Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control Systems Engineering

Reactive task planning for multi-robot systems in partial known environment

Master Candidate

Fedeli Gianmarco

Student ID 2013377

Supervisor

Prof. Chiuso Alessandro

University of Padova

KTH supervisor

Postdoc. Pian Yu

KTH Royal Institute of Technology

Academic Year
2021/2022

To my family and beyond,

of all of the support that I received over the years, the greatest undoubtedly comes from my parents. Words cannot express my gratitude to them, without whom nothing would be possible. Many thanks for the opportunities they give to me along this journey, as well as their patience and most importantly their belief in me from the beginning.

I really want to thank and acknowledge my uncles Stefano and Stefania for their emotional support and for always being there for me. A big thank you to my cousins: Giacomo, who has been a great source of encouragement along the way, Francesca and Marta for their unconditional support.

I would like to express my sincere gratitude to professor Chiuso for his availability and for making possible my experience abroad at the prestigious KTH Royal Institute of Technology in Sweden.

I really want to thank professor Dimos Dimarogonas, my examiner and head of the Division of Decision and Control Systems at KTH, who took care of my passions and ambitions.

I would like to especially thank my KTH supervisor Pian Yu, guidance and overall insights in this field who have made this an inspiring experience for me and leading me to a professional growth.

A special thank you to all my other family members and especially to my grandma Ileana.

I could not mention my colleague, friend and Stockholm flatmate Achille Policante for his advices, working mindset and moral support.

Above all, I must offer a special thank you to all my friends, you will never know how much strength and encouragement you have given me.

Abstract

The thesis investigates the planning and control problem for a group of mobile agents moving in a partially known workspace. A task will be assigned to each robot in the form of a linear temporal logic (LTL) formula. First an automaton-based method is introduced for the motion planning of a single agent, which guarantees the satisfaction of the assigned LTL task. Then a model-predictive controller considers state and input constraints leading the agent to a safe navigation. Based on a real scenario of a partial-known environment and agents can have only local sensing, two decentralized control strategies are proposed for online re-planning, which rely on a sampling-based algorithm. The first approach assumes local communication between agents, while the second one exploits a more general communication-free case. Finally, the human-in-the-loop scenario is considered, where a human may additionally take control of the agents, a mixed initiative controller is then implemented to prevent dangerous human behaviors while guarantee the satisfaction of the LTL specification.

Using the developed ROS software package, several experiments were carried out to demonstrate the effectiveness and the potential applicability of the proposed strategies.

Sommario

La tesi indaga il problema della pianificazione e del controllo per un gruppo di agenti mobili che si spostano in uno spazio di lavoro parzialmente conosciuto. Verrà assegnato un compito a ciascun robot sotto forma di una formula di logica temporale lineare (LTL). Innanzitutto viene introdotto un metodo basato sull'automa per la pianificazione del movimento di un singolo agente, che garantisce il soddisfacimento del compito LTL assegnato. Un controllore predittivo del modello considera i vincoli di stato e gli input, assicurando una navigazione sicura dell'agente. Basandosi su uno scenario reale di un ambiente parzialmente noto e in cui gli agenti possano avere solamente un rilevamento locale, vengono proposte due strategie di controllo decentralizzato per la ripianificazione online, che si basano su un algoritmo di campionamento dello spazio di stato. Il primo approccio presuppone una comunicazione locale tra gli agenti, mentre il secondo considera un caso più generale privo di comunicazione. Infine, viene considerato lo scenario human-in-the-loop, in cui un essere umano può inoltre assumere il controllo degli agenti, viene quindi implementato un controllore per prevenire comportamenti umani pericolosi garantendo allo stesso tempo il soddisfacimento della specifica LTL assegnata. Utilizzando il pacchetto software ROS sviluppato, sono stati effettuati diversi esperimenti per dimostrare l'efficacia e la potenziale applicabilità delle strategie proposte.

Contents

List of Figures

List of Acronyms

MAS Multi-agent systems

MRMC multi-robot motion coordination

LTL linear temporal logic

LT linear-time

TS Transition System

wFTS weighted finite transition system

CTS controlled transition system

NBA non-deterministic Büchi automaton

PBA product Büchi automaton

RRT rapidly exploring random tree

LQR linear quadratic regulator

LQGR linear quadratic Gaussian regulator

MIMO multi-input multi-output

LTI linear-time invariant

MPC Model Predictive Control

OMR omni-directional mobile robot

PID proportional integral derivative controller

MIC mixed-initiative controller

ROS Robot Operating System

QTM Qualisys Track Manager

Chapter 1

Introduction

During the last decade, intelligent agents and multi-agent systems have been a hot research topic and many effective applications of this technology have been developed. Multi-agent systems (MAS) are a main area of present-day artificial intelligence research. A multi-agent system consists of multiple autonomous decisions-making entities, which interact in a shared environment to achieve common or conflicting goals, having often different constraints. These entities do not have to be necessary homogeneous, they could be robots, software agents or even human beings. The control and modeling of complex systems, involving those interacting agents, is the target of MAS research.
The appeal of this new field lies on a variety of advantages. In fact, MAS can address problems that are too complex for a centralized single agent to handle (for example, due to resource constraints or robustness concerns); additionally they improve scalability, provide solutions to intrinsically decentralized problems (such as telecommunication control, workflow management), or provide solutions where expertise is spread. Because of their broad range of applications, such as autonomous driving, smart grids, power systems, automated trading, manufacturing, healthcare, e-commerce, biotechnology and many others, systems of this nature will play a crucial role in real life. Numerous technical issues have existed from the beginning, including how to create effective algorithms that allow one or more agents to accomplish specific goals, how information is shared and propagated among them, and how coordination between agents should be handled to carry out operations safely and effectively. The latest challenge has been studied since the 1980s, but its importance has expanded dramatically in recent years due to the emergence of new applications

(such as smart transportation, service robotics and collaborative robotics).
There are two types of coordination in the existing literature: path coordination and motion coordination. The former plans and coordinates all agents' paths in advance (hence offline), whereas the latter focuses on decentralized and online methods that enable robots to resolve conflicts as they occur.
The purpose of the thesis is to implement and evaluate a completely distributed control algorithm for multi-robot motion coordination, in which each agent has to accomplish a given task while avoiding collisions in a partially known environment. There are primarily two approaches to multi-robot motion coordination (MRMC) that could be distinguished: the reactive approach and the planner-based approach. Reactive methods for motion coordination are suitable in many applications, due to their quickness and ability to perform well in real-time. One possible example of them makes use of potential fields [7] - [14]. In the potential field approach, the motion is characterized by a set of forces. The agent is pulled by attractive forces in the direction of the objective, while repulsive forces push the agent away from obstacles and/or other agents. Consequently, at each location in the configuration space, the agent moves along the vector representing the combined forces acting on that point.
However, as is widely known, reactive techniques suffer from deadlocks generated by local minima and often perform better in unconstrained scenarios. Azarm and Schmidt's work [1], which proposes a strategy for online motion coordination of several mobile robots, is an early example of a planner-based method. When conflicts are found, the given solution comprises of a sequential trajectory planning problem in which a priority hierarchy among robots is assigned. Despite the fact that safety has proven difficult to ensure, future enhancements of this method have been suggested. In addition a lot of the previous works have been focused on relatively simple tasks.
As the convergence of new technologies such as artificial intelligence and big data proves to be effective, demand for robots and their capabilities is continually expanding in multiple areas such as ground, air, and water environments. As a result, in recent years, the area of robot motion planning has been inclined to assign increasingly difficult and high-level tasks, such as temporal logic specifications. The majority of previous LTL approaches are based on a global assigned task, followed by an offline centralized motion planner.
At the same time, those techniques do not account for possible environmental changes (e.g, non-static obstacles, people moving), which must be considered

in real-world applications. The degree project proposes efficient techniques addressing all previously stated challenges in order to extend multi-robot motion coordination to more complicated tasks and practical situations. Firstly in Chapter 4, an automaton-based method for a decentralized motion planning is introduced, which guarantees the satisfation of the assigned task. Such offline global planner provides the sequence of actions that the agent must follow in order to accomplish the given task, when possible collisions are not detected. Secondly in Section 4.3 different navigation methodologies for ensuring safety will be examined, with the employment of a model predictive controller as outcome of such analysis.
Theoretical evidence of safe navigation for each agent under mild constraints, acceptable in real scenarios, will be provided through model predictive control theory. Then, a partially known environment will be considered, leading to a distributed online motion re-planning approach using a sampling-based method, as discussed in Section 5. A first approach assumes local communication between agents, considering a priority hierarchy among them. On the other hand a more generic communication-free case will be analyzed, which adds a collision avoidance algorithm into the online re-planning solution. The latest method will not assume in advance the motion of other agents or non-staic obstacles involved. Finally, Section 5.3 investigates the human-in-the-loop scenario, in which a human may also assume control of the agents, extending the results obtained in [3] to multi-agent systems. Indeed, in recent years, there has been a growing emphasis on human-robot cooperation. The usage of autonomous entities alongside humans in the workplace has decreased ergonomic injuries while boosting safety and productivity.
A mixed initiative controller will be designed for this purpose to prevent dangerous human behaviors while ensuring task satisfaction.
The master's thesis concludes by reporting experimental results and associated conclusions about the overall study and proposed strategies.

Chapter 2

Motivations

This chapter will present some potential applications of multi-robot motion coordination in a partially known environment [1], as well as related failures of an offline strategy and the main difficulties involved.

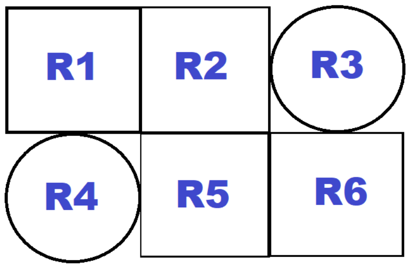
2.1 Delivery and supervision inside a warehouse

An example of a multi-agent motion coordination system is provided, along with an analysis of the main issues and the reasons why an offline approach is obviously not feasible. Consider four mobile agents operating in a shared workspace, such as the warehouse shown in Figure 2.1.

Figure 2.1: Example: warehouse

[1]Partially-known environment generally takes into consideration both the workspace's incomplete knowledge and environmental changes (for example non-static obstacles). In the thesis the workspace is assumed to be entirely known, however extensions to a partially known scenario, where the workspace is not completely known, are possible.

In the following example, the warehouse operating area will be divided into six areas for the sake of simplicity, as illustrated below:



Agents A,B are unmanned aerial vehicles (UAVs) where A is meant to monitore a specific area, while B is required to deliver a box. Agents C and D are unmanned ground vehicles (UGVs), with C having a specified area to monitor and D acting as a base for pick-up and delivery operations. To be more accurate, A and C should continuously check regions R1 and R6, respectively, while D is fixed in region R3. Agent B is then supposed to choose a box from either region R3 or region R4 and transport it to the top of agent D's base. Without going into too much detail about how motion planning is achieved, it makes perfect sense for agents A and C to go to the same region at the same time or, in any event, have trajectories that are very close to each other. Surely, an offline planner that takes into consideration the motion of other agents might be implemented, as well as a safe navigation controller that may prevent collisions between the agents. However, it is clear that even in uncomplicated cases, an offline strategy can be quite risky (as this examle shows).
In reality, the offline technique should account for all potential configurations throughout the workspace that agents may choose to move in. Such a method will be computationally expensive, incredibly ineffective, and frequently slow to react.

Figure 2.2: Example: warehouse agents

Furthermore, taking into account a real-world warehouse setting, enviroment changes might occur. The workspace is a partially-known environment with moving non-static obstacles (such as human beings or other agents), whose movements are typically unpredictable. The inability to integrate those events offline without making assumptions or knowing the motion of objects makes it difficult to ensure that potential collisions won't take place.
In order to better highlight those problems, assume that agents A and B (UAVs) are moving in the x-y plane at a fixed height. Only during the pick or delivery stage (i.e. once it has arrived at the pick/delivery x, y coordinates) Agent B may modify its height. An offline planning scenario could produce the following paths, considering the first six actions:

$$AgentA: \quad R2 \quad --> \quad R5 \quad --> \quad R6 \quad --> \quad R5 \quad --> \quad R4 \quad --> \quad R1$$
$$AgentB: \quad R6 \quad --> \quad R5 \quad --> \quad R4 \quad --> \quad R1 \quad --> \quad R2 \quad --> \quad R3$$

Regardless of the navigation technique used, it is reasonable for the two agents to reach the same area at the same instant, thereby increasing the risk of a collision. A new issue could emerge even if the two agents are equipped with sensors for detecting nearby objects and the navigation strategy takes them into account. A lack of a motion coordination algorithm might actually cause deadlocks, within which the two agents are kept in the same area for a long period of time.
To show that, assume Figure 2.3 to be the agents' configuration at certain instant of time. As can be noticed, Agent A's offline planner suggests taking a route that might be hazardous due to collisions. In fact, if Agent B is not equipped with detection sensors, it won't be able to avoid colliding with Agent A.

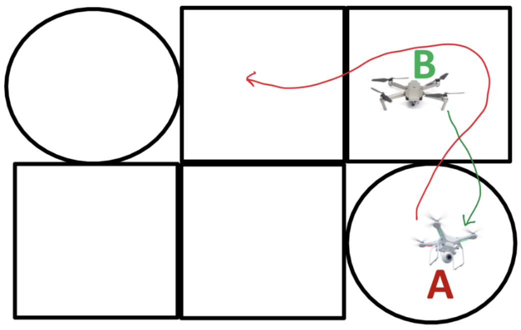


Figure 2.3: Example: possible collision

Additionally, even if collision avoidance algorithms are designed and both agents use detection sensors, the initial offline plan could not be suitable or deadlocks may occur repeatedly (leading the two agents in being stucked and not satisfying the given task).
Therefore, choosing a motion coordination technique that is effective and efficient will be crucial in order to avoid collisions and deadlocks.

2.2 Human being as obstacle

The second example explores the identical scenario as before, restricting the attention on the ground vehicles (agents C, D), with the addition of an unpredictable obstacle into the environment. Considering the first six actions, a hypothetical offline planning may result in the following paths:

$$\begin{aligned} AgentC: &\quad R4 \quad --> \quad R1 \quad --> \quad R2 \quad --> \quad R3 \quad --> \quad R6 \quad --> \quad R5 \\ AgentD: &\quad R6 \quad --> \quad R6 \quad --> \quad R6 \quad --> \quad R6 \quad --> \quad R6 \quad --> \quad R6 \end{aligned}$$

Agent D's offline plan causes to remain in region R6 at all times in order to await the package drop from agent A. In any case, Agent C's approach might be dangerous, as illustrated in the preceding example, given that the next action after region R3 is to reach region R6.
At a certain instant of time, a human being passes through one of the warehouse corridor, resulting in the following scenario:

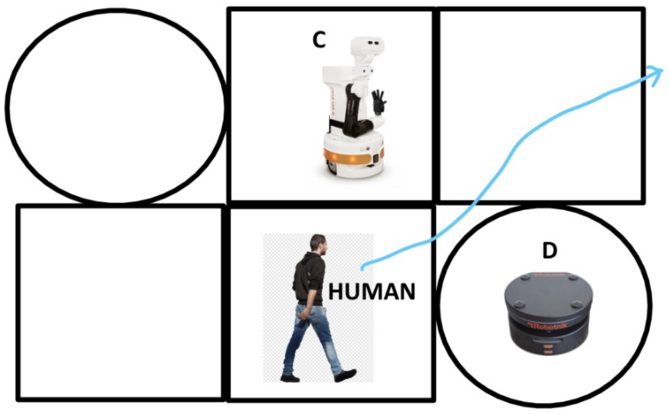


Figure 2.4: Example: human being as obstacle

In contrast to the previous example, introducing a human being into the environment exposes to face new difficulties. With respect to the preceding case, there may be no communication between mobile agents and humans (in some cases, mobile agents communication may be provided) and so the path planning must cope with an unknown and unpredictable obstacle motion.
A possible solution would be to implement a very stable and efficient collision avoidance algorithm, although deadlocks would be extremely difficult to prevent and the initial objective would likely no longer be acceptable or in the worst case violated. Task violation is a prior requirement to avoid, however as will be detailed later, in some instances preventing collisions will take precedence over task satisfaction. Overall, these examples emphasize the importance of a decentralized reactive online replan method capable of coordinating motion among agents, avoiding collisions, and perhaps completing the assigned tasks.

Chapter 3

Preliminaries

3.1 Linear Temporal Logic

This section briefly introduces linear temporal logic (LTL), a logical formalism suitable for defining linear-time (LT) properties and hence capable of specifying complex system properties and tasks. Temporal logic extends propositional or predicate logic by modalities that allow for the infinite behavior of a reactive system to be referred to. They provide a simple but mathematically rigorous syntax for expressing characteristics regarding the relationship between state labels in executions (in other words LT properties).
For example, some elementary temporal modalities included in most temporal logics are the operators:

$\Diamond$	"eventually" (eventually in the future)
$\Box$	"always" (now and forever in the future)

Although the term temporal implies a relationship with the system's real-time behavior, it is only true in an abstract sense. In essence, temporal logic allows for the specification of event order. Some examples are: "the vehicle accelerates once the driver pulls the throttle" or "the race is terminated once a car has completed all set laps", where the meaning does not apply to the specific timing of occurrences.
There are two types of time nature in temporal logics literature: linear time and branching time. The branching view relies on a tree-like structure in which time can break into alternative paths, whereas the linear view has a single successor moment at each point in time. The thesis will be built on linear-temporal logic

(LTL), which is commonly utilized for systems where all components evolved in a lock-step fashion. Furthermore, the linear sequence of time instants might be discrete or continuous, although only the discrete case will be used in the degree project (i.e. the present moment refers to the current state and the next moment corresponds to the immediate succesor state).

3.1.1 LTL syntax

The objective of this subsection is to introduce the syntactic rules based on which formulae in LTL can be established. The basic ingredients of LTL-formulae are atomic propositions (state labels $a \in AP$), the Boolean connectors like conjunction $\wedge$ and negation $\neg$ and lastly two basic temporal modalities $\bigcirc$ (pronounced "next") and U (pronounced "until"). The state label a in a transition system is represented by the atomic proposition $a \in AP$. Generally, atoms are statements regarding the values of system variables that must be true or false, such as "x > 0", "5 is prime", "cars have six wheels" or "x == y".

Definition 3.1.1. *:*
LTL formulae defined over the set AP of atomic propositions are obtained according to the following syntax:

$$\varphi ::= \ true \ \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathrm{U} \varphi_2$$

where $\varphi, \varphi_1, \varphi_2$ are LTL formulas. In addition the Boolean connector disjunction $\vee$ can be derived as $\varphi_1 \vee \varphi_2 := \neg\left(\neg\varphi_1 \wedge \neg\varphi_2\right)$.

The order of precedence on the operators is defined as follows: unary operators bind stronger than the binary ones, $\neg$ and $\bigcirc$ bind equally strong.
The temporal operator U takes precedence over $\wedge, \vee$, and $\rightarrow$, in addition parentheses are omitted whenever appropriate. The until operator allows to derive the most used temporal modalities $\Diamond$ ("eventually") and $\Box$ ("always") as follows:

$$\Diamond\varphi \overset{\text{def}}{=} \text{trueU}\ \varphi \quad \Box\varphi \overset{\text{def}}{=} \neg\Diamond\neg\varphi$$

The combination of those temporal modalities together with Boolean connectives enables the creation of more sophisticated formulas.

For instance, the dual modalities listed below can be defined:

$\Box\Diamond\varphi$	"infinitely often φ "
$\Diamond\Box\varphi$	"eventually forever φ "

According to the first one ("infinitely often φ"), there always exists a moment i in the future such that $i \geq j$ for any moment j, at which a phi-state is visited. While "eventually forever φ" refers to the property that the phi-state will be visited at some point in the future from that moment on.

3.1.2 LTL semantics

LTL formulae relies on path properties, which means that a path can either satisfied an LTL formula or not. It is necessary to establish its semantics in order to comprehend whether a path satisfies an LTL formula. The semantics of LTL formula φ is defined as a language Words(φ), that contains all infinite words over the alphabet 2^{AP} that satisfy φ. That is, a single LT property is associated to every LTL formula.

Definition 3.1.2. *Semantics of LTL (Interpretation over Words):*
Let φ be an LTL formula over a set AP of atomic propositions. The LT property induced by φ is:

$$\text{Words}(\varphi) = \left\{\sigma \in \left(2^{AP}\right)^{\omega} \mid \sigma \models \varphi\right\}$$

where the satisfaction relation $\models \subseteq \left(2^{AP}\right)^{\omega} \times LTL$ is the smallest relation with the properties in Figure 3.1.

Here, for $\sigma = A_0A_1A_2\ldots \in \left(2^{AP}\right)^{\omega}, \sigma[j\ldots] = A_jA_{j+1}A_{j+2}\ldots$ is meant the suffix of σ starting in the $(j+1)$st symbol A_j. Extending those concepts to an interpretation over paths and states, the LTL formula φ holds in state s if all paths starting in s satisfy φ.
For further details about LTL syntax, semantics and model checking refer to [2], while for extra-content about temporal modalities [9].

3.2 Transition System

Transition systems are often used in computer science as models to describe the behavior of systems. Therefore to express dynamic processes with configurations representing states and transitions, specifying how the system can evolve from one state to another. They can be seen as directed graphs where nodes represent states while edges encode model transitions. A state describes some peculiar information of the system at a certain moment of its behavior. For instance, a state of a moving vehicle indicates its current position in the environment. Similarly in a chess game the state is the current placements of pieces on the board. Another example can be a state of a sequential computer program, which indicates the current values of all program variables together with the current value of the program counter that indicates the next program statement to be executed. Regarding transitions: in the case of the moving car a transition may indicate a switch from one x,y position to another, whereas for the sequential program a transition typically corresponds to the execution of a statement and may involve the change of some variables and the program counter. Instead in a chess game, transitions describe the legal moves according to the rules of chess.
In the following transition systems will be described with action names for the transitions (state changes) and atomic propositions for the states. Action names will be used to represent communication mechanisms between processes, while atomic propositions are used to formalize temporal characteristics. More precisely, atomic propositions express simply known facts regarding the states of the system under consideration. Some examples of atomic propositions are x equals 1, or x is greater than 0 for some given integer variable x. Other examples are there is more than twelve employees in the company or there are no students in the school.

$$
\begin{array}{lcl}
\sigma & \models & \text{true} \\
\sigma & \models & a \quad \text{iff} \quad a \in A_0 \quad (\text{i.e., } A_0 \models a) \\
\sigma & \models & \varphi_1 \wedge \varphi_2 \quad \text{iff} \quad \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma & \models & \neg\varphi \quad \text{iff} \quad \sigma \not\models \varphi \\
\sigma & \models & \bigcirc\varphi \quad \text{iff} \quad \sigma[1\ldots] = A_1A_2A_3\ldots \models \varphi \\
\sigma & \models & \varphi_1 \,\mathsf{U}\, \varphi_2 \quad \text{iff} \quad \exists j \geqslant 0.\ \sigma[j\ldots] \models \varphi_2 \text{ and } \sigma[i\ldots] \models \varphi_1, \text{ for all } 0 \leqslant i < j
\end{array}
$$

Figure 3.1: LTL semantics (satisfaction relation |=) for infinite words over 2^{AP}.

Definition 3.2.1. *Transition System (TS)*
A transition system is a tuple $\mathcal{T} = (X, X_0, \Sigma, \longrightarrow, AP, L)$ where:

- *X is the set of states,*
- *$X_0 \subseteq S$ is a set of initial states,*
- *Σ is the set of actions*
- *$\longrightarrow \subseteq X \times X$ is a transition relation,*
- *AP is the set of atomic propositions,*
- *$L : X \longrightarrow 2^{AP}$ is a labeling function.*

The transition system's intuitive behavior can be summarized as follows. The transition system begins in a certain initial state $x_0 \in X_0$ and evolves according with the transition relation $\longrightarrow$. Therefore, if x is the current state, then a transition $x \xrightarrow{\alpha} x'$ obtained from x is selected non-deterministically and picked. This selection procedure is repeated in the next state x' and finishes once an accepting state is encountered (state without outgoing transitions).
It is crucial to emphasize that the "next" transition is determined in a non-deterministic manner when a state includes more than one outgoing transition. As a result, it is impossible to predict the likelihood of each transition since the outcome of the selection process is unpredictable. Similarly, when the set of initial states consists of more than one state, the starting state is selected non-deterministically. The labeling function L relates a set $L(x) \in 2^{AP}$ of atomic propositions to any state x. $L(x)$ intuitively stands for exactly those atomic propositions $a \in AP$ which are satisfied by state x. Given that φ is a propositional logic formula, then s satisfies the formula φ if the evaluation induced by $L(x)$ makes the formula φ true; that is: $x \models \varphi \quad \text{iff } L(x) \models \varphi$.

3.2.1 Weighted Finite Transition System (wFTS)

A weighted finite transition system is a tuple $\mathcal{T}_w = (X, X_0, \Sigma, \to, AP, L, W)$ where:

- $X = \{x_1, \ldots, x_N\}$ is the finite set of states,
- $X_0 \subseteq X$ is the finite set of initial states,
- Σ is the set of actions
- $\longrightarrow \subseteq X \times X$ is a transition relation,
- AP is the set of atomic propositions,
- $L : X \to 2^{AP}$ is a labeling function,
- $W : X \times \Sigma \times X \to \mathbb{R}^+$ is the weight function as cost of transition in $\longrightarrow$.

3.2.2 Controlled transition system (CTS)

Given a transition system $\mathcal{T} = (X,\ \mathrm{X}_0, \Sigma, \to, AP, L)$ and a set of atomic propositions AP, the controlled transition system (CTS) is defined $\mathcal{T}_c = (X, X_0, AP, \to, L_c)$, where $L_c : X \to 2^{AP}$ is a labeling function and the following condition hold.
The labeling function $L_c(x)$ maps a state x to the finite set of atomic propositions AP, which are true at state x.

Given a state $x \in X$, define

$$\mathrm{Post}(x) := \left\{ x' \in X : \exists\, u \in U, x \xrightarrow{u} x' \right\}.$$

An infinite path of the CTS $\mathcal{T}_c$ is a sequence of states $\rho = x_0 x_1 x_2 \ldots$ generated by an infinite sequence of inputs $\boldsymbol{u} = u_0 u_1 u_2 \ldots$ such that $x_0 \in X_0$ and $x_k \xrightarrow{u_k} x_{k+1}$ for all $k \geq 0$. Its trace is the sequence of atomic propositions that are true in the states along the path (i.e. $\mathrm{Trace}(\rho) = L_c\,(x_0)\,L_c\,(x_1)\,L_c\,(x_2)\ldots$).
Indeed the satisfaction relation $\rho = \varphi$ holds if and only if $\mathrm{Trace}(\rho) \in \mathrm{Words}(\varphi)$.

3.2.3 Non-deterministic Büchi Automaton (NBA)

A non-deterministic Büchi automaton (NBA) is a tuple $\mathrm{B} = \left(X, X_0, 2^{AP}, \delta, F\right)$, where the following conditions hold:

1. X is a finite set of states;
2. $X_0 \subseteq X$ is the set of initial states;
3. 2^{AP} is the input alphabet;
4. $\delta : X \times 2^{AP} \to 2^X$ is the transition function;
5. $F \subseteq X$ is the set of accepting states.

