University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Doctor of Industrial Innovation Engineering

XXXV cycle

Safety-Aware Control Techniques for Advanced Human-Robot Collaboration

---

Author:
Andrea Pupa

Supervisor:
Cristian Secchi

PhD Coordinator:
Franco Zambonelli

October 2022

Reggio Emilia, Italy, October 2022

To my family,

To my friends,

And anyone who

believed in me,

especially when

I did not.

Acknowledgements

"Since we all go to the same place in the end, the moments we spent with each other are the only things that do matter. The times we helped each other."

— Brandon Sanderson, *Rhythm of War*

It has been a long journey and now, having reached the end, I can only look back with a smile.

I started my PhD with only one awareness in my life: "I am not up to it". Today, after three years, my mind has changed: "Thanks to all those who are close to me, I can do it". They have been fantastic years, in which I have gained a greater awareness of what I have studied and in which I have also realised that I knew much less than I thought I did. All this would never have been possible without having had at my side fantastic people who supported me and with whom I was able to share joys and sorrows. For this reason, I can do nothing but thank them.

First of all, I thank my family, for having always believed in me unconditionally. Especially when I wanted to drop out of university because I couldn't get going. My parents, my sister and my brother were a pivotal point and creators of immense sacrifices, able to give me strength in the darkest moments.

I thank my lifelong friends, Alessandro, Andrea, Gioele, Giovanni, Marco and Vincenzo, for always being there. They never turned their backs on me, even when I could not be fully present because I had to study and had to sacrifice many moments with them. Among them a special thanks goes to Alessandro and Gioele, you helped me out of an abyss of sadness and unhappiness into which I was sinking. Words are not enough and I think I do not deserve friends like you.

I thank the whole ARSControl team for making me live the job with joy and lightness. My tutor, Cristian, demonstrates how doing a job you are passionate about leads you to never work, because it is literally what you would do in front of the fireplace during the weekend. I also owe a lot to Lorenzo and Valeria, for listening to me and helping me

make a decision for my future. Many thanks also to Federico, Federico and Marco, who are more friends than colleagues. Jokes, laughter and discussions about robotics form the basis of our conversations and have allowed us to share fantastic moments. I do not know how the future will go, but I very much hope to work with you again.

I also thank Andrea, who came into my life after all these people, but managed to leave an indelible mark. A person with an empathy outside the norm, always able to understand my state of mind and find the perfect words.

I thank Francesca, in just over a year she has turned my life upside down, bringing out what I did not think I had. "You have made a place in my heart where I thought there was no room for anything else. You have made flowers grow where I cultivated dust and stones".

I would like to thank Wietse for helping me during the visiting period. It was not easy also because of COVID and the sudden lockdown, but he did everything not to make me feel lonely.

Finally, I thank myself, for having had the strength to ask for help when it was time, for being able to take advice and for never giving up.

Ringraziamenti

> *"Dato che andiamo tutti nello stesso posto, alla fine, i momenti che passiamo insieme sono le uniche cose che hanno importanza. I momenti in cui ci siamo aiutati."*
>
> — Brandon Sanderson, *Il Ritmo della Guerra*

È stato un lungo percorso e ora, giunto alla fine, non posso fare altro che guardare indietro con un sorriso.
Ho iniziato il dottorato con una sola consapevolezza nella mia vita: "Non sono all'altezza". Oggi, dopo tre anni, la mia idea è cambiata: "Grazie a tutti coloro che mi sono vicini, posso farcela". Sono stati anni fantastici, in cui ho maturato una maggiore consapevolezza di ciò che ho studiato ed in cui ho anche compreso di sapere molto meno di quello che pensavo di conoscere. Tutto ciò, non sarebbe mai stato possibile senza aver avuto al mio fianco persone fantastiche che mi hanno supportato e con cui ho potuto condividere gioie e dolori. Per questo motivo, non posso fare altro che ringraziarli.
In primis ringrazio la mia famiglia, per aver sempre creduto in me in maniera incondizionata. Soprattutto quando volevo mollare gli studi perché non riuscivo ad ingranare. I miei genitori, mia sorella e mio fratello sono stati un punto cardine e artefici di sacrifici immensi, in grado di darmi la forza nei momenti più bui.
Ringrazio gli amici di una vita, Alessandro, Andrea, Gioele, Giovanni, Marco e Vincenzo, per esserci sempre stati. Non mi hanno mai voltato le spalle, anche quando non riuscivo ad essere pienamente presente perché dovevo studiare e ho dovuto sacrificare molti momenti con loro. Tra questi un grazie speciale va ad Alessandro e Gioele, mi avete aiutato ad uscire da un abisso di tristezza ed infelictà nel quale stavo sprofondando. Le parole non sono abbastanza e penso di non meritare degli amici come voi.
Ringrazio tutto il team dell'ARSControl per avermi fatto vivere il lavoro con gioia e leggerezza. Il mio tutor, Cristian, è la dimostrazione di come fare un lavoro per cui si ha passione ti porti a non lavorare mai, perché è letteralmente ciò che faresti davanti al

camino durante il weekend. Devo molto anche a Lorenzo e Valeria, per avermi ascoltato ed aiutato a prendere una decisione per il mio futuro. Un grazie infinito anche a Federico, Federico e Marco, che sono più amici che colleghi. Battute, risate e discussioni sulla robotica sono alla base delle nostre conversazioni e ci hanno permesso di condividere momenti fantastici. Non so come andrà il futuro, ma spero vivamente di lavorare ancora con voi.

Ringrazio poi Andrea, entrato nella mia vita dopo tutte queste persone, ma che è riuscito a lasciare un segno indelebile. Una persona con un'empatia fuori dalla norma, in grado sempre di capire il mio stato d'animo e trovare le parole perfette.

Ringrazio Francesca, in poco più di un anno ha stravolto la mia vita tirando fuori ciò che non pensavo di avere. "Ti sei creata un posto nel mio cuore dove credevo non ci fosse spazio per altro. Hai fatto crescere fiori dove coltivavo polvere e pietre".

Ringrazio Wietse, per avermi aiutato durante il mio periodo ell'estero. Non è stato facile anche a causa del COVID e dell'improvviso lockdown, ma ha fatto di tutto per non farmi sentire solo.

Infine ringrazio me stesso, per aver avuto la forza di chiedere aiuto quando era il momento, per essere riuscito ad accettare i consigli e per non aver mai mollato.

Abstract

The development and improvement of new technologies has led to a raise in the use of automated solutions inside industrial settings. Indeed, in recent years an exponential increase in the use of collaborative robotics has been observed. Collaborative robotics allow the automation of industrial process in shared environments, where humans and robots work together. Some of the reasons behind this large diffusion are due to the flexibility and versatility of collaborative manipulators. Thanks to their characteristics, they can be used without the need to install physical barriers, making them suitable for small and medium-sized enterprises. Large companies, on the other hand, still tend to automate their processes with traditional robotic solutions, despite their greater complexity and installation costs. The main reason behind this choice is related to the fact that the performance of collaborative robotics is not yet comparable to that of traditional robotics. This is because, to date, collaborative robotics applications do not fully exploit all the potential offered by this new technology. In particular, there are two key aspects on which it is important to focus: task allocation and safety. In fact, to improve the performances, it is necessary to define an approach that makes the most of the differences between operator and robot creating a synergy between these two actors. Through collaboration, in fact, the human-robot team can perform tasks that neither the operator nor the robot would be able to perform individually. Concerning the safety, on the other hand, the close contact between operator and robot and the absence of physical barriers required to pay a lot of attention to the definition of the methodologies on how to ensure safety. Therefore, regulations have been updated in order to formalise the different degrees of collaboration and how to assess safety in each collaborative mode. The mere application of these regulations, however, results in a very conservative approach: safety is treated as a barrier in front of which the robot can only slow down until stop. It would be better, indeed, to manage safety not as a constraint but as a variable in the production process. In this way, it would always be possible to have optimal collaboration while ensuring safety. The work in this thesis aims to address these two issues in order to change the paradigm of collaborative robotics. In this way, it will no longer be necessary to make a choice be-

tween performance and safety, but it will be possible to have both characteristics, bringing collaborative robotics ever closer to traditional robotic solutions. Initially, the context of the work of this thesis will be presented and the European project "RObot enhanced SenSing, INtelligence and actuation to Improve job quality in manufacturing" (ROSSINI), of which this work is a part, will be detailed. Subsequently, the two problems in question will be addressed. A first part will focus on the task allocation and scheduling, showing a framework that optimise collaboration and adapt it to the needs of the production process. The following part will focus on trajectory planning, showing solutions that can dynamically manage both safety and robot behaviour. Subsequently, the final result of the ROSSINI project will be presented, i.e. a modular and flexible architecture capable of increasing the performance of collaborative robotics without violating safety regulations. The architecture was implemented and validated on case studies proposed by the project partners, demonstrating how the proposed architecture is so general that it can be used in very different applications and hardware independent. Finally, some extensions of the architecture implemented within the ROSSINI project framework will be presented. These extensions are intended both to improve previously validated solutions and to generate new strategies to support them, making them more complete.

Sommario

Lo sviluppo ed il miglioramento delle nuove tecnologie ha portato ad un incremento dell'impiego di soluzioni automatizzate all'interno degli ambienti industriali. Negli ultimi anni, infatti, si è assistito ad un aumento esponenziale dell'impiego della robotica collaborativa, che permette di automatizzare processi di lavorazione in ambienti di lavoro condivisi con un operatore umano. Alcuni dei motivi dietro a questa ampia diffusione sono dovuti alla flessibilità e alla versatilità dei manipolatori collaborativi. Grazie alle loro caratteristiche, infatti, essi possono essere utilizzati senza la necessità di installare barriere fisiche. Questo li rende quindi adatti alle piccole e medie imprese. Le grandi imprese, invece, tendono ad automatizzare i propri processi con soluzioni robotiche tradizionali, nonostante la loro maggiore complessità e costi di installazione. Il principale motivo dietro a questa scelta è legato al fatto che le prestazioni della robotica collaborativa non sono ancora paragonabili a quelle della robotica tradizionale. Questo perché, ad oggi, le applicazioni di robotica collaborativa non sfruttano appieno tutte le potenzialità offerte da questa nuova tecnologia. In particolare, ci sono due aspetti chiave su cui è importante focalizzare l'attenzione: suddivisione dei compiti e sicurezza. Al fine di migliorare le prestazioni, infatti, è necessario definire un approccio che permetta di sfruttare al meglio le differenze tra operatore e robot creando una sinergia tra questi due agenti. Grazie alla collaborazione, infatti, il team uomo-robot può eseguire compiti che né l'operatore né il robot sarebbero in grado di svolgere singolarmente. Per quanto riguarda la sicurezza, invece, lo stretto contatto tra operatore e robot e l'assenza di barriere fisiche ha richiesto di porre molta attenzione su questo aspetto andando che ha portato alla definizione di alcune metodologie su come garantire la sicurezza. Pertanto, le normative sono state aggiornate al fine di andare a formalizzare i differenti gradi di collaborazione. La mera applicazione di queste normative, però, si traduce in un approccio molto conservativo: la sicurezza viene trattata come una barriera di fronte alla quale il robot può solo rallentare fino a fermarsi. Sarebbe meglio, infatti, gestire la sicurezza non come un vincolo ma come una variabile del processo produttivo. Così facendo, sarebbe sempre possibile avere una collaborazione ottima, mantenendo la sicurezza. Il lavoro di questa tesi si

pone come obiettivo quello di affrontare queste due problematiche al fine di modificare il paradigma della robotica collaborativa. In questo modo, non sarà più necessario effettuare una scelta tra prestazioni e sicurezza, ma sarà possibile avere entrambe le caratteristiche, portando la robotica collaborativa sempre più vicina alle soluzioni robotiche tradizionali. Inizialmente, verrà presentato il contesto in cui si colloca il lavoro di questa tesi e verrà dettagliato il progetto europeo "RObot enhanced SenSing, INtelligence and actuation to Improve job quality in manufacturing" (ROSSINI), di cui questo lavoro fa parte. Successivamente verranno affrontati i due problemi in questione. Una prima parte si concentrerà sulla suddivisione dei compiti, mostrando una soluzione in grado di ottimizzare la collaborazione e di adattarla alle esigenze del processo produttivo. La parte seguente, invece, verterà sulla pianificazione delle traiettorie, mostrando un'architettura in grado di gestire dinamicamente sia la sicurezza che la movimentazione. Successivamente, verrà presentato il risultato finale del progetto ROSSINI, ovvero un'architettura modulare e flessibile in grado di incrementare le prestazioni della robotica collaborativa senza violare le normative di sicurezza. L'architettura è stata implementata e validata sui casi di studio proposti dai partner del progetto, dimostrando come l'architettura proposta sia così generale da poter essere impiegata in applicazioni molto diverse tra loro. Infine, verranno presentate alcune estensioni dell'architettura implementata all'interno del framework del progetto ROSSINI. Queste estensioni hanno lo scopo sia di migliorare le soluzioni precedentemente convalidate che di generare nuove strategie per sostenerle, rendendole più complete.

Contents

List of Figures

List of Tables

List of Abbreviations

CBF **C**ontrol **B**arrier **F**unctions.

HD **H**and **G**uiding.

HRC **H**uman **R**obot **C**ollaboration.

I-O SFL **I**nput–**O**utput **S**tate **F**eedback **L**inearization.

MILP **M**ixed **I**nteger **L**inear **P**rogramming.

PFL **P**ower and **F**orce **L**imiting.

ROS **R**obot **O**perating **S**ystem.

ROSSINI **RO**bot enhanced **S**en**S**ing, **IN**telligence and actuation to **I**mprove job quality in manufacturing.

SMS **S**afety-rated **M**onitored **S**top.

SSM **S**peed and **S**eparation **M**onitoring.

TAMP **T**ask **A**llocation and **M**otion **P**lanning.

List of Publications

Journal Papers

- A. Pupa, M. Arrfou, G. Andreoni, and C. Secchi, "A safety-aware kinodynamic architecture for human-robot collaboration," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4465–4471, 2021
- A. Pupa, W. Van Dijk, and C. Secchi, "A human-centered dynamic scheduling architecture for collaborative application," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4736–4743, 2021
- A. Pupa, W. Van Dijk, C. Brekelmans, and C. Secchi, "A resilient and effective task scheduling approach for industrial human-robot collaboration," *Sensors*, vol. 22, no. 13, p. 4901, 2022

Book Chapters

- A. Pupa, M. Arrfou, G. Andreoni, and C. Secchi, "A human-centered dynamic task scheduling and safe task execution approach for human-robot collaboration scenarios," in *Human-Robot Collaboration: Unlocking the potential for industrial applications*, 2023

Conference Papers

- A. Pupa, C. Talignani Landi, M. Bertolani, and C. Secchi, "A dynamic architecture for task assignment and scheduling for collaborative robotic cells," in *Proceedings of the 13th International Workshop on Human-Friendly Robotics*, 2020
- A. Pupa and C. Secchi, "A safety-aware architecture for task scheduling and execution for human-robot collaboration," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1895–1902
- A. Pupa and C. Secchi, "A human-centered dynamic task planning approach for human-robot collaboration," in *2021 I-RIM Conference*. I-RIM, 2021, pp. 41–43
- A. Pupa, F. Breveglieri, and C. Secchi, "An optimal human-based control approach for mobile human-robot collaboration," in *Proceedings of the 15th International Workshop on Human-Friendly Robotics*, 2022
- A. Pupa, W. Van Dijk, and C. Secchi, "Towards an effective human-robot team

collaboration – a resilient task scheduling approach for real industrial scenarios," in *2022 I-RIM Conference*. I-RIM, 2022

- A. Pupa, M. Minelli, and C. Secchi, "A dynamic planner for safe and predictable human robot collaboration," in *Submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023

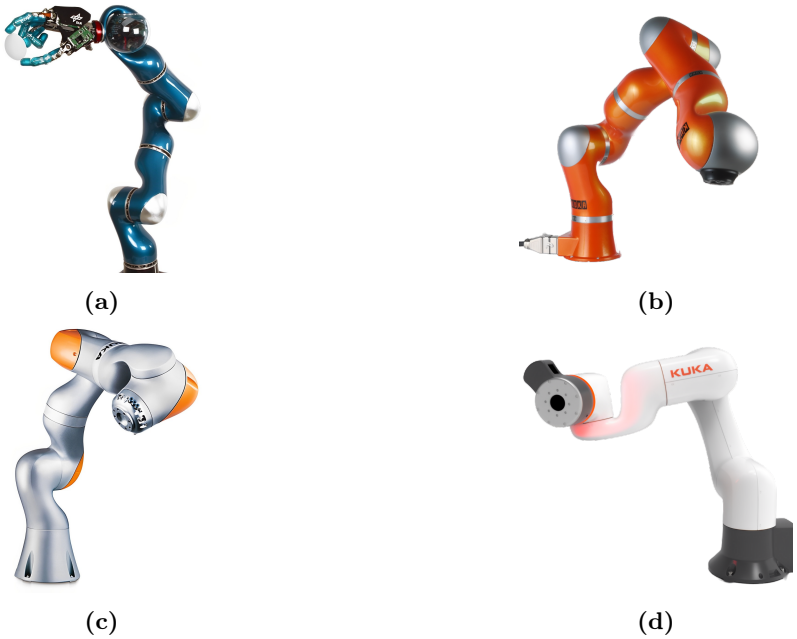
Chapter 1

Introduction

1.1 Collaborative Robotics

The term collaborative robot or **cobot** refers to a robot created to work closely with the human operator, sharing the work space. The first definition of a cobot is found in a 1997 U.S. patent [11] which reads, "An apparatus and method for direct physical interaction between a person and a general purpose manipulator controlled by a computer." The owners of the patent are James Edward Colgate and Michael Peshkin, two Northwestern University professors who are now considered the inventors of collaborative robots. The patent is the result of a research project funded by the General Motors Foundation in 1995. The goal of the project was to improve the safety of robots to enable them to collaborate with human operators. To achieve this, early cobots were designed without an internal source of motive power, which was provided directly by the human operator. The main function of these first cobot versions was to enable motion control, while compensating for a load, in a cooperative manner with the human operator. Only later, cobots were provided with their own internal motive power, ensuring safety with other intrinsic characteristics. Indeed, as professor Colgate declared [12], "limits on speed and power and good interactive design helped cobots remain safe."

In the last decades, the market of collaborative robots has grown rapidly and several cobots have been placed over the market. In 2004 KUKA, in collaboration with the German Aerospace Center institute, launched their first collaborative robot, called KUKA LBR III [13]. This robot has then been improved and it has been replaced with the KUKA LBR 4 in 2008 and the KUKA LBR iiwa in 2013. In 2022, KUKA launched a new line of cobots called KUKA LBR iisy, with the goal of having a collaborative robot that can be used quickly and productively without any prior knowledge [14]. All these KUKA models are illustrated in order in Figure 1.1. Universal Robots, which was the market leader in 2019 [15], released their first cobot in 2008, the UR5 [16]. Subsequently,



(a) **(b)** **(c)** **(d)**

Figure 1.1: Kuka models.

they enriched their catalogue with the UR10 in 2012, UR3 in 2015 and UR20 in 2022 [17]. Other important companies are FANUC and ABB, which both entered the market in 2015 with the FANUC CR-35iA [18] and the ABB Yumi [19]. From the research point of view, a great role is played by the Franka Emika Panda, collaborative robot developed by the German company Franka Emika in 2018 [20].

The main differences between traditional industrial robots and collaborative robots can be summarized as follows:

- Fast Setup.
- Easy Programmability.
- Versatility and Functionality.
- Increased Safety.

The advantages derived from collaborative robotics, such as ease of installation and its inherent safety, have led to its increasing diffusion. In fact, in 2022, the collaborative robot market is valued at USD 1.1 Billion and, over the forecast period from 2022 to 2028, it is projected to grow to USD 9.2 Billion at a Compounded Average Growth Rate of 41.5 per cent [21]. However, collaborative robots also have several disadvantages compared to traditional industrial robotics, the first of which is efficiency.

The invention and diffusion of collaborative robots has also led to the emergence of new related disciplines. In particular, Human–Robot Collaboration (HRC) is the study of collaborative processes in which human operators and robots work together to achieve common goals. The theories behind HRC derive from the study of human–human collaboration. This is because the aim of the HRC is to have a robot whose behaviour tends to emulate that of the human operator as much as possible, generating an efficient and natural collaboration. The possible approaches to HRC are [22]:

- Human Emulation.
- Human Complementary.

The goal of the human emulation approach is to make robots act like human operators or have human-like abilities as much as possible. It is mainly based on the development of formal models of human–human collaboration and their application in the case of human–robot collaboration. Humans are regarded as rational agents capable of forming and executing plans to achieve their goals and are also able to sense and understand the intentions of others. Given this ability, collaboration consists of agents helping each other to achieve their goals. The aim of the human complementary approach is to make the robot a more intelligent partner that collaborates in a complementary manner with humans. This is due in order to improve the overall human–robot interaction. The approach is mainly based on the fact that human operators and robots are different actors with complementary skills. Thus, many interaction and collaboration paradigms have been introduced with the scope of dividing the plans between the two actors in order to exploit their strengths and overcome their individual weaknesses.

Nowadays, one of the main challenges of HRC is to overcome the main disadvantage of collaborative robots, namely efficiency, while ensuring safety without physical barriers. To achieve this, the human complementary approach can be exploited, creating a synergy between the human operators and the robot such that the human-robot team performs better than the traditional solution, where the human and the robot work separately

Figure 1.2: Rossini Logo.

without interacting.

The results of this thesis are obtained addressing some of the challenging problems presented in the latter, and arising during the European funded project RObot enhanced SenSing, INtelligence and actuation to Improve job quality in manufacturing (ROSSINI), where the goal is to increase the performances of the HRC solutions giving an optimal trade-off between traditional and collaborative robots.

1.2 ROSSINI

The ROSSINI, whose logo is shown in Figure 1.2 project aims to design, develop and demonstrate a scalable and modular platform for the integration of human-centred robotic technologies in complex industrial settings. This can be achieved with two different procedures. The first one regards the development of innovative methodologies and technological components, e.g. sensing, control, risk assessment methodology. The second step, instead, is the integration of all these components in an open platform. The ultimate goal of the ROSSINI project is to make the HRC an attractive and valuable choice for all those manufacturers who have not implemented it so far, due to its more limited performance compared to traditional robotics solutions or regulatory limitations. Indeed, with a well-integrated framework it is possible to increase the reaction speed, intelligence, sensing and communication capabilities of cobots. Thus increasing the performance of the HRC, both in terms of payload and in terms of working speed and productivity, and achieving an excellent trade-off between traditional and collaborative robotics solutions.

To achieve this goal, the consortium agreed on a series of objectives which were pursued and fully satisfied during the project development. These objectives are briefly detailed below.

Objective 1: Designing a Smart and Safe Sensing System with improved detection and tracking capabilities, and a safety-graded fusion module for the processing of data
The ROSSINI platform includes sensing technologies that increase the performances of the currently available equipment, which plays a great role for close HRC. This goal

is achieved by the ROSSINI Smart and Safe Sensing System (RS4), which combines information coming from different customised sensing technologies, such as cameras, laser scanners, in order to track both the position and the speed of the human operators and the objects inside the working scenarios. The output of the RS4 is a single multidimensional image processed in real time to generate suitable 3D safety regions, called "Dynamic Safety Shells", around each person and object in the environment.

