from φ_i). Using model-checking methods [12], an accepting run can be obtained from the PBA $\mathcal{P}_i$ and projected back to the CTS $\mathcal{T}_i$ intersection $\mathcal{B}_i$. Accepting runs have a *prefix-suffix* structure of this kind: $r_{\mathcal{P}_i} = p_0 p_1 \cdots p_k (p_{k+1} \cdots p_n p_k)^{\omega}$. The output word is composed of two separate parts: a finite prefix that is executed only once from the initial state p_0 to an accepting state p_k and a suffix that is repeated infinitely from the accepting state p_k to itself. The accepting run has a corresponding action sequence that each robot must carry out in a prefix-suffix structure in order to fulfill φ_i.

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.

A. 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 $p_i([t; t + \ _i])$ the local trajectory of robot i, where $\ _i = \min_{t' \ t} \{p_i(t^0) \ 2 \ B_i(p_i(t); R_i)\}$. We say there is a *conflict* between robot i and j at time t if there exists a region $X_k \ 2$ such that $p_i([t; t + \ _i]) \setminus X_k \ 6= \ ;$ and $p_j([t; t + \ _j]) \setminus X_k \ 6= \ ;$. This means that the local trajectories of robots i and j pass through the same region X_k.

Using the local trajectory information $p_j([t; t + \ _j])$ broadcast by the neighboring robots $j \ 2 \ N_i(t) := \{j : kp_i(t) \ \ p_j(t)k \ \ R_i\}$, 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 consists of a local and a global trajectory generation algorithms, which are built upon our previous work [16].

The local trajectory generation algorithm is detailed in Algorithm 1. 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 1 localTrajectoryGeneration

Input: $\ _i$, B_i, P_i; V_{P_i}; O; O_{conf}
Return: A local transition system T_i^L and a leaf node $\ _i^f$.

1: Initialize $T_i^L = (S_i^L; S_{i;0}^L; U_i; AP; \ !_i^L; L)$ and $\ _i^f = \ ;$, where $S_i^L = S_{i;0}^L = \ _i$ and $\ !_i^L = \ ;$.
2: **for** $k = 1; : : : ; N_i^{\max}$ **do**,
3: $\ _s \ \ generateSample(SA_i)$,
4: $\ _n \ \ nearest(S_i^L; \ _s)$,
5: Solve the optimization program $P(\ _n; \ _s; \ _s)$, which returns $(\ _r; u_i)$,
6: $B_i(\ _r) \ \ trackBuchiStates(B_i)$,
7: **if** $B_i(\ _r) \ 6= \ ; \ ^ \ V_{P_i}(\ _r; B_i(\ _r)) < 1$ **then**,
8: **if** $\text{proj}_2([\ _n; \ _r])$ *isObstaclesFree*$(O; O_{conf}) \ ^$ *safeMotion*$(\ _r; R_{safe})$ **then**,
9: $S_i^L \ \ S_i^L \ [\ f \ _r g; \ !_i^L = !_i^L \ [f \ _n \ ^{u_i^*}! \ \ _r g$,
10: **end if**
11: **end if**
12: **if** $\text{proj}_2(\ _r) \ 2 \ B(p_i; R_i)$, **then**
13: $k = N_i^{\max} + 1$,
14: $\ _i^f \ \ \ _r$,
15: **end if**
16: **end for**

Algorithm 1 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 $'_i$. It takes as input the current state $\ _i = (x_i; y_i; \ _i)$ of robot i, the offline constructed NBA B_i and PBA P_i, the offline computed potential function V_{P_i} (the definition and computation can be found in [16]), the set of known static obstacles O, and the set of conflict regions O_{conf}^t as input. Firstly, a local transition system T_i^L is initialized (line 1). At each iteration, a new state $\ _s$ is taken randomly from the sampling area $SA_i := \{p; \ ) : p \ 2 \ B(p_i; R_i + \ )\}$ (line 3), where $\ > 0$ is an offline constant which ensures that one can sample outside of the sensing region $B(p_i; R_i)$, where p_i is the current position of robot i. Then through the function *nearest* $S_i^L; \ _s$ an RRT primitive is applied, which returns the nearest state to $\ _s$ in S_i^L (line 4). At this stage an optimization problem $P(\ _n; \ _s; \ _s)$ is solved, so as to find the closest reachable state from the new sample $\ _s$:

$$\min_{u_i 2 U_i} k \ _r \ \ \ _s k$$

$$\text{subject to: } \ _i(0) = \ _n;$$

$$\ _r = \ _n + \int_0^{\ _s} F_i(\ _i(s); u_i) \, ds;$$

$$u_i \ 2 \ U_i;$$

where $\ _s$ represents the sampling time, while $F_i(\ _i(s); u_i)$ describes the robot's dynamics (2) (line 5). Once $\ _r$ is obtained, the corresponding subset of valid Büchi states $B_i(\ _r)$ is computed using algorithm *trackBuchiState* (given in [22], Algorithm 1) (line 6). If both conditions $B_i(\ _r) \ 6= \ ;$ and $V_{P_i}(\ _r; B_i(\ _r)) < 1$ are satisfied (which ensures the existence of a path originating from $\ _r$ that leads to a self-reachable accepting state of P_i), then a potential new state is considered. Such state $\ _r$ is added into S_i^L and the corresponding transition relation $\ _n \ ^{u_i^*}! \ \ _r$ is added into $!_i^L$, if the path connecting $\ _n$ with $\ _r$ 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 $\text{proj}_2([\ _n; \ _r])$ and the set of obstacles $O \ [\ O_{conf}$. If the line segment $\text{proj}_2([\ _n; \ _r])$ is collision-free, we further check the safety of the motion using algorithm *safeMotion*$(\ _r; R_{safe})$ (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 $\ _i^f$ is subsequently returned (as determined by the corresponding state $\ _r$) (lines 12-15).

The global trajectory generation is similar to the initial trajectory generation, which uses the leaf node $\ _i^f$ (which is obtained by Algorithm 1) as input of the LTL core & planner, and replans the sequence of actions in order to accomplish the LTL task $'_i$.

B. 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 P_i of robot i, the set of all trap states is denoted G_i. An algorithm for computing trap states is implemented in [21].

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 $B_i(p_i; R_i)$. Let O^t_{obs} be the set of moving obstacles detected at time t. In order to avoid trap states G_i (thus guaranteeing task feasibility), the set of known static obstacles O, and the set of moving obstacles O^t_{obs} (thus guaranteeing safety), we consider a local model predictive controller:

$$\begin{aligned}\min_{u_i} \quad & \int_{t=0}^{T} (\ _i(t) \quad X_{des})^T Q(\ _i(t) \quad X_{des}) + u_i^T R u_i \\ & + (\ (T) \quad X_{des})^T Q_N(\ (T) \quad X_{des}) \\ & + \int_{t=0}^{T} w_{O_i} \frac{1}{dist(\ (t); O^t_{obs} [\ O)} + w_{G_i} \frac{1}{dist(\ (t); G_i)}\end{aligned}$$

$$\text{subject to: } \ _i(t) = \ _i(0) + \int_0^{\cdot} F_i(\ _i(s); u_i)\, ds;$$

$$u_i \ 2 \ U_i;$$

$$\ _i(T) \ 2 \ X_{des}$$

$$\ _i(0) = \ _i; \tag{3}$$

where T is the horizon and $\ _i$ 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 $X_{des} \ 2$ 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 $O^t_{obs} [O$ (which is updated at each time step) and the distance between robot i and the trap states G_i, where the distance function is defined as $dist(x; A) := \inf_{y \ 2 \ A} fjjx \quad ykg$ and $w_{O_i}; w_{G_i}$ 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_{G_i} can be set higher than w_{O_i}.

IV. Human-in-the-loop control

In this section, we consider the human-in-the-loop context, where the robots are assigned to complete complex tasks specified by an LTL formula, while a human operator can take over the control of the robot from the on-board autonomous controller. The robot is required to respect human inputs, but at the same time react to undesired (possibly dangerous) human behaviors in order to preserve safety and task satisfaction. This is useful for guiding the robot through challenging assignments.

The proposed strategy is a MIC with the intention of extending existing work [17], [18] to MRSs. The MIC, inspired by [17], [18], for robot i is given by:

$$u_i(t) \ , \ u_i^r(t) + \quad (r; O; G_i)\, u^h(t); \tag{4}$$

where $u_i^r(t)$ is a given autonomous controller, the function $(\ _i; O; G_i) \ 2 \ [0; 1]$ is a smooth function to be designed, and $u^h(t)$ is the human input function, which is uncontrollable and unknown by the robot.

The autonomous controller $u_i^r(t)$ can be a function that navigates the robot from one region X_s of the current discrete plan to the next one X_g while staying within the workspace W when there are no conflicts/obstacles are detected. Otherwise, the local trajectory generation algorithm or the model predictive controller developed in Section III is implemented to get $u_i^r(t)$. In order to guarantee the task satisfaction for all human inputs, the function $(\ _i; O; G_i)$ is designed as:

$$\begin{aligned}(\ _i; O; G_i) \ , \ & G_{\text{mix}} \ \frac{(d_o \quad d_s)}{(d_o \quad d_s) + (" + d_s \quad d_t)} \\ & + (1 \quad G_{\text{mix}}) \ \frac{(d_t \quad d_s)}{(d_t \quad d_s) + (" + d_s \quad d_t)}\end{aligned}$$

where $d_t \ , \ \min_{2 G_i} k \ _i \quad k$ is the minimum distance between robot i and any region within G_i; $d_o \ , \ \min_{2 O^t_i} k \ _i \quad k$ is the minimum distance between the robot and any obstacle within O; $(s) \ , \ e^{\ 1=s}$ for $s > 0$ and $(s) \ , \ 0$ for $s \quad 0$, and $d_s; " > 0$ are design parameters as the safety distance and a small buffer. Moreover $G_{\text{mix}} \ 2 \ [0; 1]$ represents a gain parameter, in order to manage the trade-off between two aspects: preventing trap states and obstacle avoidance.