An infinite run $\mathbf{x}$ of a NBA is an infinite sequence of states $\mathbf{x} = x_0 x_1 \ldots$ generated by an infinite sequence of input alphabets $\sigma = \sigma_0\sigma_1 \ldots \in \left(2^{AP}\right)^\omega$, where $x_0 \in X_0$ and $x_{k+1} \in \delta\left(x_k, \sigma_k\right), \forall k \geq 0$. An infinite run $\mathbf{x}$ is called accepting, if $\mathrm{Inf}(\mathbf{x}) \cap F \neq \emptyset$, where $\mathrm{Inf}(\mathbf{x})$ is the set of states that appear in $\mathbf{x}$ infinitely often.
Moreover given a state $x \in X$, define

$$\mathrm{Post}(x) := \left\{x' \in X : \exists\, \sigma \in 2^{AP}, x' \in \delta(x, \sigma)\right\}.$$

Given an LTL formula φ over AP, there is a union of infinite words that satisfy φ, i.e.

$$\mathrm{Words}\left(\varphi\right) = \left\{\sigma \in \left(2^{AP}\right)^\omega \mid \sigma \models \varphi\right\}$$

where $\models \subseteq \left(2^{AP}\right)^\omega \times \varphi$ is the satisfaction relation.

Lemma 3.2.1. *:*
Any LTL formula φ over AP can be algorithmically translated into a Büchi automaton B_φ over the input alphabet 2^{AP} such that B_φ accepts all and only those infinite runs over AP that satisfy φ.

3.2.4 Product Büchi Automaton (PBA)

Given a controlled transition system $\mathcal{T}_c = (X, X_0, AP, \rightarrow, L_c)$ and a non-deterministic Büchi automaton $\mathrm{B} = \left(S, S_0, 2^{AP}, \delta, F\right)$, the product Büchi automaton (PBA) is $\mathcal{P} = \mathcal{T}_c \times \mathrm{B} = \left(S_p, S_{0,p}, 2^{AP}, \delta_p, F_p\right)$, where $S_p := X \times S$, $S_{0,p} := \mathrm{X}_0 \times S_0$, $F_p := (X \times F) \cap S_p$ and the following condition holds:

- $\delta_p \subseteq S_p \times S_p$, defined by $((x, s), (x, s')) \in \delta_p$ if and only if $x' \in \mathrm{Post}(x)$ and $s' \in \mathrm{Post}(s)$.

Given a state $p \in S_p$, define the projection operator $pj_X(p) : S_p \rightarrow X$ as a mapping from p to its first component $x \in X$.
Given a state $x \in X$, define the function $\beta_p : X \rightarrow 2^S$, given by

$$\beta_p(x) := \left\{s \in S : (x, s) \in S_p\right\} \tag{3.1}$$

as a mapping from x to the subset of Büchi states S that correspond to x.

Denote by $D\left(p, p'\right)$ the set of all finite runs between state $p \in S_p$ and $p' \in S_p$, i.e.

$$\begin{aligned} D\left(p, p'\right) := \{&p_1 p_2 \ldots p_n : p_1 = p, p' = p_n \\ &\left(p_k, p_{k+1}\right) \in \delta_p, \forall k = 1, \ldots, n-1; \forall n \geq 2\}. \end{aligned}$$

The state p' is said to be reachable from p, if $D\left(p, p'\right) \neq \emptyset$.
The length of a finite run $\boldsymbol{p} = p_1 p_2 \ldots p_n$ in $\mathcal{P}$, denoted by $Lg(\boldsymbol{p})$, is given by

$$Lg(\boldsymbol{p}) := \sum_{i=1}^{n-1} \left\| pj_X\left(p_{i+1}\right) - pj_X\left(p_i\right) \right\|.$$

For all $p, p' \in S_p$, the distance between p and p' is defined as follows:

$$d\left(p, p'\right) = \begin{cases} \min_{\boldsymbol{p} \in D(p,p')} Lg(\boldsymbol{p}), & \text{if } D\left(p, p'\right) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases} \tag{3.2}$$

The following definitions of self-reachable set and potential functions are given in [28].

Definition 3.2.2.
A set $A \subseteq S_p$ is called self-reachable if and only if all states in A can reach a state in A, i.e. $\forall p \in A, \exists p' \in A$ such that $D(p, p') \neq \emptyset$.

Definition 3.2.3.
For a set $B \subseteq S_p$, a set $C \subseteq B$ is called the maximal self-reachable set of B if each self-reachable set $A \subseteq B$ satisfies $A \subseteq C$.

Definition 3.2.4 (Potential function of states in $\mathcal{P}$).
The potential function of a state $p \in S_p$, denoted by $V_{\mathcal{P}}(p)$ is defined as

$$V_{\mathcal{P}}(p) = \begin{cases} \min_{p' \in F_p^p} \left\{ d\left(p, p'\right) \right\}, & \text{if } p \notin F_p^* \\ 0 & \text{otherwise} \end{cases}$$

where F_p^* is the maximal self-reachable set of the set of accepting states F_p in $\mathcal{P}$ and $d(p, p')$ is defined in (3.2).

Definition 3.2.5 (Potential function of states in $\mathcal{T}_c$).
Let a state $x \in X$ and a set $M_p \subseteq \beta_p(x)$, where $\beta_p(x)$ is defined in (3.1). The potential function of x with respect to M_p, denoted by $V_{\mathcal{T}_e}(x, M_p)$ is defined as:

$$V_{\mathcal{T}_c}\left(x, M_p\right) = \min_{s \in M_p} \left\{ V_p((x, s)) \right\}$$

Remark: If $V_{\mathcal{T}_c}\left(x, M_p\right) < \infty$, it means that $\exists s \in M_p$ such that starting from (x, s), there exists a run that reaches a selfreachable accepting state of $\mathcal{P}$.

3.3 Rapidly-exploring random trees

A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces based on random sampling of the configuration space. Those samples are used to construct a tree structure in which the final path can be retrieved. Basically the algorithm can be divided into three main steps: expansion through vertex selection, expansion and terminating condition.

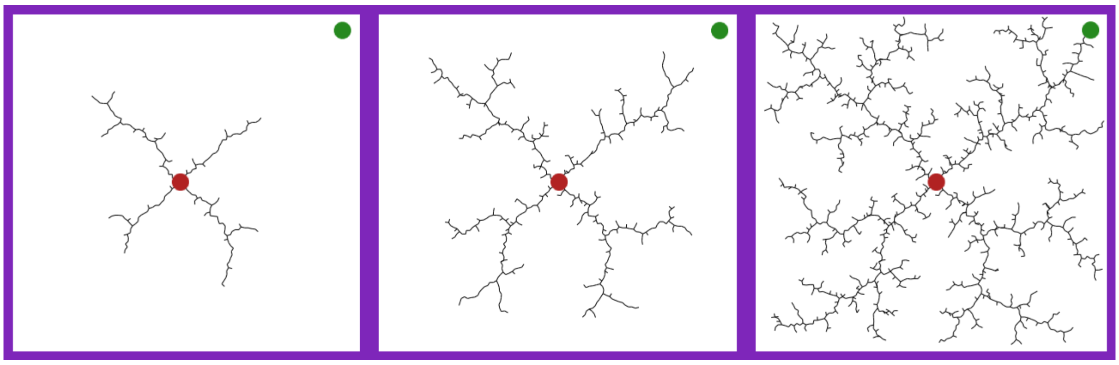


Figure 3.2: RRT algorithm idea

Firstly the RRT algorithm grows a tree, having as root the starting configuration, by randomly sampling the configuration space. Whenever a sample is taken, a connection between it and the nearest vertex in the tree is attempted. If that connection is feasible (i.e. only free space is encountered and all constraints are met), then the new vertex (corresponding to a new state) is added to the tree. When uniform sampling is used, the probability of expanding an existing state is proportional to the size of its Voronoi region. Therefore the tree preferentially expands towards large unsearch areas, since the largest Voronoi regions belong to the states on the frontier of the search. Typically the length of the connection between a new state and the tree is bounded by a growth factor. For instance if the generated sample exceeds those limits from its nearest state, then a new state at the maximum distance from the tree along the line to the random sample is considered rather than the random sample itself. The algorithm stops when the terminal condtion is satisfied, for instance when the goal state is reached or when the maximum number of iterations has been computed.
Such a method allows for biased expansion into previously undiscovered areas of the state space while limiting the size of the incremental growth.

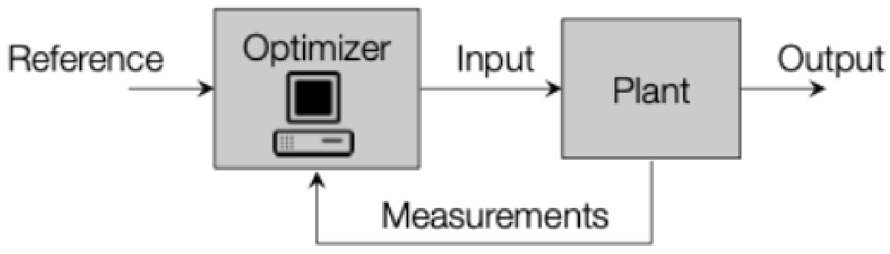
One of the most important characteristics of such algorithms is that RRTs can easily handle problems with obstacles and differential constraints (nonholonomic and kinodynamic) and are commonly utilized in autonomous motion planning.
Several effective algorithms based on RRT principle have been developed in the last years, including Fuzzy Greedy RRT, RRT-connect, RRT*, RRT-path (see [26] - [17] - [21] - [29] respectively). Moreover RRT growth can be biased by enhancing the probability of sampling states from a specific area, introducing a small probability of sampling the goal to the state expansion procedure. In practice, such ideas lead the search towards the terminal condition and higher this probability is, the greedier the tree grows on the way to the goal.

Algorithm 1 RRT pseudocode

```
$G.init(q_{init})$
for k = 1 to N do
    $q_{rand} \leftarrow RAND\_SAMPLE()$
    $q_{near} \leftarrow NEAREST\_VERTEX(q_{rand}, G)$
    $q_{new} \leftarrow GENERATE\_VERTEX(q_{near}, q_{rand}, \delta q)$
    $G.add\_vertex(q_{new})$
    $G.add\_edge(q_{near}, q_{new})$
end for
return G
```

3.4 Model Predictive Control

In the control theory field one of the most exiting result was obtained in the 1960s, with the derivation of the linear quadratic regulator (LQR) and the linear quadratic Gaussian regulator (LQGR). Kalman and others, proved that the optimal controller for a linear time-invariant MIMO system was a linear time-invariant state feedback control. Such linear feedback control law was proven to have several very desirable properties. It was guaranteed to exist, to be unique, and to be asymptotically stable under very mild and reasonable assumptions. Moreover for the feedback gain an explicit formula was provided and it was relatively easy to compute.
In particular, LQGR is one of the main achievements of the state-space control theory. Nevertheless it turned out to lack in robustness which lead to the born of robust control theory, field of particular interest in the 1980s. Although from a theoretical point of view those results solve the control problem for linear-time invariant (LTI) systems, just few real engineering implemenations were made.
The main reason of that stands on practical issues, in fact LQR regulator may badly fail in the presence of non-linearities such as input saturation (typically present in real applications) and when constraints must be imposed (for instance state or control limitations to achieve the desired goal).
Indeed, optimal operating points are often near constraints, so that classical control schemes need to be complemented with ad hoc constraints management or limit the performance to stay sufficiently far from the limits. The effort to overcome these limitations led to the born of Model Predictive Control (MPC).



3.4.1 MPC example

Consider an autonomous racing car moving on a predefined track, as shown in Fig 3.3. The goal consists to minimize the lap time while avoiding other cars, stay on the road, do not skid and with limited acceleration. One of the key ingredients of a model predictive control algorithm is the Receding Horizon (RH) principle.

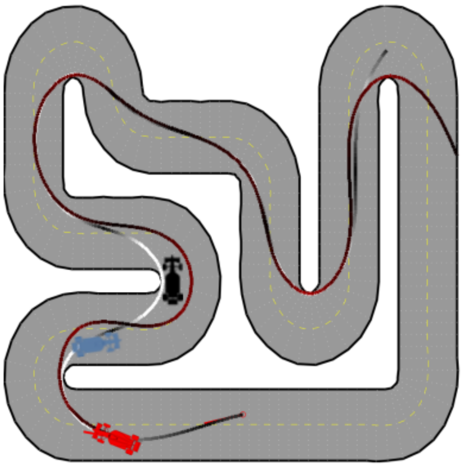


Figure 3.3: MPC algorithm idea

The main idea is to look ahead in time and plan the path based on road conditions, upcoming turns, and vehicle capabilities.
Based on these accessible data an optimization problem is formulated and solved with respect to the (future) control sequence $\mathbf{u}_k = [u(k), u(k+1), \ldots, u(k+N-1)]$, then only its first element $u_0(k)$ is applied. At the next time instant $k+1$, a new optimization problem is solved on the shifted prediction interval $[k+1, k+N+1]$, based on the available information up to time $k+1$, yielding the control $u_0(k+1)$. Afterwards, the procedure is constantly repeated so that the control action can operate on arbitrarly long time horizons.
To sum up, model predictive control theory is based on three pilars:

1. an objective function to be minimized, (in the above example the lap time);
2. an internal system model, which is used to predict the system behaviour;
3. sets of constraints to be satisfied (on state variables, control inputs, etc...)

Main advantages of an MPC strategy are its systematic approach capable to handle constraints and the ability to achieve high control performance. Furthermore, the receding horizon principle incorporates feedback into the control action, ensuring supervision of the current system condition.
Its primary challenges are related to implementation, as the MPC problem has to be solved in real-time and with available hardware (storage, CPU, etc...), as well as the fact that stability and robustness are not guaranteed. Additionally, the optimization problem may become infeasible at some future time step, and hence persistent feasibility is not guaranteed (i.e. a plan meeting all constraints may not exist).

3.4.2 Optimization problem mathematical formulation

The following subsection briefly formulates the optimization problem, which is solved at each time step in the MPC algorithm.
Considering the generic instant of time k, the optimization problem is given by:

$$\min_{\mathbf{U}} \quad p\left(x_{k+N}\right) + \sum_{i=0}^{N-1} q\left(x_{k+i}, u_{k+i}\right)$$

subject to:

$$\begin{aligned} & x_{k+1+i} = f(x_{k+i}, u_{k+i}), \qquad \forall i,\ 0 \le i \le N-1 \\ & (\text{linear case: } x_{k+1+i} = Ax_{k+i} + Bu_{k+i}, \qquad \forall i,\ 0 \le i \le N-1) \\ & x_{k+i} \in X, x_{k+i} \in \mathcal{U} \qquad \forall i,\ 0 \le i \le N-1 \\ & x_{k+N} \in X_f \\ & x_0 = x(k) \end{aligned}$$

where U = $\{u_k, u_{k+1}, ..., u_{k+N-1}\}$ and N represents the receding horizon length (number of steps in the future that the MPC strategy should consider).
As can be noticed, the cost function presents two terms: one related to the terminal cost and one associated with all other instants of time.
Moreover there is an additional constraint $x_{k+N} \in X_f$, which when combined with the terminal cost guarantees persistent feasibility and stability of the MPC strategy under some conditions that will be discussed later on. For further details about model predictive control theory refer to [23] and [6].

3.4.3 Lyapunov stability

Consider a dynamical system which satisfies

$$\dot{x} = f(x,t) \quad x(t_0) = x_0 \quad x \in \mathbb{R}^n.$$

Assume that $f(x,t)$ is Lipschitz continuous with respect to x, uniformly in t, and piecewise continuous in t and $x^* = 0$ is an equilibrium point for the system.

Definition 3.4.1 (Lyapunov stability).
The equilibrium point $x^* = 0$ is stable (in the sense of Lyapunov) at $t = t_0$, if for any $\epsilon > 0$ there exists a $\delta(t_0, \epsilon) > 0$ such that:

$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \|x(t)\| < \epsilon, \quad \forall t \geq t_0.$$

Lyapunov stability is a very mild requirement on equilibrium points.
In particular, it does not require that trajectories starting close to the origin tend to the origin asymptotically.

Definition 3.4.2 (Asymptotic stability).
An equilibrium point $x^* = 0$ is asymptotically stable at $t = t_0$, if
1. $x^* = 0$ is stable, and
2. $x^* = 0$ is locally attractive;
i.e., there exists $\delta(t_0)$ such that:

$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \lim_{t \to \infty} x(t) = 0$$

Definitions 3.4.1 and 3.4.2 are local definitions; in fact they describe the behavior of a system near an equilibrium point. An equilibrium point x^* is said to be globally stable if it is stable for all initial conditions $x_0 \in \mathbb{R}^n$ (see more details in [18] - [20]).

3.4.4 Persistent feasibility

Some basic concepts and defintions are provided to help readers understand what persistent feasibility means in the context of MPC.

Definition 3.4.3 (One-step backward reachable set).

$$\text{Pre}(\mathcal{S}) = \{x \in \mathbb{R}^n : \exists u \in U \ s.t. \ f(x, u) \in \mathcal{S}\}$$

Definition 3.4.4 (One-step forward reachable set).

$$\text{Reach}(\mathcal{S}) = \left\{x \in \mathbb{R}^n : \exists u \in U, z \in \mathcal{S} \text{ s.t. } x = f(z, u)\right\}$$

Definition 3.4.5 (N-step backward reachable set subject to system constraints).

$$\mathcal{R}^-_{j+1}(\mathcal{S}) = \text{Pre}\left(\mathcal{R}^-_j(\mathcal{S})\right) \cap X, \quad \mathcal{R}^-_0(\mathcal{S}) = \mathcal{S}$$

Definition 3.4.6 (N-step forward reachable set subject to system contraints).

$$\mathcal{R}^+_{j+1}(\mathcal{S}) = \text{Reach}\left(\mathcal{R}^+_j(\mathcal{S})\right) \cap X, \quad \mathcal{R}^+_0(\mathcal{S}) = \mathcal{S}$$

Definition 3.4.7 (Positive Invariant Set).
Given a constrained autonomous system $x(k+1) = f_a(x(k)), \quad t \in \mathbb{Z}_+, x(k) \in X$.
A set $\mathcal{O} \subset X$ is called a positive invariant set if

$$x(0) \in \mathcal{O} \Rightarrow x(k) \in \mathcal{O}, \forall k \in \mathbb{Z}_+$$

Definition 3.4.8 (Maximal Positive Invariant Set $\mathcal{O}^*$).
A set $\mathcal{O}^*$ is said to be the maximal positive invariant set, if it is positive invariant and includes all invariant sets contained in X.

Definition 3.4.9 (Control Invariant Set).
A set $C \subseteq X$ is called a Control Invariant Set for a dynamic system if

$$x(k) \in C \Rightarrow \exists u(k) \in U \ \textit{ such that } \ f(x(k), u(k)) \in C, \forall k \in \mathbb{Z}_+$$

Definition 3.4.10 (Maximal control invariant set C^*).
A set C^* is said to be the maximal control invariant set, if it is control invariant and includes all control invariant sets contained in X.

A feasible set X_k is the set of feasible state x_k at prediction step k for which the optimization problem (see 3.4.2) is feasible:

$$X_k = \left\{x \in X : \exists u \in U, \text{ s.t. } f(x,u) \in X_{k+1}\right\}, \quad \text{with } X_N = X_f$$

Or equivalently $X_k = \text{Pre}\,(X_{k+1}) \cap X$, with $X_N = X_f$.

Once all the previous concepts have been introduced, the notion of persistent feasibility can be formalized.

Definition 3.4.11 (MPC persistent feasibility).
Starting from any intial state $x(0) \in X_0$, persistent feasibility is achieved if the MPC control law ensures that feasibility is guaranteed at all time (i.e. $x(k) \in X_0$, for all $k \in \mathbb{Z}_+$).

Chapter 4

Planning and control for a single robot

In this chapter a detailed description about the theoritical background used in the degree project is presented, in order to formalize the main concepts and techniques needed for practical applications.

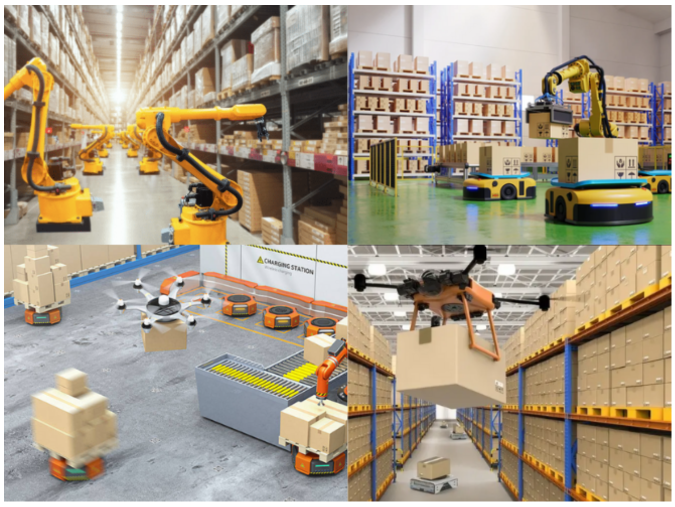
4.1 Linear Temporal Logic planner

First of all, as stated in the introduction, more complex and high-level tasks are desired, which has been a major focus in recent years. In fact robots can operate in risky conditions, such as poor lighting, toxic substances, or tight spaces saving workers from dangerous situations. For such reason the thesis focuses on temporal logic specifications, which can thoroughly cover a variety of challenging robot assignments. Therefore each agent is meant to fullfill a task, assigned through a LTL formula.
Some examples are:

1. $\varphi = (\Box\Diamond\ \text{"loaded"}) \wedge (\Box\Diamond\ \text{"unloaded"}) \wedge (\Box\neg\ r_4)$
2. $\varphi = (\Box\neg\ r2) \wedge ((\Diamond\Box\ r4) \vee (\Diamond\Box\ r6))$

In the first task, the agent must be loaded infinitely often (for example, to pick up a package), unloaded infinitely often and always avoid region r_4 (unsafe area). Such a task is a high-level pick and place task that can be encoded using an LTL formula. The latter requires the agent to always avoid region r_2 while also visiting and remaining in either region r_4 or region r_6 in the future. Aside

from supervision and pick and place tasks, it is simple to grasp the potential of LTL formulas, which can appear in even more complex and difficult agent assignments with a simple logical formalization.
Once the task has been assigned to an agent, a global planner must be implemented to produce the sequence of actions required to meet the given specification. A pre-built ROS package [3] was used for this purpose, and a brief explanation of how a global planner can be built using the given LTL formula is provided.



4.1.1 Software architecture

The developed planner relies on a node-based architecture and is integrated within the well-known open-source framework ROS. The LTL (core) planner determines the robot agent plan given an LTL specification, the agent model, and its workspace (i.e. the sequence of actions in order to fullfill the given LTL task in its environment). The agent model is represented by a weighted finite transition system (wFTS), and depending on the wFTS state retrieved by the agent node, the plan is updated or the next action command to be executed is output by the global planner. Each agent has its own node, which communicates with the planner node and converts the high-level plan into agent-specific low-level commands. The proposed planner sends high-level commands (of the string type) to the agent node, based on the correspondence between the assigned LTL specification and a finite Büchi automaton. Such high-level commands are translated into low-level inputs for the specific agent.
A state monitor node has also been implemented to simplify agent physical state

localization and mapping it into the corresponding abstract wFTS state.
The wFTS is a ROS parameter that can be accessed by any ROS node. To run additional code inside the planner node, a plugin functionality has also been added, offering a modular way to implement new features, such as the HIL perspective that will be discussed in Section 5.3.

4.1.2 LTL CORE AND PLANNER

The general planner, which is not agent-specific, is included in the core package. Software integration on a variety of robot platforms, including mobile robots and multi-DOF robot manipulators, is made possible by separating the LTL core planner from the agent-specific node. The input to the planner node consists of an agent model wFTS and an LTL task φ.

LTL TO NBA

First, the LTL formula φ is used to produce the corresponding NBA $\mathcal{B}_\varphi$ via the LTL2BA software [12]. The generated plan is published on a ROS topic, further information can be found in the documentation [4].

AGENT MODEL

The agent's workspace has been divided into regions, with permissible transitions between them and the different actions that can be taken. Remember that a wFTS, which encodes all potential agent transitions, is utilized for agent modeling. Furthermore, the developed ROS package considers not only 2D motion models, but also multi-dimensional models and more complicated action models for a more generic framework. As long as the state-space can be represented as a finite transition system, the planner can be applied with any sort of agent, simply by mixing any type of discretizable state-space in the agent model (e.g., 3D motion, battery status, pick/drop state, and so on...).
Notably, the LTL planner always produces a sequence of actions in which each action is carried out one at a time, forcing that only one dimensional states change by performing one action. In general, such an assumption is not restrictive, because different motion/action models describe different aspects of the agent state (for example pose, battery condition, loaded/unloaded).

PRODUCT BÜCHI AUTOMATON AND PLAN GENERATION

The previously obtained NBA$_\varphi$ and the wFTS intersection $\mathcal{D}$ is used by the LTL planner to generate a discrete plan. A previous work [25] is specifically investigated in order to get the code for plan generation. A PBA $\mathcal{A}_\mathcal{P}$ is defined as a tuple:

$$\mathcal{A}_\mathcal{P} = B_\varphi \otimes \mathcal{D} = (S_\mathcal{P}, \delta_\mathcal{P}, S_{\mathcal{P},0}, \mathcal{F}_\mathcal{P}, \mathcal{W}_\mathcal{P})$$

where $S_\mathcal{P} = S \times Q^P$; transition relation $(\langle s, q\rangle, \langle s', q'\rangle) \in \delta_\mathcal{P}$ iff $\exists \sigma \in \delta, s' \in \delta(s, \sigma)$ and $\exists \sigma_a \in \Sigma^P, (q, \sigma_a, q') \in \rightarrow_\mathrm{P}$; the set of initial states $S_{\mathcal{P},0} = S_0 \times Q_0$; the set of accepting states $\mathcal{F}_\mathcal{P} = (F \times Q^P)$; the weight function $W_\mathcal{P} \supset: \delta_\mathcal{P} \rightarrow \mathbb{R}^+, W_\mathcal{P}(\langle s, q\}, \langle s', q'\rangle) = W_P(q, q')$.
Using model-checking methods [5], an optimal run could be obtained from the defined PBA and projected back to the wFTS intersection $\mathcal{D}$. Accepting runs have a prefix-suffix structure of this kind: $r_\mathcal{P} = p_0, p_1 \cdots p_k (p_{k+1} \ldots\ldots p_n p_k)^\omega$, where $p_0 \in S_{\mathcal{P},0}$ and $p_k \in \mathcal{F}_\mathcal{P}$.
The output word is composed of two separate parts: a finite prefix part that is executed only once from the initial state p_0 to an accepting state p_k and a suffix part that is repeated infinitely from the accepting state p_k to itself.
Additionally, the optimal accepting run minimizes a cost function based on transition weights. The optimal accepting run has a corresponding action sequence that the agent must carry out in a prefix-suffix structure in order to fulfill its LTL specification.
The planner node also monitors the execution of the generated plan and provides the next action command to the agent node.