Objective 2 - To develop a Safety Aware Control Architecture for robot cognitive perception and optimal task planning and execution

In order to enable also the perception of the working environment, the project exploits artificial intelligence techniques to build a semantic scene map that adapts to dynamic working conditions. This improves the data coming from the RS4, by adding also semantic information to the geometric data. Moreover, the artificial intelligence techniques are also exploited to give cognitive capabilities to the robot. In this way, the ROSSINI Safety Aware Control Architecture is capable of optimally scheduling the tasks among the robots and the human operators, optimising the trade-off between safety and manufacturing productivity. Each robot action is forwarded to a dynamic planner that dynamically optimize its execution considering both the path that the robot has to follow and the interactive behaviour that the robot must reproduce while considering the safety conditions in the working scenario.

Objective 3 - To develop a "Collaborative by Birth" Robotic Arm range with novel built-in safety features

The standard collaborative robots available in the market are mainly standard manipulators on top of which specific technologies have been added in order to reduce risks for humans during the execution of collaborative operation. Thus, a great improvement in the HRC can be achieved designing a new robotic technology from scratch, considering safe interaction with humans as the main guiding principle at each design step. In the ROSSINI project a new robotic manipulator have been developed and exploited as the platform actuation layer.

Objective 4 – To develop a framework for Human-Robot Mutual Understanding in collaborative operations

In the project, the human factors like job quality, user experience, trust, feeling of safety, and liability, are incorporated in a proper human-centered design process. In this way, all the desired factors are considered directly from the early design phase and not afterwards. Furthermore, by monitoring the progression of the collaboration, such as task progression or human emotions, it is possible to change online the original task planning exploiting a proper dynamic allocation strategy. Lastly, a proper communication level allows to enable a mutual understanding between robots and people making improving the predictability of the collaboration.

Objective 5 – To integrate all the ROSSINI technological components into one inherently safe platform for HRC applications development

Lastly, all the technologies developed during the project are made available to both integrators and manufacturers, which are the end users, in order to simplify the deployment of the HRC inside the industrial settings. Thus, all the components have been adequately

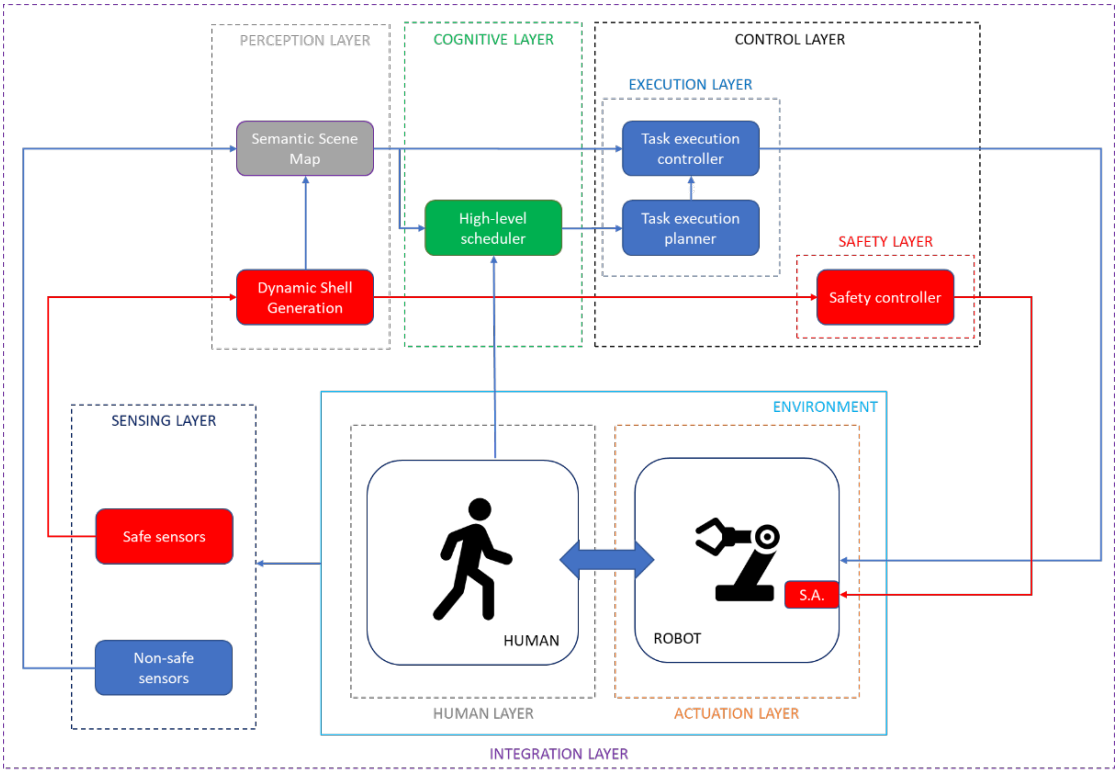


Figure 1.3: The overall Rossini Architecture.

interfaced among each others. Starting from this, the ROSSINI Integration and Validation Layer provides both a design tool, which is used to easily configure all the platform modules, and a set of new methodologies for risk assessment and validation, which are exploited to speed up the validation and increase the efficiency of HRC applications. With this layer, the ROSSINI platform ensures an intrinsic safety to all the application developed through it, i.e. it provides safety compliant design guidelines and an optimised set of components and methodologies that ensures safety.

1.2.1 ROSSINI Architecture

To pursuit and achieve the objectives the overall ROSSINI platform, represented in Figure 1.3, has been developed by all the partners. In particular, the platform is the integration of a set of layers, each of which is dedicated to a specific function:

- The **Sensing Layer**, which aims at combining all the information from both safe and non-safe sensors in a fusion module to provide a unique and complete flow of data to the Safety Aware Control Architecture.
- The **Perception Layer**, which exploits artificial intelligence techniques to generate a Semantic Scene Map that integrates both geometric and semantic information. Furthermore, the Semantic Scene Map creates a set of virtual "Dynamic Shells" surrounding each object in the scene that can be exploited to implement safe control actions.
- The **Cognitive Layer**, which aims at providing a high-level scheduler in order to both dynamically plans a set of cooperative tasks that the robot and the human operator should execute and to adapt them online when a change in the working

conditions is detected by the Semantic Scene Map.

- The **Control Layer**, also called **Flexible Execution and Safety Layers**, whose role is to convert the high-level action into the most efficient and safety compliant low-level action for the robot. Thus, the Control Layer aims at optimizing the trade-off between safety and productivity for the HRC environment.
- The **Actuation Layer**, which aims at increasing the degree of freedom for robotic applications design by introducing a novel concept of manipulators with built-in safety features that allow to reduce the separation distance between the human operator and the robot during the collaboration.
- The **Human Layer**, whose role is to both include the human-related factors defined in the early design phases of collaborative applications design and to constantly monitor the factors that affects the job quality.
- The **Integration Layer**, which provides manufacturers with the tools and guidelines to ensure intrinsic safety in the design of HRC applications and to accelerate the configuration and reconfiguration of the application.

Cognitive and control layers are the core of the overall architecture. Indeed, providing cognitive capabilities to the robot it is possible to create a synergy between the human operator and the robot, obtaining a more natural collaboration. The control layer, on the other hand, implements a collision-free safe dynamic behavior, giving flexibility to the robot within the working area. The integration of these layers, therefore, allows to obtain an intelligent and adaptive behavior, leading the robot to adapt to the operator and not the other way around as usually happens. UNIMORE was the leader of the development of these two layers and of their integration with also the perception and the sensing layer, which are the main topics of this thesis work.

1.2.2 Consortium

In addition to UNIMORE, responsible of the cognitive layer and the control layer, the ROSSINI consortium is composed of multiple European partners, illustrated in Figure 1.4 and listed below [23]:

- **Datalogic**, which is a global leader in the automatic data capture and process automation markets, specialized in the designing and production of bar code readers, mobile computers, sensors for detection, measurement and safety, RFID vision and laser marking systems. Its role in the project is manly focused in the technical development of bth the sensing and safety layers and in coordination activities.
- **Datasensing**, which is developing, manufacturing, and supplying Machine Vision, Sensor and Safety. Its payoff is "easing automation challenges", thanks to solutions focused on industrial automation, from production to internal logistics of goods. Datasensing has a global presence, with HQ in Italy and main offices in Europe, China and US, with over 350 employees worldwide. Its role in the project is focused on the development of the sensing and safety layers.
- **Whirlpool**, which is the world's leading major home appliance company. The company markets Whirlpool, KitchenAid, Maytag, Consul, Brastemp, Amana, Bauknecht, Jenn-Air, Indesit, Hotpoint and other major brand names in nearly every country throughout the world. It provided one of the use cases of the project.

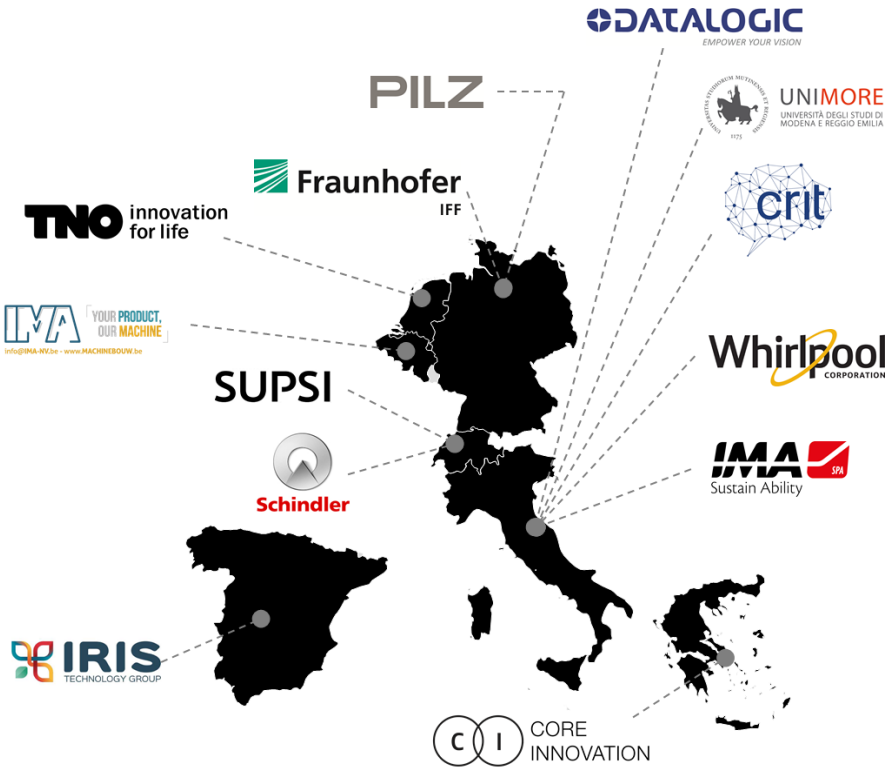


Figure 1.4: The Rossini Consortium.

- **SUPSI**, which is one of the 7 UAS in Switzerland and has a university statute focused on applied research. Key research areas are: Automation and Control, Mechatronic, Robotic, Artificial Vision Systems. In the project, it is mainly responsible of the actuation layer.
- **Pilz Group**, which is a global supplier of products, systems and services for automation technology. Based in Ostfildern, Pilz supplies safe solutions for people, machinery and the environment. Together with Datalogic and Datasensing, in the project is responsible of developing safety technologies.
- **IRIS**, which delivers R&D projects in the area of Process Analytical Technology, Photonics, IoT and ICT. IRIS specializes in the manufacture and integration of real-time in-process monitoring solutions applied to the process, food, pharma, and manufacturing industries. IRIS widely applies Artificial Intelligence and Data Mining techniques for predictive and active process control, as well as for decision support systems. In the project, IRIS is responsible of the perception layer, developing the semantic scene map.
- **VINTIV**, which is specialised in providing the right industrial technology, developing and building quality machines, and offering technological services. VINTIV aims to be the first choice for converting business cases into the right technical solutions. The main role of VINTIV in the project regards the activity of system integrator and development of a new custom gripper.
- **IMA**, which is world leader in the design and manufacture of automatic machines for the processing and packaging of pharmaceuticals, cosmetics, food, tea and coffee. In the project, it provides one of the use cases.
- **Fraunhofer IFF**, which is an institute of the non-profit Fraunhofer-Gesellschaft, a

provider of contract-applied research. It specializes in research and development in the fields of robotics, sensor systems, virtual engineering, logistics and process and plant management. In the project, it is responsible of developing a robust human module that can be exploited for the physical human–robot interaction.

- **CRIT**, which is a private company specialized in the research and analysis of technical and scientific information and in the development of research project activities. Its role is mainly focused on coordination activities.
- **Schindler**, which is one of the world's leading providers of elevators, escalators, and moving walks. It provides one of the use cases.
- **TNO**, which aims at connecting people and knowledge to create innovations that boost the competitive strength of industry and the well-being of society in a sustainable way. It is responsible of analyzing the human-factors for evaluating the job quality.
- **Core**, which is a start-up company offering tailor-made solutions to attach meaning to data in several industrial sectors and use cases using Artificial Intelligence technologies. It is responsible of the dissemination activity of the project.

1.3 Contribution and Thesis Outline

This thesis focuses on the development of novel control strategies to have an efficient and natural human–robot collaboration. In particular, starting from the architecture illustrated in Figure 1.3, this thesis mainly focuses on the development and the validation of both the Cognitive Layer and the Flexible Execution and Safety Layers. The integration of these two architectures together with both the perception layer and the sensing layer gives the Safety Aware Control Architecture, which is validated in both experimental scenarios and the the ROSSINI use cases. Furthermore, this thesis also addresses some research activities born following the development of the ROSSINI project, with the aim of improving the developed technologies.

The main contribution of this thesis are:

- A novel adaptive framework for task assignment and scheduling that takes into account real execution time, the job quality of the human, and the communication with human and robot for dynamic rescheduling
- A novel adaptive framework for trajectory planning and scaling that takes into account the high dynamicity of the environment, adapting in real-time the trajectory.
- A modular and integrated framework for addressing the Task Assignment and Motion Planning (TAMP) problem and the experimental validation in real and complex industrial scenarios .
- A novel adaptive task scheduling framework that is effective and applicable to most of the scenarios that could occur in a real HRC.
- A framework for trajectory planning that considers the high dynamism of the environment and the human preferences to generate collision free-trajectory in a HRC scenario.
- A control architecture that allows the human operator to lead the collaboration,

forcing the robot to stay close, while ensuring safety.
The rest of the thesis is organized as follows.

Chapter 2 reports the development of the Cognitive Layer for the ROSSINI project. Problems related to the optimal task allocation and scheduling and online rescheduling for complex HRC scenarios are addressed. The tasks are firstly assigned exploiting both the intrinsic capability of each agent and trying to maximize the parallelism, i.e. reducing the waiting times. Subsequently, the tasks are dynamically scheduled and rescheduled taking into account the real execution times of the human operator. Finally, the proposed architecture is validated in a custom collaborative scenario.

Chapter 3, instead, reports the development of the Flexible Execution and Safety Layers. Problems related to the safe trajectory planning and replanning and the online safety aware control for HRC scenarios are addressed. Initially, the proposed framework plans a trajectory that the robot could ideally execute at maximum speed, i.e. without considering the safety of the human operator. Subsequently, a scaling layer reduce online the magnitude of the velocity in order to be compliant with the safety standards. These two procedures are also mutually integrate to improve the performances. Lastly, the validation of the framework is presented.

Chapter 4 reports the development of the Safety Aware Control Architecture the ROSSINI project. This chapter focuses on the integration of all the layers. Firstly, it presents the integration of the Cognitive Layer and the Flexible Execution and Safety Layers. Subsequently, it reports a brief explanation and the integration of both the Perception Layer and of the RS4 system. Finally, the chapter focuses on extensive experimental validation in both custom environments and project use cases, demonstrating how all architecture can be applied to different scenarios.

Chapter 5 presents a resilient and effective task scheduling framework that improves the results obtained with the cognitive layer. Firstly, how an industrial HRC process can be formalized into a set of interdependent tasks is presented. Secondly, the chapter details the proposed framework, which leverages a database to understand how the collaborative job is composed. Moreover, at runtime, the framework monitors the task execution to understand the human operator skills and the task result to adapt online the schedule. Finally, the the experimental validation is reported.

Chapter 6 reports two architectures that follow the work on the flexible execution and safety layers. The first one is a natural extension of the architecture validated in the ROSSINI project. It improves the planning strategy by adapting it to the characteristics of the human operator in order to make the robot's behavior more predictable. The second architecture allows the human operator to directly control the collaboration, while maintaining safety. Since it uses a nominal trajectory, it can be seen as a support architecture for the one developed in the ROSSINI project. Both architectures are experimentally validated.

Chapter 7 reports the conclusions of the entire work of thesis and the future direction on which researches could be driven

Chapter 2

Cognitive Layer

This chapter reports the development of the Cognitive Layer for the ROSSINI project. Problems related to the optimal task allocation and the task scheduling for complex HRC scenarios are addressed. To this aim, the overall architecture leverages two main aspects: the online human monitoring, to estimate the real execution time of the operator, and the definition of the job quality metrics, that allow to quantitatively evaluate the the job quality aspects. The overall architecture is experimentally validated on a custom assembly scenario.

The work presented in this chapter is published in [2, 5].

2.1 Introduction

In recent years, industrial setting has been supported by a constant increase in the use of collaborative robotics [24].

The shift towards collaborative robotics can significantly change the quality of the job for the human. In fact, collaborative robots can take over dull, heavy or dangerous tasks making the life of the human easier. To ensure that the re-distribution of tasks is favorable to the human, the job quality aspect has to be taken into account when dividing tasks between human and robot [25]. This process can be guided by using job-quality metrics. The job quality framework, as used by the OECD [26], is a multidimensional concept that covers various topics. The part of job quality that concerns the quality of the working environment, e.g. time pressure, physical risks and work autonomy, is of specific interest for human–robot collaboration. At an even lower level, aspects of task load are governed by guidelines and regulations, for example for lifting loads [27] and noise exposure [28].

In the industrial scenarios, collaborative cells are built in order to enable the human and the robot to work together on various jobs, each of which composed by a set of tasks. The distribution of task determines the load to which the human is subjected [25, 29]. Additionally, the distribution of tasks determines influences the fluency, which in turn is strongly related to job quality aspects [30].

A lot of research has been done in the multi-agent task allocation problem in the industrial cases, see e.g. [31–33]. In general, these solutions cannot be directly applied in a human-robot collaboration application, as they consider the presence of homogeneous agents.

Task allocation for collaborative cells has been modeled as a nonlinear optimization problem, see e.g. [34–37], but the computational complexity of the problem is often high and it does not explicitly allow to take into account variable job-quality parameters. In [38] a two-level feedforward optimization strategy for offline subtask allocation between human and robot is presented. This strategy is integrated with a feedback procedure based on mutual trust to re-allocate the subtasks online. In [39] the authors propose a multi-criteria decision-making framework for task allocation and online re-scheduling which generates a solution that best matches the criteria you want to optimize. The re-scheduling procedure, however, happens only in case of unexpected events, without directly considering the real tasks duration. In [40] a two-level framework for task assignment is presented that dynamically handles task failures in a HRC scenario. Nevertheless, job quality over several jobs is not considered. Task assignment strategies for HRC assume that both the human and the robot require a constant amount of time to perform a task. This assumption may lead to inefficiencies. In reality, the human does not always take the same amount of time to accomplish the same task. Several works that consider task rescheduling with a variable human execution time are available in the literature, see e.g. [41, 42], but the operator and the robot are treated as two separate entities and human-robot interaction and communication is not considered.

In order to make the human–robot collaboration as natural as possible, it is necessary

to improve mutual awareness and communication both at the task execution level and at the task planning and scheduling level. Thus, it is important to have a monitoring strategy that makes the scheduler aware of the real duration of the tasks executed by the human and about the job quality of the human. This strategy allows the scheduler to adapt the assigned tasks improving the job quality and the efficiency of the human-robot collaboration. Moreover, both the human and the robot should be able to communicate through the scheduler in order to improve the collaboration and to make it as natural as possible. The human, due to its expertise and experience, should be able to decide to execute a task that was previously assigned to the robot. While the robot should be able to assign a specific task to the human, if some failure occurs, e.g. a robot tool broke. All these decisions should be handled by the scheduler which re-assigns the tasks accordingly.

This chapter proposes a novel framework for task assignment and scheduling for collaborative cells that is aware about the activity of the human and that allows the human and the robot to take decisions about the tasks they need to execute. This framework intrinsically considers job quality in the scheduling algorithm, in order to improve job quality of human workers. The proposed framework is composed by two layers. The first layer assigns, off-line, the tasks within a job to either the human or the robot by providing a nominal schedule. It considers the actual job quality indexes and the dependencies between the tasks. The second layer, the scheduler layer, reschedules the tasks considering the real execution time of the human operator, if needed, and the decisions taken online both by the human and the robot.

The main contributions of this chapter are:

- A novel adaptive framework for task assignment and scheduling that considers into account real execution time, the job quality of the human, and the communication with human and robot for dynamic rescheduling
- A strategy for dynamic rescheduling that is effective and computationally cheap, i.e. suitable for industrial applications, and that allows human and robot to communicate their needs to the scheduler.

2.2 Problem Statement

A collaborative industrial workspace is characterized by the presence of two different agents, a human operator H and a robot R, that must cooperate during a work shift in order to perform S jobs $(J_1, \ldots, J_S)$. Each job consists out of one or more tasks[1] $(T_{1_j}, \ldots, T_{N_j})$, each of which is characterized by an intrinsic cost w_{Ai} and by a *nominal* execution time t_{ai}, where $a \in A = \{H, R\}$ represents the agent that executes the task i. For ease of notation, in the rest of the chapter the double index is removed and the set of tasks $(T_{1_j}, \ldots, T_{N_j})$ is referred to as $(T_1, \ldots, T_N)$. The real task execution time can differ from the nominal execution time, due to uncertainties in the human behavior. Therefore, the workspace is equipped with a monitoring unit that, for each task T_i assigned to the

[1]The choice of the specific technique for splitting a job into several tasks is out of the scope of this work. Several strategies are available in the literature, see e.g. [40] for assembly tasks.

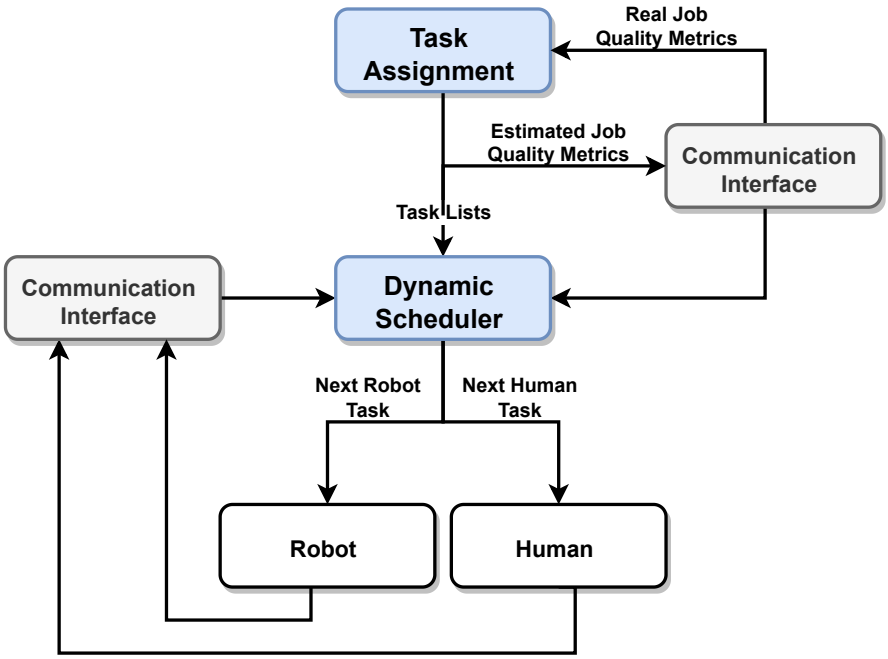


Figure 2.1: The Cognitive Layer architecture.

human, estimates online the real execution time. To achieve this, many solutions can be found in literature, e.g. sequential interval networks [43], interaction probabilistic movement primitives [44], Open-Ended Dynamic Time Warping (OE-DTW) [45]. The output from the monitoring unit is used to update all relevant parameters over the entire work-shift.