V. Experimental results

In this section, we present experimental studies to validate our results, which is also the main contribution of this paper. 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. 1 left), which is an omnidirectional mobile platform with three omnidirectional wheels. 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 $5m \quad 6m$ region as shown in Fig. 1 right, which is abstracted into 30 $1m \quad 1m$ squares. We consider a group of 2 or 3 HEBI Rosie robots (Rosies 0, 1, and 2), and the dynamics of each robot is given by (2). In addition, each Rosie is subject to the inputs constraints $jv_ij \quad 0{:}35m{=}s$ and $jw_ij \quad 0{:}35rad{=}s$. The sensing radius of each robot is $R_i = 0{:}8m; 8i$.

A. Online replanning

In this section, we consider $N = 2$ HEBI Rosie robots. The LTL specification for each robot is given by

$$'_0 = ((\quad R8) \ ^ \ (\quad R20)),$$
$$'_1 = (\quad R17) \ ^ \ (\quad R21).$$

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

Fig. 1. Rosie HEBI mobile base (left) and workspace discretization (right).

1) Local communication case: Initially each robot synthesizes a satisfying trajectory using the LTL core & planner [21]. 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. 2. During the experiment time horizon 140s, both robots successfully complete the surveillance 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. A video demonstration of this experiment can be found at `https://www.youtube.com/watch?v=mKWpqvMrW9Y`.

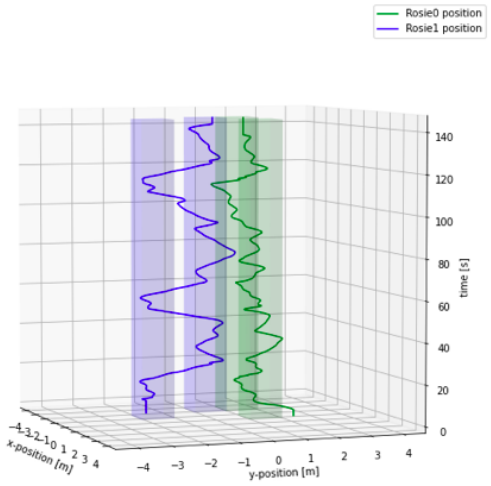

Fig. 2. Local communication case: the real-time position trajectories of Rosies 0 (green line) and 1 (purple line). The light green and purple regions represent the target regions of Rosies 0 and 1, respectively.

2) Communication-free case with human as moving obstacle: In this case, we additionally consider a human walking in the workspace W to evaluate the effectiveness of the model predictive controller.

Fig. 3 depicts the real-time position trajectories of the 2 Rosies and the human, where the proposed model predictive controller (3) is applied for both Rosies. Differently from the local communication case where replanning is activated

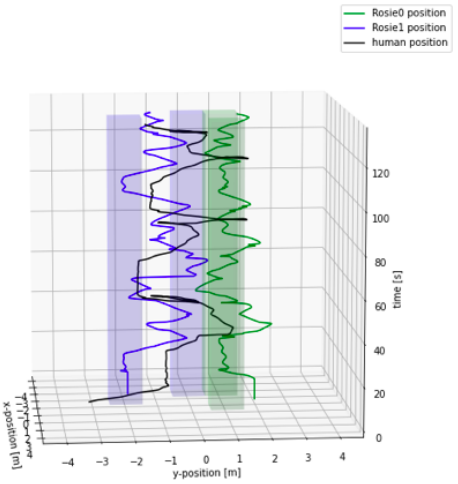

Fig. 3. Communication-free case: the real-time position trajectories of Rosies 0 (green line), 1 (purple line), and human (black line).

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 (3) 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 video demonstration of this experiment can be found at `https://youtu.be/TYgfbrk7hDs`.

B. Human in-the-loop control

Fig. 4. Human in-the-loop control.

This section aims to evaluate the MIC explained in Section V in the human in-the-loop context, where a human may take control of one or more robots during the motion. We consider $N = 3$ HEBI Rosie robots and the LTL specification for each robot is given by:

$$'_0 = (\quad R8) \wedge (\quad R20),$$
$$'_1 = (\quad R22) \wedge (\quad R28),$$
$$'_2 = (\quad R10) \wedge (\quad R11).$$