AGENT-LEVEL SOFTWARE

The implementation of agent-level nodes (see Figure 4.1) makes them agent-specific (or even scenario-specific). The agent node is responsible for translating the action command (string type) and sending the appropriate low-level commands to the actual system. State monitors also keep track of physical system configurations and link them to the proper weighted finite transition system state (whether they operate as separate nodes or are integrated into the agent node). Finally the agent node aggregates these states to a TS state and publishes it to the planner node.

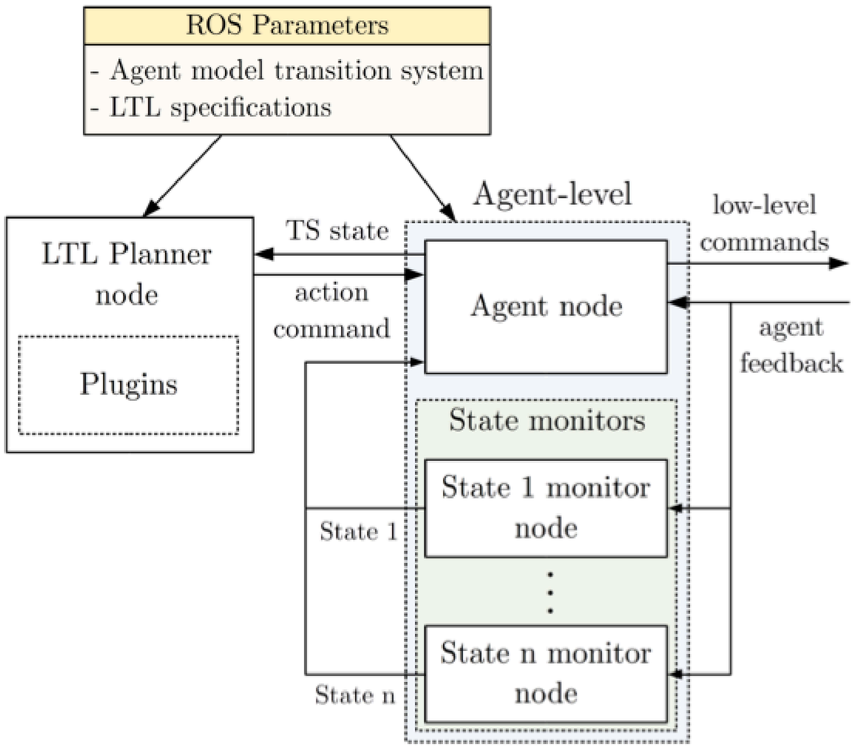


Figure 4.1: LTL automaton stack node graph

4.2 Rosie HEBI mobile robot model

The current section introduces the primary agent used in the master degree project. The Rosie HEBI mobile robot is an omnidirectional mobile platform with three omnidirectional wheels that has two goals: to facilitate the design of control algorithms and to facilitate the development of mobile robots in general by providing a ready-to-use motion solution. Over the last few decades, the research community [31] - [15] has paid increasing attention to and investigated omni-directional mobile robot (OMR). One of the benefits of an OMR is that it does not have non-holonomic constraints, which are present in differentially driven mobile robots [10] - [27].
The mobile platform may be simply moved wherever the user wishes thanks to the ability to regulate the rotation of each omnidirectional wheel, exemplifying the control law design.

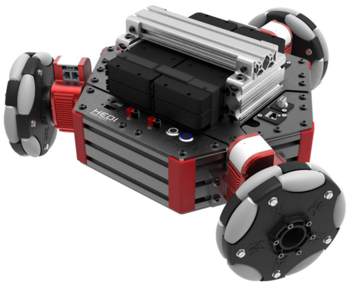


Figure 4.2: Rosie HEBI mobile base

An omnidirectional wheel is made up of a wheel and rollers, as shown in Figure 4.3. Hence the omnidirectional wheel speed is obtained as a combination of wheel speed and roller speed. Refer to [24] for more information on HEBI platforms and hardware components.
To execute the proposed plan retrieved by the LTL core planner and thus complete the assigned task, two navigation approaches will be proposed. For such purpose the Rosie HEBI mobile base's kinematics model is derived.

Figure 4.3: Rosie HEBI wheel

4.2.1 Kinematics model

Even though an accurate model can be obtained, for practical motivations a relatively simple model can be considered. Thus in order to design a good navigation controller that ensures safety, a kinematics model of the HEBI Rosie mobile base is sufficient.
A kinematics model, as is widely known in the literature, represents the motion of mechanical points, bodies, and systems without taking into account the forces acting on them and their corresponding physical properties.
In particular for the specific case, the following model is used:

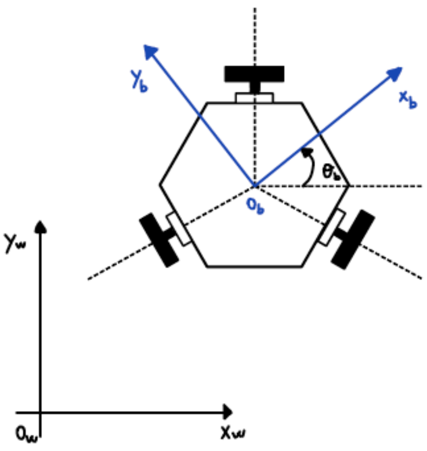


Figure 4.4: Rosie HEBI kinematics model

Denoting with $\mathcal{F}_w = O_w - x_w y_w$ and $\mathcal{F}_b = O_b - x_b y_b$ respectively the world and the body reference frames, the kinematics is derived:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{4.1}$$

where:

$\mathbf{x}(t) = \left[x_b;\ y_b;\ \theta_b\right]^\top$:= state vector in the world frame

$\mathbf{u}(t) = \left[v_{x,b};\ v_{y,b};\ w_b\right]^\top$:= input vector (robot velocities) in the body frame

$\mathbf{x}_0 = \left[x_{0,b};\ y_{0,b};\ \theta_{0,b}\right]^\top$:= represents the initial state

Hence equation (4.1) can be written as:

$$\begin{cases} \dot{x}_b = \cos\left(\theta_b\right) \cdot v_{x,b} - \sin\left(\theta_b\right) \cdot v_{y,b} \\ \dot{y}_b = \sin\left(\theta_b\right) \cdot v_{x,b} + \cos\left(\theta_b\right) \cdot v_{y,b} \\ \dot{\theta}_b = w_b \end{cases} \tag{4.2}$$

In matrix form:

$$\implies \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta_b & -\sin\theta_b & 0 \\ \sin\theta_b & \cos\theta_b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{x,b} \\ v_{y,b} \\ w_b \end{bmatrix} \tag{4.3}$$

Essentially, it is the input vector $u(t)$ multiplied by a rotation matrix, resulting in a transformation of the input vector coordinates from the body frame to the world frame. As can be seen, the presented model is a high-level representation of the system that excludes electrical and mechanical aspects. The decision not to include them in the model is based on the fact that the real robot already includes low-level controls that convert linear and angular velocities ($\left(v_{x,b}, v_{y,b}\right)$ and w_b respectively) into the necessary voltage for the actuators.

4.3 Navigation

The navigation problem will be addressed in this section, with two different approaches analyzed and two final proposed autonomous controllers.
The first relies on RRT theory and a PID controller for point-to-point tracking, while the second is based on MPC theory (see Section 3.4). To understand why one was chosen over the other, the environment in which the agents would move must be described.

4.3.1 Agents workspace

The agents workspace is assumed to be bounded and it is denoted by $\mathcal{W} \subset \mathbb{R}^2$. It consists of $N > 0$ regions of interest, denoted by $\Pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$, where $\pi_n \subset \mathcal{W}$. Furthermore, there is a set of $M > 0$ properties (atomic propositions) associated with Π, denoted by $AP = \{a_0, a_1, \cdots, a_M\}$, e.g. "the current location is an unknown region".
The agent's motion within the workspace is abstracted as a labeled transition system $\mathcal{T} \triangleq (\Pi, \rightarrow, \Pi_0, AP, L)$, where Π, AP are defined above, $\rightarrow \subseteq \Pi \times \Pi$ is the transition relation that $(\pi_i, \pi_j) \in \rightarrow$ if the robot can move from region π_i to region π_j without crossing other regions in Π, $\Pi_0 \in \Pi$ is where the robot starts initially, $L : \Pi \rightarrow 2^{AP}$ is the labeling function where $L(\pi_i)$ returns the set of properties satisfied by π_i. Moreover the environement is assumed to be partially-known and dynamic, more precisely the workspace is assumed to be fully-known but environmental changes can take place (e.g. non-static obstacles, other agents and objects). As illustrative example, assume that the environment is a 2D space (i.e. $\mathcal{W} \subset \mathbb{R}^2$) and can be divided into six square regions:

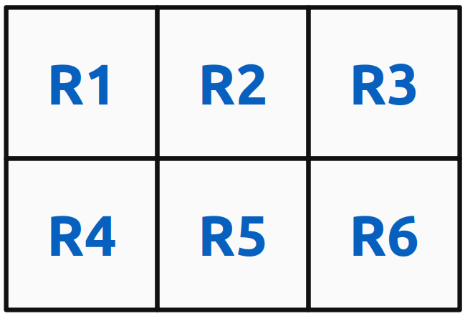
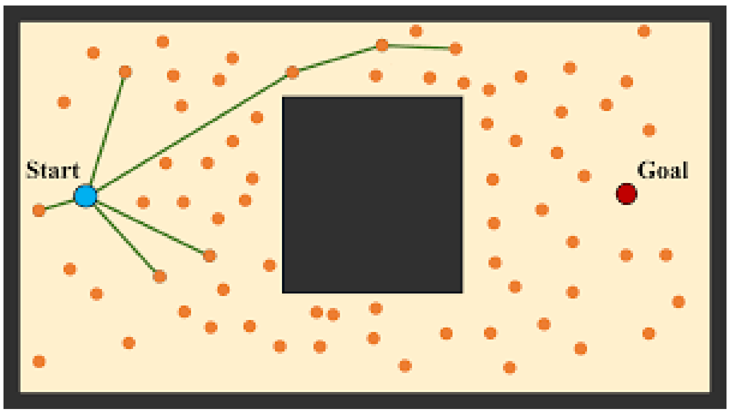


Figure 4.5: Example: workspace discretization

As stated above, those regions can be denoted by $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6$. Additionally, suppose that transitions between regions are allowed only when areas are adjacent and not diagonally close. For instance π_1 and π_2 are connected, while π_1 and π_5 are not.
If the current agent region is π_s and the next action retrieved by the LTL planner is "goto_r$_g$" (i.e. move to region π_g), a prior requirement to satisfy is that the robot navigation from region π_s to π_g must remain within $\mathcal{W}$ and without crossing other regions in Π. Such an example highlights a prerequisite that the navigation autonomous controller must achieve.
In fact, if the initial region is π_1 and the LTL task encodes "it is forbidden to access region π_4", the navigation path from π_1 to π_2 must never cross through that area.

4.3.2 RRT planning and point-to-point tracking controller

To handle navigation at first an autonoumous controller, obtained as combination of state-space sampling and a point-to-point tracking controller, is developed.
Primarly for each possible region-to-region transition ($\pi_s \rightarrow \pi_g$) , an RRT based algorithm is used to generate the shortest path connecting π_s and π_g. Consequently the state-space is sampled and each sample (x,y position) is used to construct a graph tree structure, from which the path plan is obtained (for instance by inspecting Dijkstra as done in [30] and computing in that way the shortest path). All referred computations are made offline, otherwise a large workspace discretization will turn to be intractable in an online approach.



After the online process, where all the region-to-region path trajectories have been computed, a PID controller is developed as follows to track each trajectory point. Thanks to the updated x,y agent position, at each instant of time t the position error is considered:

$$\mathbf{e}(t) = \begin{bmatrix} e_x(t) \\ e_y(t) \\ e_\theta(t) \end{bmatrix} = \begin{bmatrix} x_{\text{des}} - x(t) \\ y_{\text{des}} - y(t) \\ \theta_{\text{des}} - \theta(t) \end{bmatrix}$$

where:
x_{des}:= is the point trajectory x-coordinate
y_{des}:= is the point trajectory y-coordinate
θ_{des}:= is the point trajectory theta-coordinate (which will be imposed to be zero, i.e. the robot will be aligned with the world reference frame for the whole path)

Such error is then used to tune and design a PID controller, which will end up to have the following structure:

$$\mathbf{u}(t) = K_\text{P} \cdot \mathbf{e}(t) + K_\text{I} \cdot \int_0^t \mathbf{e}(\tau) \mathrm{d}\tau + K_\text{D} \cdot \frac{\mathrm{d}\mathbf{e}(t)}{\mathrm{d}t}$$

where K_P, K_I, and K_D, all non-negative, stand the coefficients for the proportional, integral, and derivative terms respectively.
A PID controller contains three control terms, however depending on the desired result, just two terms may be enough to provide a suitable control law.
Numerous simulations and tests have been performed in order to evaluate performances and choose the best gains. Even though this type of navigation algorithm has been confirmed to be effective and reliable, there are no theoretical assurances that the agent won't move outside the start and target areas (issue mentioned in the illustrative case in 4.3). Although it can be demonstrated theoretically that the integral gain allows convergence towards the objective point, the agent's path to that point cannot be predicted.
An alternative navigation strategy has been designed and then put into practice in order to address the latter stated issue, as will be outlined in Chapter 6.

4.3.3 Model Predictive Controller

As mentioned in Section 3.4, model predictive control (MPC) allows to insert both state and input constraints within the control law. Such control strategy enables to have a theoretical proof and solve the previous disccussed problem.

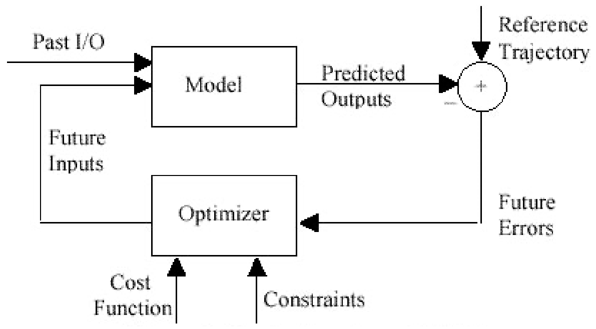


Figure 4.6: MPC controller

The degree project focalizes on a squares division of the workspace, similarly as that of Figure 4.5 and the model chosen for prediction purposes is the one formulated in Section 4.2.1. Each action retrieved by the LTL core planner mainly will be the next region the agent should move to, encoded through a string (e.g. "goto_r5). In pursuit of ensuring a safe navigation without violating the assigned LTL task, the extreme corners of the start and goal regions are added as hard constraints in the cost function, resulting in the optimization problem:

$$\begin{aligned} \min_{\mathbf{U}} \quad & [\mathbf{x}(k+N) - \mathbf{x}_{des}]^T Q_N [\mathbf{x}(k+N) - \mathbf{x}_{des}] \quad + \\ & \sum_{i=0}^{N-1} \Big[[\mathbf{x}(k+i) - \mathbf{x}_{des}]^T Q [\mathbf{x}(k+i) - \mathbf{x}_{des}] + \mathbf{u}(k+i)^T R \mathbf{u}(k+i) \Big] \\ \text{subject to:} & \\ & \mathbf{x}_{k+1+i} = f(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}), \qquad \forall i,\ 0 \le i \le N-1 \\ & \mathbf{x}_{k+i} \in X, \mathbf{u}_{k+i} \in \mathcal{U} \qquad \forall i,\ 0 \le i \le N-1 \\ & \mathbf{x}_{k+N} \in X_f \\ & \mathbf{x}_0 = \mathbf{x}(k) \end{aligned} \tag{4.4}$$

A quadratic cost function of both state and input has been selected for the specific case, with the matrices $Q \in \mathbb{R}^{n \times n}$, $Q_N \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ set to be positive definite.

As can be noticed, the cost function involves two terms: one related to $\mathbf{x}(k+i) - \mathbf{x}_{des}$ (i.e. the distance between the state vector and the desired pose that the agent should reach, thus the next region given by the planner) and another that is related to the input effort required to get there. State constraints are represented by:

$$x_{lower\_bx} + \frac{base\_footprint}{2} \leq \; x_{k+i} \; \leq x_{upper\_bx} - \frac{base\_footprint}{2} \qquad \forall i, \;\; 0 \leq i \leq N$$
$$y_{lower\_bx} + \frac{base\_footprint}{2} \leq \; y_{k+i} \; \leq y_{upper\_bx} - \frac{base\_footprint}{2} \qquad \forall i, \;\; 0 \leq i \leq N$$

These constraints account for the lower and upper bounds of the agent's allowed range of motion. To further guarantee that the body's agent completely satisfies such motion limits, the base footprint is taken into account. As can be observed, adding state constraints to the cost function enables safe navigation while preventing possible agent task violations. However, the MPC technique arises two additional issues that need to be properly addressed: stability and persistent feasibility.
Lyapunov stability is used to define stability, and persistent feasibility is the principle that the autonomous controller guarantees feasibility at all future times (i.e., there is a feasible solution at every step in the future).
One solution to ensure (or enforce) stability for any step horizon is to include a terminal constraint, which forces the state to assume a particular value at the end of the prediction horizon. Proof of MPC stability and persistent feasibility under mild constraint are reported in Appendix A and Appendix B.

MPC DESIGN PARAMETERS

The choice of MPC parameters will affect not only the control performances but also the computational complexity of the MPC algorithm, which solves an online optimization problem at each time step. Firstly the sampling time T_s must be chosen properly, which determines the rate at which the controller executes the control algorithm.
If it is too big, when a disturbance comes in, the controller won't be able to react to the disturbance fast enough. On the other hand if the sampling time is too small, the controller can react faster to disturbances and setpoint changes, but this causes an excessive computational load. To find the right balance between performance and computational effort, a good choice is to fit 10 to 20 samples

within the rise time of the open-loop system response.
As reported in Section 3.4, at each time step the MPC controller makes predictions about the future plant output, and the optimizer finds the optimal sequence of control inputs that drives the predicted plant output as close to the setpoint as possible.
The number of predicted future time steps is called the prediction horizon and shows how far the controller predicts into the future. The prediction horizon should cover the significant dynamics of the system, otherwise some important dynamics aspects may not be taken into account and it might be too late to correct them. At the same time a large prediction horizon will arise on a computational waste of time, since some environmental changes can take place.

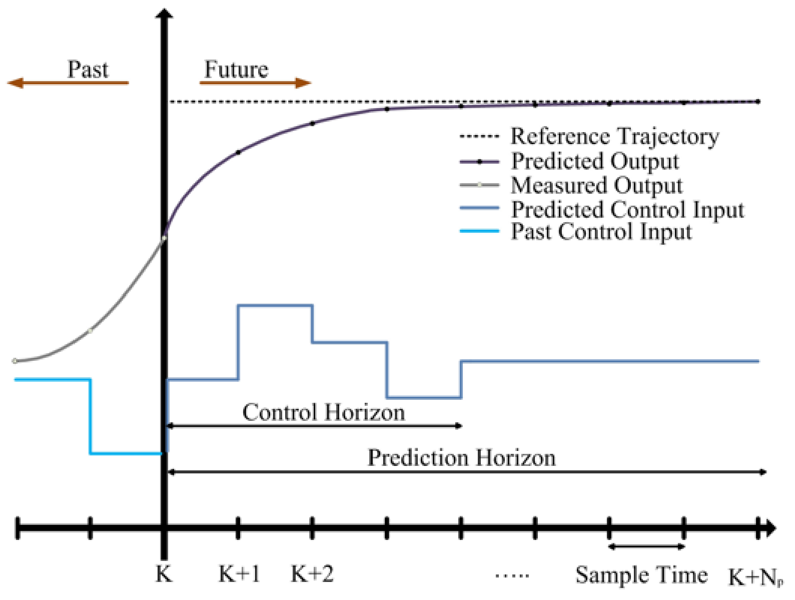


Figure 4.7: MPC: prediction and control horizon

Another important design parameter is the control horizon, if Figure 4.7 represents the set of future control inputs leading to that predicted plan output, the number of control actions to time step m is called the control horizon.
Smaller the control horizon is, fewer are the computations. However increasing such parameter could lead to get better predictions but at the cost of increasing the complexity. Certainly the control horizon can be chosen the same as the prediction horizon, nevertheless usually only the first couple of control inputs have a significant effect on the predicted output behavior, while the remaining

actions have only a minor effect. Furthermore MPC can include input and state constraints, which can be either thought as soft or hard constraints. Hard constraints cannot be violated, whereas soft constraints can be violated (often used to avoid conflicting requirements in the optimization).
The design of the MPC controller has been built by carefully selecting all parameters for the particular experiments that have been taken, keeping in mind the above theoretical guidelines.

Chapter 5

Online replanning for multi-robot systems

Robots and autonoumos vehicles are opening up new routes to success for many industries. But to smooth the way to commercial advantage, innovators must negotiate a challenging problem: how to enhance navigation and collision avoidance. Sitting on a production line is one thing, but it is quite another when an autonomous system ventures out into crowded and dynamic environments. In most solutions it has to freeze, unable to progress until the path forward has cleared. To be fit for purpose in environments such as streets, farms, construction sites, space surfaces, underwater areas and many other, robots and autonomous vehicles must react to the random movements of people and objects.

Figure 5.1: Partially-known environment

For example, an autonomous vehicle moving in a crowded city can encounter pedestrians crossing the street, distracted drivers, new routine utility maintenance, unreliable traffic signals, or blind curves. Hence, there are many different factors to take into consideration, which open up new and different environomental conditions where the autonomous agent should move and deal with. Multiple issues might emerge, considering an autonomous agent in a dynamic environment. But extending such randomness to a multi-agent system, where motion coordination is fundamental to achieve complex and challenging goals, it may increase the solution's complexity.
The degree project aims to tackle those conditions and objectives, using a decentralized approach which will be explained in details.
First of all, a multi-robot system is introduced, which consists of multiple autonomous agents or robots. Additionally, the agent's workspace is now supposed to be partially-known. The term partially-known environment refers to a situation in which the agent is unaware of the physical laws governing the environment and hence of the environmental changes that will take place (such as human beings, non-static obstacles, etc...).
The degree project will specifically consider two or three HEBI Rosie mobile bases as multi-agent systems.

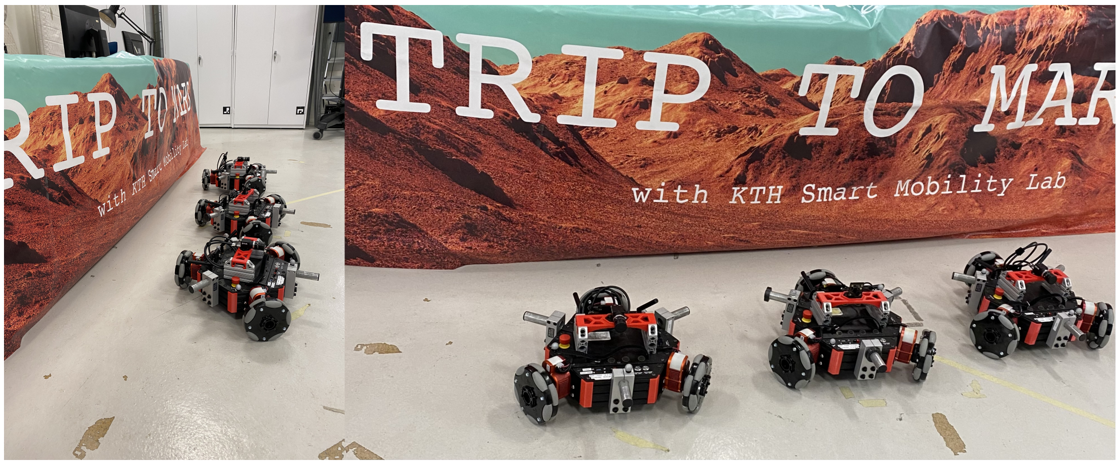


Figure 5.2: Thesis multi-agent system

The initially planned trajectory for each robot, retrieved by the LTL core planner to fullfill the assigned task, does not account for the motion of other robots or moving obstacles. Moreover, each robot only has access to local view and data. As a result, a motion coordination algorithm is required during the online implementation.

Two distinct cases will be treated using an appropriate technique. The goal of this section is to examine in depth the control algorithm that enables reactive online replanning of each robot. Both solutions are built with a decentralized approach, keeping in mind the local view and limited knowledge about the surroundings. Every agent is thought to have a local view and access to information within a circle with a preset radius; for the degree project, such radius will be between 0.5 and 1 meter. Each robot may identify conflicts in its neighborhood, using sensing data about the workspace and broadcast data from other robots. The robots then do motion replanning to ensure that conflicts are avoided and the given task is satisfied.

Algorithm 2 localTrajectory algorithm

```
Input  : $\xi_i(t_k), B_i(\xi_i(t_k)), \mathrm{Post}(B_i(\xi_i(t_k))), \mathcal{P}_i, V_{\mathcal{P}_i}, \mathbb{O}$, and $r_{\mathrm{safe}}$.
Output: A local transition system $\mathcal{T}_{c,i}^L$
1  Initialize $\mathcal{T}_{c,i}^L = \left(S_i^L, S_{i,0}^L, AP_{\varphi_i}, \rightarrow_{c,i}^L, L_{c,i}\right)$ and $\xi_i^f = \emptyset$, where $S_i^L = S_{i,0}^L = \xi_i(t_k)$
   and $\rightarrow_{c,i}^L = \emptyset$.
2  for $k \leftarrow 1$ to $N_i^{max}$ do
3      $\xi_s \leftarrow$ generateSample $(SA_i(t_k))$,
       $\xi_n \leftarrow$ nearest $\left(S_i^L, \xi_s\right)$,
       Solve the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$, which returns $\xi_r$,
       $B_i(\xi_r) \leftarrow \beta_{\mathcal{P}_i}(\xi_r) \cap \{B_i(\xi_n) \cup \mathrm{Post}(B_i(\xi_n))\}$
       if $B_i(\xi_r) \neq \emptyset \wedge V_{T_{c,i}}(\xi_r, B_i(\xi_r)) < \infty$, then
4          if $\mathrm{proj}_\ell([\xi_n, \xi_r])$ $isObstFree(\mathbb{O}_i(t_k)) \wedge safeMotion(\xi_r, r_{\mathrm{safe}})$ then
5              $S_i^L \leftarrow S_i^L \cup \{\xi_r\}; \rightarrow_{c,i}^L = \rightarrow_{c,i}^L \cup \left\{\xi_n \xrightarrow{u_i} \xi_r\right\}$
6          end if
7      end if
8      if $\mathrm{proj}_l(\xi_r) \notin \mathcal{B}(\mathrm{proj}_\ell(\xi_i(t_k)), R)$ then
9          $k = N_i^{\max} + 1$
           $\xi_i^f \leftarrow \xi_r$
10     end if
11 end for
12 return $\mathcal{T}_{c,i}^L$
```

The online motion replanning structure consists of a local and a global trajectory generation algorithms, where the global trajectory is the one provided by the model predictive controller explained in Section 4.6. The local trajectory generation process is built starting from the work made in [22], highlighted in Algorithm 2, which relies on a sampling based method that operates online during motion. In fact, whenever a conflict is detected, the algorithm starts to randomly sample the state-space and construct a graph to find a path-free trajectory, which leads the agent outside of the sensing area and ensures the satisfaction of the given task.