The tasks composing a job may depend on each other, i.e. there could be precedence constraints, and the execution order must be considered when assigning tasks.

The aim of this work is to design a task assignment and dynamic scheduling architecture that:

- Builds optimal nominal task schedules for the human and the robot, i.e. two task schedules such that, considering the nominal execution times, the precedence constraints, and job quality metrics, minimizes the job makespan, maximizing the parallelism between human and robot (i.e. minimizes waiting time), and optimizing the job quality for the human operator over the entire work shift.
- Starting from the nominal task schedules, reschedules both the human and the robot tasks according to the real execution time detected by the monitoring unit and the decisions taken by the human and the robot for task swapping. The rescheduling aims at minimizing the job makespan and improving the collaboration between the two agents.

2.3 Architecture

The proposed task assignment and dynamic scheduling strategy, i.e. the Cognitive Layer of the ROSSINI project, is shown in Figure 2.1, where the gray blocks symbolize the strategies implemented to provide richer information to the architecture and the white

blocks represent the two agents. Furthermore, the blue blocks represent the two main layers:

1. **The Task Assignment Layer** is responsible of generating initial nominal schedules for the robot and the human, based on the maximum parallelism criterion, taking into account precedence constraints and job quality metrics over the entire work shift.
2. **The Dynamic Scheduler Layer** is responsible of scheduling the tasks, considering the real execution time and the requests coming from the human and from the robot.

Once the task assignment layer computes the initial nominal schedules, an estimation of the human job quality parameters is performed. This estimation is based on the nominal execution time t_{Hi}.

The resulting estimate represents the expected job quality parameters if the behavior of the human was exactly as the nominal one. To accommodate for (expected) deviations from the nominal schedule, the estimated job quality parameters are given as input to the **Human Monitoring** block, which is responsible for supporting and improving the scheduling procedure.

The human monitoring block aims to track the real execution time of the human operator during the execution of assigned tasks. The information about the real human behavior is then leveraged to update the estimated parameters. Subsequently, the real parameters that come out of the human monitoring block are used as input for the task assignment layer when calculating the nominal schedules of the new job. This feedback procedure allows to keep track of the evolution of job quality throughout the entire work shift, adapting each schedule accordingly.

The real execution time of the human is also exploited by the dynamic scheduler, which aims to reschedule, in real-time, the nominal tasks schedules. Frequently changing the order of the tasks assigned to the human can lead to confusion and poor efficiency of the human [46]. Thus, it has been chosen to focus the rescheduling strategy primarily on robot tasks. The list of tasks assigned to the human changes only when required by the **Communication Interface** block, namely when the robot cannot execute a task and a failure occurs, and when the human decides to perform a task instead of the robot. In all these cases, the changes in the human schedule are necessary and minimal.

2.4 Job Quality Metrics

Job quality is considered in the task assignment via two mechanisms. The first mechanism ensures that various metrics related to job-quality do not exceed threshold values via optimization constraints. The second mechanism considers the overall attractiveness of the task-set that is assigned to the human. The attractiveness of the task-set is made part of the optimization objective.

Various metrics can be defined that describe a certain task load, e.g. the weight that is lifted, or the noise that is experienced during task execution. To monitor these metrics each task is assigned a set of weights $(k_{i1}, \ldots, k_{iM})$. The metrics $(K_1, \ldots, K_M)$ can be calculated using two different generic representation of a job quality metric:

- *summed weight:*

$$K_m = K_{m,0} + \sum_{l=1}^{L} \sum_{i=1}^{N} (x_{Hil} k_{im}) \tag{2.1}$$

- *average weight:*

$$K_{m,av} = \frac{t_e K_{m,0} + \sum_{l=1}^{L} \sum_{i=1}^{N} (x_{Hil} t_{Hi} k_{im})}{t_m + c} \tag{2.2}$$

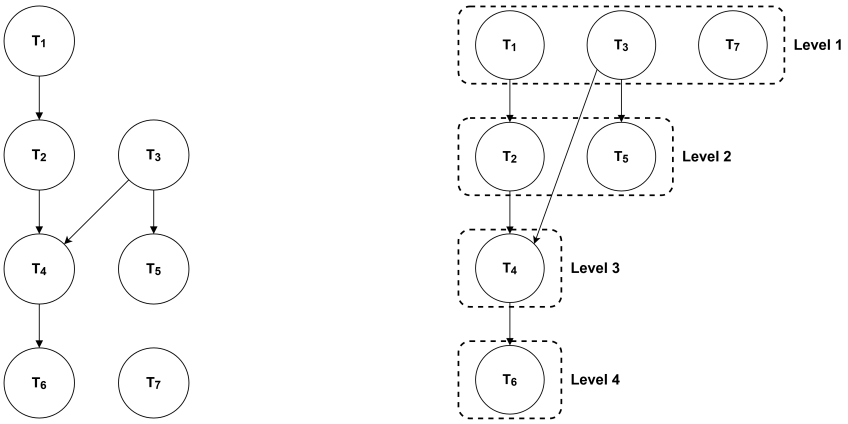
Where $c = \sum_{l=1}^{L} c_l$ is the cycle time and x_{Hil} is a boolean variable that allow to select only the tasks assigned to the human operator, as deeply explained in Section 2.4. Typically, job quality metrics are calculated over a time span longer than the execution of one task, t_m is the elapsed time of the time frame that is relevant for the metric, e.g. elapsed time since the human started the work shift. $K_{m,0}$ is the cumulative costs from previous jobs within the time frame t_e. The set of cumulative costs of each metric represents the job quality metrics that come out from the human monitoring block as shown in Figure 2.1. It is worth noting that this cumulative cost guarantees that all the desired job quality metrics are estimated and constrained over the relevant time frames t_m and not just only over the single schedule.

Ensuring that pre-set thresholds on job-quality aspects are not exceeded does not automatically assign the tasks that are preferred by the human to the human. This aspect is governed by a general attractiveness factor that is assigned to each task. The attractiveness of the tasks is included in the optimization problem (2.3), i.e. in the evaluation of w_{HI}, so the scheduler tries to assign those tasks to the human, that the human likes to perform the most.

2.5 Task Assignment

The role of the task assignment layer is to build, for each job, the nominal task schedules for the human and the robot, taking into account job quality and precedence constraints. This relation of dependency between tasks can be represented with a directed acyclic graph $G = (T, E)$, as shown in Figure 2.2a. Each vertex represents a task T_i while each directed edge E_{ij} means that the task T_i must be executed before the task T_j. Some tasks could be independent of each other, since there is not a path that goes from one task to the other, see e.g. T_1 and T_3. The graph can then be rearranged so that all the parallel tasks are grouped together into several sets called levels L_l, as shown in Figure 2.2b. The choice of how the tasks are assigned to each level has a large impact on the schedules.

In this work, both the problem of allocating the tasks to each agent and the way in which the tasks are distributed over the levels are addressed by solving the following multi-objective Mixed Integer Linear Programming (MILP) problem:



(a) **(b)**

Figure 2.2: Directed acyclic graph of a Job composed by seven tasks and the division into four levels.

$$
\begin{aligned}
&\min_{x,c_l} \quad G = \sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{a\in A} w_{ai}x_{ail} + \frac{1}{t_{max}}\sum_{l=1}^{L} c_l \\
&\text{s.t.} \\
&\qquad \sum_{a\in A}\sum_{l=1}^{L} x_{ail} = 1 && \forall i \in \{1,\dots,N\}, \\
&\qquad \sum_{i=1}^{N} t_{ai}x_{ail} \le c_l && \forall l \in \{1,\dots,L\}, \forall a \in A, \\
&\qquad \sum_{a\in A}\sum_{l=1}^{L} l \cdot x_{ail} < \sum_{a\in A}\sum_{l=1}^{L} l \cdot x_{ajl} && \forall i \to j, \\
&\qquad K_m \le K_{m,max} && \forall m \in \{1,\dots,M\}, \\
&\qquad K_{m,av} \le K_{m,av,max} && \forall m \in \{1,\dots,M\},
\end{aligned}
\tag{2.3}
$$

where $w_{ai} > 0$ represents the cost required by the actor a, i.e. human or robot, to execute T_i. The Boolean variable $x_{ail} \in \{0, 1\}$ determines if T_i is assigned to actor a at the level l. $t_{ai} > 0$ represents the nominal execution time of the actor a in executing T_i, while t_{max} is the maximum of these nominal times. $c_l > 0$, instead, denotes the cycle time of the l^{th} level. K_m and $K_{m,av}$ are quantitative parameters used to evaluate a desired qualitative job quality metric m, where M is the number of analyzed metrics, as detailed in Section 2.4.

w_{ai} is exploited for encoding the cost required by each agent to perform the task, e.g electrical cost, tool wear, or risk assessment. Very high costs communicate to the task assignment algorithm that the agent is unsuitable for the execution of the task. Moreover, the human costs are also exploited to embed and evaluate in a quantitative way the job quality. The calculation method of these costs is a design parameter, see e.g. [47, 48]. In a general way it is possible to define $w_{Ri} = h(costs)$, $w_{Hi} = g(costs, job\ quality)$.

The first constraint guarantees that each task is assigned either to the robot or to the human. The second constraint maximizes the parallelism between the human and the

robot. In fact, since all the terms in the quantity to minimize are positive, the optimization problem would tend to choose $c = \sum_{i=1}^{L} c_l$ as small as possible and the lower bound for this sum is given by the third constraint and corresponds to the maximum parallelization of the activities of the human and of the robot. The third constraint ensures the respect of the precedence relationship, since all the tasks that should be executed before another are assigned to an upper level. The last constraints impose that the job quality metrics for the human operator will not violate the upper bounds.

The outcome of the optimization problem (2.3) are the nominal schedules, i.e. two ordered tuples S_H and S_R containing the tasks that have to be sequentially executed by each agent in each level.

2.6 Dynamic Scheduler

Starting from the output of the task assignment, the goal of the dynamic scheduler is to adapt online the two nominal schedules S_H and S_R taking into account the uncertainty of the human behavior. When two humans collaborate, their natural synergy allows them to reach high team performance. If one human gets slower, the other can compensate by speeding up. Furthermore, more complex unexpected difficulties are handled by communication. Experienced team members can exploit their knowledge to reorganize the work or decide to take over a difficult task. Less experienced team members can ask the expert member for some help when problems occur. The dynamic scheduler aims at reproducing this kind of behavior in human-robot collaboration in order to create an effective and natural cooperation.

This is achieved exploiting two different strategies. Firstly, the human operator is monitored in real time when performing the task in order to estimate the real execution time and, if necessary, to reschedule the future activities of the robot, reducing waiting time. Secondly, the communication between the human and the robot is enabled, allowing the two agents to take decisions about their activities through the scheduler. In particular, the robot delegates a task it cannot momentarily execute to the human. The human, instead, can decide to execute the task that the robot is performing, because, e.g., from its experience, it knows that the robot is not executing the task properly or to speed up the workflow. Moreover, the human can decide to re-assign some of its tasks to the robot. The dynamic scheduler is implemented according to the pseudo-code reported in Algorithm 1. The dynamic scheduler needs as input the nominal task schedules S_H and S_R (Line 1). It firstly sets to false two variables End_R and End_H, which identify if the respective agent has concluded its task, and it initializes the job at the first level (Lines 2-3). Subsequently, if applicable, the algorithm assigns the first tasks of $S_R(l)$ and $S_H(l)$ to the human and to the robot (Lines 4-7). At this point, it starts two loops to check the end of the job (Line 8) and the actual level (Line 9), respectively. Inside the second loop, the scheduler first checks if the robot has performed all its tasks in the actual level. If this is true, the robot is in idle, waiting for the human to finish its task, and some tasks may be rescheduled (Line 11), maximizing the parallelism between the two agents. In the other cases the algorithm exploits the function **monitoR**(T_R) to

Algorithm 1 DynamicScheduler()

1: **Require:** S_H,S_R
2: $End_R, End_H \leftarrow false$
3: $l \leftarrow 1$
4: **if** $S_R(l) \neq \emptyset$ **then** $T_R \leftarrow S_R(l,1)$
5: **end if**
6: **if** $S_H(l) \neq \emptyset$ **then** $T_H \leftarrow S_H(l,1)$
7: **end if**
8: **while** $l \leq L$ **do**
9: **while** ($T_R \neq \emptyset$ **and** $T_H \neq \emptyset$) **do**
10: **if** $T_R = \emptyset$ **then**
11: $S_R \leftarrow$ **reschedule**(T_H, S_R)
12: **else**
13: $End_R \leftarrow$ **monitorR**(T_R)
14: **end if**
15: $End_H \leftarrow$ **checkEndH**()
16: $M_H \leftarrow$ **read**$_H()$, $M_R \leftarrow$ **read**$_R()$
17: $(End_H, End_R, S_H, S_R) =$ **communication**(M_H, M_R, S_H, S_R)
18: **if** End_H **then** $T_H \leftarrow$ **next**$(T_H, S_H(l))$
19: **end if**
20: **if** End_R **then** $T_R \leftarrow$ **next**$(T_R, S_R(l))$
21: **end if**
22: **end while**
23: $l \leftarrow l + 1$
24: **end while**
25: **updateJQ**()

check if the robot has finished the assigned task (Line 13). The **monitorR**() function can be implemented using standard procedures, available for robotic applications, see e.g. [49]. If the robot cannot succeed to execute T_R, e.g. a timeout error, a *delegate* message M_R is communicated. Subsequently, the algorithm checks if the human has completed its task, e.g. exploiting an HMI, and all the messages generated by the human and the robot are considered for task swapping (Lines 17). Afterwards, the algorithm checks if the two agents have concluded their tasks and, if it is the case, assigns them the next task in the level (Lines 18, 20). If no tasks are scheduled in the actual level, the function **next**$(T, S(l))$ returns $\emptyset$. When both T_R and T_H are empty, then the level is concluded and the job moves on to the next one (Line 23). Finally, when all the tasks have been performed, the job quality parameters are updated through the function **updateJQ**() (Line 25) and used as input for the task assignment of the next job.

The rescheduling algorithm is represented in Algorithm 2. The algorithm requires as input the task T_H is currently assigned to the human and the current robot schedule S_R (Line 1). It exploits the human monitoring strategy to estimate the remaining time t_{res} for the accomplishment of T_H (Line 2). The procedure **monitorH** can be implemented using several strategies available in the literature as, e.g., [43–45]. If t_{res} is greater than the time necessary for the execution of some tasks in S_R, then these tasks may be executed in parallel with T_H and, therefore, the rescheduling procedure starts (Line 3). First, the schedule S_R is split into two sub-lists: pS_R contains all the robot assigned tasks in the actual level while fS_R contains all the tasks of the next levels, which still

Algorithm 2 Reschedule()

1: **Require:** T_H, S_R
2: $t_{res} \leftarrow$ **monitorH**(T_H)
3: **if** $t_{res} > t_{Ri}$ **then**
4: $\quad (pS_R, fS_R) \leftarrow$ **split**(T_R, S_R)
5: $\quad fS_R^r \leftarrow$ **fill**$(fS_R, t_{res} - t_{Ri})$
6: $\quad S_R \leftarrow$ **concat**$(pS_R.S_R^r, fS_R/fS_R^r)$
7: **end if**
8: **return** S_R

need to be performed (Line 4). fS_R is then used to create another sub-list S_R^r, which contains all the tasks that can be executed in the extra time available $t_{res} - t_{Ri}$ and whose precedences have already been executed (Line 4). Subsequently, $S_R(l)$ is updated by concatenating pS_R, with the list of the rescheduled tasks fS_R^r (Line 6). Finally, the new schedule S_R is returned.

During the execution of the job the human and the robot can generate messages in order to communicate to the dynamic scheduler the intention or need to swap their tasks. A detailed representation of this communication layout is shown in Figure 2.3. In particular, the message M_R, sent by the robot, indicated with the brown lines, can be either empty or containing the value "delegate T_R" and it is generated by the **monitorR** function when the robot cannot succeed in executing the assigned task. The message M_H can be either empty or it can assume two values: "reassign T_{reas}", indicated with the dotted lines, or "delegate T_{del}", indicated with the dashed lines. The first message is generated when the human decides to execute the task that the robot is executing, because, e.g., the robot is not doing the assigned work properly or in the best way. The second message is generated when the human decides to delegate some tasks in S_H to the robot. This message has an argument, that specifies the task to be delegated. The human can enter the messages through a proper, job dependent, input interface. The messages generated by the human and by the robot are handled by Algorithm 3. The algorithm requires the messages generated by human and robot M_R and M_H, the current schedules S_R and S_H and the task T_R currently assigned to the robot (Line 1). The message M_H is the first to be handled in order to give priority to the decisions taken by the operator. If the human decides to execute a task that was initially assigned to robot, it is necessary to check if the robot already started this task. If this is true, the robot task execution is aborted,

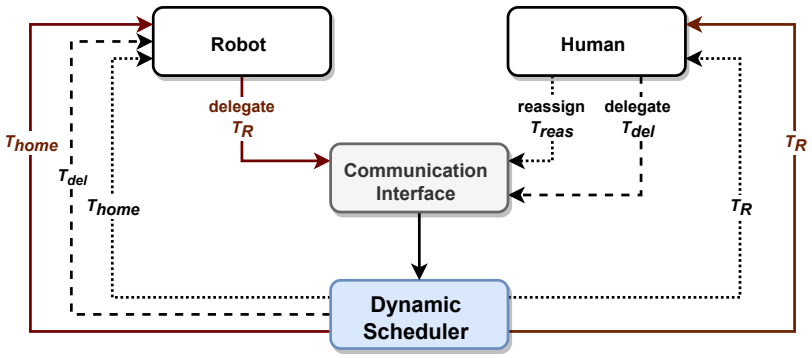


Figure 2.3: Communication Layout.

Algorithm 3 Communication()

1: **Require:** M_H, M_R, T_R, S_H,S_R
2: **if** $M_H = reassign(T_{reas})$ **then**
3: $End_H \leftarrow true$
4: **if** $T_{reas} = T_R$ **then**
5: $End_R \leftarrow true$
6: $S_R \leftarrow \textbf{push}(T_{home}, S_R)$
7: **end if**
8: $S_R \leftarrow \textbf{delete}(T_{reas}, S_R)$
9: $S_H \leftarrow \textbf{push}(T_{reas}, S_H)$
10: **else if** $M_H = delegate(T_{del})$ **and exRobot**(T_{del}) **then**
11: $End_H \leftarrow true$
12: $S_H \leftarrow \textbf{delete}(T_{del}, S_H)$
13: $S_R \leftarrow \textbf{push}(T_{del}, S_R)$
14: **end if**
15: **if** $M_R = delegate(T_R)$ **and exHuman**(T_R) **then**
16: $End_R \leftarrow true$
17: $S_R \leftarrow \textbf{delete}(T_R, S_R)$
18: $S_R \leftarrow \textbf{push}(T_{home}, S_R)$
19: $S_H \leftarrow \textbf{push}(T_R, S_H)$
20: **end if**
21: **return** End_H, End_R, S_H, S_R

moving the robot in an home safe position. T_{reas} is then deleted from S_R and pushed in the first position of the human schedule (Lines 3-9). The human can also decide to delegate a task $T_{del} \in S_H$. If T_{del} is executable by the robot, it is deleted from S_H and transferred into the robot schedule (Lines 10-13). The robot, instead, could detect that it cannot fulfill the assigned task and, if the task is executable by the human operator, T_R is deleted from the robot schedule and inserted in the schedule of the human, while a homing mission T_{home} is added as the next task for the robot (Lines 15-19). Finally the procedure returns the updated end of task variables and schedules (Line 21). The procedures **exRobot** and **exHuman** exploit prior information about the job and the tasks, e.g. the weights w_{Ri} and w_{Hi} in (2.3), to detect if a task can be executed by the robot or by the human. Note that the communication algorithm allows to manage only temporary errors. If the human perceives that a certain type of tasks should not be performed by the robot, e.g. the screwing tool broke, w_{Ri} must be adapted and the task assignment procedure must be re-executed.

Swapping the tasks between the human and the robot directly affects the job quality. Thus, it may happen that the final job quality indices do not respect the constraint imposed in the MILP problem (2.3). However, since the optimization problem requires as input the real job quality parameters, namely $K_{m,0}$, the possible work overload for the operator will be mitigated with subsequent task schedules. Another possible way to ensure the optimum of the job quality is the implementation of a function that checks if the task swapping will violate the constraints and, if required, prevents it. However the latter strategy is very conservative, as it does not take into account the possibility of correcting the job quality parameters in subsequent optimization problems. Furthermore, at the communication level, the job quality is treated as a soft constraint with respect

(a) **(b)**

Figure 2.4: Setup of the Cognitive Layer experiments.

to the need to swap a task, e.g. the robot failing a task is a more critical situation. For these reasons, in this work it was decided not to investigate the latter solution.

The Dynamic Scheduler is computationally cheap. The heaviest part is represented by the **fill**() function inside the **Reschedule**() algorithm (see Algorithm 2, Line 4). This function, in its worst implementation, has a linear complexity equal to $O(n+l)$, i.e. when analyzing every single element of the tuple with two *for* loops. Since the number of tasks and levels in an industrial application is not that large, the algorithm is very reactive.

2.7 Validation

The Cognitive Layer has been experimentally validated in a custom collaborative assembly work shift, that has been set up only for evaluation purpose. During the experiments, the human operator cooperated with a UR10e, a 6-DoF collaborative robot. To monitor the human task execution it has been used a Kinect V2 RGB-D Camera with the official APIs for the skeleton tracking and to evaluate the remaining task time it has been implemented the OE-DTW algorithm, which is already available in literature [50], to both operator wrists. This algorithm returns the percentage completion of the task $\%_{compl}$, comparing the actual time series with the reference ones. This percentage was then exploited to estimate the remaining time of the task as $t_{res_i} = (1 - \%_{compl})t_{Hi}$. The communication interface has been implemented exploiting a simple HMI that sends the desired signals though the computer keyboard. The complete setup for the experiment is shown in Figure 2.4.

All the software components were developed using Robot Operating System (ROS) Melodic Morenia and they ran on a Intel(R) Core(TM) i7-10510U with Ubuntu 18.04. The optimization problem was implemented using Python-MIP [51], a collection of Python tools for the modeling and solution of MILP programs, and solved with Gurobi solver [52]. The UR10e is position controlled using the ROS interface which accepts a desired final position while the overall trajectories are directly generated by the low level controller.

Table 2.1: Tasks Description of the Cognitive Layer experiments.

Task Index	Description	Job
1	Pick&Place square shape.	J_1, J_2
2	Pick&Place U shape.	J_1, J_2
3	Pick&Place circular shape.	J_1, J_2
4	Pick&Place cross shape.	J_1, J_2
5	Pick&Place weight.	J_1, J_2
6	Pick&Place weight.	J_1, J_2
7	Packaging USB key.	J_1
8	Packaging USB key.	J_1
9	Packaging USB key.	J_1

Table 2.2: Task Assignment Data of the Cognitive Layer experiments.