The above algorithm takes as input the current i-th robot pose $\xi_i\left(t_k\right)$, $B_i\left(\xi_i\left(t_k\right)\right)$, $\operatorname{Post}\left(B_i\left(\xi_i\left(t_k\right)\right)\right)$, the offline precomputed product Büchi automaton $\mathcal{P}_i$ (obtained combining the agent transition system and the assigned LTL task), the potential function $V_{\mathcal{P}_i}$ and the set of obstacles $\mathbb{O}$ known in advance. Firstly a local transition system $\mathcal{T}_{c,i}^L$ is initialized, it will be constructed throughout the algorithm and returned as final outcome. At each iteration a new state ξ_s is taken randomly from the sampling area $SA_i\left(t_k\right) := \left\{(p,\theta) \in \mathbb{X}_i : p \in \mathcal{B}\left(\operatorname{proj}_l\left(\xi_i\left(t_k\right)\right), R+\eta\right)\right\}$ (line 3), where $\eta > 0$ is an offline constant which ensures to get out from $\mathcal{B}\left(\operatorname{proj}_\ell\left(\xi_i\left(t_k\right)\right), R\right)$. Such detail is fundamental to check the terminal condition (line 12). Then through the function $\operatorname{nearest}\left(S_i^L, \xi_s\right)$ an RRT primitive is applied, which returns the nearest state to ξ_s in S_i^L.

At this stage an optimization problem $\mathcal{P}\left(\xi_n, \xi_s, \tau_s\right)$ is solved, so as to find the closest reachable state from the new sample ξ_s:

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

subject to:

$$\xi_i(0) = \xi_n$$

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

where τ_s represents the system sampling time, while $F_i\left(\xi_i(s), u_i\right)$ describes the robot's kinematics (described in Section 4.2.1).

Once ξ_r is obtained, the corresponding subset of valid Büchi states $B_i(\xi_r)$ is computed (line 6). After that, if both conditions $B_i(\xi_r) \neq \emptyset$ and $V_{T_{e,i}}(\xi_r, B_i(\xi_r)) < \infty$ are met (which guarantees that there exists a path, starting from ξ_r, that reaches 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 $\xi_n \overset{u_i^*}{\longrightarrow} \xi_r$ is added into $\rightarrow_{c,i}^L$, if the path connecting ξ_n with ξ_r is obtacles free and the motion is considered safe. Two requirements checked through the function $isObstFree(\mathbb{O}_i(t_k))$ and $safeMotion(\xi_r, r_{\text{safe}})$ (tested considering the base footprint of the HEBI Rosie base, set to 0.64 meters).
The algorithm is stopped, when the local sampling tree reaches the outside of the sensing area and the leaf node ξ_i^f is then returned (given by the corresponding state ξ_r) (line 12-14). Finally, the algorithm output ξ_r is used as starting position from LTL core planner, which replans the sequence of actions in order to accomplish the assigned agent specification.
The described online procedure contains, in the local transition system $\mathcal{T}_{c,i}^L$, the sequence of positions which the associated robot must follow.
Such sequence can be tracked in different ways: applying a PID controller or for instance a more advanced tracking algorithm. The next subsections will present two scenarios, leading to different strategies to track those robot positions retrieved by the online planner.

5.1 Local communication case

The first case assumes local communication of the nearby robots and knowledge of obstacle locations. Regarding the latter assumption, it means that the obstacle area is assumed to be known in advance. Such hypothesis is included in Algorithm 2 by adding those locations in the set of obstacles $\mathbb{O}$.
To coordinate the robots' motion, local communication is instead accomplished by creating a priority hierarchy. Therefore whenever a new robot enters the sensing area, the localTrajectory algorithm is executed directly afterward the priority hierarchy has applied. To clarify it, consider the given priority hierarchy example: $agentA$ can move freely, $agentB$ before moving waits for $agentA$ communication, lastly $agentC$ waits $agentA$ and $agentB$ communication. Hence $agentA$ will move "liberally" despite the presence of the surrounding robots. If agent B is close to agent A, then agent B must wait for agent A to communicate its motion (i.e., what will be the agent's future locations), and those future positions are included as obstacles in the local trajectory algorithm. In all other scenarios, where just agent C is present or there are no agents detected, $agentB$'s local trajectory process is directly applied.
The same reasoning applies to the final agent. If both agents are detected, it awaits local communication; if only one agent is identified, it will include exclusively such agent motion in Algorithm 2, otherwise it will move freely.
Different tracking controllers can be used in the given scenario to follow the sequence of points provided by the local trajectory generation process. Because those locations are always close each other, even a PID controller with accurate gains tuning is adequate to achieve the goal (see Section 4.3).

5.2 Communication-free case

A communication-free context has been investigated in pursuit of extending the degree project work to more practical and real-world scenarios. A collision avoidance algorithm is added into the local navigation controller to deal with it. Before proceeding, the concept of "trap state" need to be introduced. Trap states are PBA states from which the Büchi acceptance condition cannot be fulfilled, hence states that cannot reach accepting states that appear infinitely often.
The approach to deal with trap states and avoid obstacles considers a local model

predictive controller, in which the cost function is set to be:

$$\begin{aligned}\min_{\mathbf{U}} \quad & [\mathbf{x}(k+N) - \mathbf{x}_{des}]^T Q_N [\mathbf{x}(k+N) - \mathbf{x}_{des}] \ + \\ & \sum_{i=0}^{N-1} \Big[[\mathbf{x}(k+i) - \mathbf{x}_{des}]^T Q [\mathbf{x}(k+i) - \mathbf{x}_{des}] + \mathbf{u}(k+i)^T R \mathbf{u}(k+i) \Big] + \\ & \sum_{j=1}^{N_{\mathbb{O}}} P_{\mathbb{O}} \frac{1}{dist_j} + \sum_{\ell=1}^{N_{\mathbb{O}_t}} P_{\mathbb{O}_t} \frac{1}{dist_\ell} \\ \text{subject to:} \quad & \\ & \mathbf{x}_{k+1+i} = f(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}), \qquad \forall i,\ 0 \le i \le N-1 \\ & \mathbf{x}_{k+i} \in X, \mathbf{u}_{k+i} \in \mathcal{U} \qquad \forall i,\ 0 \le i \le N-1 \\ & \mathbf{x}_{k+N} \in X_f \\ & \mathbf{x}_0 = \mathbf{x}(k)\end{aligned} \tag{5.1}$$

The cost function is fairly similar to the one proposed in Equation (4.4), but two more terms are inserted. Essentially at every instant of time, the local MPC updates the obstacle positions and computes the distance $dist_j = \|p_{\text{agent}} - p_{\text{j}}\|$ between agent and obstacle position. The same has been made for the trap states, where in this case $dist_\ell = \|p_{\text{agent}} - p_\ell\|$ denotes the distance between agent and trap state position.
The suggested technique opens up a discussion on what is more appropriate to choose, in the sense that depending on the weights $P_{\mathbb{O}}, P_{\mathbb{O}_t}$ distinct solutions can be preferred. For instance if it prioritized a satisfaction of the given LTL task rather than not collide with obstacles, the weight $P_{\mathbb{O}_t}$ will be set higher than $P_{\mathbb{O}}$. On the other hand the degree project was constructed with the primary intention of guaranteeing safety (therefore avoiding collisions with objects and other agents) and, if possible, to fulfill the assigned task. Clearly, trap states should be avoided whenever possible, but safety is considered more important and hence higher will be set gain $P_{\mathbb{O}}$ (related to obstacle avoidance) compared with $P_{\mathbb{O}_t}$.
However, depending on the individual objective, an alternative trade-off might be devised and different outcomes can be produced.

5.3 Human-in-the-loop

Because manual jobs are painful, businesses are continuously seeking for methods to increase their productivity through automation. Not only it is really time-consuming to do each of these individual tasks, but they end up incurring tons of different inefficiencies. When a completely automated process is considered, incorrect information and a difficult data cleansing challenge are frequently the results.
A lot of examples of autonomous systems can be found in our daily life, such as self-driving vehicles, package delivery drones and household service robots [8]. However certain applications are more oriented towards a human-robot interaction, where autonomous systems often perform the intended tasks under the supervision or collaboration with human operators [11]. Hence the right balance between intend automation and automation in which people are more taken into consideration is the exact idea behind this new research branch. Human-in-the-loop aims to achieve what neither a human being nor a machine can achieve on their own.

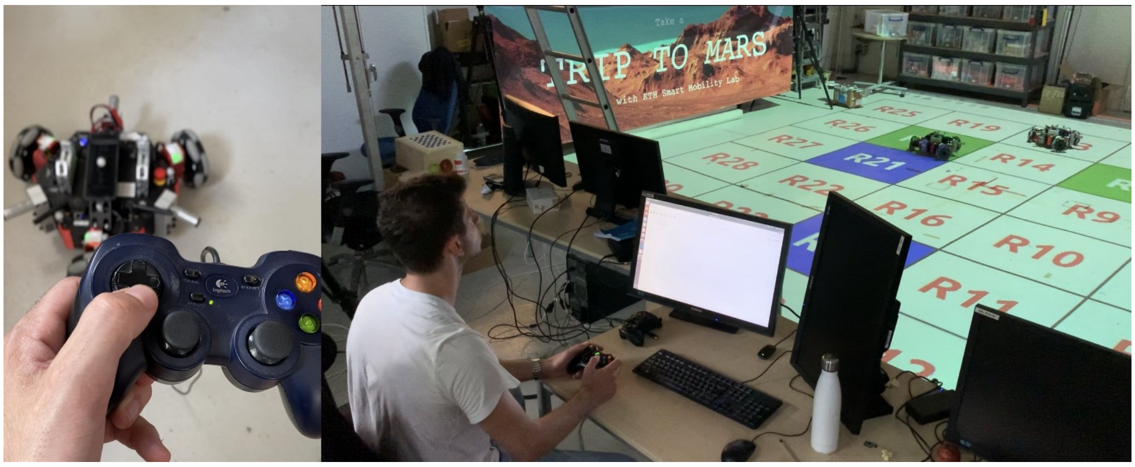


Figure 5.3: Human in-the-loop scenario

The thesis analyzes a situation where autonomous robots are designed to complete difficult tasks specified by an LTL formula, while a human operator can simultaneously monitor progress, take action when necessary, or even take over the control of the robot from the on-board autonomous controller. The robot, on the other hand, can react to human inputs while preserving constant safety, which is useful for guiding the robot through challenging assignments [11]- [19].

The proposed strategy is a mixed-initiative controller, with the intention of extending previous works [19] - [13] to multi-agent systems. The mixed-initiative controller relies on the following built-in function, which allow to satisfy the given assigned task for all human inputs:

$$k\left(x,\mathbb{O},\mathbb{O}_{\mathrm{t}}\right) \triangleq G_{\mathrm{mix}}\cdot\frac{\rho\left(d_o-d_s\right)}{\rho\left(d_o-d_s\right)+\rho\left(\varepsilon+d_s-d_t\right)}+(1-G_{\mathrm{mix}})\cdot\frac{\rho\left(d_t-d_s\right)}{\rho\left(d_t-d_s\right)+\rho\left(\varepsilon+d_s-d_t\right)}$$

where $d_t \triangleq \min_{\langle\pi,q_1,q_2,c\rangle\in\mathbb{O}_{\mathrm{t}}\|x-\pi\|}$ is the minimum distance between the robot and any region within $\mathbb{O}_{\mathrm{t}}$; $d_o \triangleq \min_{\langle\pi,q_1,q_2,c\rangle\in\mathbb{O}\|x-\pi\|}$ is the minimum distance between the robot and any obstacle within $\mathbb{O}$; $\rho(s) \triangleq e^{-1/s}$ for $s>0$ and $\rho(s)\triangleq 0$ for $s\leq 0$, and d_s, $\varepsilon>0$ are design parameters as the safety distance and a small buffer. Moreover $G_{\mathrm{mix}}\in[0,1]$ represents a gain parameter, in order to manage the trade-off between two aspects: preventing trap states and obstacles avoidance. Thus the mixed-initiative controller is given by:

$$u \triangleq u_r\left(x,\pi_s,\pi_g\right)+\kappa\left(r,\mathbb{O},\mathbb{O}_{\mathrm{t}}\right)u_k(t).$$

Such controller is a combination of the autonomous navigation controller, proposed in Section 4.6 and Section 5.2, with the human input command filtered by the function $\kappa\left(r,\mathbb{O},\mathbb{O}_{\mathrm{t}}\right)$. The developed structure, which is presented throughout the degree project, enables a safe multi-motion coordination strategy and, as a result, the expansion of a mixed-integer controller (applied to each agent) to multi-agent systems.

Chapter 6

Experimental results

Experiments were carried out at the KTH Royal Institute of Technology's Smart Mobility Lab in Stockholm, Sweden, to verify the overall techniques described in Chapter 4. All tests were anticipated by a successfull simulation, which takes out some potential bugs and difficulties were addressed and fixed in the final version of the code. The developed code and the necessary package dependencies can be found at the GitHub repository [1]. Through the use of an optical motion capture system with twelve cameras arranged over the lab surface, the motion of the involved rigid bodies was tracked. Additionally as already stated, the software development is attained through the usage of Robot Operating System (ROS) and its node-based architecture.
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.

Figure 6.1: Smart Mobility Lab, KTH Royal Institute of Technology

[1]URL GitHub: https://github.com/Gianmarco2410/master_thesis_repository.git

6.1 Motion capture system: Qualisys

Qualisys Track Manager (QTM) is a Windows-based data acquisition software with an interface that allows the user to perform 2D and 3D motion capture. QTM is designed to provide both advanced functionality required by technically advanced users and a simple application approach for novice users. In combination with the Qualisys line of optical measurement hardware, QTM will streamline the coordination of all characteristics in a sophisticated motion capture system, allowing for the quick creation of unique and precise 2D, 3D, and 6D data. Real-time 2D, 3D, and 6D camera information is displayed during the capture, offering instant confirmation of accurate data acquisition. Advanced algorithms that are flexible to different movement characteristics rapidly process and convert each 2D camera data into 3D or 6D data. The data can then be exported to analysis software via several external formats.
All rigid body positions are shared as ROS topics within the ROS framework, the built software then access the appropriate topic in order to use the actual location of each entity.

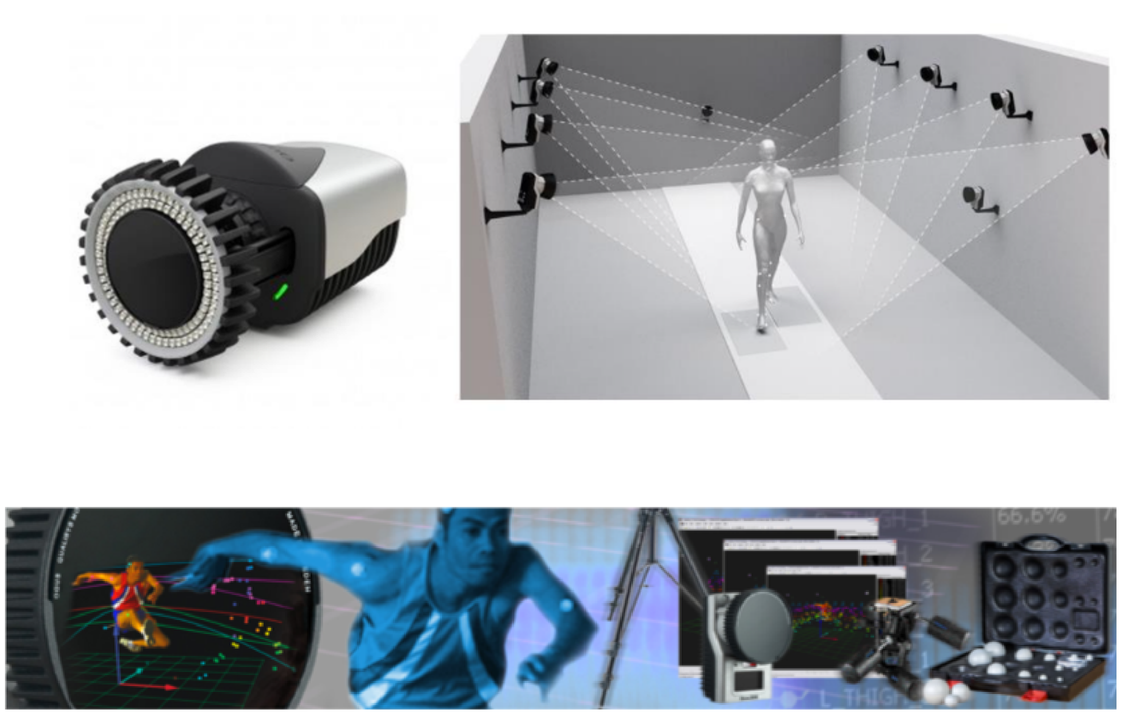


Figure 6.2: Qualisys Motion Capture system

6.2 Workspace discretization

The Smart Mobility Lab (SML) is a hub for the development and experimentation of intelligent transportation solutions, located at KTH Royal Institute of Technology and it consists of a 7×10 meters area. Given that the Qualisys Motion capture system is constrained by ambient occlusions (PC desks, unused objects, etc...), the laboratory configuration calls for a workspace spanning area of 5×6 meters. As a result, the workspace has been discretized into thirty squares, each of one meter side. The QTM reference frame is situated in region R15's center, with the y-axis pointing toward the SML access door. (see Figure 6.3).

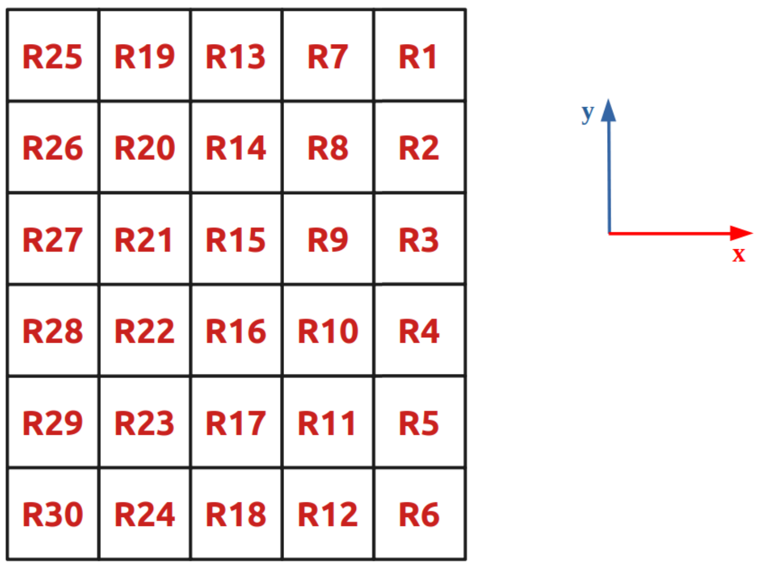


Figure 6.3: Workspace discretization

Such choice is not only based on the potential area tracked by the Qualisys Motion capture system, but also relies on the experimental goals and the assigned LTL tasks of the degree project. Due of the restricted motion surface, the presented experiments are making use of two or three HEBI Rosie mobile bases. Therefore the workspace W under consideration is a $5 \times 6\ \text{m}^2$ square area, where in all future plots gray areas represent prohibited areas or static obstacles, colored circles represent robots' initial positions, and light colored areas represent a set of target regions in the workspace.

6.3 Experiment 1

Consider a multi-agent system consisting of $N = 2$ HEBI Rosie robots, the dynamics of robot i is given by Equation (4.1), and the inputs constraints are set as follows. For all robots the maximum permissible linear velocity is set to 0.35 m/s, whereas for the angular velocity a maximum speed of 0.35 rad/s. The sensing radius of each robot is $R = 0.8\ m$, while the safe radius is set at 0.6 m. Initially, $\theta_i = 0$ and $v_i(0) = 0$.
Experiment 1 assigns each robot to constantly survey two target regions in the workspace (regions R8 and R20 for Rosie 0, regions R17 and R21 for Rosie 1) with both local communication and free-communication cases investigated.
The table below lists the associated LTL formulas, starting regions, and the offline plan generated by the LTL core planner:

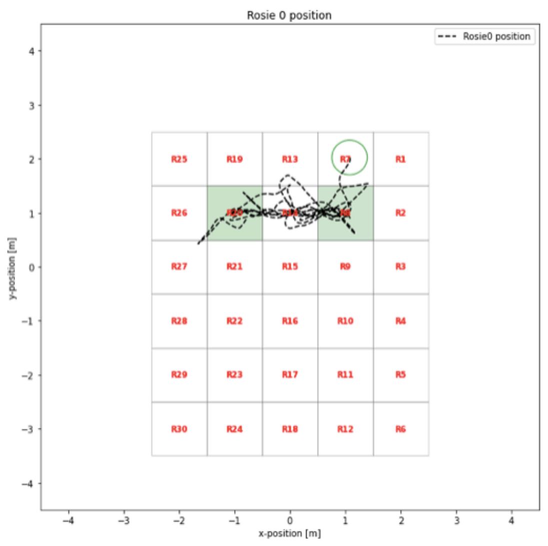
EXPERIMENT #1 - SETTING		
Agent	Rosie 0	Rosie 1
LTL task	$(\Box\Diamond R8) \wedge (\Box\Diamond R20)$	$(\Box\Diamond R17) \wedge (\Box\Diamond R21)$
Starting region	$R7$	$R23$
Preffix plan	$[R8, R14, R20, R14]$	$[R17, R16, R15, R21, R15]$
Suffix plan	$[R8, R14, R20, R14]$	$[R16, R17, R16, R15, R21, R15]$

Table 6.1: Experiment #1

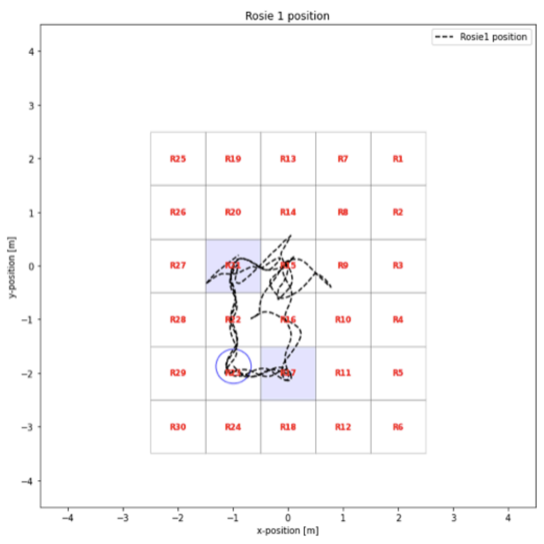
The NBA B_i associated with $\varphi_i, \forall i$ (robot LTL task) has been computed and it consists of 3 states and 8 edges using [12] after the workspace has been discretized in thirty area as specified in the preceding section.
The CTS $\mathcal{T}_i$ and the PBA $\mathcal{P}_i$ for each robot are constructed using LTLCon toolbox [16] and finally the potential function $V_{\mathcal{T}_i}$ is calculated. All these quantities are obtained offline and have no effect on the online computational cost of the local trajectory generation described in Algorithm 2.

Local communication case

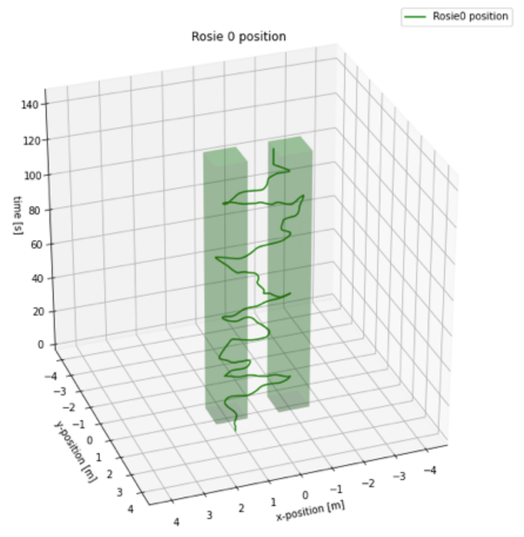
The local communication case involves an offline hierarchy between the robots which must be chosen in advance. In this situation, Rosie 0 has complete freedom of motion, but Rosie 1 must wait for the other robot's local trajectory generation. Initially from the starting region, each robot i navigates safely by following the preffix-suffix sequence of regions returned by the global planner (see Table 6.1). An MPC controller, as described in Section 4.3, ensures that the given task will not be violated and that each region is reached.



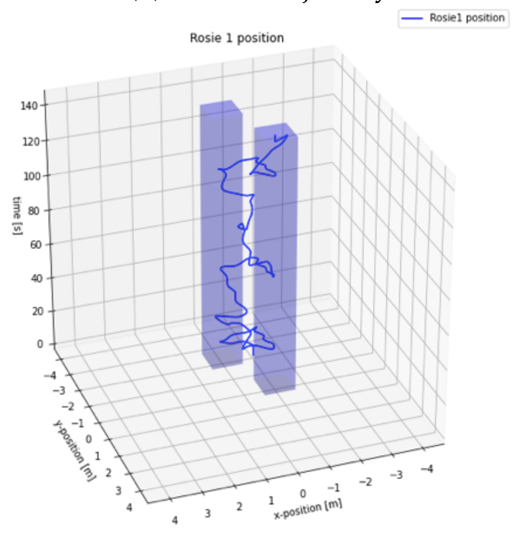
(a) Rosie 0 trajectory



(b) Rosie 1 trajectory



(c) Rosie 0 trajectory with respect to time



(d) Rosie 1 trajectory with respect to time

Figure 6.4: Experiment 1.1 - Rosie position trajectory

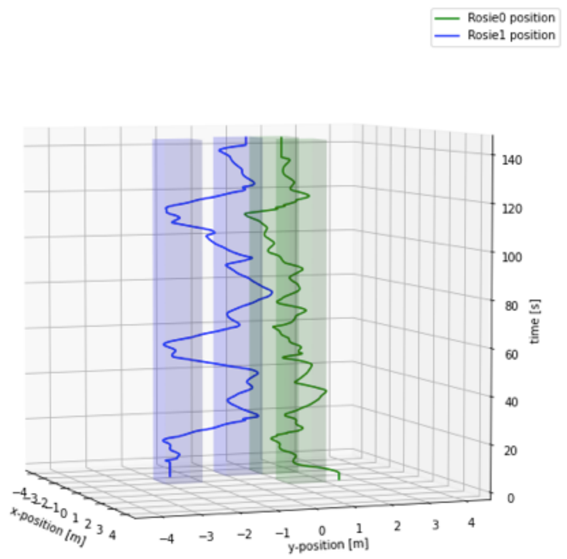
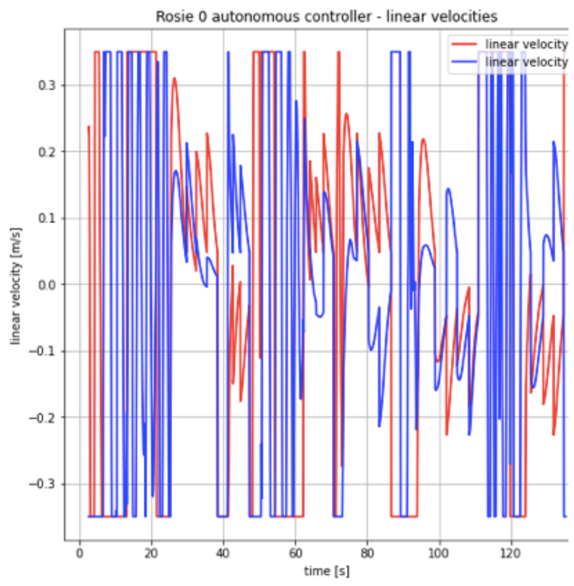
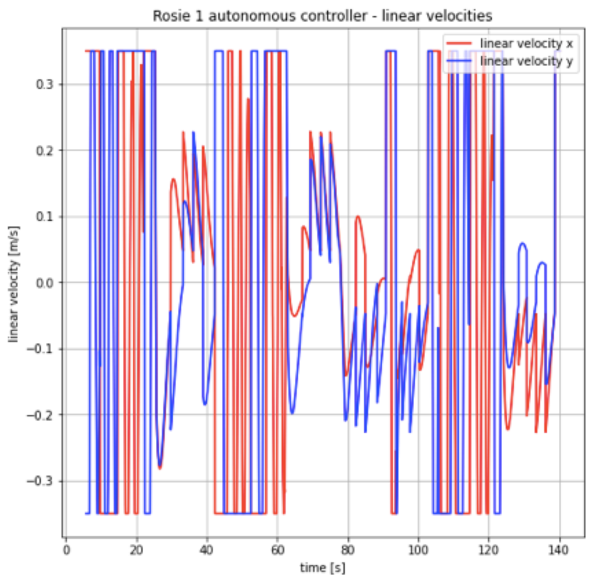


Figure 6.5: Experiment 1.1 - Evolution of position trajectories with respect to time

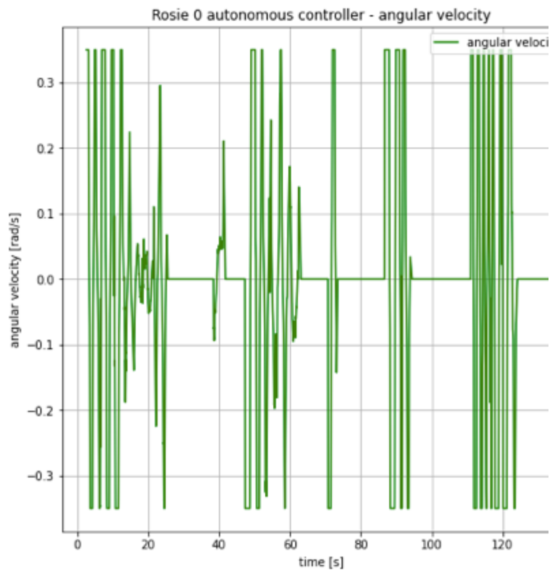
The motion coordination algorithm (see Section 5) is applied for each robot during online execution when potential collisions are identified.
Specifically, as far as Rosie 1 approaches the other robot, Algorithm 2 waits for the computation of Rosie 0's trajectory and then treats each point as an obstacle to produce a motion-free path (described throughout Section 5.1).
The real-time position trajectories of each robot are depicted in Figure 6.4. It is evident that each robot successfully completes its surveillance mission during the 140 second testing. Figure 6.4 (c)-(d) show the evolution of each robot trajectory in time and highlight that both rosies are visiting infinitely often the assigned target regions. Figure 6.5 shows the evolution of the position trajectories of all robots over time, where it is clear that the real-time position trajectories are collision-free. During the experiment, conflicts are detected in total 4 times. The hierarchical motion order and the local trajectory generating mechanism are both activated once a conflict is noticed in order to resolve it. Hence Rosie 0 never waits and moves without considering the future points in which Rosie 1 will be. While Rosie 1 starts its online replan anytime Rosie 0 enters the sensing area, viewing all of its replan trajectory as a moving obstacle. Figure 6.5 illustrates the online replan procedure, which guarantees collision-free motion in less than a second before the robot moves again. The real-time velocities of each Rosie are reported in Figure 6.6, where those velocities are exactly the inputs applied by the proposed control strategy.



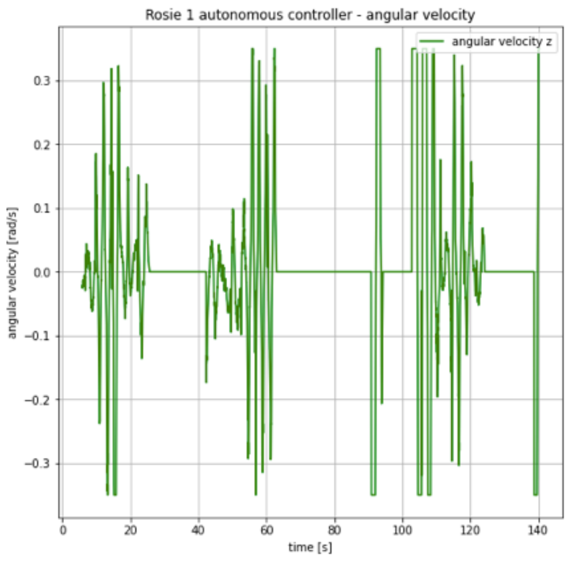
(a) Rosie 0 linear velocities



(b) Rosie 1 linear velocities



(c) Rosie 0 angular velocity



(d) Rosie 1 angular velocity

Figure 6.6: Experiment 1.1 - Rosies velocities

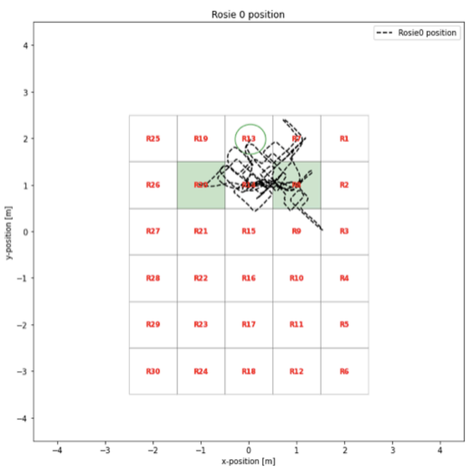
The fact that all robots satisfy input requirements at all times is evident, notably in Figure 6.5 (a)-(b). Saturation applies everty time a change of direction is required, indeed as soon as the robot reaches the given region, it immediately changes the target to the LTL planner's subsequent action.

Because of this, a large linear velocity is needed in the initial time instants due to a big initial error in the MPC strategy. However, input constraints inserted in the optimazion problem allow one to avoid exceeding the maximum velocity specified. Instead Figure 6.5 (c)-(d) represent the real-time angular velocity of each robot along the z-axis and the input action related to this velocity component, which is more powerful whenever the online replan takes place. A video demonstration of Experiment 1.1 can be found at the YouTube channel: Smart Mobility Lab - Gianmarco Fedeli [2].

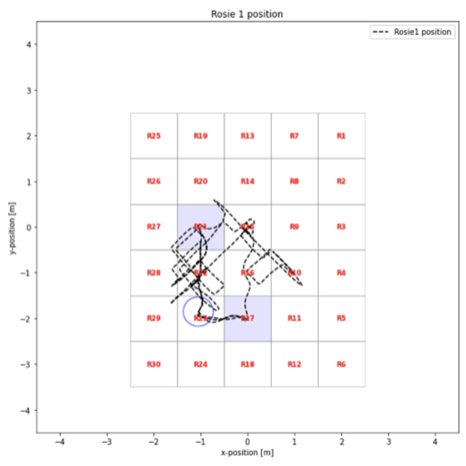
[2]URL video: https://youtu.be/mKWpqvMrW9Y

Communication-free case with human as obstacle

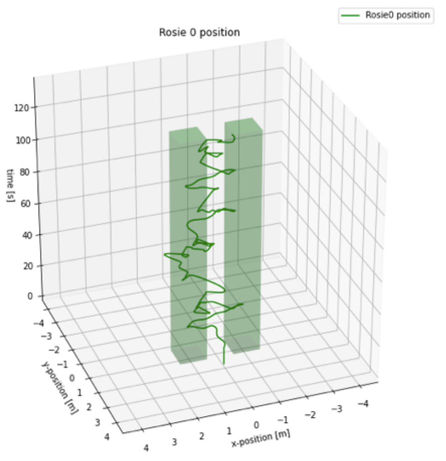
A more general communication-free case has been investigated in order to expand the application of multi-robot motion coordination. All rosie specifications presented for Experiment #1 are still applied, however, in comparison to the previous scenario, a human being is included to evaluate the effectiveness of the technique suggested in Section 5.2. As before from the starting region (Rosie 0 now has region $R13$ as starting region), each robot i follows the sequence of regions obtained by the global planner in the form of preffix-suffix and safe navigation is ensured by an MPC controller, exactly as in the previous case. Additionally, the person involved in the experiment begins in a neutral location, far from both robots and therefore having no influence on their motion.



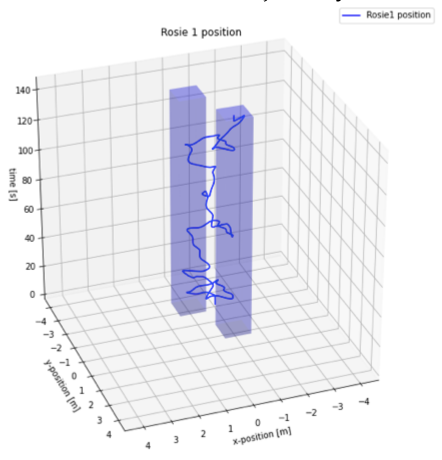
(a) Rosie 0 trajectory



(b) Rosie 1 trajectory



(c) Rosie 0 trajectory with respect to time



(d) Rosie 0 trajectory with respect to time

Figure 6.7: Experiment 1.2 - Rosie position trajectory

Figure 6.7 shows rosie positions evolution in the 2-D space, underlining the satisfaction of the predefined LTL tasks. Differently from the local communication case, the online replan algorithm applies whenever a robot detects obstacles in its sensing area. As a result, the monitoring activity is affected not only by other robot's motion, but also by a human randomly walking through the environment.
Figure 6.8 (a)-(b)-(c), where the evolution of all entities involved with respect to time is depicted, which reveals the power of the approach discussed throughout Section 5.2.

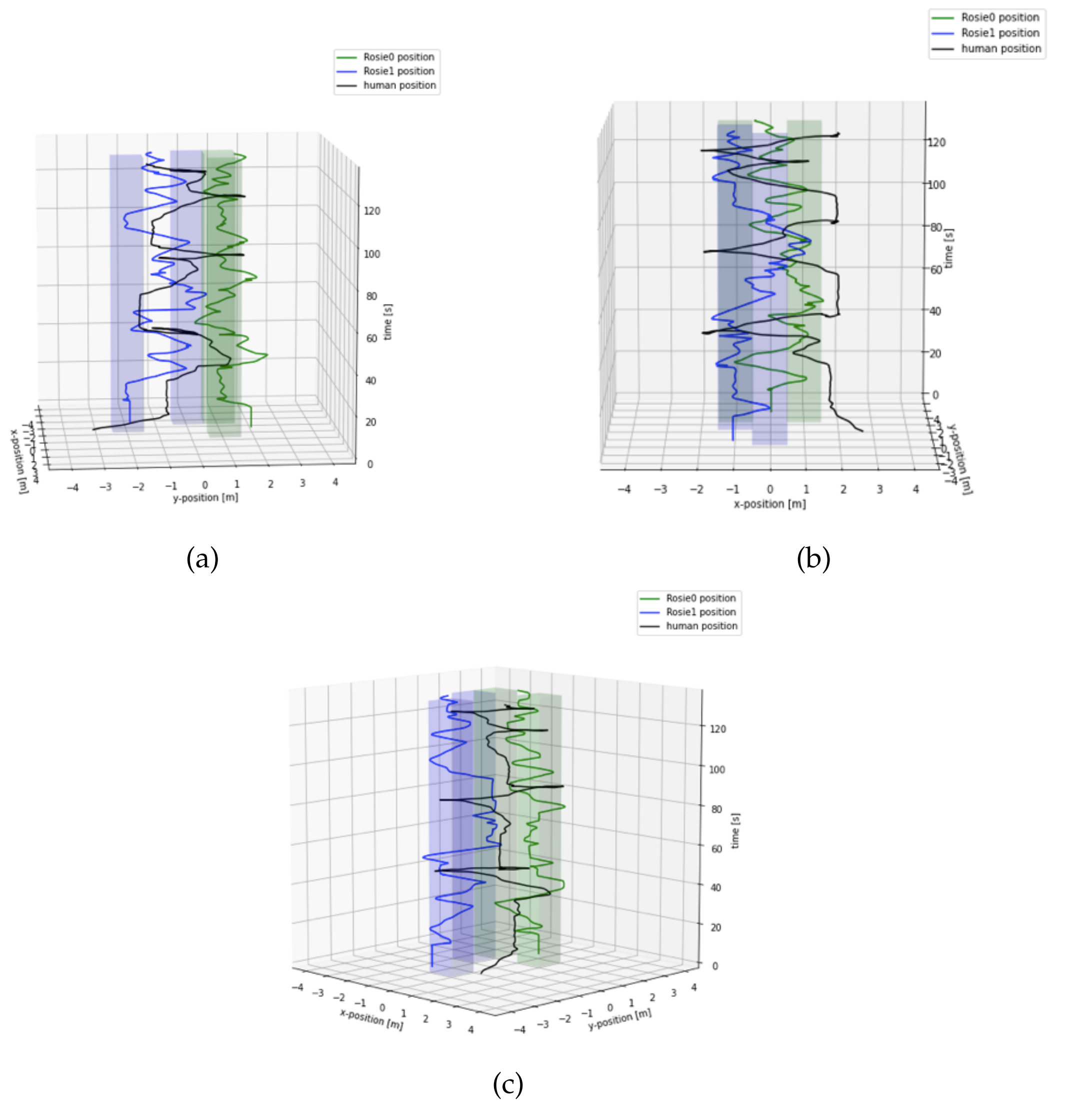
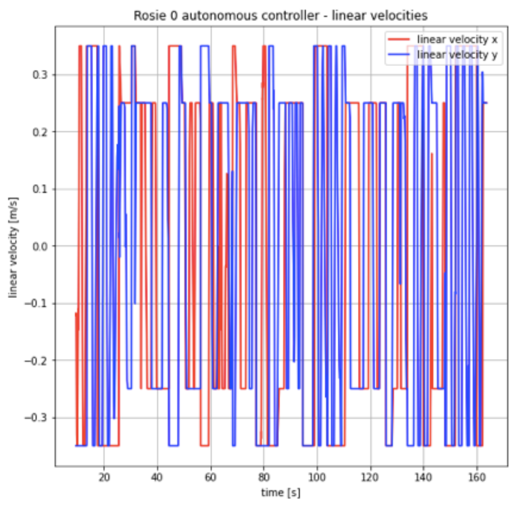
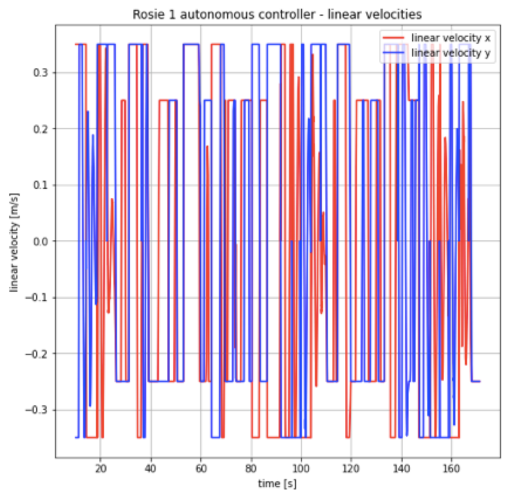


Figure 6.8: Experiment 1.2 - Evolution of rosie position trajectories and human position with respect to time, from different points of view

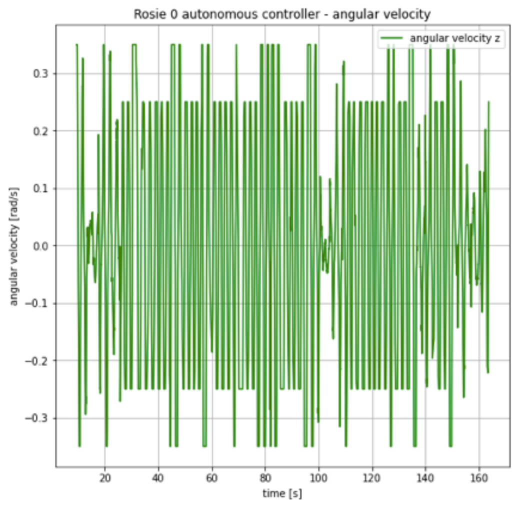
These plots demonstrate that collisions are prevented and that both rosies correctly survey the target locations. Conflicts are identified 14 times in total over the 120 seconds of the experiment, accounting for both robots. The online replan stage is completely different from the local communication scenario, where the motion is carried out based on priority assignments and most significantly, moving obstacles must be known in advance, which is typically not achievable in real scenarios.
In the communication-free case once the local trajectory has been generated, a local MPC is responsable for tracking such points while at the same time avoiding collisions, and hence react to environmental changes of any kind.



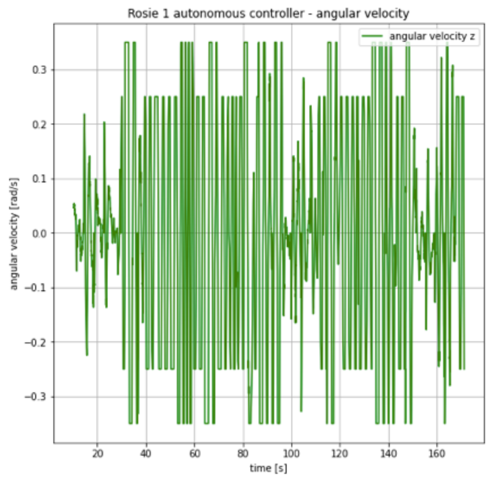
(a) Rosie 0 linear velocities



(b) Rosie 1 linear velocities



(c) Rosie 0 angular velocity



(d) Rosie 1 angular velocity

Figure 6.9: Experiment 1.2 - Rosies velocities

The real-time velocities of both Rosies are depicted in Figure 6.9, where it can be seen that all constraints are respected at any instant of time. The maximum allowed linear and angular velocities are set to 0.25 m/s and 0.25 rad/s for the specific case during the online replan phase. The decision is supported by the intention of achieving a smoother trajectory; indeed, relaxing too many such constraints would result in a very reactive response, but on the other hand, choppy motion is the price to pay. Clearly, the higher the velocity, the shorter the reaction time to environmental changes; thus, these limitations must be determined while considering the velocity at which possible obstacles might move (for instance how fast a human being can walk or run). A video demonstration of Experiment 1.2 can be found at the YouTube channel: Smart Mobility Lab - Gianmarco Fedeli [3].

[3]URL video: https://youtu.be/TYgfbrk7hDs

6.4 Experiment 2

Consider a multi-agent system consisting of $N = 3$ HEBI Rosie robots, the dynamics of robot i is given by Equation (4.1), and the inputs constraints are set as follows. For all robots the maximum allowed linear velocity is set to 0.35 m/s, while for the angular velocity a maximum speed of 0.35 rad/s is reachable. The sensing radius of each robot is $R = 0.8$ m and the safe radius is chosen to be 0.6 m. Initially, $\theta_i = 0$ and $v_i(0) = 0$. In Experiment 2, one robot has to monitore at least one of the two target regions (regions R14 and R15 for Rosie 0) while Rosie 2 is required to survey regions R11 and R27 and finally Rosie 1 is supposed to supervise only region $R26$. Furthermore, all three Rosies must avoid region $R20$, taught as dangerous area or where a non-static obstacle can be collocated.
The table below reports the related LTL formulas, the starting regions and moreover the offline plan generated by the LTL core planner in the preffix-suffix form:

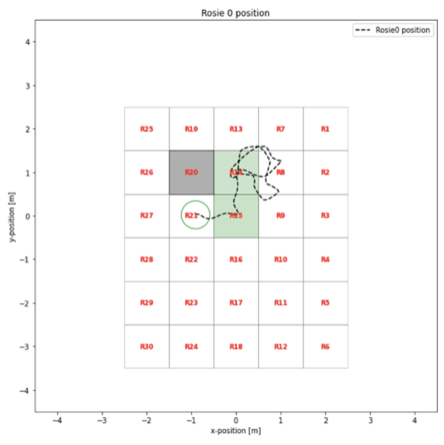
EXPERIMENT #2 - SETTING			
Agent	Rosie 0	Rosie 1	Rosie 2
LTL task	$((\Box\Diamond R14) \vee (\Box\Diamond R15)) \wedge (\Box\neg R20)$	$(\Box\Diamond R26) \wedge (\Box\neg R20)$	$((\Box\Diamond R11) \wedge (\Box\Diamond R27)) \wedge (\Box\neg R20)$
Starting region	$R21$	$R13$	$R17$
Preffix plan	$[R15, R14, R15]$	$[R19, R25, R26]$	$[R11, R10, R9, R15, R21, R27, R28]$
Suffix plan	$[R14, R14]$	$[R26, R26]$	$[R22, R16, R10, R11, R10, R9, R15, R21, R27, R28]$

Table 6.2: Experiment #2

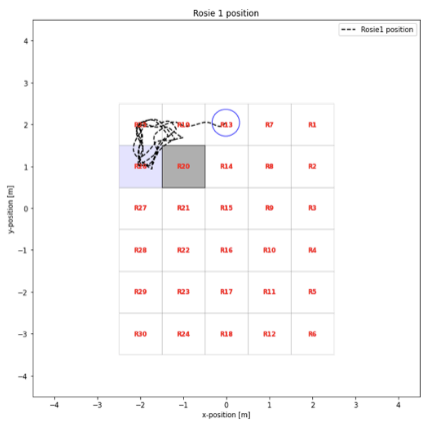
The workspace has been discretized into thirty regions, the NBA B_i associated with $\varphi_i, \forall i$ (robot LTL task) has been computed and it consists of 2 states and 4 edges by [12]. The CTS $\mathcal{T}_i$ and the PBA $\mathcal{P}_i$ for each robot are constructed using LTLCon toolbox [16] and finally the potential function $V_{\mathcal{T}_i}$ is calculated.

Local communication case

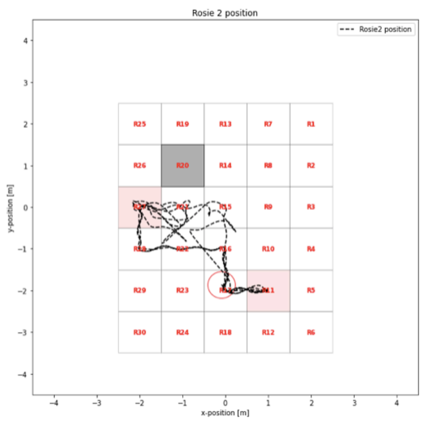
The local communication case involves an offline hierarchy between the robots which must be chosen in advance.Rosie 0 has complete freedom of motion in this situation, Rosie 1 must wait for Rosie 0's local trajectory generation, and Rosie 2 must be provided with the other robot trajectories before moving. Remind that only those robots detected in the sensing area are the source of wait; if no robots are present even Rosie 2 can directly move. Moreover target regions are marked with light colors, while the inaccessible region is identified with gray.
Initially from the starting region, each robot i follows the sequence of regions retrieved by the global planner in the form of preffix-suffix (see Table 6.2), safe navigation is then ensured by an MPC controller as described in Section 4.3, which guarantees to not violate the given task and to track such sequence of actions.



(a) Rosie 0 trajectory

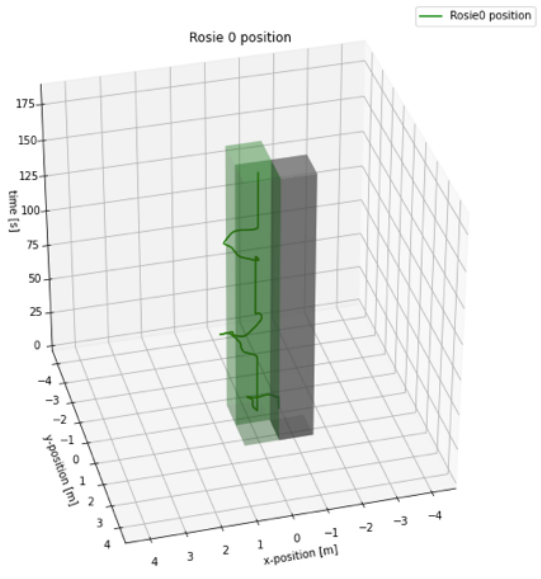


(b) Rosie 1 trajectory

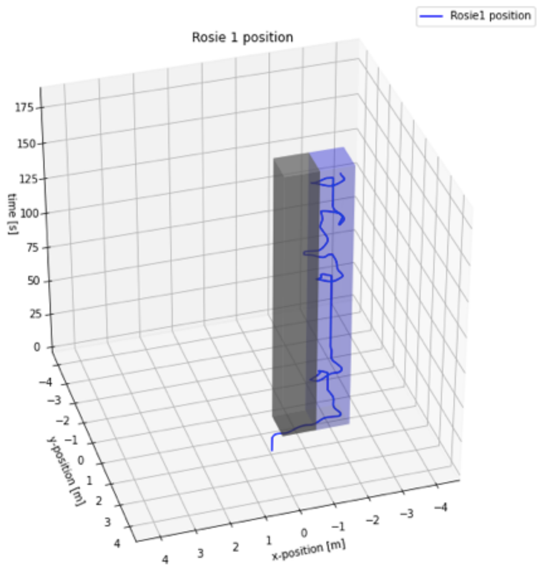


(c) Rosie 2 trajectory

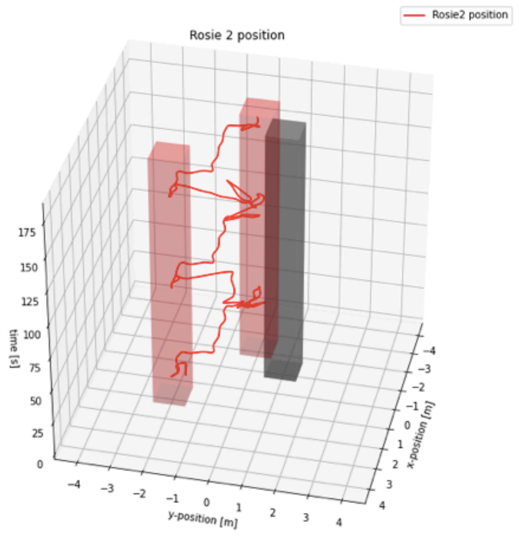
Figure 6.10: Experiment 2.1 - Rosie position trajectory



(a) Rosie 0 trajectory with respect to time



(b) Rosie 1 trajectory with respect to time

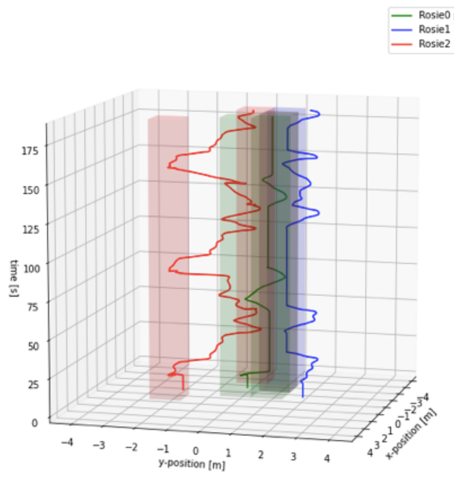


(c) Rosie 2 trajectory with respect to time

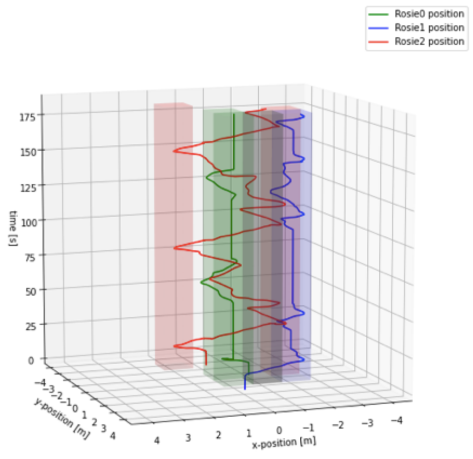
Figure 6.11: Experiment 2.1 - Evolution of robot trajectory with respect to time

Contrarily to Experiment 1, one robot must now survey at least one of the target regions but not both, as shown by the use of the ∨ ("or") operator in the LTL formulas rather than the ∧ ("and") operator used in the previous experiment. Clearly, the addition of one agent in the workspace might result in additional robots identified in the sensing area and, as a consequence, higher engagement of the online replan module. Possible collisions are detected whenever a higher priority robot enters within the sensing area of a lower priority one, then the motion coordination algorithm (see Section 5) is executed taking into

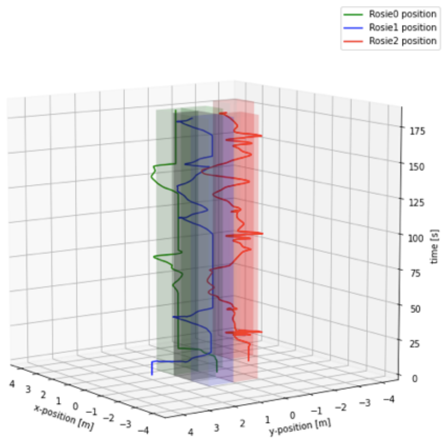
consideration the acquired agents' future trajectories. The real-time position trajectories of each robot are depicted in Figure 6.10 and Figure 6.11, where the latter shows the evolution of the Rosie positions over time.
During the 185 seconds of the experiment, the surveillance task of each robot is accomplished, with Rosie 0 monitoring largely region $R14$ rather than region $R15$ and Rosie 2 supervising both target regions equally. Besides all robots avoid the inaccessible region $R20$ as requested. Only once does Rosie 1 come quite near to that region (particularly see Figure 6.10 (b)).
The reason of that, relies on a locally computed trajectory in which the path connecting all points is collision-free and fulfills the task. On the other hand, it is quite close to region $R20$, and the PID controller used to track those points produces a borderline violating trajectory.
Figure 6.6 shows the evolution of the position trajectories of all robots with respect to time, where all trajectories are collision-free at every instant of time.
Conflicts are detected 9 times in total, and as soon as they arise the hierarchy motion order and the local trajectory generation process are activated to resolve them. As result of the assigned robot goals, a limited amount of online replan processes are taking place.
Due to the restricted workspace area for three or more Rosies, a huge number of these operations may occur depending on the tasks.