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	k_{i1}	$i \rightarrow j$
1	0.1	12	0.4	15	0	-
2	0.1	12	0.4	15	0	-
3	0.1	12	0.4	15	0	1, 2
4	0.1	12	0.4	15	0	1, 2
5	0.5	25	0.8	10	9	-
6	0.5	25	0.8	10	9	-
7	1000	-	0.4	25	0	-
8	1000	-	0.4	25	0	-
9	1000	-	0.4	25	0	-

The work shift was composed by two jobs divided into multiple tasks. These are listed in Table 2.1.

In the experiments it has been considered the average weight the human has to lift during the execution of the job $K_{1,av}$ as a job quality metric, with its respective upper bound $K_{1,av,max} = 1.1$.

The other inputs required by the task assignment layer (see Section 2.4) are shown in Table 2.2. The nominal durations were estimated by computing the average value of multiple measurements, while the intrinsic costs were calculated with the following equations:

$$w_{Ri} = 0.7D_{Ri} + 1000(1 - capability_i) \tag{2.4}$$

$$w_{Hi} = u_i \tag{2.5}$$

where D_{Ri} represents the distance that the robot has to perform during the execution of the task T_i and $capability_i$ is a Boolean variable that indicates if the robot is capable

of perform the task, e.g. the robot is not able to execute tasks T_7, T_8, T_9. u_i is the attractiveness factor for the human in executing the task T_i and it was defined using our experience and knowledge in the work. k_{i1} are defined taking into account the weight of the objects, i.e. holding an USB stick or a shape does not affect the analyzed job quality metric. The only precedence constraints are related to the shapes and they are necessary to avoid the robot making a wrong pick with the magnets.

When the work shift starts. the job J_1 must be executed and the task assignment layer is initialized with the cumulative cost $K_{1,0} = 0$. The optimization problem is solved in $100\,ms$ and the schedule is composed by the following two tuples:

- $S_H = \{[7, 8, 5], [9]\}$
- $S_R = \{[3, 4, 6], [1, 2]\}$

with an estimated job quality metric equals to:

$$K_{1,av} = \frac{t_{H5} k_{15}}{c} = 1.1 \tag{2.6}$$

Starting from the output of the task assignment layer, the dynamic scheduler was then initialized and the two agents began to perform the collaborative job. The snapshots of the experiments are illustrated in Figure 2.5. The first part of the experiments is dedicated to the execution of the *"nominal schedule"* of the job J_1, i.e. the two agents perform exactly the assigned tasks, as shown in Figure 2.5a. When the robot concludes all the tasks of the first level, the dynamic scheduler exploits the information coming from the OE-DTW to reschedule some tasks (see Algorithm 2, Line 3 – 6). Since the monitoring algorithm returns an estimated remaining time $t_{res} > t_{R1}$, the robot can anticipate the task T_1 in the first level, instead of waiting for the level to finish as scheduled. This procedure, executed in $4\,ms$, is illustrated in Figure 2.5b. It is worth noting that the estimation of the real execution time is the only variable that is evaluated at runtime, i.e. when the robot concludes the level. All the other metrics are evaluated after the execution of the entire job and used as input for the task assignment of the subsequent job.

Once J_1 is concluded the real execution time $t_{H5} = 15\,s$ and the real duty cycle $c = 79\,s$ are exploited to calculate the real job quality metric $K_{1,av} = 1.7$. The real job quality metric is then used as input for the assignment procedure of the next job J_2. The new optimization problem is solved in $90\,ms$ and the resulting schedules are:

- $S_H = \{[3], [1, 2]\}$
- $S_R = \{[4], [5, 6]\}$

As illustrated in Figure 2.5c, thanks to the job quality constraint no tasks affecting the weight metric are assigned to the human and the new estimated job quality metric is below the upper bound:

$$K_{1,av} = \frac{K_{m,0}}{t_e + c} = 0.96 \leq 1.1 \tag{2.7}$$

The second part of the experiments is then dedicated to the execution of J_2, without any relevant result.

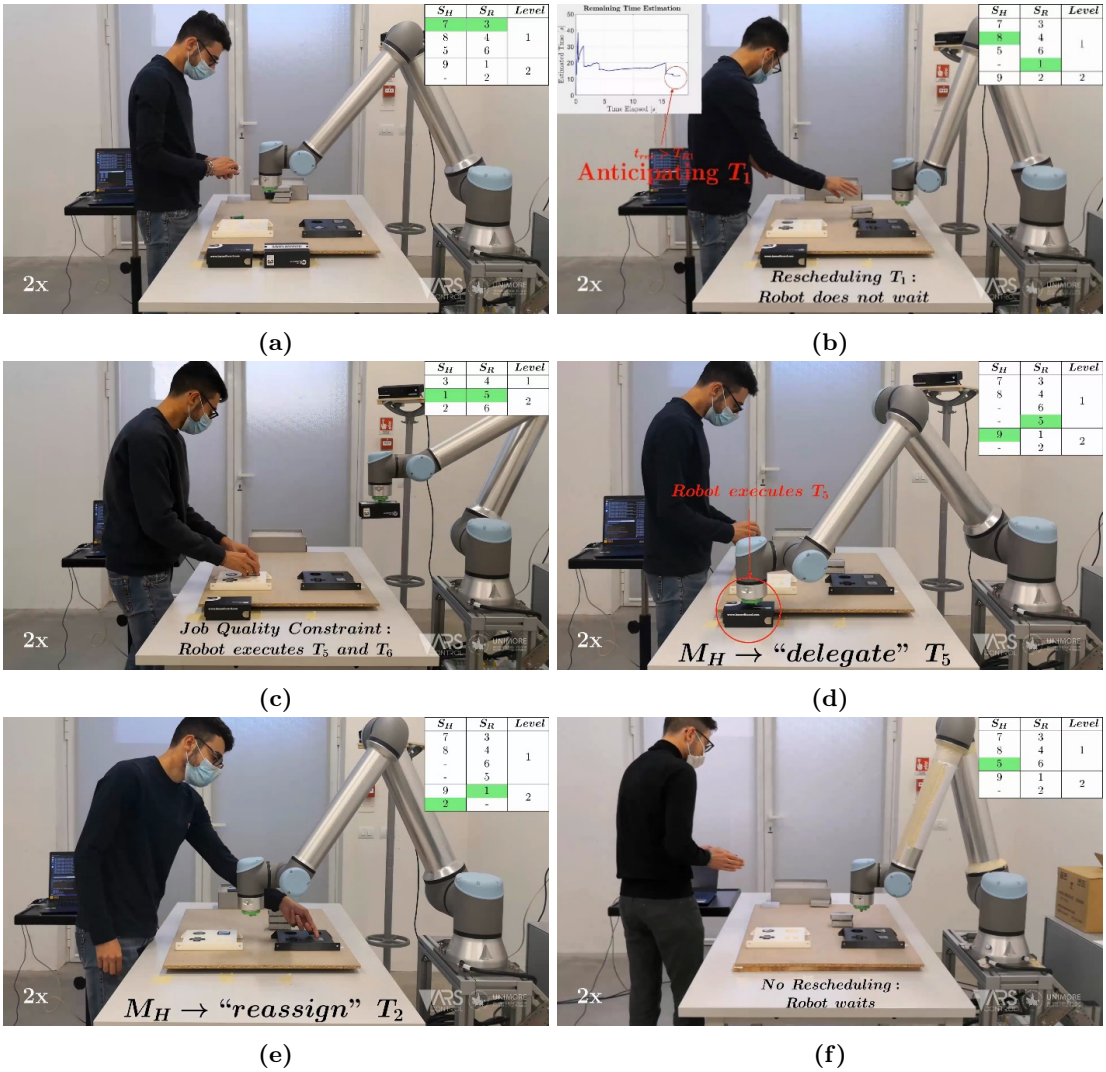
The communication strategy is then exploited. The work shift is reinitialized and the

operator starts to execute the nominal schedule of the job J_1. After concluding T_7 and T_8 the operator sends a message "delegate T_5" to the dynamic scheduler and the robot starts to execute this task instead of the human. This is shown in Figure 2.5d. At the end of the schedule, the human asks to the dynamic scheduler to reassign the task T_2. As illustrated in Figure 2.5e that task is performed by the operator instead of the robot. Since the human did not perform any task that affects the weight metric, the cumulative cost for the assignment of the J_2 is $K_{1,0} = 0$. For this reason, the output of the Task Assignment for J_2 is:

- $S_H = \{[\,], [6]\}$
- $S_R = \{[3, 4], [1, 2, 5]\}$

with an estimated weight metric $K_1 = 1$. This solution is obtained in $100\,ms$. It is worth noting that $c = 90\,s$ this is due to the fact that the task assignment schedules a pause before starting the execution of the second level. This pause is necessary to obtain an admissible value for the weight metric.

In order to demonstrate the effectiveness of the architecture, J_1 is performed without rescheduling the tasks. As happened before, while the human performs its tasks the robot concludes the first level as planned. Since the rescheduling is not active, the robot stops, waiting for the human to conclude the first level of the schedule. This is illustrated in Figure 2.5f. After the human executes T_5, the schedule passes to the second level and two agents resume the expected behavior. As expected, the trial without the rescheduling takes more time, $c = 85\,s$. Moreover, a great improvement can be seen in the robot idle times: $T_{R,idle} = 12\ s$ with the proposed framework and $T_{R,idle} = 20\ s$ without using the rescheduling procedure. It is worth noting that in both trials the human operator introduced approximately the same delay in the first level. The deviation from the nominal durations is in fact equal to $\mu_{proposed} = 1.67 \pm 6.2\,s$ and $\mu_{baseline} = 1.33 \pm 4.7\,s$. This means that the performance of the baseline architecture is not affected by further delays introduced by the human.



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure 2.5: Snapshots of the Cognitive Layer experiments.

Chapter 3

Flexible Execution and Safety Layers

This chapter reports the development of the Flexible Execution and Safety Layers, also called Control Layer, for the ROSSINI project. Problems related to the safe trajectory planning for complex HRC scenarios are addressed. Firstly, an algorithm that allow to dynamic plan an optimal trajectory for the robot manipulator is presented. Subsequently, starting from the safety limits imposed by the safety regulations, an online trajectory scaling strategy is presented. This strategy is mutually integrated with the dynamic planner to improve the performances while maintaining safety for the human operator.

The work presented in this chapter is published in [1].

3.1 Introduction

The introduction and diffusion of collaborative robotics within the industrial environments has allowed to create shared workspace where humans and robots can work closely. While this new paradigm has led to an increase in the flexibility of production lines, the lack of physical barriers requires to pay more attention on how to guarantee human safety. Therefore, the robot safety standards have been updated to address this new collaborative scenarios. In particular, the ISO 10218-1 and the ISO 10218-2 [53, 54] standards classify the collaborative modes in four different categories: *safety-rated monitored stop* (SMS), *hand guiding* (HD), *speed and separation monitoring* (SSM) and *power and force limiting* (PFL). Additionally, the technical specification ISO/TS 15066 [55] provides further information to assess the risk for each collaboration mode. In case of applications where industrial robots are used, the SSM is typically adopted. In this collaborative mode the speed of the robot is reduced according to the relative human-robot velocity and position. However, this approach is overly conservative, since the robot speed should not be limited if its motion is directed away from the human. Moreover, by monitoring the human speed the performance of the robot can be further increased without violating the safety constraints.

Different approaches were presented in the literature to deal with human safety and collision avoidance in a human–robot collaboration (HRC) scenario. In [56] the authors propose a real-time solution to evaluate the future human occupancy and scale the robot speed accordingly, ensuring safety. The idea is to use a 3D camera and a simple human kinematic model to predict the future human occupancy. In [57] an optimization which treats safety as an hard constraint to be satisfied is presented. This strategy leads to obtain a proportional reduction of the speed, with a consequent higher productivity, while ensuring safety. In [58] the authors present a safety framework for collaborative tasks where multiple robots have to share the workspace with human operators. The idea is to scale the velocity preventing that a safety index falls below a certain value. When the scaling procedure is not enough, an emergency stop is applied.

Reducing the speed of the robot is not always the best solution, especially when the workspace conditions allow the robot to modify the pre-planned path. In [59] the authors exploit the concept of static and kinetostatic danger field on a mobile robot in order to prevent collision with human operators in a tire workshop. In [60] the concept of potential field around the whole robot body is used to generate collision-free trajectory. The entire workspace is surrounded by multiple depth sensors that track both dynamic and static object. In [61] authors implement virtual fixtures, which combine attractive and repulsive potential field, in a teleoperated environment. Even if these methods are effective in guaranteeing safety requirements, potential fields can easily cause the system to be stuck in local minima, compromising the task execution.

For this reason, optimization-based algorithms have been exploited to achieve a collision free behavior by applying the minimum correction to the desired path. Safety is embedded through the constraints in the optimization problem. In [62] an optimization problem is solved in real-time in order to force the robot to stay inside a safe set, evalu-

ating the variation of a safety index. In [63], the authors propose an optimization-based control algorithm that explicitly considers safety in order to avoid the human operator while trying to preserve the desired path. Their strategy exploits the use of control barrier functions [64] around the robot body to maintain a collision-free trajectory while fulfilling the ISO/TS 15066.

Adopting the optimal behavior to avoid collision in highly dynamic environments could be computationally challenging, especially in a real industrial scenario where the number of obstacles to be considered is very high. In [65, 66] the authors use kinodynamic rapidly-exploring random tree (RRT) to plan collision free trajectory under kinodynamic constraints. However, these solutions are only suitable for constraints that do not change during the execution of the path, while the safety kinodynamic constraints change in real time based on human behavior.

This chapter proposes a novel framework for trajectory planning and velocity scaling for HRC scenario that is aware of the highly dynamic of the environment and ensures safety for the human operator by explicitly considering safety regulations. The proposed framework is composed by two layers. Given a desired configuration to reach, a trajectory planner layer computes and adapts online the trajectory that the robot has to follow. The trajectory scaling layer, according to the safety constraints imposed by the safety standards, scales the robot velocity ensuring safety for the human operator. Moreover, in order to avoid drastic drops of the robot velocity with consequent poorly efficient robot behaviors, mutual communication between the two layers is enabled. When required, the trajectory scaling can request for a replan of a new trajectory, increasing the robot performances.

The main contributions of this chapter are:

- A novel adaptive framework for trajectory planning and scaling that takes into account the high dynamicity of the environment, adapting in real-time the trajectory.
- A strategy for trajectory scaling that is computationally cheap, i.e. suitable for real industrial application, and that explicitly considers the kindoynamic safety constraint.
- The overall architecture that integrates the trajectory planning and scaling strategies in order to improve the efficiency of the system.

3.2 Problem Statement

Consider a HRC application where a robot manipulator with n joints has to follow a trajectory $\mathbf{q}_{des}(t) \in \mathbb{R}^n$ that goes from an initial configuration $\mathbf{q}_{des}(t_i) = \mathbf{q}_i \in \mathbb{R}^n$ to a desired final configuration $\mathbf{q}_{des}(t_f) = \mathbf{q}_f \in \mathbb{R}^n$ in order to execute a task. In this work, the trajectory $\mathbf{q}_{des}(t)$ is considered admissible if these two conditions hold:

1. It does not collide with the human operator.
2. It is compliant with the safety limits imposed by the ISO/TS 15066.

In particular, defining m as the number of the human limbs, the trajectory $\mathbf{q}_{des}(t)$ is

considered collision-free when

$$d(\sigma_{Ri}(\mathbf{q}(\bar{t})), \sigma_{Hj}(\bar{t})) \geq d_{min} \qquad \begin{aligned} &\forall i \in \{1, \dots, n\}, \\ &\forall j \in \{1, \dots, m\}, \\ &\forall \bar{t} \in \left[t_i, t_f\right], \end{aligned} \tag{3.1}$$

where $d(\sigma_{Ri}(\mathbf{q}(\bar{t})), \sigma_{Hj}(\bar{t}))$ is the distance between the $i-th$ robot link and the $j-th$ human limb at time $\bar{t}$ and d_{min} is the minimum admissible distance. For this reason, the shared workspace is equipped with a monitoring system that allows to track the human movements and estimate the human speed. Several strategies to track the human body are available in literature: skeleton tracking with multiple cameras [67], placing markers on the human body [68], machine learning techniques [69], to name a few.

Since the SSM collaboration mode is considered, to be compliant with the safety standards the robot velocity must be lower than an upper bound that depends on the human-robot distance. This is further explained in Section 3.3. To this purpose, it is possible to explicitly isolate the magnitude of the velocity along the trajectory applying a path-velocity decomposition:

$$\mathbf{q}_{des}(t) = \mathbf{q}_{des}(s(t)) \qquad t \in \left[t_i, t_f\right], \tag{3.2}$$

where s is the curvilinear abscissa that parametrizes the geometrical path $\mathbf{q}_{des}(s(t))$, while the variation of s represents the time law of the desired path, e.g. the velocity profile along the desired path.

Differentiating (3.2) the velocity profile can be isolated:

$$\dot{\mathbf{q}}_{des}(t) = \mathbf{q}'_{des}(s(t))\dot{s} \qquad t \in \left[t_i, t_f\right], \tag{3.3}$$

where $\mathbf{q}'_{des}(s(t))$ is the vector tangent to the desired path, while $\dot{s}$ constitutes the magnitude of the robot velocity. By acting on $\dot{s}$ it is possible to ensure the compliance with the safety standards, without modifying the overall path.

Goal of this work is to design a safety kinodynamic architecture that:

- Computes a nominal trajectory that is always collision-free, i.e. a trajectory that the robot can execute at maximum speed. Exploiting the tracking of the human movements, the planning strategy aims at preserving the feasibility of the trajectory, replanning a new trajectory when the actual trajectory becomes infeasible.
- Starting from the nominal trajectory, scales the robot velocity according to the limits imposed by the ISO/TS 15066 standard. The scaling aims at maintaining safety for the human operator taking into account both the distance between human and robot and the velocity of the human towards the robot.

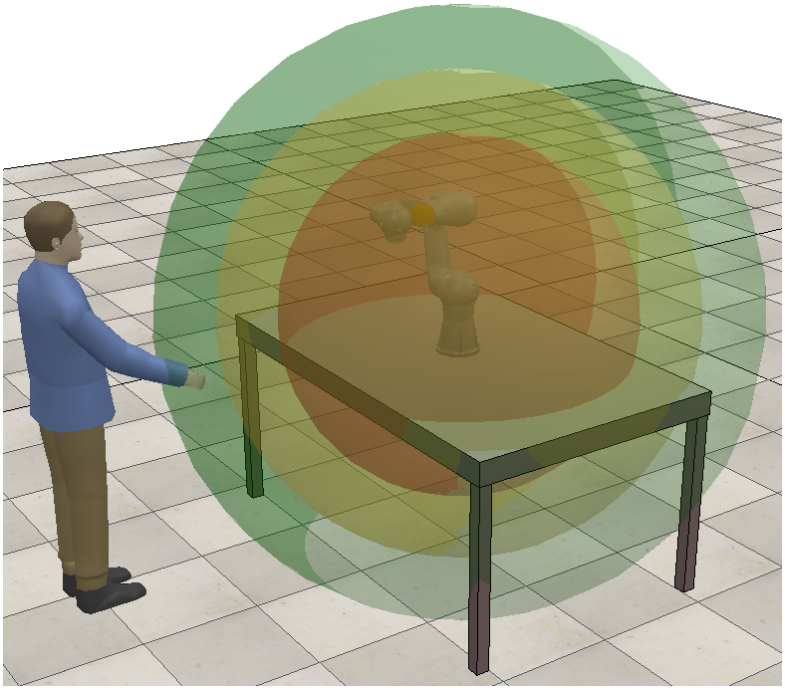


Figure 3.1: Representation of different safety zones with SSM collaboration mode.

3.3 Speed and Separation Monitoring

In modern industrial applications of collaborative robotics, the Speed and Separation Monitoring collaboration mode is widely used. In this collaborative mode, the speed of the robot is continuously adapted depending on the position and velocity of the human operator into the collaborative workspace. Typically the human velocity is not monitored and the workspace is divided into three different areas based on the distance between the human and the robot. This scenario is represented in Figure 3.1. The robot is allowed to operate at full speed when the human is in the green area, at reduced speed when the human is in the yellow area and it stops when the human is in the red area. The ISO/TS 15066 provides the guidelines for calculating the sizes of these areas, namely the minimum protective separation distance S_p, considering also the relative speed between the robot and the human operator. S_p can be computed as:

$$S_p(t_0) = S_h + S_r + S_s + C + Z_d + Z_r. \tag{3.4}$$

$S_p(t_0)$ is the protective separation distance at time t_0, while t_0 is the current time. S_h represents the contribution to the protective separation distance due to the operator's movements, S_r is the one derived from the robot reaction time and S_s is the contribution caused by the robot stopping time. C represents the intrusion distance, i.e. the distance that a part of the body can intrude into the sensing field before it is detected. Z_d and Z_r are the position uncertainties of the human operator inside the workspace and of the robot system respectively.

The first terms of (3.4) can be expressed as:

$$S_h = \int_{t_0}^{t_0+T_s+T_r} v_h(t)\,dt, \tag{3.5}$$

$$S_r = \int_{t_0}^{t_0+T_r} v_r(t)\,dt, \tag{3.6}$$

$$S_s = \int_{t_0+T_r}^{t_0+T_s+T_r} v_s(t)\,dt, \tag{3.7}$$

where T_s and T_r represents the robot stopping time and the robot reaction time respectively. v_h is the directed speed of the human operator towards the robot, v_r is the directed speed of the robot towards the human operator and v_s is the speed of the robot in the course of stopping.

Under the assumptions that the velocity of the robot is constant during the robot reaction time, that the acceleration remains constant during the stopping phase and that the dynamics of the human operator is slower than the robot dynamics, which is true in the case of a generic HRC application, the equations (3.5) – (3.7) can be approximated as follow:

$$S_h = v_h(t_0)(T_s + T_r), \tag{3.8}$$

$$S_r = v_r(t_0)T_r, \tag{3.9}$$

$$S_s = v_r(t_0)T_s - a_{max}\frac{T_s^2}{2}, \tag{3.10}$$

where a_{max} is the maximum robot deceleration expressed as absolute value.

Remembering that the robot stopping time can be expressed as a function of the actual robot velocity, $T_s = \frac{v_r(t_0)}{a_{max}}$, it is possible to further expand (3.8) – (3.10) as:

$$S_h = v_h(t_0)\left(\frac{v_r(t_0)}{a_{max}} + T_r\right), \tag{3.11}$$

$$S_r = v_r(t_0)T_r, \tag{3.12}$$

$$S_s = \frac{v_r(t_0)^2}{2a_{max}}. \tag{3.13}$$

Substituting (3.11) – (3.13) in (3.4), it is possible to obtain an upper bound robot velocity:

$$v_{r_{max}}(t_0) = \sqrt{v_h(t_0)^2 + (a_{max}T_r)^2 - 2Ka_{max}} - a_{max}T_r - v_h(t_0), \tag{3.14}$$

where $K = C + Z_d + Z_r - S_p(t_0)$.