(a)



(b)

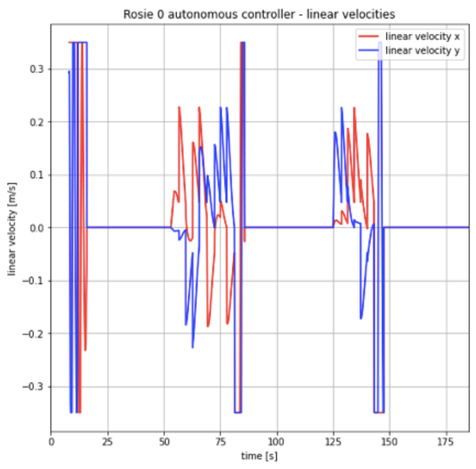


(c)

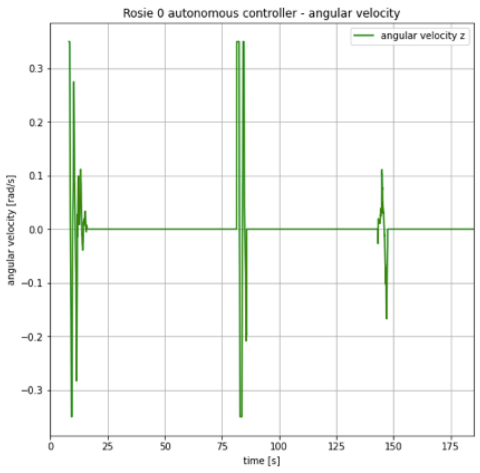
Figure 6.12: Experiment 2.1 - Evolution of Rosie trajectories with respect to time from different points of view

The real-time velocities of each Rosie are depicted in Figure 6.13. Because no input is applied, Rosie 0 and Rosie 1 linear velocities highlight time intervals where they are monitoring a specific area. Regardless, if more robots reach their sensing area, the multi-robot motion coordination technique is activated, and new inputs are applied to avoid collisions and fulfill the initial LTL task. A video demonstration of Experiment 2.1 can be found at the YouTube channel: Smart Mobility Lab - Gianmarco Fedeli [4].

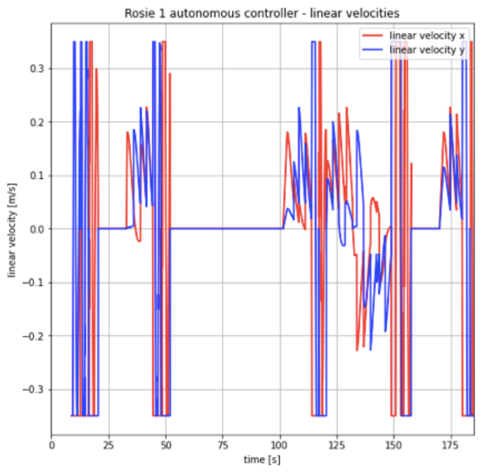
[4]URL video: https://youtu.be/wZHnku3nsHo



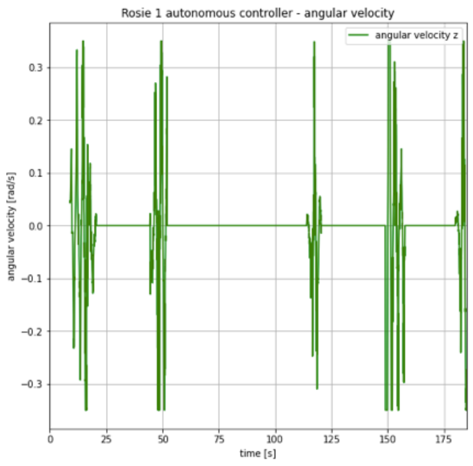
(a) Rosie 0 linear velocities



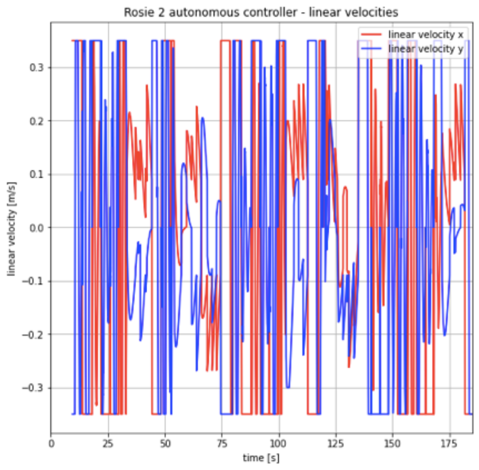
(b) Rosie 0 angular velocity



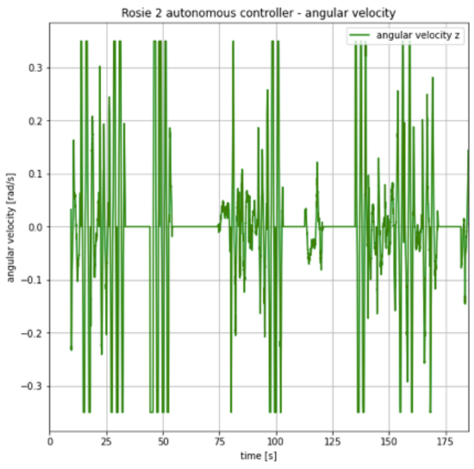
(c) Rosie 1 linear velocities



(d) Rosie 1 angular velocity



(e) Rosie 2 linear velocities

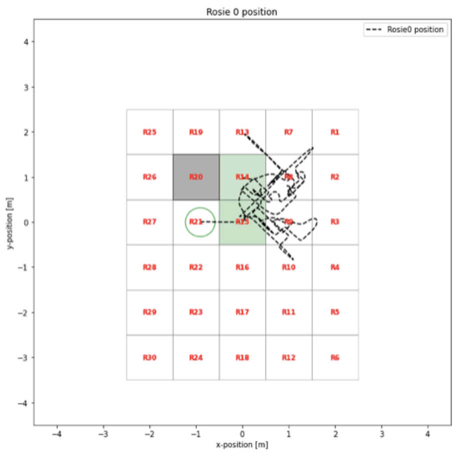


(f) Rosie 2 angular velocity

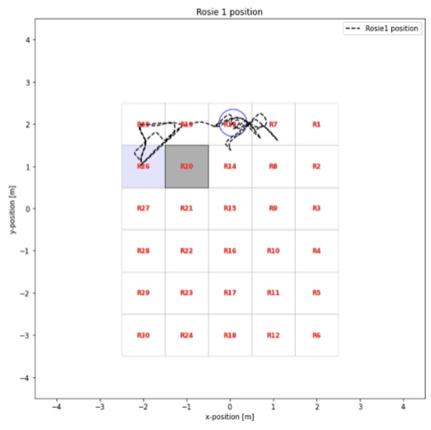
Figure 6.13: Experiment 2.1 - Rosies velocities

Communication-free case

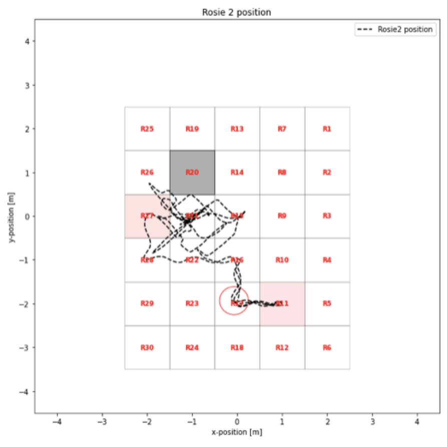
The same scenario is provided to test and demonstrate the reliability of the strategy proposed for the communication-free case, even as the number of robots involved in the multi-agent system grows.
All rosie specifications are those stated in Table 6.2, initially from the same starting regions (circles in the 2D plots), each robot i follows the sequence of regions obtained by the global planner in the form of preffix-suffix, with safe navigation ensured by an MPC controller. There are no additional obstacles introduced, however restricting access to region $R20$ makes it easier you to figure out how the trade-off between trap states and obstacle avoidance works, as specified in the local MPC cost function (see Section 5.2).



(a) Rosie 0 trajectory

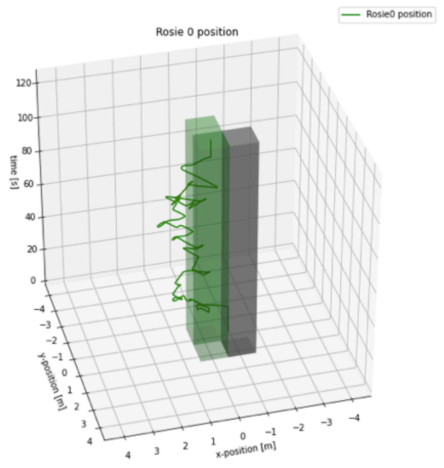


(b) Rosie 1 trajectory

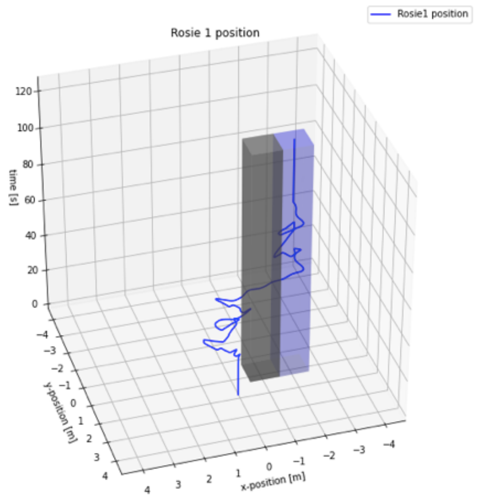


(c) Rosie 2 trajectory

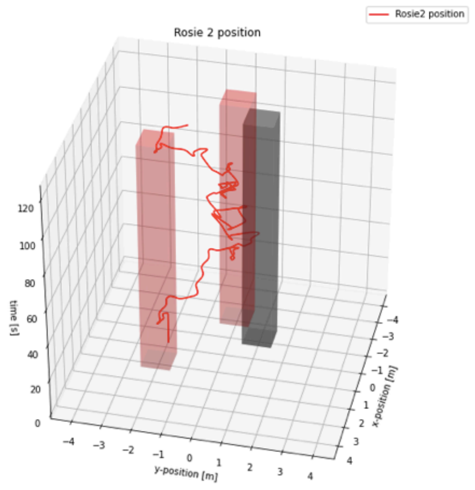
Figure 6.14: Experiment 2.2 - Rosie position trajectory



(a) Rosie 0 trajectory with respect to time



(b) Rosie 1 trajectory with respect to time



(c) Rosie 2 trajectory with respect to time

Figure 6.15: Experiment 2.2 - Evolution of robot trajectory with respect to time

Figure 6.14 and Figure 6.15 reveal subsequent hurried robot movements, causing not particularly smooth trajectories in the online replan stage. When a robot gets close to region $R20$ (trap state) and finds other obstacles within its sensing area, such behavior becomes more evident. On the other hand, as a consequence of such circumstance, the role of the gains $P_{\mathbb{O}}$, $P_{\mathbb{O}_t}$ contained in Equation 5.1 will be more clear.
As mentioned throughout the degree project, it is more crucial to prevent collisions than to fulfill the assigned task; clearly, all scenarios in which both requirements might be met are not part of this discussion. Gain $P_{\mathbb{O}}$ is set to be

bigger in order to prioritize obstacle avoidance, and suchăeffect can be noticed at various instants of time when two or three Rosies are detected by each other and are bordering on the dangerous area (region $R20$). When compared to the local communication case, the number of occurring conflicts is undoubtedly higher, and several small movements are produced as a result of the local MPC controller, which attempts to avoid both obstacles and trap states. Meanwhile, when the detection occurred for the first time, each point of the generated local trajectory is set as goal location.

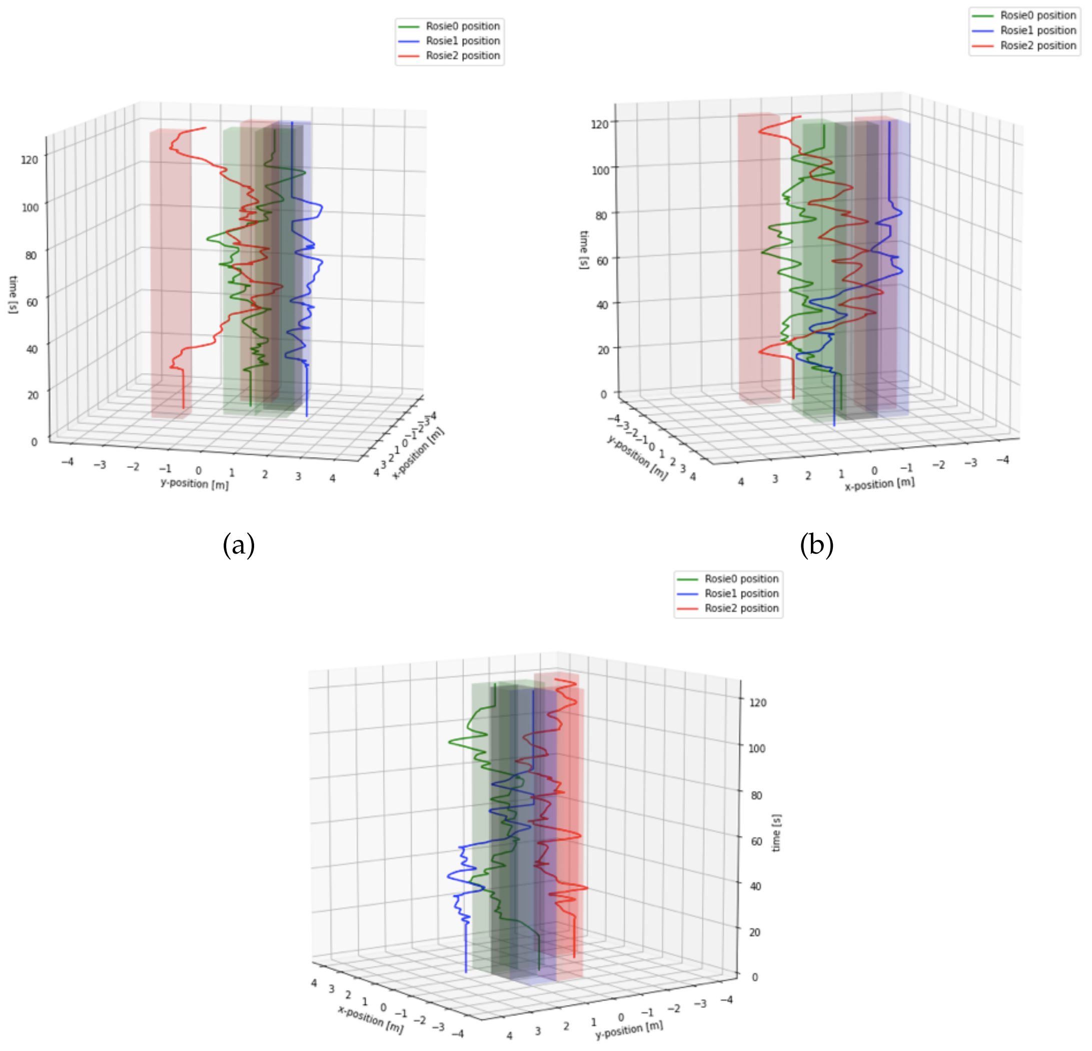
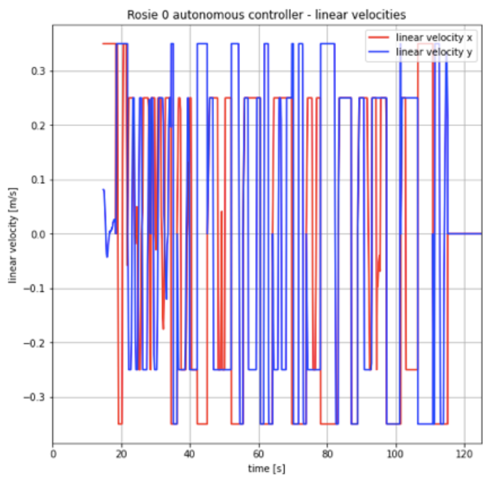


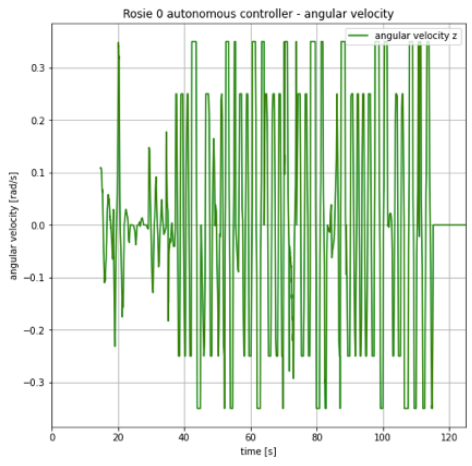
Figure 6.16: Experiment 2.2 - Evolution of Rosie trajectories with respect to time from different points of view

Whereas other strategies cause multiple deadlocks, the suggested strategy leads in no robots getting stuck in the same place for an extended period of time. The Rosie real-time velocities are depicted in Figure 6.17, where the angular velocity component is more stressed than in the previous example, because no yaw angle limits were imposed in the local MPC controller . Hence each Rosie is free to modify its orientation during the online replan stage, to facilitate the collision avoidance algorithm incorporated within the local MPC controller. A video demonstration of Experiment 2.2 can be found at the YouTube channel: Smart Mobility Lab - Gianmarco Fedeli [5].

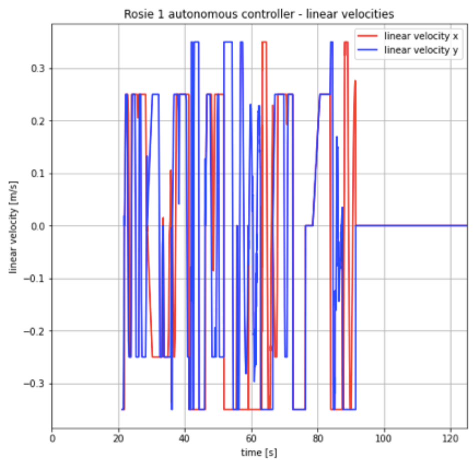
[5]URL video: https://youtu.be/q5_b41fprJU



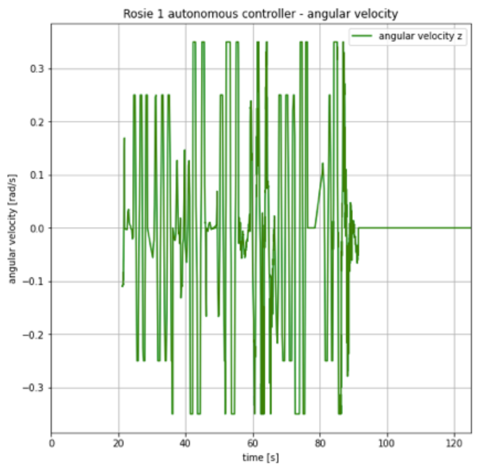
(a) Rosie 0 linear velocities



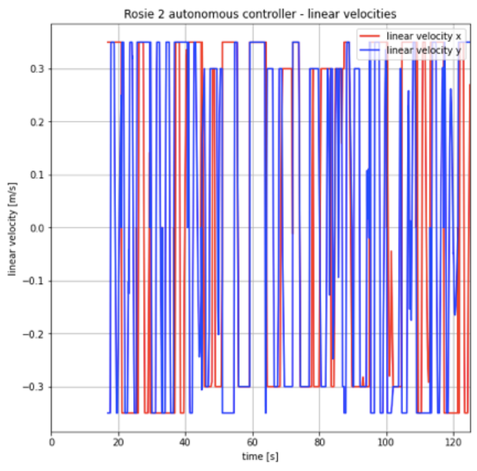
(b) Rosie 0 angular velocity



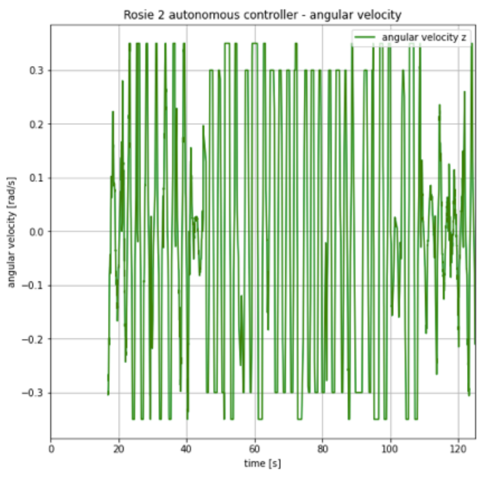
(c) Rosie 1 linear velocities



(d) Rosie 1 angular velocity



(e) Rosie 2 linear velocities



(f) Rosie 2 angular velocity

Figure 6.17: Experiment 2.2 - Rosies velocities

6.5 Experiment 3: Human in-the-loop

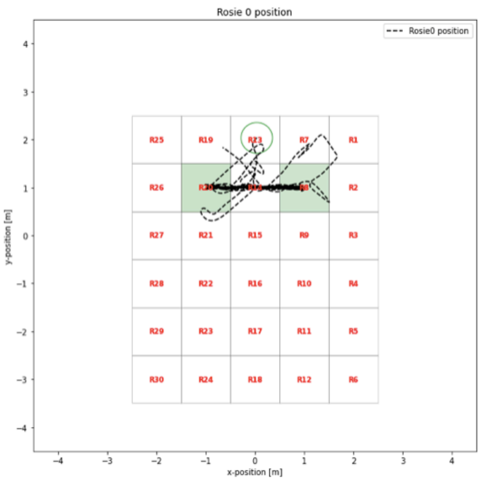
Consider a multi-agent system consisting of $N = 3$ HEBI Rosie robots, the dynamics of robot i is given by Equation (4.1), and the inputs constraints are set as follows. For all robots the maximum allowed linear velocity is set to 0.35 m/s, whereas the maximum admissible angular velocity is 0.35 rad/s. The sensing radius of each robot is $R = 0.8$ m and the safe radius is chosen to be 0.6 m. Initially, $\theta_i = 0$ and $v_i(0) = 0$.
Experiment 3 aims to evaluate the mixed-initiative controller explained in Section 5.3 in the human in-the-loop context, where a human may take control of one or more robots during the motion. All three robots are intended to survey some area in the worspace. Rosie 0 has target regions $R8$ and $R20$, Rosie 1 has $R22$ and $R28$ as goal regions, lastly Rosie 2 must monitore regions $R10$ and $R11$. The following table lists the associated LTL formulas, starting regions, and the offline plan generated by the LTL core planner in the prefix-suffix form:

EXPERIMENT #3 - SETTING			
Agent	Rosie 0	Rosie 1	Rosie 2
LTL task	$(\square\lozenge R8) \wedge (\square\lozenge R20)$	$(\square\lozenge R22) \wedge (\square\lozenge R28)$	$(\square\lozenge R10) \wedge (\square\lozenge R11)$
Starting region	$R13$	$R22$	$R10$
Preffix plan	$[R7, R8, R14, R20, R14]$	$[R28, R22]$	$[R11, R10]$
Suffix plan	$[R8, R14, R20, R14]$	$[R28, R22]$	$[R11, R10]$

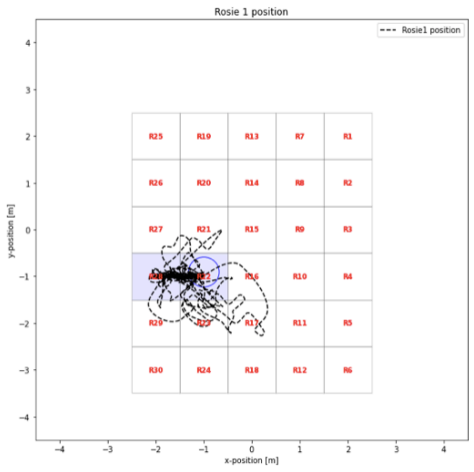
Table 6.3: Experiment #3

One agent under human control

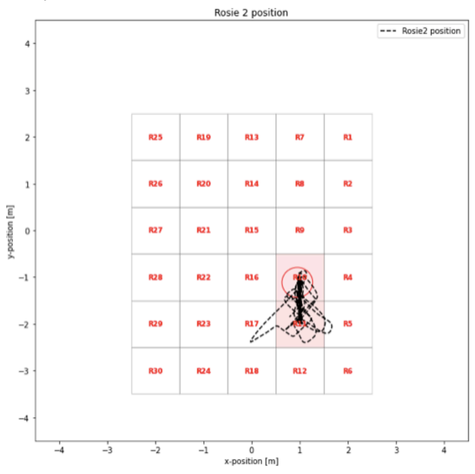
Initially, just Rosie 1 can be controlled by a human, with control inputs executed through a joystick equipped with bluetooth. Human initiative certainly influences robot autonomy when evaluating the performance of the proposed mixed-initiative controller, where operator inputs and autonomous commands are fused, as detailed in Section 5.3. Moreover, an obstacle is introduced in the form of the human who controls the robot movements in the shared workspace. At the beginning the human being stands in a neutral area, away from the robot's motions and without acting on Rosie 1. The real-time positions of each Rosie is depicted in Figure 6.18 and Figure 6.19, initially all robots are not affected by the human obstacle and no joystick inputs are applied during the first 60 seconds.



(a) Rosie 0 trajectory

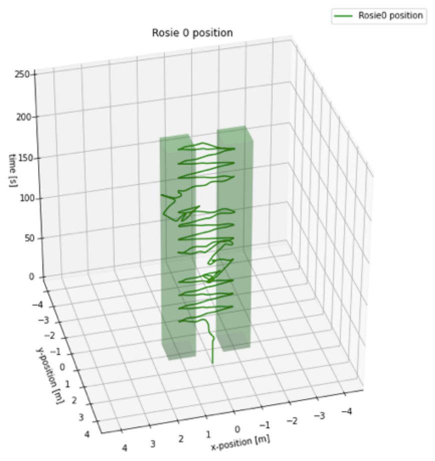


(b) Rosie 1 trajectory

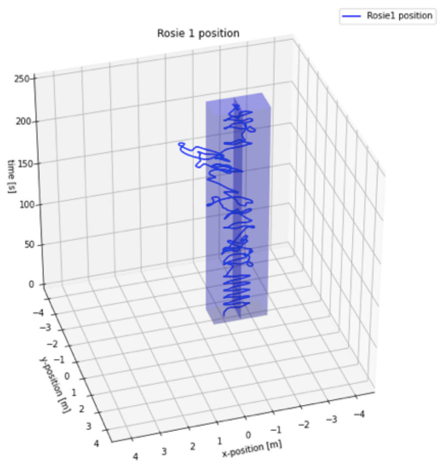


(c) Rosie 2 trajectory

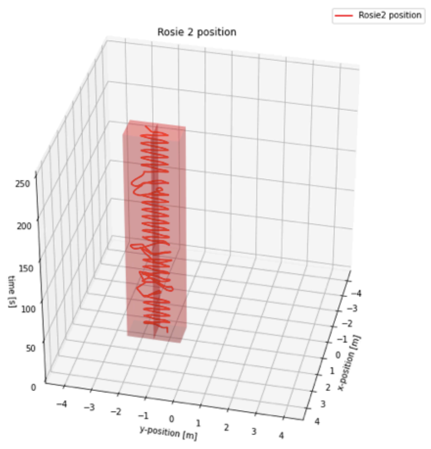
Figure 6.18: Experiment 3.1 - Rosie position trajectory



(a) Rosie 0 trajectory with respect to time



(b) Rosie 1 trajectory with respect to time



(c) Rosie 2 trajectory with respect to time

Figure 6.19: Experiment 3.1 - Evolution of robot trajectory with respect to time

The plots above demonstrate that all robots are efficiently monitoring the given target regions as a result of the autonomous controller action. In this specific scenario, the communication-free strategy is used to deal with an individual obstacle who is randomly walking in the environment. Particular emphasis should be paid to the controlled robot Rosie 1; in fact, throughout the time intervals [100 s, 145 s], human inputs are dominating, allowing the robot to explore other areas while obstacles and other robots are far away. When the person is walking towards Rosie 1 or one of the other robots approaches, human initiative is considered less and less as the distance from the obstacle increases. As the obstacle gets closer to the safe distance, the operator's inputs are no longer considered.

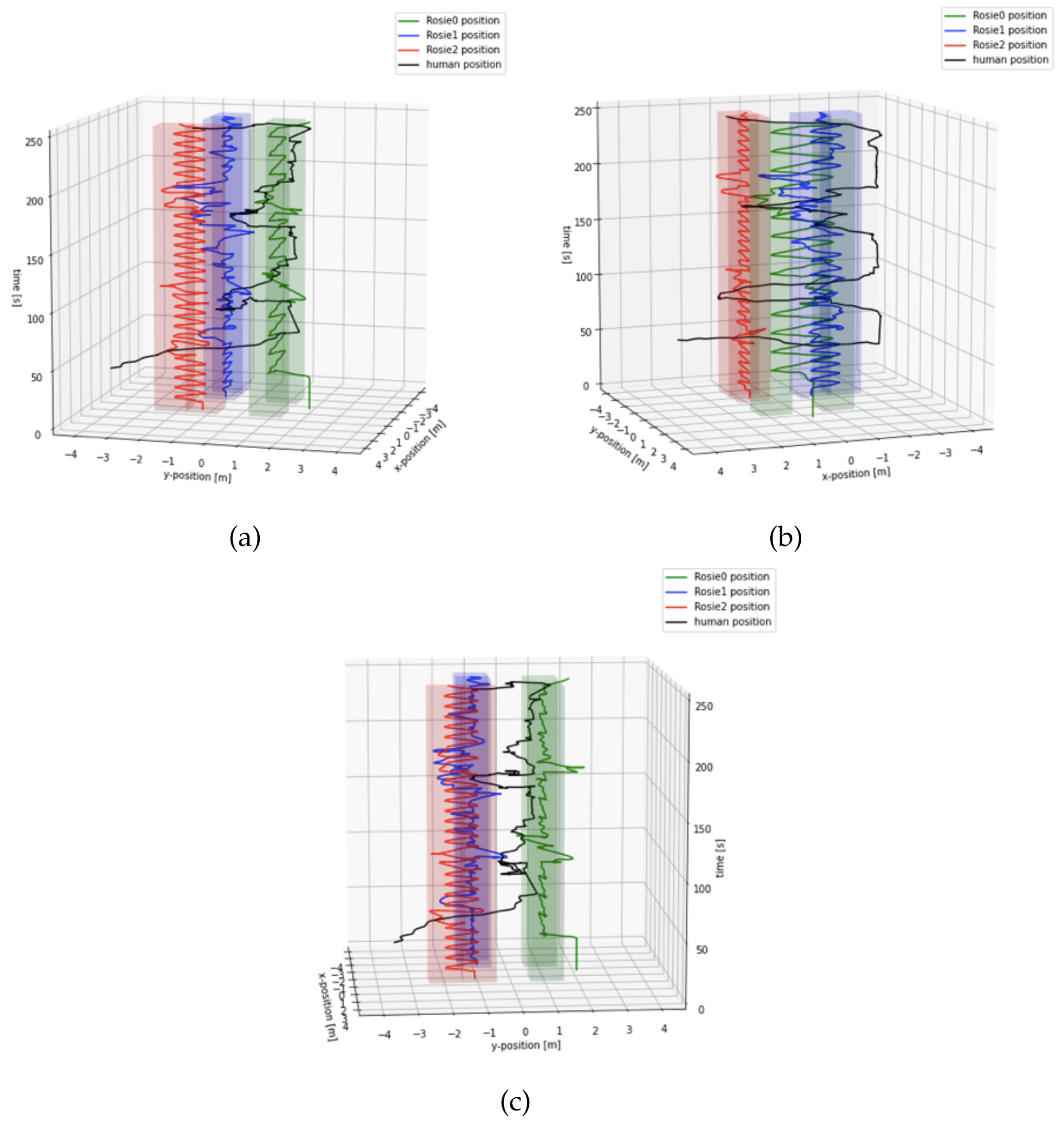
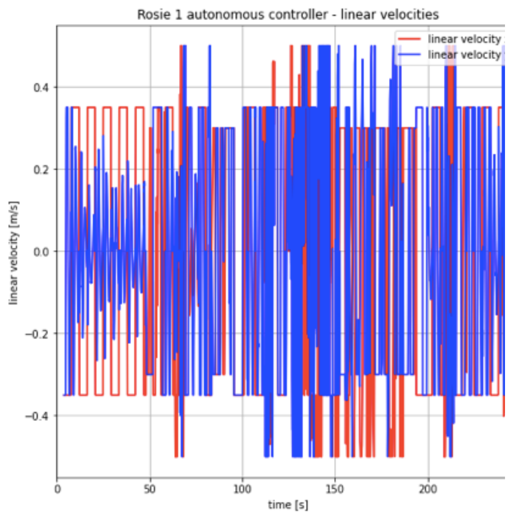


Figure 6.20: Experiment 3.1 - Evolution of Rosie position trajectories and human obstacle with respect to time from different points of view

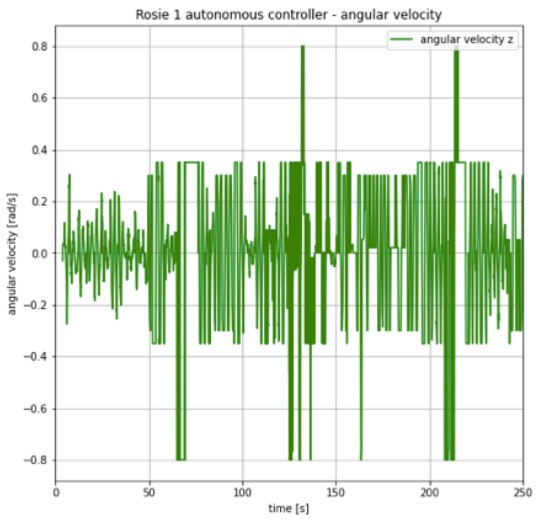
Additionally, some dangerous behaviours are tested, including the human attempting to guide the robot towards itself, however the mixed-initiative controller prevents these actions ensuring safety. The function $k(x, \mathbb{O}, \mathbb{O}_t)$ described in Section 5.3 is the key of such results; it may be selected differently depending on the final desired behavior and the attention given to operator's inputs.

Conflicts are detected 13 times in total, considering all robots. Furthermore, Figure 6.20 shows, from different points of view, that no collisions occur during the experiment. The real-time velocities of the controlled robot Rosie 1 are depicted in Figure 6.21, including human inputs and the autonomous controller action. A video demonstration of Experiment 2.2 can be found at the YouTube channel: Smart Mobility Lab - Gianmarco Fedeli [6].

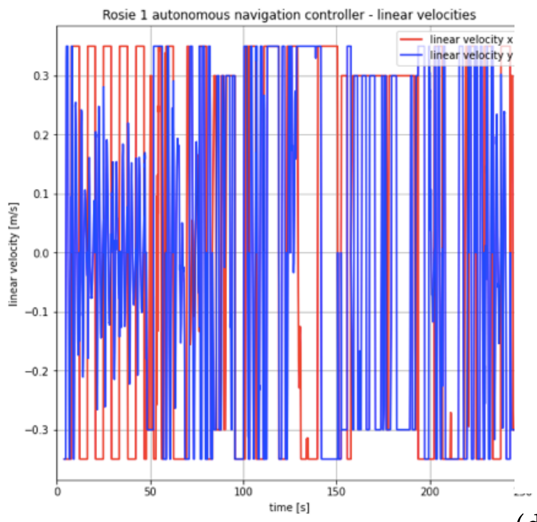
[6]URL video: https://youtu.be/2hWe5Wu52Bg



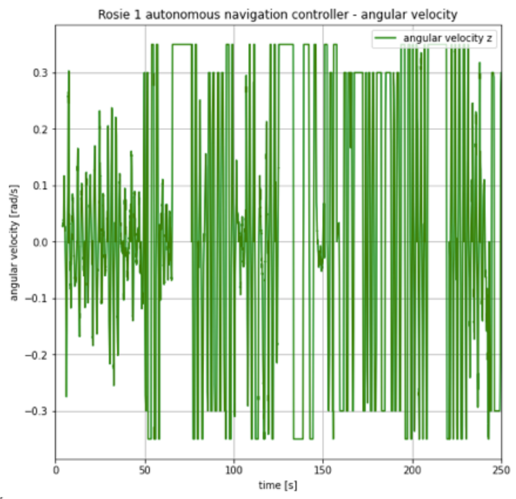
(a) Rosie 1 MIC linear velocities



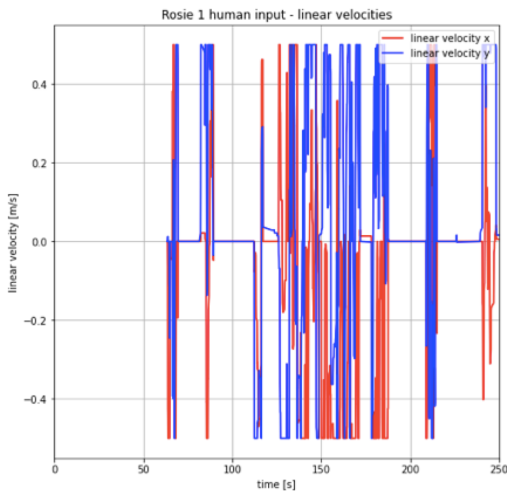
(b) Rosie 1 MIC angular velocity



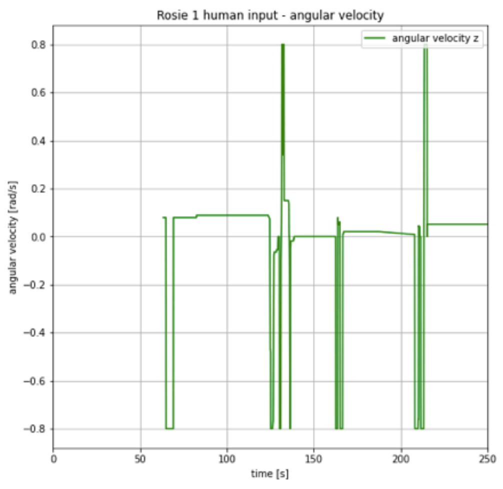
(c) Rosie 1 autonomous linear velocities



(d) Rosie 1 autonomous angular velocity



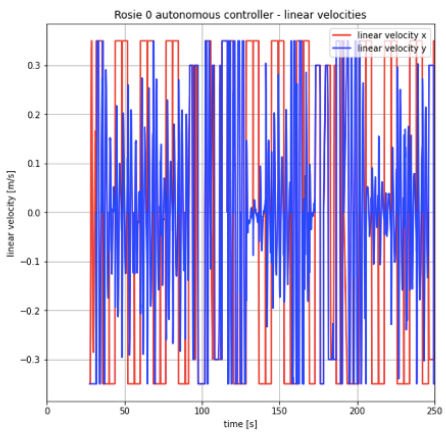
(e) Rosie 1 human linear velocities



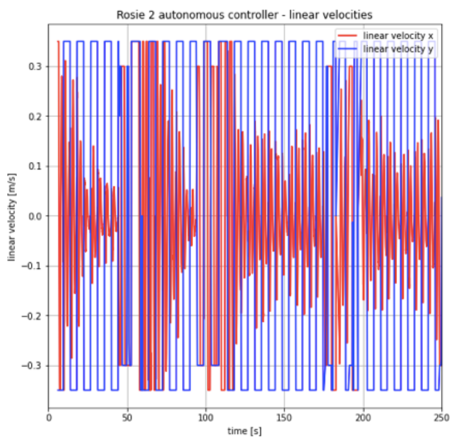
(f) Rosie 1 human angular velocity

Figure 6.21: Experiment 3.1 - Rosie 1 velocities

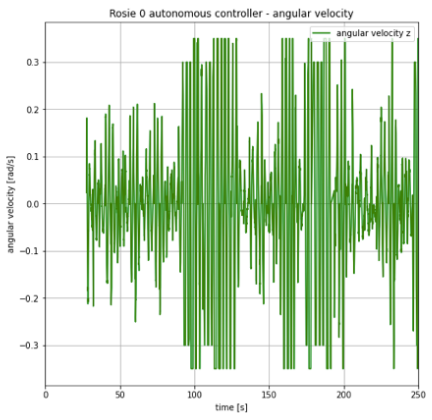
The operator turns the robot of 180 degrees after 60 seconds, to demonstrate that the outer MPC restores the appropriate orientation and then takes the robot to the next region retrieved by the LTL planner. Notice that the maximum allowed human linear velocity is set to 0.5 m/s, while the highest permissible human angular velocity is 0.8 rad/s. Whenever Rosie 1 is close to an obstacle or another agent, it is clear that the resulting velocities are those computed by the autonomous controller; conversely, when the controller robot is far from those objects, the resulting velocities are entirely the operator velocities. In all other circumstances, a fusion of the autonomous controller's and the human's velocities is actuated, with the combination result guided by the function $k(x, \mathbb{O}, \mathbb{O}_t)$. Finally the real-time velocities of Rosie 0 and Rosie 2 are reported in Figure 6.22.



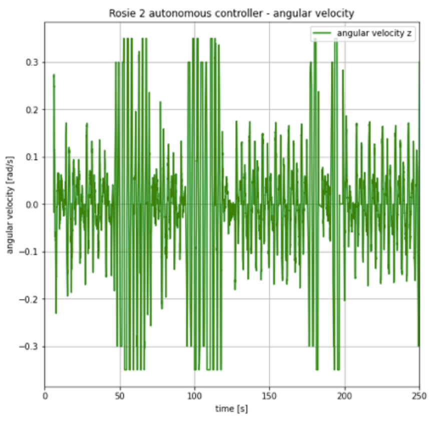
(a) Rosie 0 linear velocities



(b) Rosie 2 linear velocities



(c) Rosie 0 angular velocity

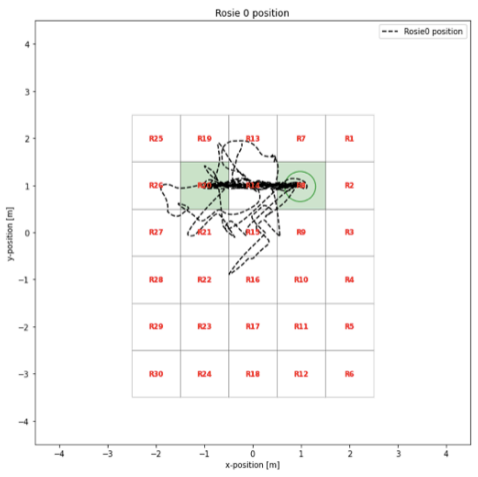


(d) Rosie 2 angular velocity

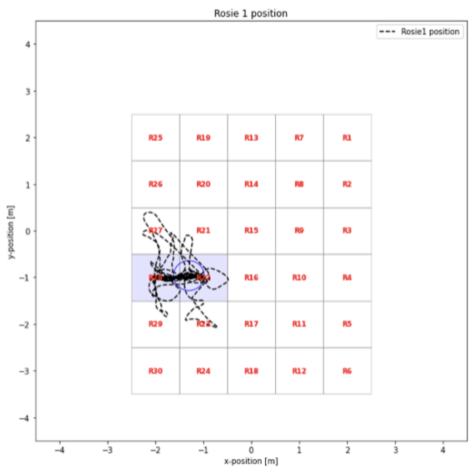
Figure 6.22: Experiment 3.1 - Rosie 0, Rosie 2 velocities

Two agents under human control

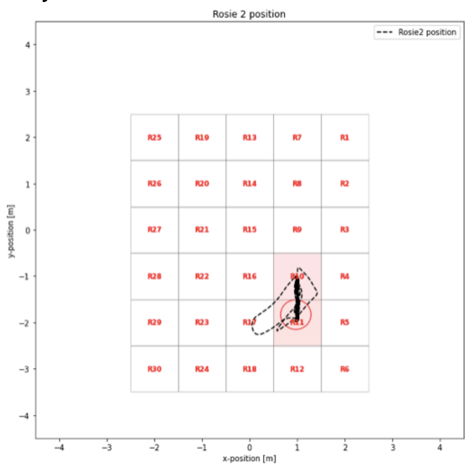
Both Rosie 0 and Rosie 1 may now be controlled by a human, as previously operator control inputs are executed via two independent joysticks connected with bluetooth. In addition, no obstacles are considered and the communication-free case is adopted. The main purpose of of enabling more agents to be controlled is to validate and understand potential functionalities of the developed work.
The real-time positions of each Rosie is depicted in Figure 6.23 and Figure 6.24, and as can be noticed the starting regions differ from the last example. Initially Rosie 0 is not affected by human inputs because they are not applied for the first 30 seconds.



(a) Rosie 0 trajectory

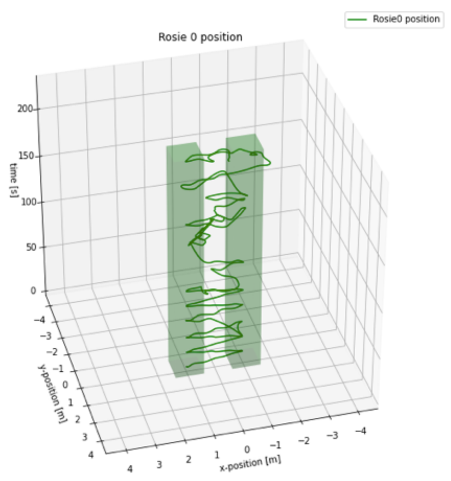


(b) Rosie 1 trajectory

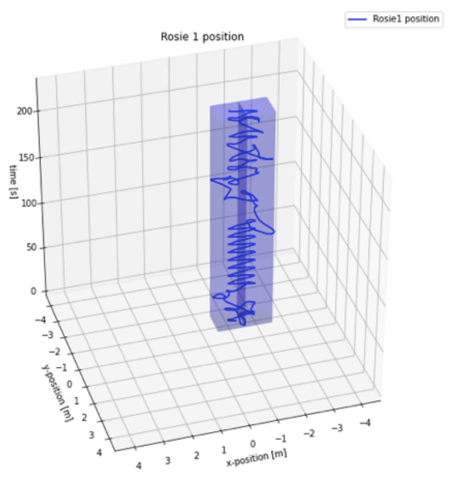


(c) Rosie 2 trajectory

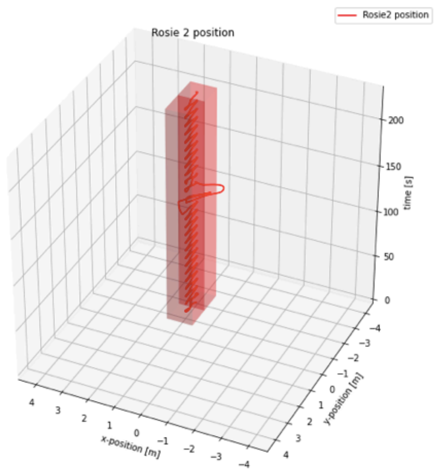
Figure 6.23: Experiment 3.2 - Rosie position trajectory



(a) Rosie 0 trajectory with respect to time



(b) Rosie 1 trajectory with respect to time



(c) Rosie 0 trajectory with respect to time

Figure 6.24: Experiment 3.2 - Evolution of robot trajectory with respect to time

Concerning the prescribed tasks, it is worth noting that they are all completed without collisions and always through safe navigation as guaranteed by the MPC controller. In more occassions, the two operators try to crush into all moving robots in the environment; nevertheless, the mixed-initiative controller precludes those risky situations and exclusively uses the autonomous navigation inputs arising from the local MPC computations. Human inputs are particularly effective to explore new locations and extend the range of surveillance in specific time periods. Indeed, Rosie 0 motion area covers not only the two target regions but also middle regions like $R15$ and $R16$.