Using equation (3.14) it is possible to compute the safety limit imposed by the ISO/TS 15066, i.e. the maximum velocity that the robot could reach in the direction of human operator. Figure 3.2 shows the trend of the velocity limit $v_{r_{max}}(t_0)$ as a function of the

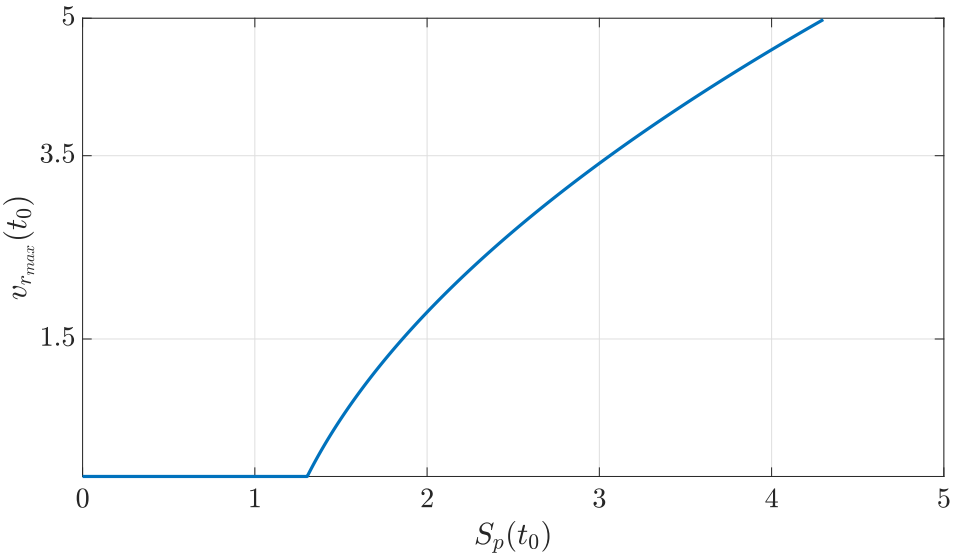


Figure 3.2: Maximum allowed robot velocity towards the human operator.

actual separation distance between the human operator and the robot $S_p(t_0)$ in case of $v_h(t_0) = 2\,m/s$, $a_{max} = 7.5\,m/s^2$, $T_r = 0.002\,s$, $C + Z_d + Z_r = 1.3\,m$.

3.4 Architecture

The Flexible Execution and Safety Layers are represented by the architecture in Figure 3.3. In particular, in the architecture the black lines symbolize the data exchange and the red one constitutes the signal that request for a replan of a new trajectory. The grey blocks symbolize the strategies implemented to provide richer information to the layers while the white block represents the agent. Moreover, in blue the two main layers can be distinguished:

1. **The Trajectory Planning layer**. It is responsible of generating the initial nominal trajectory that the robot can execute at maximum speed, i.e. it considers only the robot limits. Subsequently, it continuously adapts this trajectory exploiting the human tracking information.

2. **The Trajectory Scaling layer**. It is responsible of scaling the robot velocity along the planned path, explicitly taking into account the velocity limits imposed by the safety constraints.

Once the trajectory planning computes the initial nominal trajectory, it sends it to the trajectory scaling and it remains active until the robot reaches the desired final configuration $\mathbf{q}_f$. The trajectory planning layers does not take into account the safety regulation, i.e. it computes a trajectory that the robot could ideally execute at maximum speed.

The trajectory scaling firstly applies a path-velocity decomposition to the desired trajectory as shown in (3.2)–(3.3). Subsequently, it computes online the optimal scaled

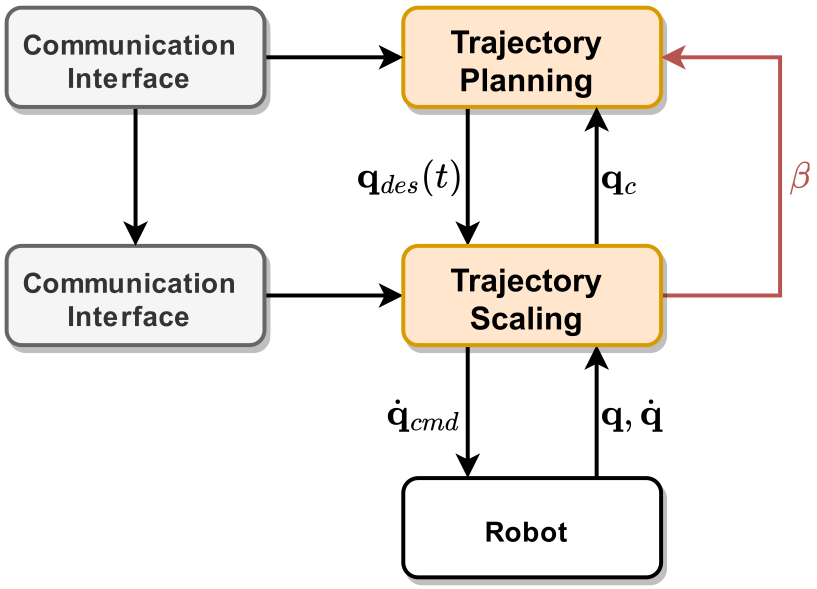


Figure 3.3: The Flexible Execution and Safety Layers architecture.

velocities in order to satisfy the constraint imposed by ISO/TS 15066 (3.14).

During the execution of the motion, mutual communication between the two layers is enabled. The trajectory planning exploits the human tracking information and replans a new trajectory when the previous one becomes infeasible, as explained in Section 3.4.1. The trajectory scaling immediately parametrizes the new trajectory and starts following the new path. At each iteration, it returns to the trajectory planning the actual state of the trajectory. Moreover, when the scaling factor decreases too much, the trajectory scaling sends a signal to the trajectory planning requesting for a new trajectory to be planned, as it is becoming inefficient, see Section 3.4.2.

It is worth noting that during the real-time execution the two algorithms work in parallel, relying on the last available data sent by the other algorithm. For an optimal behavior, the scaling algorithm should work at a frequency at most equal to that of the robot control.

3.4.1 Trajectory Planning

The role of this layer is to find a trajectory $\mathbf{q}_{des}(t)$ for the robot that is collision-free and that the robot can execute at maximum speed. Since the human behavior is in general unpredictable it is not possible to use a strategy that computes offline an optimal trajectory, as in short time it could become infeasible causing collisions between the human and the robot. The trajectory planning aims at continuously maintaining a collision-free trajectory, adapting it online when required.

The trajectory planning is implemented according to the pseudo-code reported in Algorithm 4. The trajectory planning needs as input the initial and the final configuration, respectively $\mathbf{q}_i$ and $\mathbf{q}_f$, and the length of the horizon trajectory that will be checked

Algorithm 4 TrajcetoryPlanning()

1: **Require:** $\mathbf{q}_i, \mathbf{q}_f, N$
2: $\mathbf{q}_{des}(t) \leftarrow \mathbf{plan}(\mathbf{q}_i, \mathbf{q}_f)$
3: $\mathbf{send}(\mathbf{q}_{des}(t))$
4: $\mathbf{q}_c \leftarrow \mathbf{q}_i$
5: **while** $\mathbf{q}_c \neq \mathbf{q}_f$ **do**
6: $h \leftarrow \mathbf{horizon}(\mathbf{q}_c, N)$
7: **for** $i = 1 : N$ **do**
8: **if not_feasible**$(h(i))$ **then**
9: $\mathbf{q}_{des}(t) \leftarrow \mathbf{replan}(\mathbf{q}_{des}(t), \mathbf{q}_{des}(h(i-1)), \mathbf{q}_f)$
10: **break**
11: **end if**
12: **end for**
13: **if** β **then**
14: $\mathbf{q}_{des}(t) \leftarrow \mathbf{replan}(\mathbf{q}_{des}(t), \mathbf{q}_c, \mathbf{q}_f)$
15: **end if**
16: **update_qc**()
17: **end while**

N (Line 1). It immediately plans the maximum speed trajectory $\mathbf{q}_{des}(t)$ that the robot could perform (Line 2). The function **plan** can be implemented using different strategies available for robotic applications, see e.g. [70–72]. Subsequently it sends the trajectory to the trajectory scaling layer (Line 3) and it sets the current trajectory state $\mathbf{q}_c$ equal to the initial configuration $\mathbf{q}_i$ (Line 4). From this point the algorithm starts to loop until the entire trajectory has been executed (Line 5). In the loop, the dynamic planner first creates the horizon h starting from the actual state (Line 6). This horizon represents the set of the future configuration that are analyzed to check if the trajectory is still feasible (Line 7 – 8). In case an infeasible configuration is found a new feasible trajectory is planned through the function **replan** (Line 9). The **replan** function is responsible of planning a new trajectory that goes from a desired configuration, in this case the last feasible one $\mathbf{q}(h(i-1))$, to the final goal. Moreover, the **replan** function merges the new trajectory with the previous one and sends the resulting trajectory to the trajectory scaling. Subsequently, the dynamic planning algorithm reads if there is a request to replan a new trajectory due to the inefficiency of the current one, i.e. β is equal to one (Line 13). This request is given by trajectory scaling layer, as described in Section 3.4.2. If there is the request, a new trajectory starting from the actual configuration is computed (Line 14). Lastly, the actual configuration is updated exploiting the information coming from the trajectory scaling in (3.20) (Line 16).

The replan algorithm is presented in Algorithm 5. The algorithm takes as input the

Algorithm 5 replan()

1: **Require:** $\mathbf{q}_{des}(t), \mathbf{q}_{rp}, \mathbf{q}_f$
2: $\mathbf{q}_{new}(t) \leftarrow \mathbf{plan}(\mathbf{q}_{rp}, \mathbf{q}_f)$
3: $\mathbf{q}_{des}(t) \leftarrow \mathbf{merge}(\mathbf{q}_{des}(t), \mathbf{q}_{new}(t))$
4: $\mathbf{send}(\mathbf{q}_{des}(t))$
5: $\mathbf{return}(\mathbf{q}_{des}(t))$

actual planned trajectory $\mathbf{q}_{des}(t)$, the starting configuration of the new trajectory $\mathbf{q}_{rp}$

and the final desired configuration $\mathbf{q}_f$ (Line 1). It firstly plan a new trajectory $\mathbf{q}_{new}(t)$ that goes from the starting configuration of the new trajectory $\mathbf{q}_{rp}$ to the desired goal $\mathbf{q}_f$ (Line 2). The new trajectory is then merged with the old one (Line 3). This merging procedure replaces the part of the old trajectory from $\mathbf{q}_{rp}$ to $\mathbf{q}_f$ with the new trajectory such that:

$$\begin{cases} \mathbf{q}_{new}(t_{i,new}) = \mathbf{q}_{rp}, \\ \dot{\mathbf{q}}_{new}(t_{i,new}) = \dot{\mathbf{q}}_{rp}, \\ \ddot{\mathbf{q}}_{new}(t_{i,new}) = \ddot{\mathbf{q}}_{rp}, \\ t_{i,new} = t_{rp}, \end{cases} \tag{3.15}$$

where $t_{i,new}$ and t_{rp} are the initial time of the new trajectory and the time corresponding to the replanning configuration $\mathbf{q}_{rp}$, respectively. $\dot{\mathbf{q}}_{new}(t_{i,new})$ and $\ddot{\mathbf{q}}_{new}(t_{i,new})$ are the velocity and the acceleration of the new trajectory at the starting time, while $\dot{\mathbf{q}}_{rp}$ $\ddot{\mathbf{q}}_{rp}$ are the velocity and the acceleration that the robot should have when it reaches the replanning configuration. Both t_{rp}, $\dot{\mathbf{q}}_{rp}$ and $\ddot{\mathbf{q}}_{rp}$ can be obtained analyzing $\mathbf{q}_{des}(t)$. Lastly, the updated trajectory $\mathbf{q}_{des}(t)$ is sent to the trajectory scaling (Line 4) and returned to the dynamic planner (Line 5).

3.4.2 Trajectory Scaling

Starting from the output of the dynamic planner, the goal of the trajectory scaling is to regulate the robot velocity without violating the safety constraint expressed in (3.14). When a human and a robot cooperate the environment could be highly dynamic, for this reason the robot must follow exactly the same path coming from the upper layer, since a deviation from the planned path could cause a collision. The trajectory scaling aims at scaling only the magnitude of the velocity $\dot{s}$, assuring that the executed path is collision-free.

This is achieved in two steps. Firstly, by applying the path-velocity decomposition as shown in (3.2) – (3.3). Secondly, by solving the following optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & -\alpha, \\ \text{s.t.} \quad & \\ & J_{r_i}(\mathbf{q})\mathbf{q}'(s)\dot{s}\alpha \leq v_{max_i} \qquad \forall i \in \{1, \dots, n\}, \\ & \dot{\mathbf{q}}_{min} \leq \mathbf{q}'(s)\dot{s}\alpha \leq \dot{\mathbf{q}}_{max}, \\ & \ddot{\mathbf{q}}_{min} \leq \frac{\mathbf{q}'(s)\dot{s}\alpha - \dot{\mathbf{q}}}{T_r} \leq \ddot{\mathbf{q}}_{max}, \\ & 0 \leq \alpha \leq \delta_s. \end{aligned} \tag{3.16}$$

$\alpha \in [0, \delta_s]$ is the optimization variable and represents the scaling factor. δ_s is a variable that will be exploited in Chapter 4, while for this moment it can be considered always equal to 1. $J_{r_i}(\mathbf{q}) \in \mathbb{R}^{1 \times n}$ is a *modified jacobian* that takes into account only the scalar velocity towards the human operator of the i-th link. This modified version of the

jacobian is required as the velocity constraint imposed by the ISO/TS 15066 (3.14) limits only the velocity that reduce the human-robot distance, i.e. the velocity towards the human. $v_{max_i} \in \mathbb{R}$ is the velocity limit imposed by the ISO/TS 15066 for the i-th link. $\dot{\mathbf{q}}_{min} \in \mathbb{R}^n$ and $\dot{\mathbf{q}}_{max} \in \mathbb{R}^n$ are the joint velocity lower bounds and the joint velocity upper bounds, respectively. While $\ddot{\mathbf{q}}_{min} \in \mathbb{R}^n$ and $\ddot{\mathbf{q}}_{max} \in \mathbb{R}^n$ are the acceleration limits. $\dot{\mathbf{q}} \in \mathbb{R}^n$ is the actual robot velocity and T_r is the robot execution time.

The modified jacobian $J_{r_i}(\mathbf{q})$ is expressed as:

$$J_{r_i}(\mathbf{q}) = \vec{n}_i^T \begin{bmatrix} J_i(\mathbf{q}) & \bar{0} \end{bmatrix} \tag{3.17}$$

where $\vec{n}_i = \{n_{x_i}, n_{y_i}, n_{z_i}, 0, 0, 0\}$ is the unit vector representing the direction that goes from the i-th robot link to the human. The method used to compute this unit vector is a design parameter, e.g. it can be found representing both the robot and the human links as capsules and computing the minimum distance [73]. $J_i(\mathbf{q}) \in \mathbb{R}^{6\times i}$ is the i-th jacobian, i.e. the jacobian matrix that relates the firsts i joints velocity to the linear and angular velocity of the i-th link, and $\bar{0} \in \mathbb{R}^{6\times(n-i)}$ is a matrix with all zero elements.

The optimization problem (3.16) is a convex problem and computationally cheap, since the only factor that affects the convergence is the problem dimension, i.e. the number of joints and links. Thanks to its convexity, the solution obtained by the solver is always the global minimum of the cost function, i.e. the maximum admissible scaling factor. Moreover the problem has always a feasible solution. When the human operator is very far from the robot, the robot is allowed to move at the desired speed, i.e. $\alpha = 1$ that is the maximum speed as seen in Section 3.4.1. When the human approaches the robot, the safety standards require to decrease the velocity until, in the worst case, stopping the robot. This is guaranteed by the solution $\alpha = 0$.

The output of the trajectory scaling is then used to send the desired velocity to the robot:

$$\dot{\mathbf{q}}_{cmd} = \mathbf{q}'(s)\dot{s}\alpha, \tag{3.18}$$

and the curvilinear abscissa s is updated accordingly:

$$s_{new} = s + \dot{s}\alpha T_r, \tag{3.19}$$

Lastly, the new robot configuration that is given to the dynamic planner for the replanning procedure (see Algorithm4, Line 16) is equal to:

$$\mathbf{q}_c = \mathbf{q}'(s_{new}). \tag{3.20}$$

However, greatly reducing the robot velocity is a very conservative strategy and it strictly decreases the overall efficiency. Sometimes it could be more convenient for the robot to move away from the human and execute another trajectory. For this reason it has been implemented a step signal that requests to the dynamic planner the replan of new

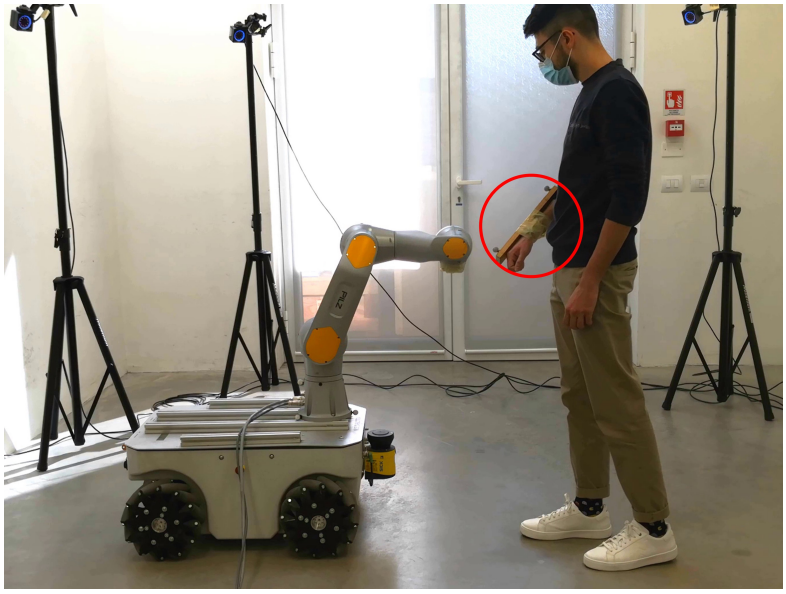


Figure 3.4: Setup of the Flexible Execution and Safety Layers experiments.

trajectory:

$$\beta = \begin{cases} 1 & \alpha \leq \alpha_{min} \\ 0 & otherwise \end{cases} \tag{3.21}$$

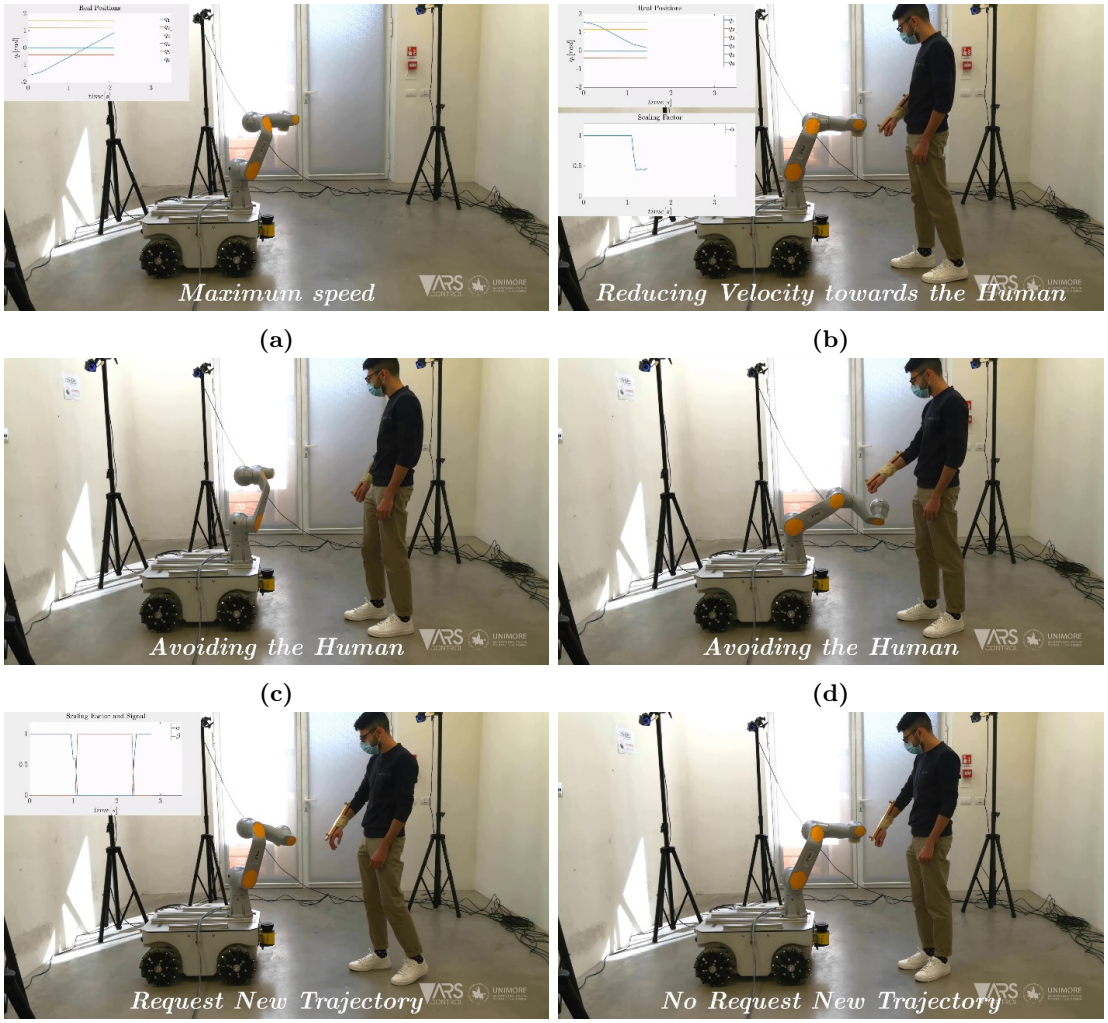
where α_{min} is a predefined threshold and represents the lower desired bound for the scaling factor.

When β is high a replan request is sent to the trajectory planning and a new trajectory is planned (see Algorithm 4, Line 13).

3.5 Validation

The proposed two-layers framework has been experimentally validated on a Pilz PRBT, a 6-DoF manipulator for industrial application. To track the movements of the human right arm it has been decided to exploit six OptiTrack Prime^x cameras with the OptiTrack Motive software [74]. A complete setup of the experiments is shown in Figure 3.4. Where it is possible to see the Pilz PRBT manipulator, which is placed on a mobile robot, three of the six OptiTrack Prime^x cameras and a wooden rod with the OptiTrack markers to track the right arm of the human operator (red circle). All the software components were developed using ROS Melodic Morenia meta-operating system and they ran on a Intel(R) Core(TM) i7-10510U with Ubuntu 18.04. The dynamic planner layer is based on the RRT-Connect algorithm [75] and it is implemented using MoveIt Motion Planning Framework [76]. The trajectory scaling layer exploits the C code generated by CVXGEN [77] to solve the optimization problem (3.16). For simplicity, the modified jacobian J_{r_i} is applied only to the end-effector while $\vec{n}_i$ is the unit vector of minimum distance between the $i-th$ robot link and the human operator arm. The minimum distance is computed representing both the robot links and the human arm as capsules, see [73].

Concerning the frequencies, the communication with the robot works at 50 Hz while the



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 3.5: Snapshots of the Flexible Execution and Safety Layers experiments.

optimization problem converges in 1 ms. The OptiTrack, instead, works at a frequency of 240 Hz. Since the PRBT has not a real-time velocity ROS interface, it has been decided to position control the robot integrating the solution coming from (3.16) at 20 Hz.