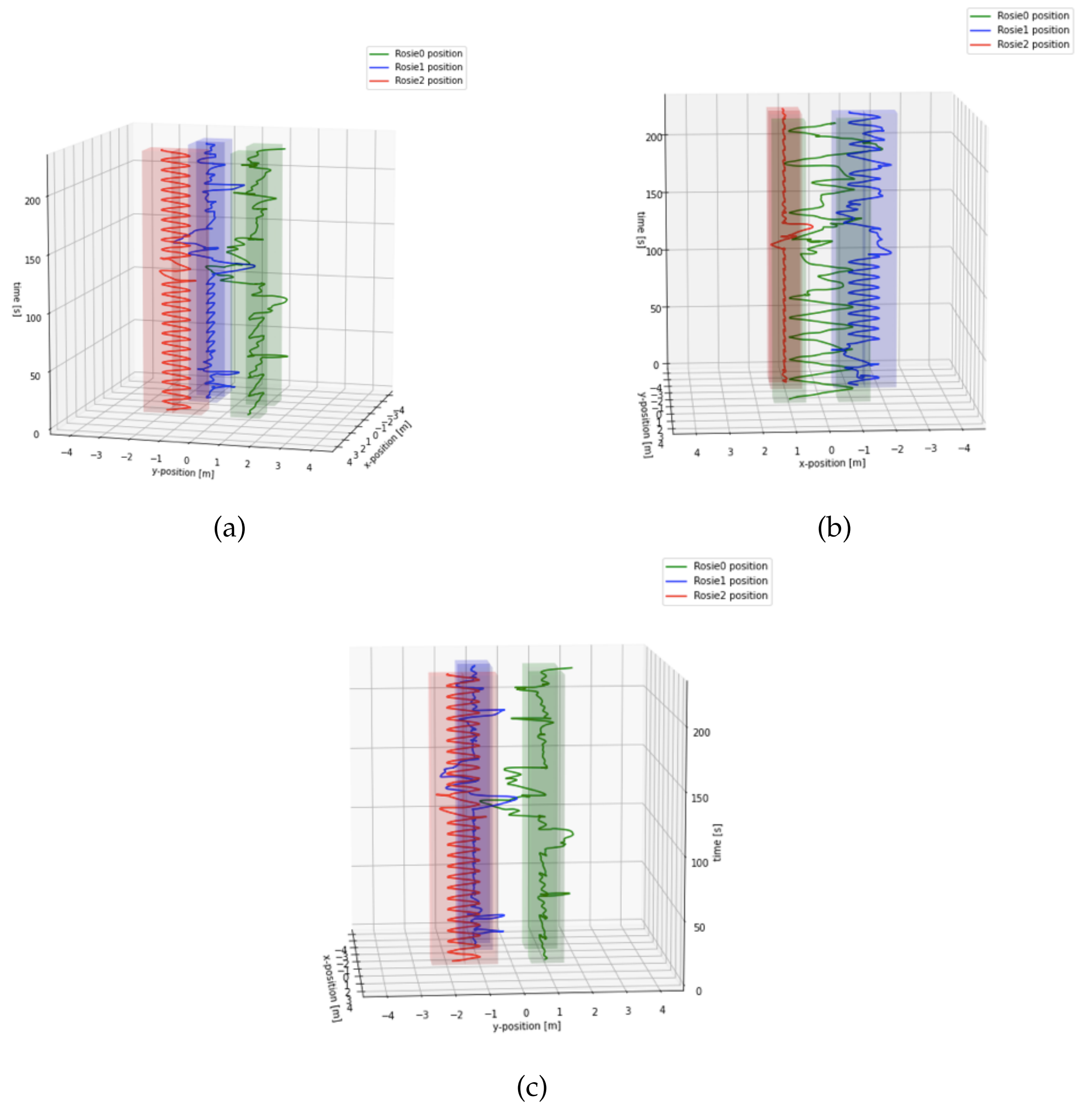
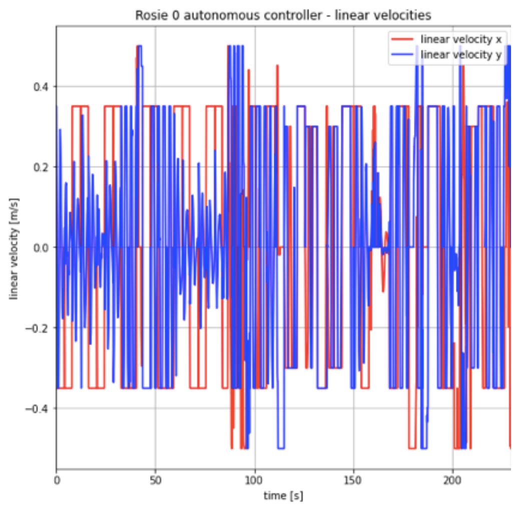
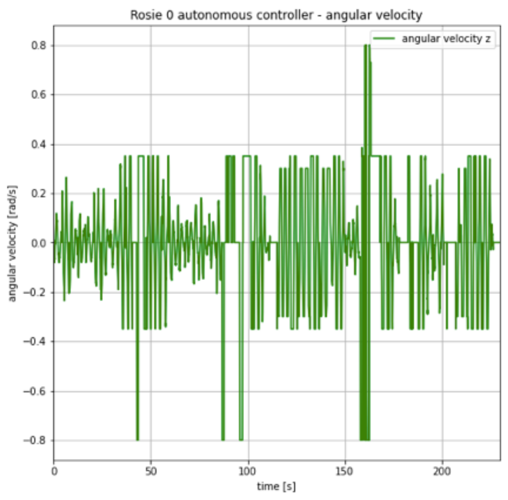


Figure 6.25: Experiment 3.2 - Evolution of Rosie position trajectories with respect to time from different points of view

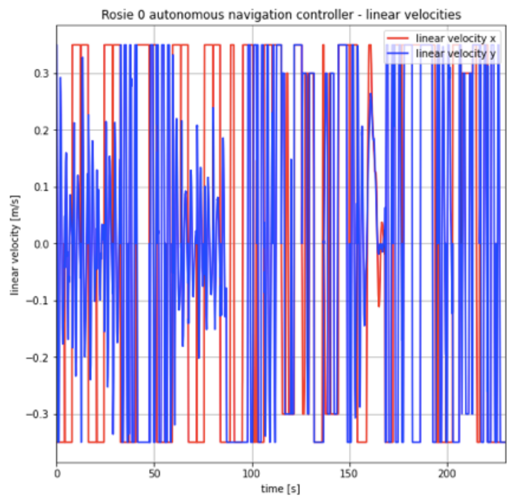
The subsequent plots depicted in Figure 6.26, Figure 6.27 and Figure 6.28 show the real-time velocities of each Rosie. Where the operator's maximum allowed linear velocity is set to 0.5 m/s and the angular velocity limit is set to 0.8 rad/s. While the same limitations that were imposed in all experiments are inserted as constraints to the appropriate MPC controller. Even in this case, the operator may sometimes turn the robot 180 degrees to highlight how the MPC reestablish the correct orientation before proceeding.



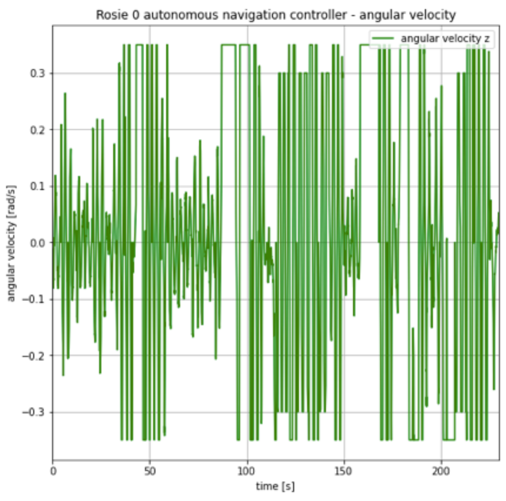
(a) Rosie 0 MIC linear velocities



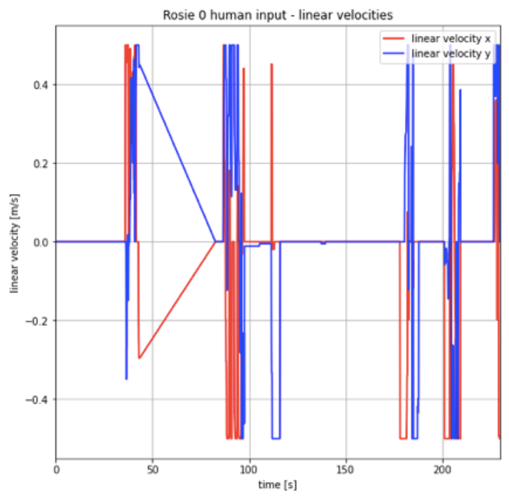
(b) Rosie 0 MIC angular velocity



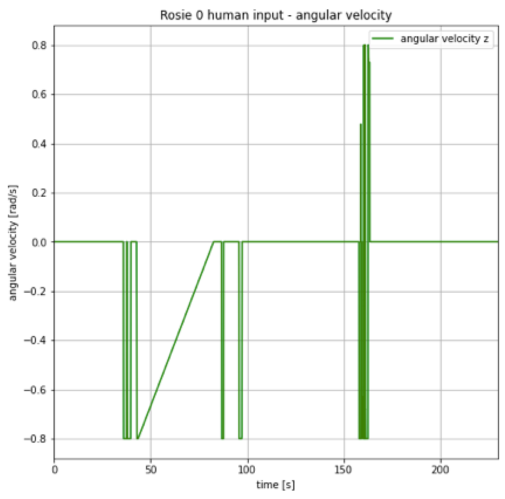
(c) Rosie 0 autonomous linear velocities



(d) Rosie 0 autonomous angular velocity

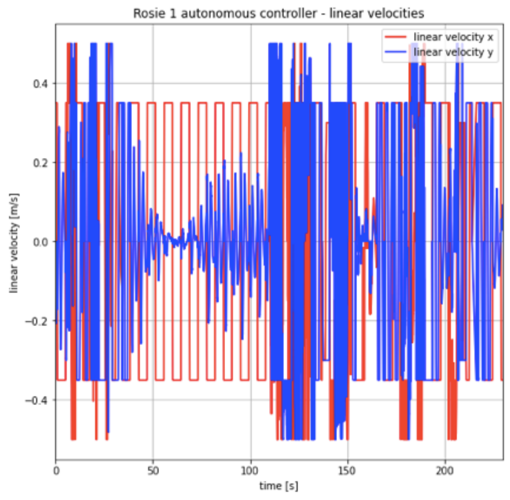


(e) Rosie 0 human linear velocities

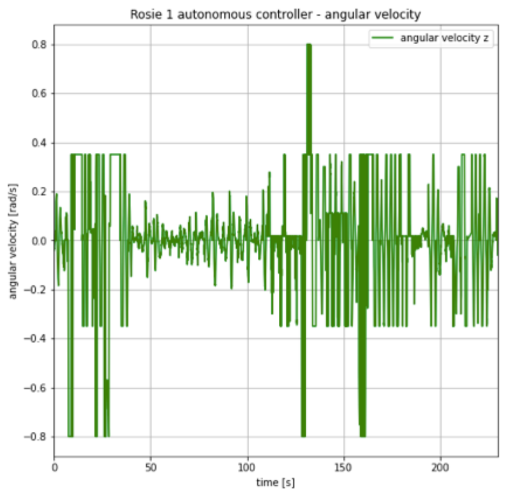


(f) Rosie 0 human angular velocity

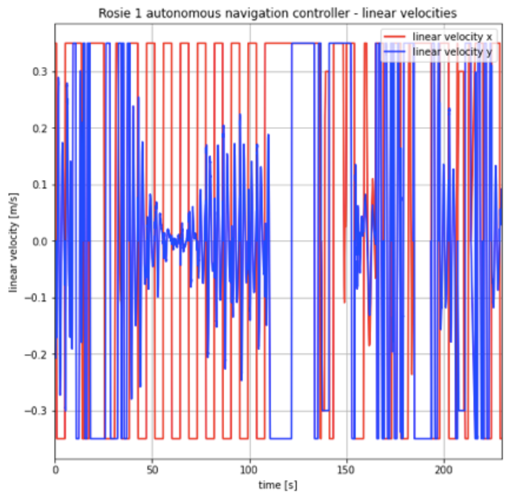
Figure 6.26: Experiment 3.2 - Rosie 0 velocities



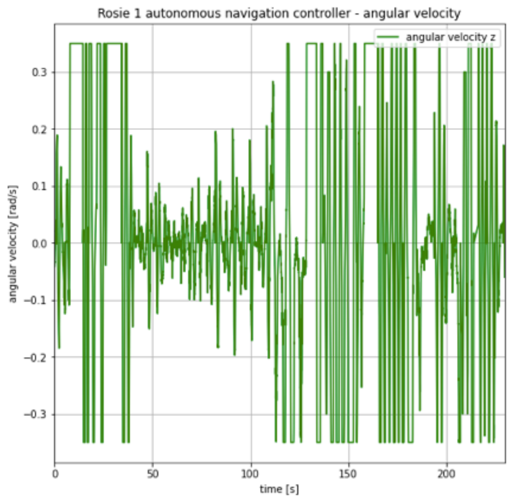
(a) Rosie 1 MIC linear velocities



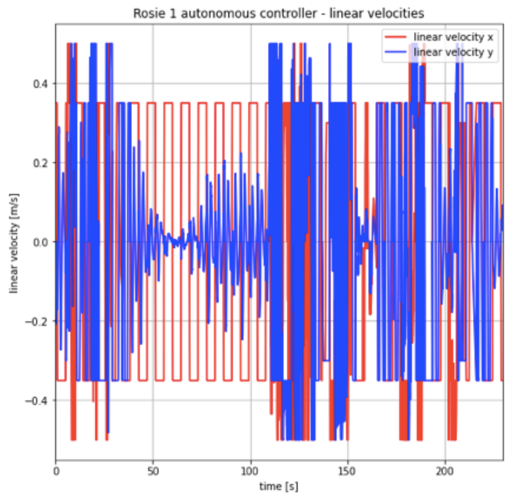
(b) Rosie 1 MIC angular velocity



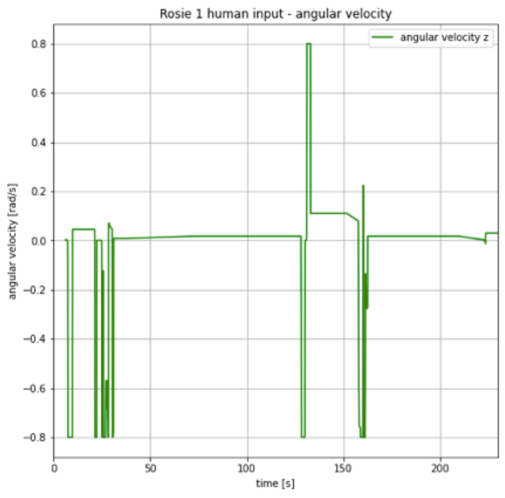
(c) Rosie 1 autonomous linear velocities



(d) Rosie 1 autonomous angular velocity

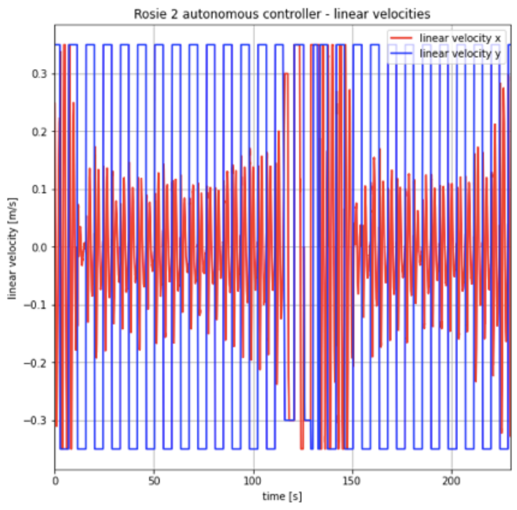


(e) Rosie 1 human linear velocities

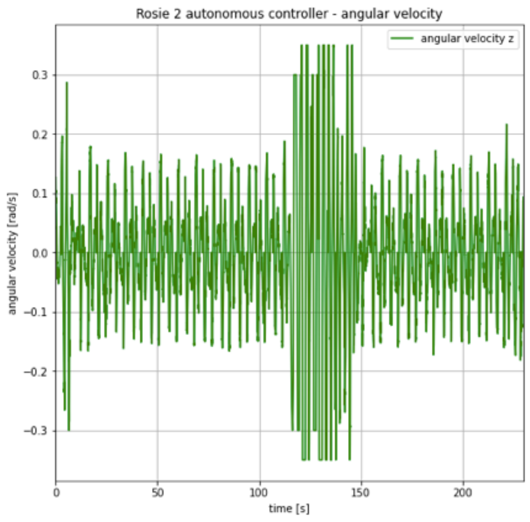


(f) Rosie 1 human angular velocity

Figure 6.27: Experiment 3.2 - Rosie 1 velocities



(a) Rosie 2 linear velocities



(b) Rosie 2 angular velocity

Figure 6.28: Experiment 3.2 - Rosie 2 velocities

6.6 Results discussion

The original objective of the degree project was to design an efficient and effective motion coordination algorithm for multi-robot systems, including parallel experiments to validate the entire work. Experiments presented in these Chapter demonstrate that the overall strategies are very valuable in both basic scenarios and more complicated situations where additional obstacles are represented by people walking in the workspace. Testing the method described throughout Section 5.2, is considered essential to make a motion coordination strategy applicable in settings such as industries, companies, warehouses, and within novel technologies.
The experiments performed at the KTH Royal Institute of Technology's Smart Mobility Laboratory compare both cases addressed in the thesis. The local-communication case provides a very reliable and effective strategy for overcoming even the most difficult tasks, resulting in very smooth movements. When obstacles or additional agents are introduced in the same working area, some assumptions must be made. Local communication between robtos may be made available in a variety of conditions; however, prior knowledge of non-static obstacles moving randomly is more complicated. Concerning the communication-free case implementation, it allows the motion coordination algorithm's applicability to be extended, particularly in circumstances where people can pass through the motion area or when prior information is unavailable. Analyzing all tests where communication-free case has been appplied, a more choopy motion is obtained to prevent collisions, but the final result is still reliable and safe. Another factor to consider is the maximum allowable velocity, which influences robot movement behaviors during the online replan process, particularly in avoiding obstacles. Clearly higher velocity permission is given to the robot, faster it may react to environmental changes with compulsive responses to them as drawback. Because all of the robots are sensorless, their positions are determined by an external system, as detailed in Section 6.1. Nonetheless, thanks to the ROS framework, where the actual position of each object is shared as a ROS topic, the proposed solutions are unaffected by the source of data acquisition. All experiments were carried out in an indoor setting, there are some extra factors to consider in outdoor environments, however the developed approaches may be applied adjusting some specific details. No collisions occured in both cases, showing the reliability

and safety of all techniques, achieving one of the primary starting objectives. Furthermore, even when additional obstacles are included and the workspace gets crowded, all tasks have been fulfilled in an acceptable computational time. The human-in-the-loop scenario the designed mixed-initiative controller has proven the chosen function success, preventing dangerous actions and satysfing the assigned tasks. The proposed strategies can handle more challenging tasks, and they also apply to heterogeneous multi-agent systems consisting of entities of different type (for instance ground, air and water vehicles).

Chapter 7

Conclusions

The degree project focuses on the design of an effective and efficient motion coordination algorithm for multi-robot systems, where its application is supposed to deal with a partial known environment. Firstly a safe navigation controller for a single robot, is designed to ensure that the given LTL task is fullfilled during the entire motion (see Chapter 4). Two approaches are shown, however the final autonomous navigation controller used is a model predictive controller, which ensures safety even with theoretical demonstration. Then, a multi-robot system is considered, in which all of the robots are intended to move in a partially known environment. This scenario assumed to deal with non-static obstacles such as people walking in the environment, new obstacles appearing during the experiments. In Chapter 5 Along the way two cases were addressed, in which the corresponding motion coordination algorithm for multi-robot systems slightly differ. Initially local communication between robots is assumed. Each robot is given with a sensing area with a set radius; communication is shared among them within this area and used to find a path-free trajectory, where the main goal is to coordinate the motion meanwhile not violating the assigned LTL task. The latter case represents a more realistic scenario, in which no communication is provided and the robots should deal with random environmental changes. An MPC controller with a collision avoidance term and a term related with trap states addresses this issue, in order to prevent if possible unacceptable runs. Finally, a mixed-initiative controller was adopted to handle the human-in-the-loop scenario, and its applicability was extended to multi-robot systems, where the operator may assume control of one or more robots in certain conditions.

Finally, the tests conducted (see Chapter 6) suggest some potential improvements connected to the presented approaches as well as some future work that may be pursued.

7.0.1 Future Work

Future works include to tackle a global task, where all the involved agents must cover the workspace and optimize the surveillance activity throughout the whole surface. Indeed, an optimization problem can be formulated and then solved anytime environmental changes take place. Drones can be used to evaluate each solution in 3D space, and manipulators mounted on top of the mobile platform to perform pick and place routines. Concerning the human in-the-loop scenario, future studies should predict operator movements through a reinforcement learning approach or high-level AI techniques.

Appendix A

MPC stability

Theorem 1 (Maciejowski)**.**
Assume model predictive control is appplied at time instant k to the system dynamics:

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), u(k))$$

by minimizing the cost function

$$V(k) = \sum_{i=1}^{N} J(x(k+i), u(k+i-1))$$

subject to:

$$\begin{aligned}
&x_{k+1+i} = f(x_{k+i}, u_{k+i}), && \forall i,\ 0 \le i \le N-1 \\
&x_{k+i} \in X, x_{k+i} \in \mathcal{U} && \forall i,\ 0 \le i \le N-1 \\
&x_{k+N} = 0 \\
&x_0 = x(k)
\end{aligned}$$

where $J(\boldsymbol{x}, u) \ge 0$ and $J(\boldsymbol{x}, u) = 0$ only if $x = 0$ and $u = 0$, and J is monotonically decreasing. Suppose an equilibrium condition is defined by $\boldsymbol{x} = 0$, and $u = 0$. The receding horizon method is applied, with only the first element used from the optimizing input sequence.
Then the equilibrium point is stable providing that the optimization problem is feasible and is solved at each step.

Proof.
Let $V_0^*(k)$ be the optimal value of V corresponding to the optimal input signal u_0^*. Clearly, $V_0^*(k) \geq 0$ and $V_0^*(k) = 0$ only if $\boldsymbol{x}(k) = 0$. In fact if $\boldsymbol{x}(k) = 0$, then the optimal solution is to set $u(k+i) = 0$ for all i.
To show that $V_0^*(k+1) \leq V_0^*(k)$, and hence that $V_0^*(k)$ is a Lyapunov function.
Let consider:

$$\begin{aligned}
V_0^*(k+1) &= \min_u \sum_{i=1}^{N} J(x(k+1+i), u(k+i)) \\
&= \min_u \Big[\sum_{i=1}^{N} J(\boldsymbol{x}(k+i), u(k-1+i)) \;-\; J(\boldsymbol{x}(k+1), u(k)) + \\
&\quad J\left(\boldsymbol{x}\left(k+1+N\right), u\left(k+N\right)\right) \Big] \\
&\leq -J\left(\boldsymbol{x}(k+1), u_0^*(k)\right) + \; V_0^*(k) + \\
&\quad \min_u \Big[J\left(x\left(k+1+N\right), u\left(k+N\right)\right) \Big]
\end{aligned}$$

since the optimum cannot be worse that keeping the optimal solution found at time k. But assuming as terminal constraint $\boldsymbol{x}\left(k+N\right) = 0$, thus $u\left(t+N_p\right) = 0$ and remain at $x = 0$.
This gives

$$\min_u \left\{ J\left(\boldsymbol{x}\left(k+1+N\right), u\left(k+N\right)\right) \right\} = 0$$

Since $J\ (x(k), u_0^*(k)) \geq 0$, then

$$V_0^*(k+1) \leq V_0^*(k)$$

and therefore, $V_0^*(k)$ is a Lyapunov function. ■

A generalization of the terminal constraint idea is to specify a terminal contstraint set $\mathcal{X}_f$, which contains the origin rather than just a single point. More details and proof can be found in [6].

Appendix B

MPC persistent feasibility

Lemma B.0.1.
Given the receding horizon control problem $V_N(x(k))$:

$$\begin{aligned}\min_{\mathbf{U}} \quad & [\boldsymbol{x}(k+N) - \boldsymbol{x}_{des}]^T Q_N [\boldsymbol{x}(k+N) - \boldsymbol{x}_{des}] \quad + \\ & \sum_{i=0}^{N-1} \Big[[\boldsymbol{x}(k+i) - \boldsymbol{x}_{des}]^T Q [\boldsymbol{x}(k+i) - \boldsymbol{x}_{des}] + \boldsymbol{u}(k+i)^T R \boldsymbol{u}(k+i) \Big]\end{aligned}$$

subject to:

$$\begin{aligned} & \boldsymbol{x}_{k+1+i} = f(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}), && \forall i,\ 0 \le i \le N-1 \\ & \boldsymbol{x}_{k+i} \in X, \boldsymbol{u}_{k+i} \in \mathcal{U} && \forall i,\ 0 \le i \le N-1 \\ & \boldsymbol{x}_{k+N} \in X_f \\ & \boldsymbol{x}_0 = \boldsymbol{x}(k) \end{aligned}$$

If X_1 is control invariant, then the receding horizon control problem defined above is persistently feasible.

Proof.
$\mathrm{X}_1 \subseteq \mathrm{Pre}\,(X_1) \cap X = X_0, \forall x \in X_0$. Then applying the first-step of the model predictive control u_0^*, the resulting state is given by $x^+ = f\left(x, u_0^*\right) \in X_1 \subseteq X_0$. ■

Theorem 2.
If X_f is control invariant, then the receding horizon optimization $V_N(x(k))$ is persistently feasible.

References

[1] K. Azarm and G. Schmidt. "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation". In: 4 (1997), 3526–3533 vol.4. DOI: 10.1109/ROBOT.1997.606881.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[3] Robin Baran et al. "A ROS Package for Human-In-the-Loop Planning and Control under Linear Temporal Logic Tasks". In: (2021), pp. 2182–2187. DOI: 10.1109/CASE49439.2021.9551648.

[4] Robin Baran et al. "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)*. IEEE. 2021, pp. 2182–2187.

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

[6] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[7] D.Panagou. "Motion planning and collision avoidance using navigation vector fields". In: (2014), pp. 2513–2518. DOI: 10.1109/ICRA.2014.6907210.

[8] Matthew Dunbabin and Lino Marques. "Robots for environmental monitoring: Significant advancements and applications". In: *IEEE Robotics & Automation Magazine* 19.1 (2012), pp. 24–39.

[9] E Allen Emerson. *Temporal and modal logic*. Elsevier, 1990, pp. 995–1072.

[10] Rafael Fierro and Frank L Lewis. “Control of a nonholomic mobile robot: Backstepping kinematics into dynamics”. In: *Journal of robotic systems* 14.3 (1997), pp. 149–163.

[11] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. “A survey of socially interactive robots”. In: *Robotics and autonomous systems* 42.3-4 (2003), pp. 143–166.

[12] Paul Gastin and D Oddoux. *LTL 2 BA: fast translation from LTL formulae to Büchi automata*. 2001.

[13] Meng Guo, Sofie Andersson, and Dimos V Dimarogonas. “Human-in-the-loop mixed-initiative control under temporal tasks”. In: (2018), pp. 6395–6400.

[14] Yong Koo Hwang, Narendra Ahuja, et al. “A potential field approach to path planning.” In: *IEEE transactions on robotics and automation* 8.1 (1992), pp. 23–32.

[15] Kiattisin Kanjanawanishkul. “Omnidirectional wheeled mobile robots: wheel types and practical applications”. In: *International Journal of Advanced Mechatronic Systems* 6.6 (2015), pp. 289–302.

[16] Marius Kloetzer and Calin Belta. “A fully automated framework for control of linear systems from temporal logic specifications”. In: *IEEE Transactions on Automatic Control* 53.1 (2008), pp. 287–297.

[17] James J Kuffner and Steven M LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: 2 (2000), pp. 995–1001.

[18] Joseph P La Salle. *The stability of dynamical systems*. SIAM, 1976.

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

[20] Anthony N Michel, Ling Hou, and Derong Liu. *Stability of dynamical systems*. Springer, 2008.

[21] Iram Noreen, Amna Khan, and Zulfiqar Habib. “A comparison of RRT, RRT* and RRT*-smart path planning algorithms”. In: *International Journal of Computer Science and Network Security (IJCSNS)* 16.10 (2016), p. 20.

[22] Dimos Dimarogonas Pian Yu. “Distributed motion coordination for multirobot systems under LTL specifications”. In: *IEEE Transactions on Robotics* 38.2 (2021), pp. 1047–1062.

[23] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2017.

[24] HEBI Robotics. URL: `https://www.hebirobotics.com` (visited on 08/22/2022).

[25] Stephen L Smith et al. “Optimal path planning for surveillance with temporal-logic constraints”. In: *The International Journal of Robotics Research* 30.14 (2011), pp. 1695–1708.

[26] Ehsan Taheri, Mohammad Hossein Ferdowsi, and Mohammad Danesh. “Fuzzy greedy RRT path planning algorithm in a complex configuration space”. In: *International Journal of Control, Automation and Systems* 16.6 (2018), pp. 3026–3035.

[27] Herbert G Tanner, Savvas G Loizou, and Kostas J Kyriakopoulos. “Nonholonomic navigation and control of cooperating mobile manipulators”. In: *IEEE Transactions on robotics and automation* 19.1 (2003), pp. 53–64.

[28] Cristian Ioan Vasile and Calin Belta. “Reactive sampling-based temporal logic path planning”. In: (2014), pp. 4310–4315. DOI: `10.1109/ICRA.2014.6907486`.

[29] Vojtch Vonásek et al. “RRT-path–a guided rapidly exploring random tree”. In: (2009), pp. 307–316.

[30] Huijuan Wang, Yuan Yu, and Quanbo Yuan. “Application of Dijkstra algorithm in robot path-planning”. In: *2011 second international conference on mechanic automation and control engineering*. IEEE. 2011, pp. 1067–1069.

[31] Robert L Williams et al. “Dynamic model with slip for wheeled omnidirectional robots”. In: *IEEE transactions on Robotics and Automation* 18.3 (2002), pp. 285–293.