In the experiment the robot has to go continually from the initial configuration,

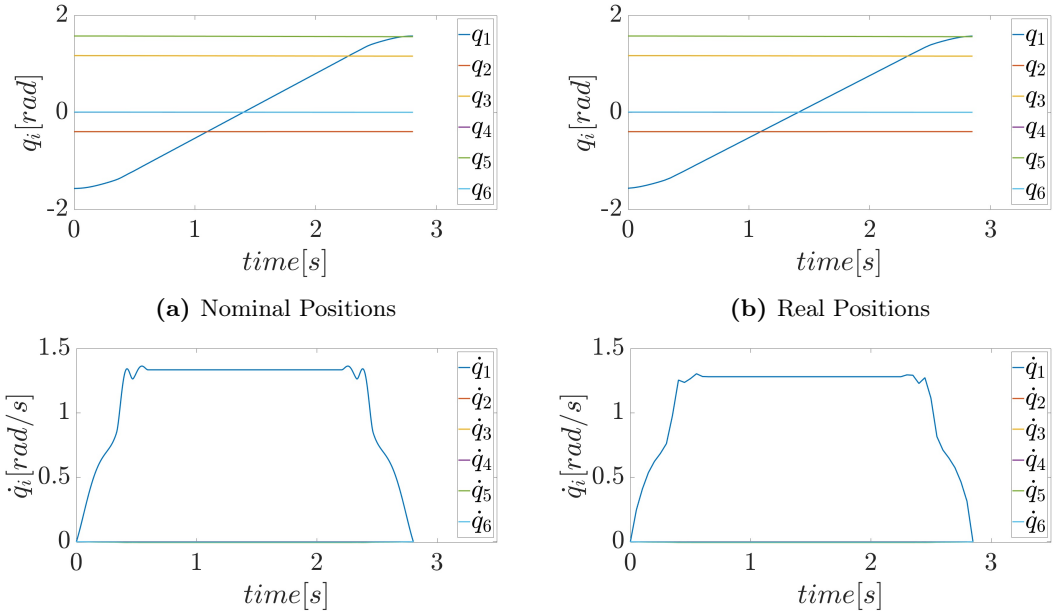
$$\mathbf{q}_i = \{1.57, -0.4, 1.17, 0.0, 1.57, 0.0\}$$

to the final configuration

$$\mathbf{q}_f = \{-1.57, -0.4, 1.17, 0.0, 1.57, 0.0\},$$

and vice versa.

The snapshots of the experiments are illustrated in Figure 3.5. Initially, the human operator is very far from the robot, i.e. he is in the green area of the SSM (see Figure 3.1). In this phase, illustrated in Figure 3.5a, the robot is allowed to move at maximum speed, following the nominal planned trajectory. The resulting trajectory is shown in Figure 3.6. Subsequently, the human operator approaches the robot causing the scaling



(a) Nominal Positions

(b) Real Positions

(c) Nominal Velocities

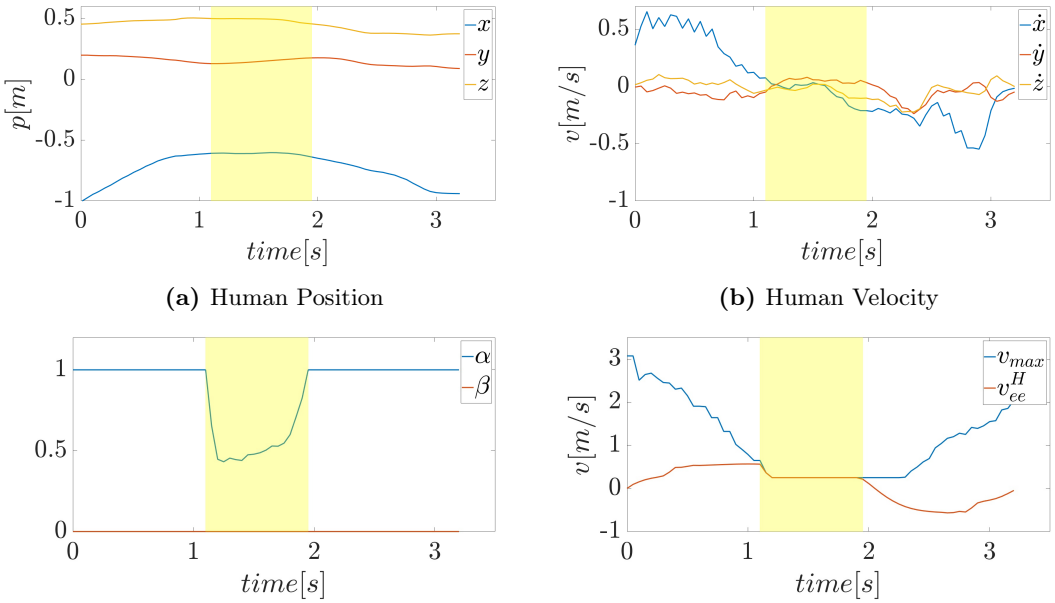
(d) Real Velocities

Figure 3.6: First part of the experiment, nominal execution.

of the trajectory, as shown in Figure 3.5b. This is due to the fact that, according to the safety limit imposed by ISO/TS 15066 (3.14), the maximum speed allowed towards the human operator decreases. The Figures 3.7a and 3.7b show the position and the velocity of the nearest human point in the robot reference frame, respectively. As a consequence of the approaching behavior, in the first phase the x component increases and its velocity is positive. While during the scaling, the velocity components are very low. Figures 3.7c and 3.7d show the behaviour of the scaling factor and demonstrate that the safety constraint is not violated, respectively. In the graph only the velocity of the end-effector towards the human v_{ee}^{H} is shown. It is worth noting that the robot slows down only in the first part of the trajectory, i.e. from $t = 1.1$ *sec* to $t = 1.95$ *sec*. This is because the robot is going towards the human operator. In the second part, i.e. when it moves away, it goes at higher speed, restoring the nominal behavior. As a matter of fact, at $t = 1.95$ *sec* the scaling factor increases. A comparison between the planned trajectory and the scaled one can be found in Fig 3.8.

In the next part of the experiment the human operator hinders the robot, making the trajectory infeasible. The dynamic planner layer takes care of planning a new one, avoiding the human operator, and the robot is able to reach the desired configuration. This is demonstrated with the snapshots in Figures 3.5c and 3.5d.

In the last part of the experiment, the human operator goes very close to the robot, causing a drop of the scaling factor. When $\alpha \leq \alpha_{min} = 0.2$ the trajectory scaling layer sends a step signal $\beta = 1$ to the dynamic planner requesting for a replan of a more efficient trajectory, as explained in Section 3.4. The evolution of the scaling factor and the signal is shown in Figure 3.9, while the replanning is illustrated in Figure 3.5e. The replanning strategy is successful and the robot is free to restore its behavior, completing the trajectory.



(a) Human Position

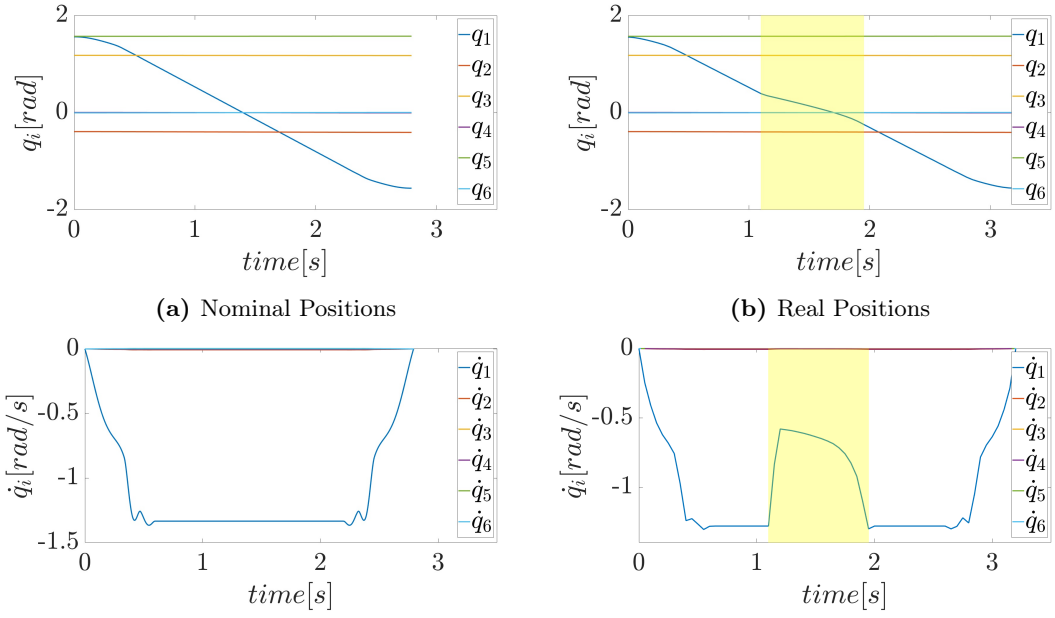
(b) Human Velocity

(c) Scaling Factor and Step Signal

(d) Velocity Constraint on the EE

Figure 3.7: First part of the experiment, scaling of the trajectory.

In order to demonstrate the effectiveness of the architecture, the same experiment is performed without sending the replan request when the scaling factor is too small. As shown in Figure 3.5f, when the human operator goes very close, the robot stops and it stays stuck until the human operator leaves. Intuitively, this solution without the replan signal strictly depends on how long the human operator stays close to the robot. For this reason, a comparison on the execution times of the different strategies would not be very interesting.



(a) Nominal Positions

(b) Real Positions

(c) Nominal Velocities

(d) Real Velocities

Figure 3.8: Second part of the experiment, comparison of the trajectories.

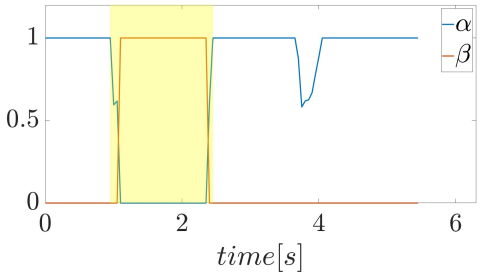


Figure 3.9: Last part of the experiment, replanning request.

Chapter 4

Safety Aware Control Architecture

This chapter reports the development of the Safety Aware Control Architecture the ROSSINI project. Problems related to the integration of the Cognitive Layer and the Flexible Execution and Safety Layers are reported. Furthermore, a brief explanation of the Perception Layer and of the RS4 system and subsequently of their integration in the architecture is reported. Finally, the chapter concludes with the experimental validation in the use cases of the project.

The work presented in this chapter is published in [4, 6, 7].

4.1 Introduction

After the development of the cognitive layer and the flexible execution layer, the next step of the ROSSINI project is to integrate the two architectures into one large framework called safety aware control architecture. Furthermore, this architecture also contains the perception layer and the RS4 safety architecture developed within the project. The purpose of this framework is to collect and unite all technologies concerning both the control and safety features of the robot to create a framework that may lead to make a general robot more autonomous and safe.

To this purpose it has been necessary to modify both the cognitive layer and the flexible execution and safety layers. This is because both the scheduling and the planning strategies are two independent modules and it was necessary to enable not only the mutual integration between them but also to exploit the information provided by all the other layers to improve the behavior of the architecture. Moreover, these modules must be integrated with the output of the RS4, responsible of the safety, and of the perception layer, i.e. the semantic scene map, responsible of enriching the data with semantic information.

Integrating all these modules it has been possible to create an integrated and safe architecture that can be used to solve the task allocation and motion planning problem (TAMP) in real industrial scenarios. The framework combines the estimation of the real execution time of the human operator and the safety limits imposed by the regulations to ensure a natural, flexible and safe collaboration. In literature, the TAMP problem is usually solved by using a centralized approach, where a single entity is responsible of the task allocation and motion planning, see e.g. [78–80]. However this approach was outside the scope of the ROSSINI project, which aims at creating a scalable and modular architecture.

The main contributions of this work are:

- A modular and integrated framework for addressing the TAMP problem.
- The experimental validation in real and complex industrial scenarios.

4.2 Architecture

The proposed architecture is represented in Figure 4.1, where three components may be distinguished:

1. **RS4 Layer**, which is responsible of tracking the human operator and the robot inside the collaborative setting in a ***safe certified*** way. It takes as input all the data coming from the safe sensors, combines them to define the zones occupied by the two actors and defines the maximum allowed robot speed according to the ISO/TS 15066.

2. **Perception Layer**, which is responsible of tracking the human operator and the robot in the collaborative setting in a ***non-safe*** **certified** way. It takes as input the

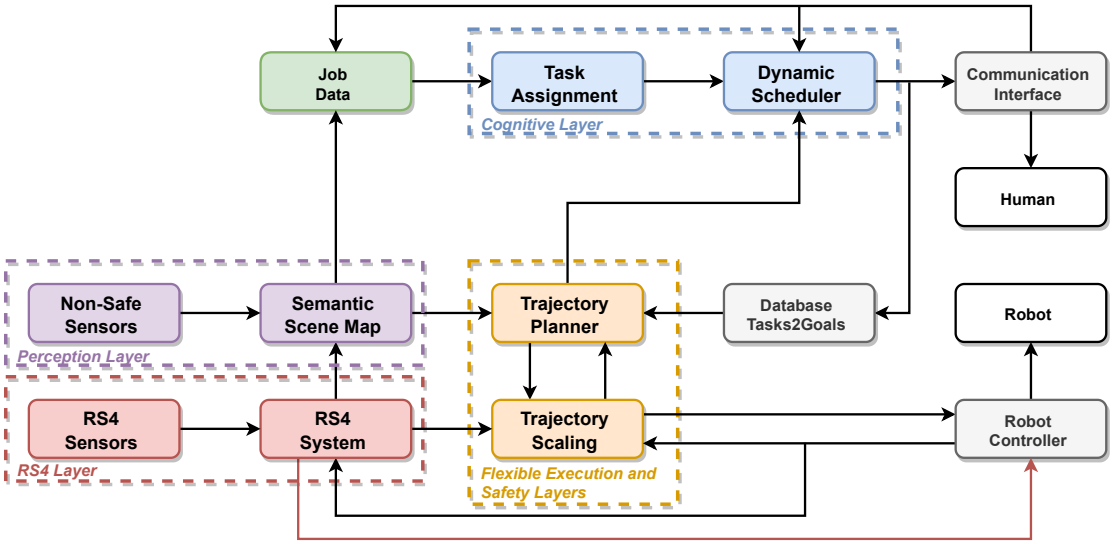


Figure 4.1: The Safety Aware Control Architecture.

data coming from the non-safe sensors and the output of the RS4 layer to generate the dynamic safety shell for both agents. Moreover, it is capable of detecting also semantic information and understand if something in the job has changed and update the job data accordingly.

3. **Cognitive Layer**, which is responsible of scheduling the tasks between the two actors. It firstly build the optimal nominal schedule tasking as input the data of the job to be performed. Subsequently it adapts online the schedule to face the possible deviations that may arise during the real collaboration/interaction.

4. **Flexible Execution and Safety Layers**, which is responsible of planning the collision-free trajectories for the robot at runtime. It firstly exploits the knowledge of the scene and the human zone to compute the trajectory. Subsequently, the robot speed is adapted along the path accordingly to the velocity limit computed by the safety layer and, if necessary, the overall trajectory is replanned.

The entire procedure starts with the definition of the data composing the job, i.e. t_{ai} and w_{ai}. Initially, the task assignment block solves an optimization problem to build a nominal schedule that minimizes the overall cost, while be compliant with both the precedences and job quality constraints. Subsequently, the task scheduling assigns at runtime the tasks to each actor, and if necessary, locally adapts the nominal schedule considering the real execution time or the human preferences. The human tasks, is forwarded to a proper communication interface, that allows the user to interact with the framework. The goal of this communication interface is the same of the one implemented in chapter 2, but the final implementation is case dependant, e.g. for the IMA use case the partner TNO developed a smartphone application. The robot task, instead, is firstly given to a database, in order to convert the single task into a set of intermediate goals that the robot has to execute. Defined the goals, the planners plan a collision-free trajectory that the robot could ideally execute at maximum speed. Lastly, the trajectory scaling adapts at runtime the velocity along the path, ensuring both the collision avoidance, i.e.

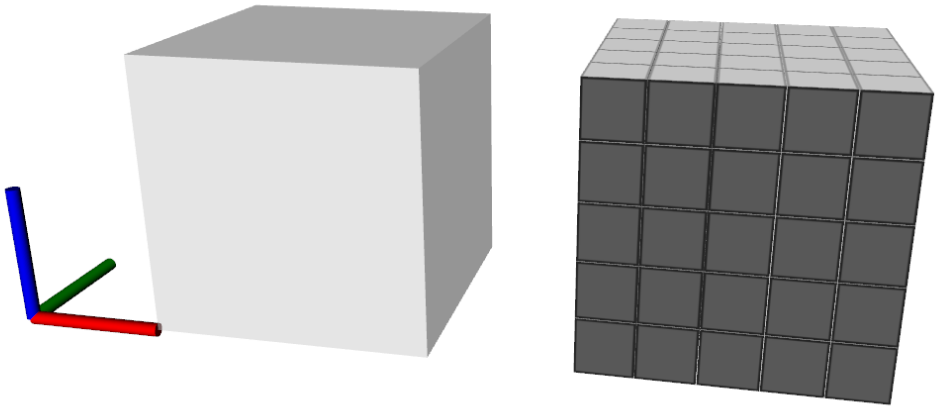


Figure 4.2: Discretization of a 3D area as a Voxel Map.

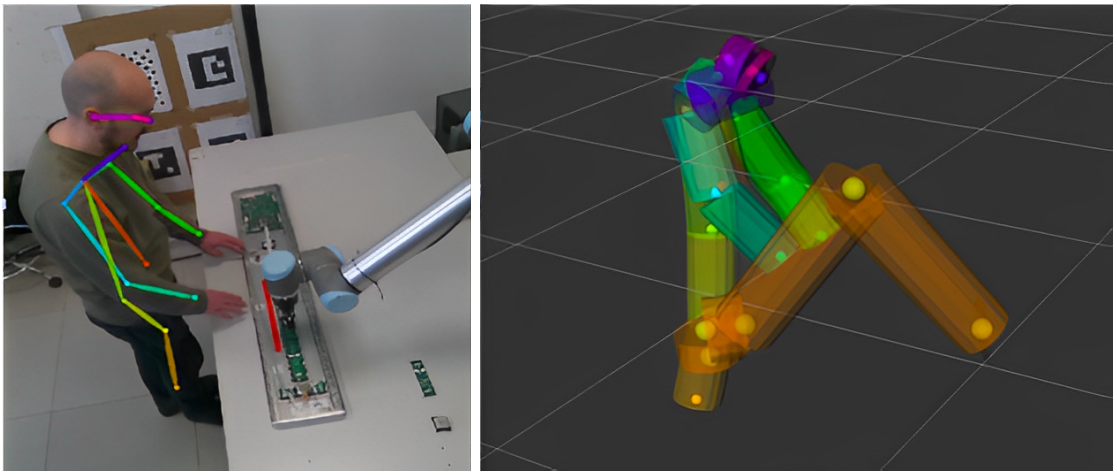
the collision-free path is not changed, and the compliance with the safety regulations and sends the desired velocity to the low-level robot controller. If necessary, e.g., trajectory becomes infeasible, the planner will change online the trajectory.

During the overall procedure, both the RS4 and the perception layers are always active. In particular, the RS4 generates compute at runtime the maximum allowed robot speed that is given to the scaling. Moreover, when it is necessary to stop the robot, the safety layer implements two different strategies: it sends a safe stop to the robot controller, ensuring the safety, and it sends a maximum allowed robot speed equal to 0.

It is worth to underline the differences between the final safety aware architecture and the one that was originally defined in the ROSSINI project, see Figure 1.3. In particular the dynamic shell generator has been included directly in the semantic scene map. Furthermore, the safety layer is still part of the control layer, through the trajectory scaling module, but it has also partially decomposed in a standalone module called RS4 layer.

4.2.1 RS4 Layer

The RS4, developed by the partner Datalogic, is in charge of generating a safe reconstruction of the monitored environment using safe sensors data. All the data coming from the sensors are fused and elaborated to generate a spatial representation of the working environment called Voxel map [81]. This map is a discretization of the 3D space in unit elements, called Voxels. A voxel represents a volume unit on a regular grid in three-dimensional space as shown in Figure 4.2. A data structure associated with a voxel can include information related to its status. As pixels in a 2D bitmap are represented in memory as a 2D array of values or data structures accessed using couples of 2 coordinates, voxels are represented as 3D arrays and accessed using triplets of 3 coordinates. An important point to highlight is related to the procedure used to assign the occupancy status to each voxel belonging to the map. Indeed, each voxel can have three different status: free, occupied or occluded. Using this method, a conservative approach is always ensured during the definition of the space that need to be considered as a possible



(a) **(b)**

Figure 4.3: Dynamic safety shells.

collision object for the robot. Indeed, considering critical also the space that cannot directly being monitored, i.e. the occluded voxels, the worst-case scenario is considered. After having the occupancy status of each voxel, a meaning label is assigned based on a human–robot detection algorithm, where each voxel is labelled as human or as a robot voxel. After that, minimum distance is calculated as the Euclidean distance between the closest voxels belonging to the clusters of voxels which are related to the human and to the robot and the robot-human direction can found as:

$$\vec{n}_{RH} = \frac{\mathbf{x}_H - \mathbf{x}_R}{d_{min}}, \tag{4.1}$$

where $\mathbf{x}_H$ and $\mathbf{x}_R$ are the position of the closest human and robot voxel, respectively, and $\vec{n}_{RH}$ is the unit vector representing the robot-human direction. Moreover, the closest human point and the closest robot point are also exploited to compute the maximum velocity limit for the robot substituting in (3.14).

Equation (4.1) may not always have a solution, i.e. $d_{min} = 0$. This occurs when the human operator touches the robot and the two voxel maps converge into one[1]. In this case, the algorithm developed by Datalogic recognizes this situation and provides a stop signal to the robot via its safety interface. This ensure a safe stop. Furthermore, this signal is also forwarded to the flexible execution and safety layers which sets $\delta_S = 0$ in (3.16).

4.2.2 Perception Layer

The perception, developed by the partner Iris, has the task of generating an unsafe reconstruction of the entire working scenario by merging both the data of the unsafe sensors and the output of the RS4 layer. In particular, the perception layer takes advantage of artificial intelligence techniques to perform two different tasks. First, both the operator and the robot are represented with dynamic safety shells, i.e. variable radius capsules as

[1]This may happen in any HRC scenario without violating the safety, e.g. the robot is moving away and the human operator decides to touch it.

shown in Figure 4.3. These shells are exploited by the flexible execution and safety layers to perform the collision avoidance. The other task performed by the perception layer is the detection of the objects in the scene. With this process, the semantic map returns, for each object, its position in space with a good precision and, if possible, its semantic status. Thanks to the addition of this semantic information, the perception layer is able to evaluate changes in working conditions, e.g. an object is damaged. Variations of this type can be used to reinitialize the overall procedure, starting from a new task assignment.

4.2.3 Cognitive Layer

The role of this layer is to optimally schedule the tasks between the human and the robot, in order to reduce the waiting time and to improve the overall performances of the HRC. This is achieved in two steps. The first, carried out by the task assignment block, with the aim of assigning and scheduling tasks based on nominal execution times. The second, performed by the dynamic scheduler, that aims at adapting the schedule in real-time taking into account the deviations from the nominal behavior. This is achieved based on the Cognitive Layer presented in Chapter 2. Specifically, the rescheduling strategy is improved in order to embed also the planning information.

Starting from the representation of the job as an acyclic graph $\mathcal{G} = (T, E)$, as shown in Figure 2.2, the task assignment distributes the tasks over the levels and generates the first nominal schedule by solving the optimization problem in (2.3). The output of the optimization problem are the two nominal S_H and S_R.

Starting from this, the dynamic scheduler is responsible for adapting online the two nominal schedules taking into account all the deviations from the nominal behavior and exploiting the task planning information. When two humans collaborate, they are able to adjust their behavior according to what is currently happening. If one human being is slow, the other can compensate by speeding up. If one human is working in a certain area, the other can perform tasks that are far away first, so as not to get in the way. The dynamic scheduler aims at reproducing this natural synergy with a consequent improvement of the HRC.

To achieve this, firstly the human operator is monitored in real time when performing the task in order to estimate the real execution time. This data is then exploited to evaluate online the real value of the cost function in (2.3). Since the cost function value has changed, it may be convenient to reschedule some tasks for the robot. The rescheduling procedure exploits a penalty cost γ_i, defined for each task, allowing to choose the most suitable task for the robot at that moment. The higher this penalty cost, the lower the priority of executing the task at that time. The penalty function is implemented according to the following equation:

$$\gamma_i = \Delta G_i + isHA_i + isFA_i + M \cdot prec_i - m \cdot succ_i \tag{4.2}$$

where ΔG_i represents the variation of the cost function in (2.3) that would occur if

the robot performed T_i at this moment. $isHA_i$ and $isFA_i$ are two boolean variables exploited for encoding the planning information inside the scheduling procedure. If the human operator is occupying the area where the robot will have to perform the task, it is very likely that the planner will not be able to plan a collision-free trajectory, failing to execute it, or that it will have to significantly reduce its speed, causing a drop in performance. This is handled by setting the variable $isHA_i$ equal to one. A different situation occurs when the planner has failed to perform a task. It may happen that the human operator is not exactly in the area that the robot needs to reach, but his current position makes it impossible to plan a feasible trajectory. In this case, it is preferable to assign tasks that take the robot elsewhere, setting $isFA_i$ to one for all the tasks in that area. Finally, the variables $prec_i$ and $succ_i$ represent the number of precedence and successor tasks that have not yet been concluded. These variables are multiplied respectively by a large constant M and a small constant m, guaranteeing a high cost if the precedences have not yet been executed and favoring tasks with more successors.

The procedure that takes into account this penalty cost, namely the dynamic scheduler, is implemented according to the pseudo-code reported in Algorithm 6. The algorithm

Algorithm 6 DynamicScheduler()

```
1: Require: S_H,S_R
2: End_H, End_R ← false
3: Fail ← false
4: l ← 1
5: while l ≤ L do
6:     if End_H then T_H ← next(S_H)
7:     end if
8:     (t_real, HA) ← monitorH()
9:     updateG(t_real)
10:    if Fail then FA ← setFA(T_R)
11:    end if
12:    Γ_R ← computeGamma(S_R, HA, FA)
13:    if End_R or Fail then
14:        T_R ← assign(Γ_R, T_H, S_R, S_H)
15:        Fail ← false
16:    end if
17:    End_H ← checkEndH()
18:    (End_R, Fail) ← checkPlanner()
19:    if T_H = ∅ and T_R = ∅ then
20:        l ← l + 1
21:    end if
22: end while
```

needs as an input the nominal tasks schedules S_H and S_R (Line 1). It immediately sets to false the two variables End_H and End_R, which identify when the respective agent has concluded the task and it is available to start a new one, and the variable $Fail$, which identifies when the planning level has failed (Lines 2-3). At this point, it initializes the overall schedule at the first level and it starts a loop to iterate thorough all the levels of the schedule (Lines 4-5). Inside the loop, the algorithm firstly checks if the human has finished in order to assign a new task (Line 6). Subsequently it exploits the

human monitoring data to get bot the real execution time of the human operator and its occupancy area and it updates the cost function G (Lines 8-9). At this points, defining k as the number of completed tasks, the algorithm checks if it is necessary to update the failing area and computes

$$\Gamma_R = \{(T_{k+1}, \gamma_{k+1}), \ldots, (T_N, \gamma_N)\}$$

namely the tuple containing the tasks not yet completed with the corresponding penalty costs[2] (Lines 10-12). It is worth noting that γ_i is calculated for all remaining tasks, regardless of whether they were initially assigned to the robot or the human operator. This means that the robot, if necessary, can perform a task that was initially assigned to the human. If the robot is available, then the algorithm assigns the cheapest task to the robot, updating the schedules S_R and S_H, and sets to false the variable $Fail$ (Lines 14-15). The **assign**() function can be implemented using a well known sorting algorithm, but it must also ensure that the same task is not assigned to both agents and that the next robot task is not too expensive, i.e. precendences not yet completed or task incompatibility. Finally, the algorithm checks if the human operator and the robot have concluded their tasks or the planner has failed (Lines 17-18) and if the level of the schedule has been concluded, i.e. the two agents have no tasks (Lines 19-20).

Each task T_R is then converted into a desired final configuration, whose information is stored into a database.

4.2.4 Flexible Execution and Safety Layers

The role of this layer is to execute the task while ensuring safety for the human operator. This is achieved in two steps. The first, carried out by the trajectory planner block, with the aim of planning a trajectory that is always collision-free. The second, performed by the trajectory scaling, with that of adapting the velocity along the path, ensuring safety. To this aim the flexible execution layer presented in Chapter 3 has been extended. In particular, these are the features changed:

- The dynamic planner exploits the capsules developed by the perception layer to detect collisions with the human operator.
- The dynamic planner returns if the task has been completed or the trajectory is failed.
- The trajectory scaling exploits the data computed by the RS4 and $\delta_S \in (0, 1)$ to ensure safety for the human operator.
- The signal β of the trajectory scaling has been deactivated to avoid the request of a better trajectory.

[2]If the number of tasks is large, evaluating γ_i for all the tasks could be a very demanding procedure. This can be addressed by limiting the computation to only the tasks that were originally scheduled in the actual level and in a future desired horizon.

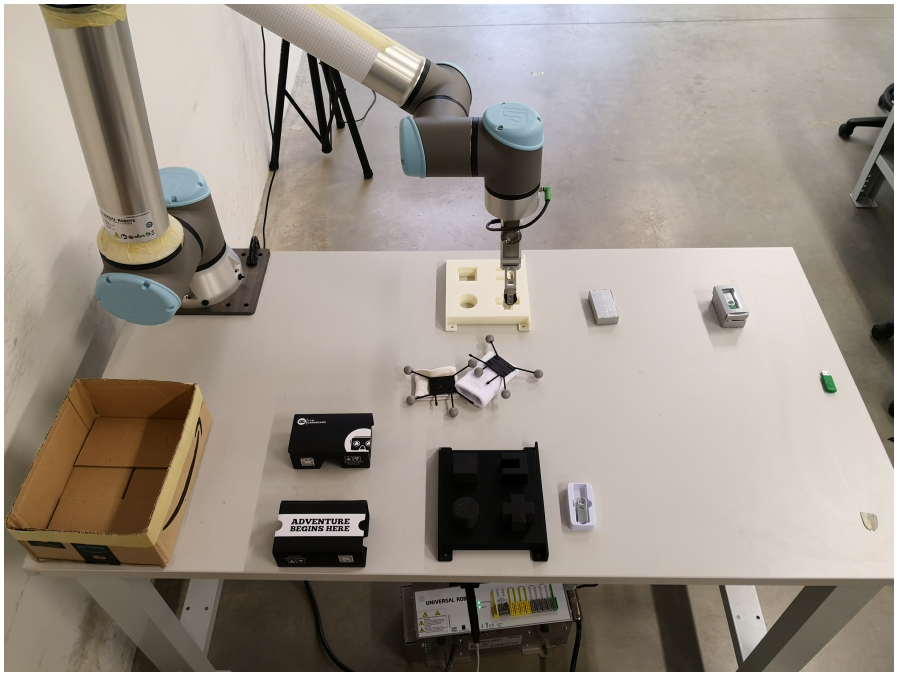


Figure 4.4: Setup of the Lab experiments.

4.3 Validation

The proposed overall architecture has been experimentally validated on five different scenarios. The first one has been carried out in a custom collaborative scenario and focused on validating the integration between the cognitive layer and the flexible execution and safety layers. The second one validate the overall integration inside the company Datalogic, partner of the project. The other three scenarios are the industrial use cases of the project, i.e. Whirlpool, Schindler and IMA.

4.3.1 Lab Scenario

The first experiment regard only the integration of the technologies developed by UNIMORE. To this aim, both the perception layer and the rs4 layer have been substituted with the Optitrack system, as previous done in Chapter 3. The complete setup for the experiments is shown in Figure 4.4, where it is possible to notice two wristbands used to track both human wrists with the OptiTrack.

Table 4.1 reports a detailed description of all the tasks composing the collaborative job, while the data required by the scheduling layer is shown in Table 4.2, where the same heuristics of Section 2.7 are used. The column $\boldsymbol{i \rightarrow j}$ represents the precedence constraint, e.g. tasks T_6 and T_7 must be executed before tasks T_4 and T_5. The nominal durations have been defined by performing the tasks multiple times and computing the average time. The intrinsic costs, instead, have been defined using our experience and knowledge in the collaborative job. It is worth noting that $T_1 = T_2 = T_3$, e.g. there are three tasks that are the same, and for these tasks w_{Ri} is very high, this is because the robot is not capable in executing the USB packaging action. Table 4.3, instead, reports the data used in the scaling optimization problem.

Table 4.1: Lab Tasks Description

Task Index	Description
1	Packaging USB key.
2	Packaging USB key.
3	Packaging USB key.
4	Pick&Place circular shape.
5	Pick&Place cross shape.
6	Pick&Place square shape.
7	Pick&Place U shape.
8	Pick&Place weight.
9	Pick&Place weight.

Table 4.2: Lab Task Assignment Data

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	$i \rightarrow j$	Area
1	1000	-	0.4	25	-	1
2	1000	-	0.4	25	-	1
3	1000	-	0.4	25	-	1
4	0.1	12	0.4	15	-	2
5	0.1	12	0.4	15	-	2
6	0.1	12	0.4	15	4, 5	2
7	0.1	12	0.4	15	4, 5	2
8	0.5	25	0.8	10	-	3
9	0.5	25	0.8	10	-	3

Exploiting this data, the task assignment block solves the optimization problem (2.3) in $100\,ms$ whose result are the following optimal nominal schedules:

- $S_H = \{[2, 3, 8], [1]\}$
- $S_R = \{[6, 7, 9], [4, 5]\}$

Starting from the output of the task assignment, the dynamic scheduler is then tested into three different scenarios: when the human operator is very slow in executing the tasks, the human hinders the robot during the collaboration, the human is inside the same working area of the next robot task. In particular, the first scenario is performed twice: first with a baseline architecture and, later, with the proposed architecture. The baseline architecture is a simplified architecture that does not implement any rescheduling strategy and, therefore, waits for the completion of all the tasks of each level. The other two scenarios are necessary to show how the proposed architecture dynamically adapts the schedule both in the case of planning failures and in the case of already occupied working area, validating the overall architecture. A set of snapshots illustrating the experiments can be found in Figure 4.5.

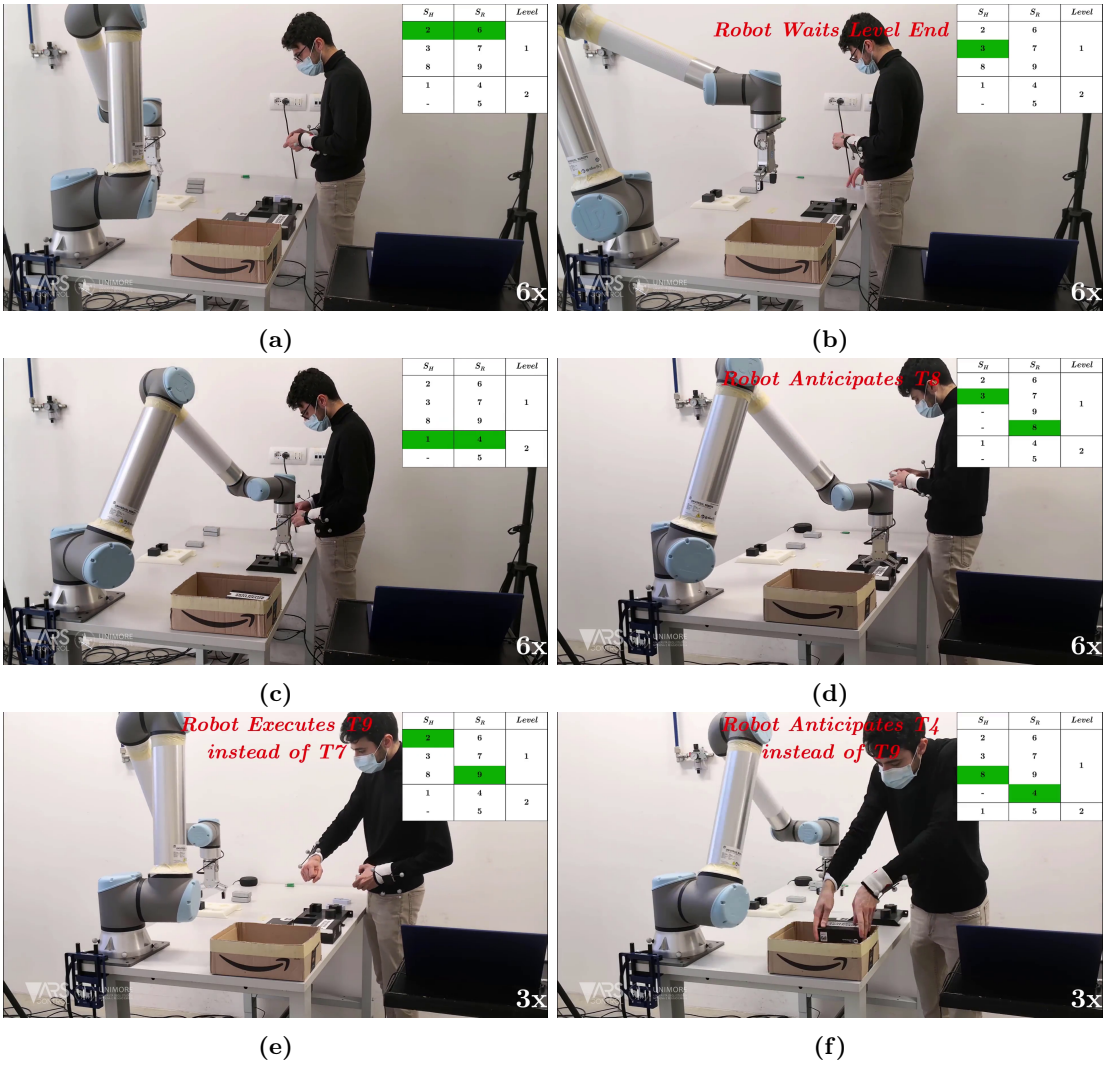


Figure 4.5: Snapshots of the Lab experiments.

Table 4.3: Lab Planning Data Description

Data	Value
a_{max}	$-3.71\ m/s^2$
T_r	$0.002\ s$
C	$0.21\ m$
Z_d	$0.0\ m$
Z_r	$0.01\ m$
$\dot{\mathbf{q}}_{max} = -\dot{\mathbf{q}}_{min}$	$3.14\ rad/s$
$\ddot{\mathbf{q}}_{max} = -\ddot{\mathbf{q}}_{min}$	$6.28\ rad/s^2$

A detailed comparison of the timing of the experiments in the first scenario is reported in Figure 4.6. Figure 4.6a shows the ideal optimal nominal schedule computed by the task assignment block exploiting the nominal durations, i.e. the schedule that would be obtained if there were no deviations from the nominal behavior. In Figure 4.6b, instead, the real schedule obtained with the baseline architecture is reported. Initially, both the human operator and the robot start to follow the ideal nominal schedules, as shown in Figure 4.5a. However, according to the proposed scenario, the human operator is slow in executing the assigned tasks, and in the meanwhile the robot is able to conclude the first three tasks, i.e. tasks T_6, T_7, T_9. Since no rescheduling procedure is allowed, the robot stops and waits for the human to conclude all the tasks in first level of the schedule. This is illustrated in Figure 4.5b. After the completion of task T_8, the schedule passes to the second level and two agents start again to execute their tasks, resuming the expected behavior, as illustrated in Figure 4.5c. The total duration of the schedule obtained with the baseline architecture is $t_{tot} = 110s$.

The proposed architecture, instead, relies on the rescheduling strategy discuss in Section 4.2.3 to dynamically adapt the schedules, compensating for the deviations from the nominal behaviors and improving the performances of the HRC. The resulting schedule is shown in Figure 4.6c. Unlike what happened with the baseline architecture, during the execution the dynamic scheduler exploits both the OE-DTW algorithm to compute the variation of the cost function G in (2.3) caused by the human delay and the skeleton tracking information to update the tuple of the penalty costs Γ_R (see Algorithm 6). After the robot has conclude all its tasks in the first level, the delay introduced by the human operator would cause a drop of the performances, i.e. the parallelism criterion in the first level is not achieved. For this reason the dynamic scheduler calculates that is more convenient to reschedule T_8 to the robot, helping the human and making the collaboration more efficient. This is illustrated in Figure 4.5d. At this point, the robot completes T_8 and both agents are able to conclude the remaining schedules. It is worth noting that in the graph the final schedule is composed by only one level. This is because, according to Algorithm 6, the dynamic scheduler anticipated all the tasks to the first level, making the others empty. The total duration of the schedule obtained with the proposed architecture is $t_{tot} = 100s$. Comparing the two graph it is possible to see that the proposed framework allows to reduce the total execution time by:

$$KPI = \frac{T_{BL} - T_{PR}}{T_{PR}} = \frac{110 - 100}{100} = 9\% \tag{4.3}$$

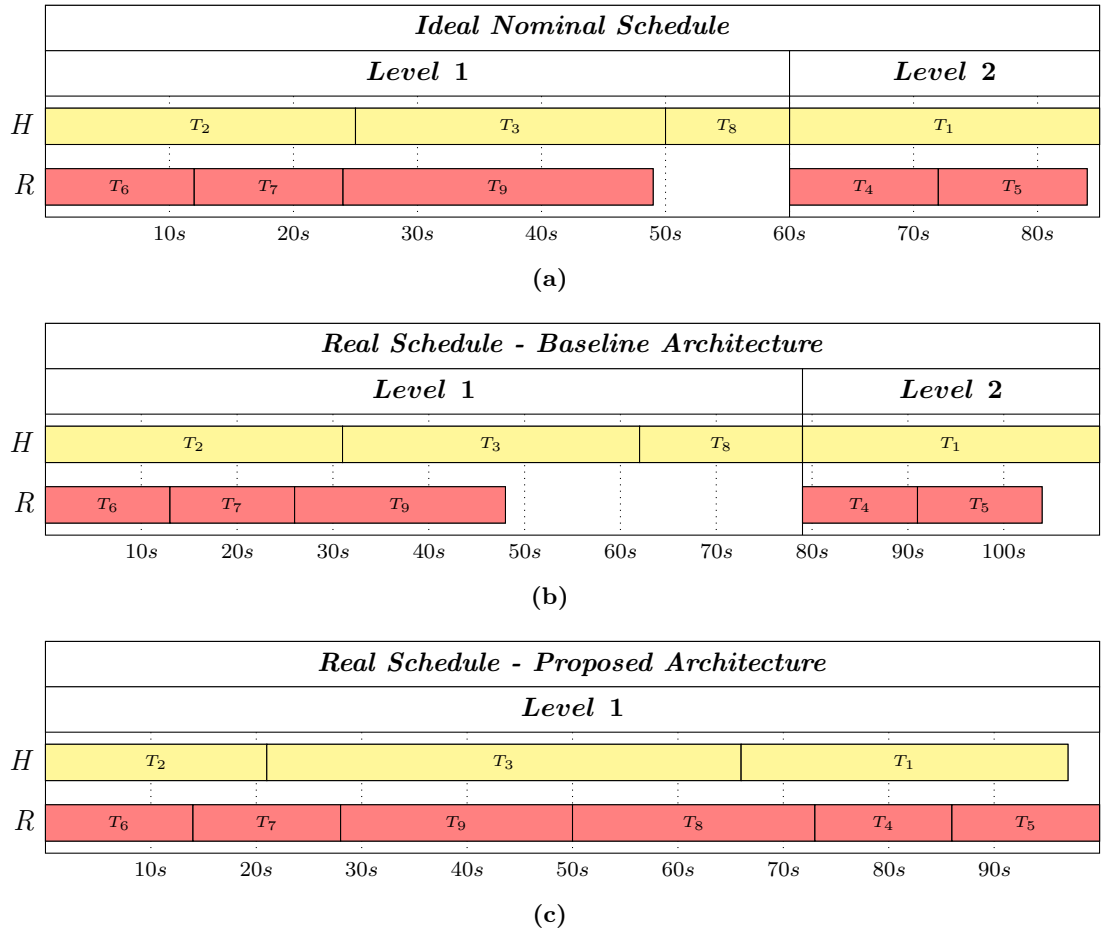


Figure 4.6: Gantt Chart of the schedules for the first scenario.

In addition to the total time, a great improvement with the proposed architecture can be seen in the robot behavior. With the presented dynamic rescheduling strategy, the robot is able to execute a task instead of the human, helping in the job. With the baseline architecture, instead, the robot waits until the entire level is concluded, without compensating any human delay. Furthermore, the baseline architecture is not capable of assigning a new task if the human obstructs the robot, introducing useless waiting time, i.e. the planner continues to search for a feasible trajectory. This is shown in Figures 4.5e and 4.5f.

4.3.2 Datalogic Scenario

The first integration test of the entire architecture has been carried out in a human-robot collaboration scenario in Datalogic that has been set up for experimental purpose. The robot used is a UR10, that is velocity controlled at $f = 250\ Hz$. A complete setup of the experiments is shown in Figure 4.7, where the safety cameras developed by Datalogic can be distinguished. These cameras works at $f = 24\ Hz$ and returns both the voxels maps and the closest human operator and robot points. These points are then exploited to compute the robot-human direction required by the flexible execution layer.

During the experiments, the human operator and the robot have to fill two different boxes with fruit juices, in one box and in the other. A detailed description of the tasks

(a) (b)

Figure 4.7: Setup of the Datalogic experiments.

Table 4.4: Datalogic Tasks Description

Task Index	Description
1,2,3,4	First box: Pick & Place of the juice.
5,6,7,8	Second box: Pick & Place of the juice.

Table 4.5: Datalogic Task Assignment Data

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	$i \rightarrow j$
1	0.6	10	0.1	15	-
2	0.6	10	0.1	15	-
3	0.6	10	0.1	15	-
4	0.6	10	0.1	15	-
5	0.6	10	0.1	15	1,2,3,4
6	0.6	10	0.1	15	1,2,3,4
7	0.6	10	0.1	15	1,2,3,4
8	0.6	10	0.1	15	1,2,3,4

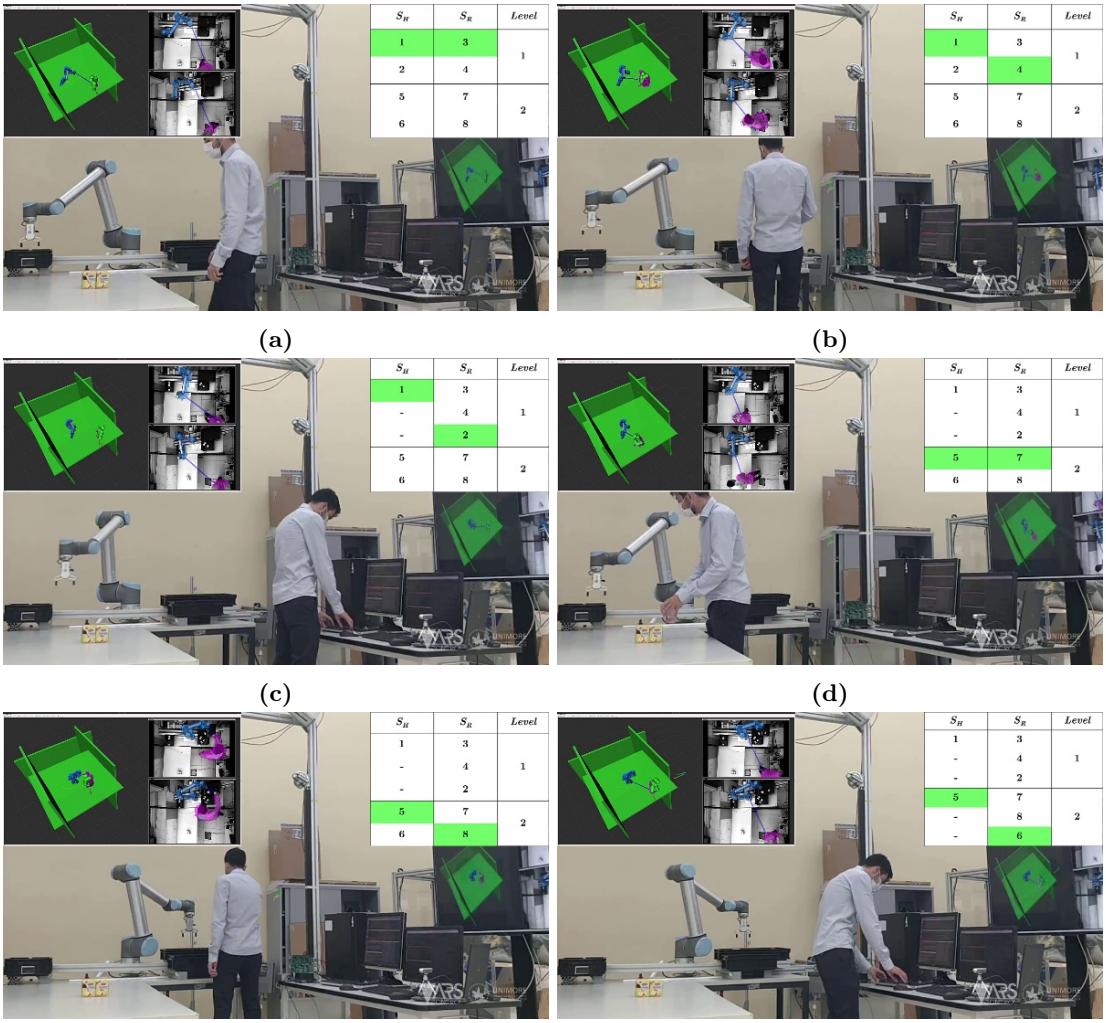


Figure 4.8: Snapshots of the Datalogic experiments.

can be found in Table 4.4. Moreover, the first box must be filled before the two agents can start to put the fruit juices inside the second box. All the data required by the task assignment are listed in Table 4.5.

The schedule resulting from the optimization problem is the following:

- $S_H = \{[T_1, T_2], [T_5, T_6]\}$
- $S_R = \{[T_3, T_4], [T_7, T_8]\}$

Starting from the output of the task assignment layer, the dynamic scheduler was then initialized and the two agents began to perform the collaborative job. The workflow of the experiments is shown in Figure 4.8. Initially the two agents start to execute the nominal schedule, i.e. the two agents perform exactly the assigned tasks, as shown in Figures 4.8a – 4.8b. However, during the execution of T_1 the human operator is late. For this reason, the dynamic scheduler reschedules T_2, assigning it to the robot, as it can be seen in Figure 4.8c. At this point the two agents start to fill the second box, moving the schedule to the second level. This is illustrated in Figures 4.8d – 4.8e. Again, during the execution of T_5 the human operator is late and T_6 is executed by the robot, as shown in Figure 4.8f. The resulting schedule is shown in Figure 4.9a.

In order to demonstrate the effecctiveness of the framework, the same experiment has

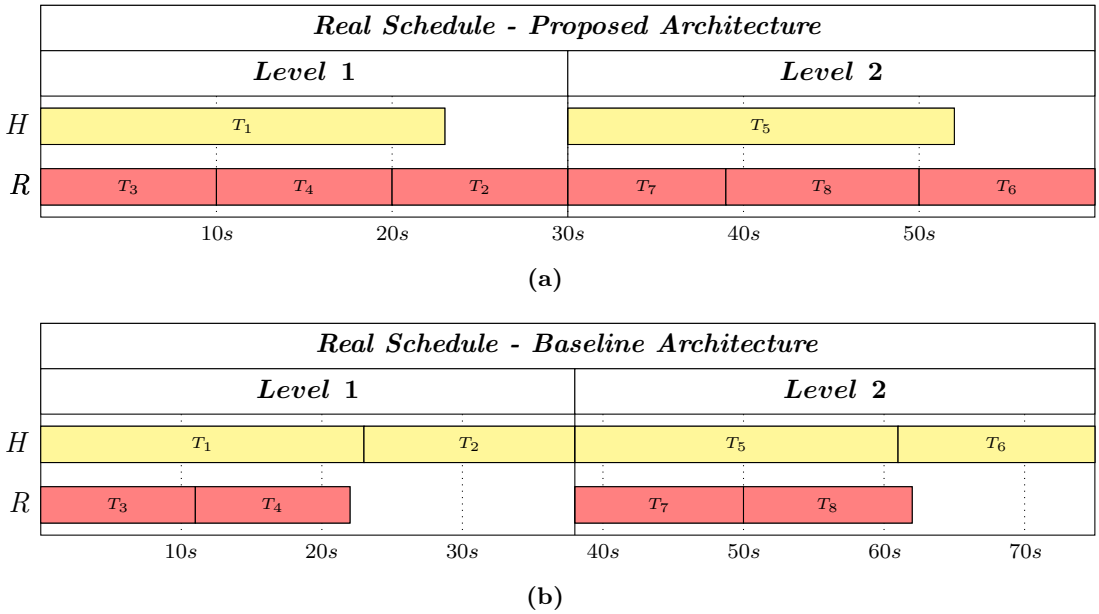


Figure 4.9: Gantt Chart of the Datalogic experiments.

been performed with a baseline scheduling architecture. With this architecture, the rescheduling procedure is not allowed. As before, the two agents start to execute the nominal schedule and the human operator is slow. With this baseline architecture, when the robot concludes all the tasks in the level it waits until the human operator executes its own tasks. This lead to a consequent reduction of the performances. The resulting schedule without the rescheduling procedure is shown in Figure 4.9b.

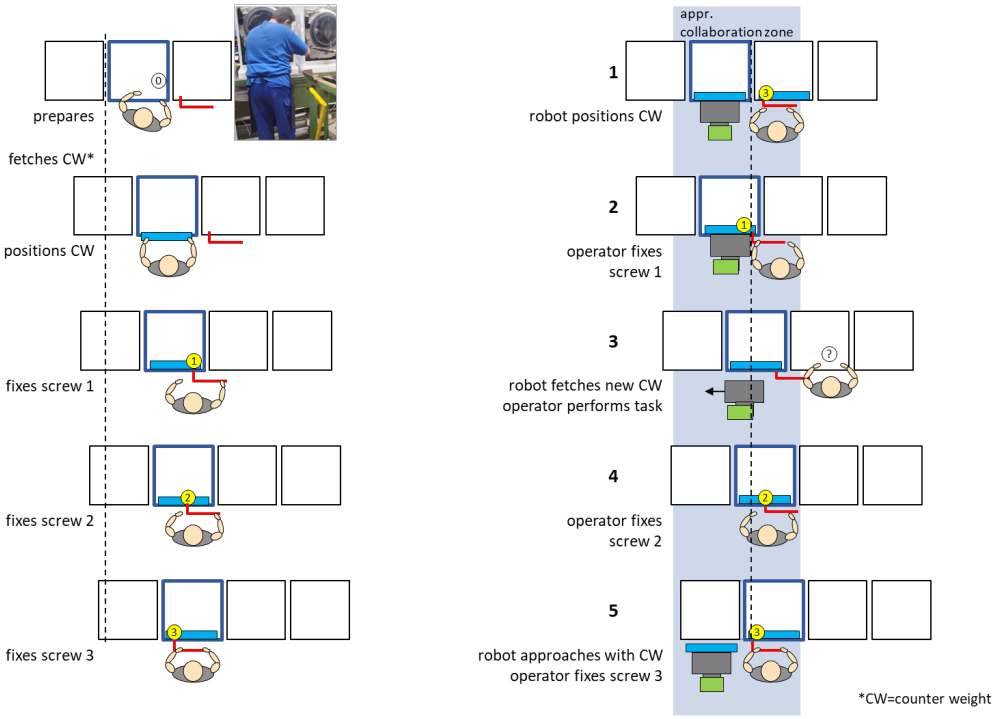
Comparing the two graph it is possible to see that the proposed framework allows to reduce the total execution time by:

$$KPI = \frac{T_{BL} - T_{PR}}{T_{PR}} = \frac{75 - 60}{60} = 25\% \tag{4.4}$$

4.3.3 Whirlpool Use Case

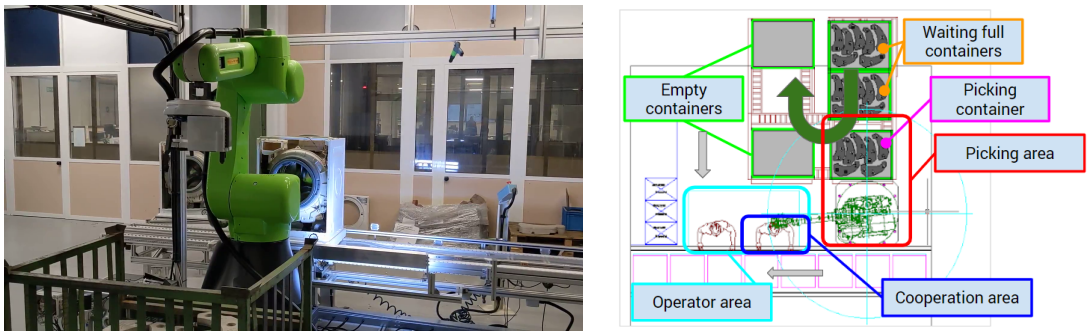
The Whirlpool use case consists of the assembly of a washing machine on a production line. Currently, this operation is done manually. The washing machine is placed over a conveyor belt and, while it is moving, the human operator pick and place the required counterweight inside the washing machine. Subsequently, while holding the counterweight, the human operator screw and fix it inside the washing machine. This is illustrated in Figure 4.10a.

This operation however is very exhausting for the human operator and, due to the high weight of the counterweight, it may lead to back problems. The goal of the ROSSINI project is to partially automatise this procedure, creating a collaborative scenario. The situation is illustrated in Figure 4.10b. In particular, a human operator and a Fanuc CR35-iA must work together in order to insert the counterweight inside the washing machine. The robot is equipped with a custom gripper developed within the ROSSINI project by VINTIV. The gripper is enriched with both an industrial 2D camera, which is exploited to estimate the counterweight pose, and a safety skin developed by PILZ,



(a) **(b)**

Figure 4.10: Working cycle of the Whirlpool use case.



(a) **(b)**

Figure 4.11: Setup of the Whirlpool use case.

which detects if some contact occurs. Since no velocity interface is available, the robot is position controlled at $T_r = 100\ ms$ integrating the desired velocities computed by the flexible execution layer. A photo of the setup is shown in Figure 4.11a. The main challenge of this use case is to ensure a safe collaboration while the robot is carrying such a heavy object. Moreover, the precision required during both the picking and the insertion phase is very high. For use case requirements, the workspace has been divided in three different zones: red, where only the robot can work , blue, where the collaboration is enabled, and cyan, where only the human operator can work and the robot is not allowed to enter. This is illustrated in Figure 4.11b.

The detailed list of all the tasks composing the job can be found in Table 4.6, while the data required by the task assignment are listed in Table 4.7. It is worth noting that both the costs and the times are already balanced; some tasks have a very high cost for one agent over another. This is because, due to the characteristics of the use case, the

Table 4.6: Whirlpool Tasks Description

Task Index	Description
1	Wait new washing machine.
2	Pick the counterweight.
3	Track the washing machine.
4	Insert the counterweight.
5	Screw the counterweight.
6	Release the counterweight.

Table 4.7: Whirlpool Task Assignment Data

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	$i \rightarrow j$
1	0.1	20	999	999	-
2	0.1	20	999	999	1
3	0.1	20	999	999	2
4	0.1	20	999	999	3
5	999	999	0.1	20	5
6	0.1	20	999	999	6

tasks allocation has been chosen a priori. Moreover, it has been chosen not to allow any rescheduling.

When the collaboration starts, the task assignment computes the following nominal schedules:

- $S_H = \{T_5\}$
- $S_R = \{T_1, T_2, T_3, T_4, T_6\}$

At this point, the dynamic scheduler dynamically allocate the two tasks to each agent. Since no rescheduling is allowed, the dynamic scheduler simply forwards the tasks synchronizing the execution. A complete and detailed video of the use case can be found on the ROSSINI YouTube channel[3]. Furthermore a series of snapshots of the working cycle is shown in Figure 4.12. When the job starts, the robot waits for the washing machine to be available. In this phase, if it is not in the home position, it goes over the counterweight container, see Figure 4.12a. When it reaches the initial configuration and the assembly line gives the signal of a new washing machine, the robot can start the execution of T_2. In this task, illustrated in Figure 4.12b, it exploits the 2D camera to acquire the pose of the counterweight and picks the counterweight. Subsequently, it approaches the assembly line, going outside the red area and entering in the collaborative zone. Once it reaches the washing machine, it starts the execution of T_3, which consists in tracking the washing machine again exploiting the 2D camera, see Figure 4.12c. Subsequently, it aligns with the washing machine and starts to insert the counterweight, i.e. execution of T_4. This is shown in Figure 4.12d. At this point the real collaboration starts:

[3] `https://www.youtube.com/watch?v=B2NanPArV6A`

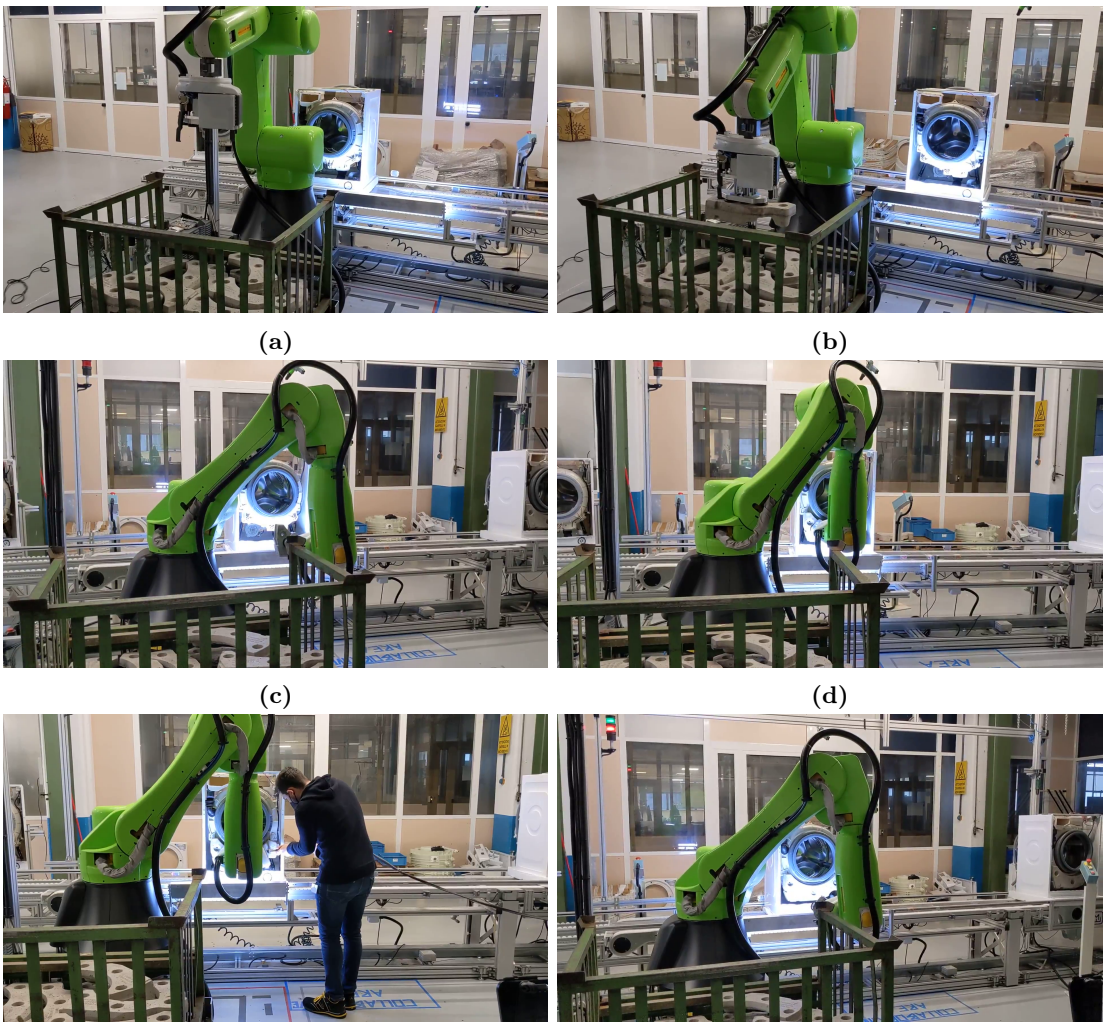


Figure 4.12: Snapshots of the Whirlpool use case.

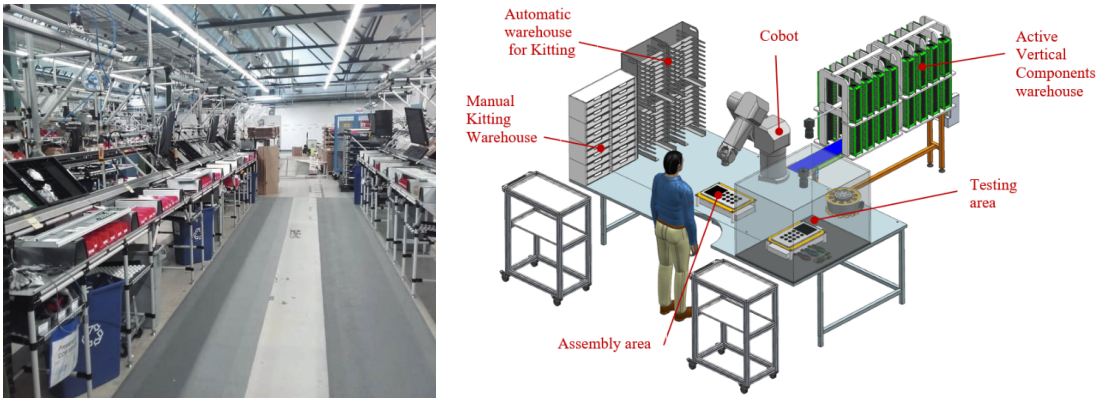
the human operator screws the counterweight while the robot holds it firmly inside the washing machine. This is the execution of T_5, illustrated in Figure 4.12e. After the screwing is completed, the human operator press the button over the gripper and the robot release the counterweight going back in the home position, ready to start a new job, see Figure 4.12f.

As already mentioned all the trajectories are computed through the flexible execution layer, which is also responsible of adapting the speed of the robot if necessary. This is demonstrate in Figure 4.13, where the robot is stopped during the execution of T_6 in order to be compliant with the safety standards.

Schindler Use Case

The Schindler use case consists of the assembly of an elevator panel. Currently, this operation is done entirely by human operators who are guided step by step by the instructions on the screen. The assembly is made in two macro steps: the first one consists in assembling the bottom part of the panel, composed by the buttons, while the second one consists in assembling the top part of the panel, composed by the display. The actual

Figure 4.13: Robot stopped during the execution of T_6.



(a) **(b)**

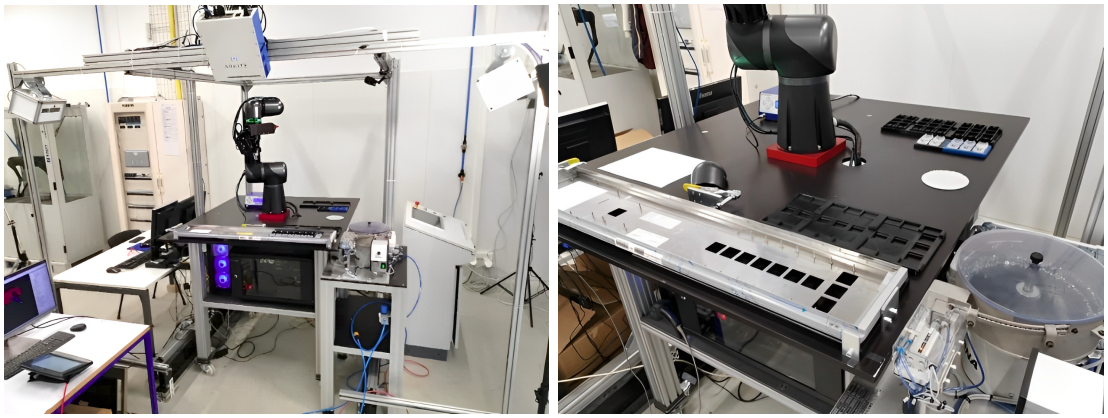
Figure 4.14: Working cycle of the Schindler use case.

station is illustrated in Figure 4.14a.

Schindler wants to entirely or partially automate this process, by exploiting the robot capabilities. Since the assembly of the display requires high manipulation skills, e.g. manipulation of cables, and there are also difficulties of electrostatic nature, it is not possible to automate the entire process. For this reason, the ROSSINI consortium decided to focus on the assembly of the bottom part of the panel, i.e. pick and place of the buttons, creating a collaborative station as shown in Figure 4.14b. In particular, the human operator has to collaborate with a new collaborative robot entirely built inside the ROSSINI project by the partners SUPSI. This robot is a 6-axis robot with a payload of 13.5 Kg and a reach of 1.0 m without a velocity interface. For this reason, the desired velocities computed by the flexible execution layer are integrated and the robot is position controlled at $T_r = 20\ ms$. The robot is also equipped with a Robotiq 2-finger gripper and a 2D camera, exploited to detect the number or symbol of the buttons. A photo of the setup is shown in Figure 4.15a.

The detailed list of all the tasks composing the job can be found in Table 4.8, while the data required by the task assignment are listed in Table 4.9. It is worth to underlie that even if the human operator is capable of executing the assembly of the buttons, it has been chosen an very high cost. The reason behind this choice is that, as already explained, Schindler requested for the automation of the assembly of the buttons. For this reason, changes in the task allocation, i.e. the rescheduling, are not allowed.

When the collaboration starts, the task assignment computes the following nominal schedules:



(a) **(b)**

Figure 4.15: Setup of the Schindler use case.

Table 4.8: Schindler Tasks Description

Task Index	Description
1	Pick and place of the panel board.
2	Scan of the barcode.
3	Order the buttons.
4	Fix the buttons to the frames.
5	Pick and place of the frames in the panel.
6	Screw the frames.
7	Pick and place of the display.
8	Wiring of the electronic parts.

Table 4.9: Schindler Task Assignment Data

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	$i \rightarrow j$
1	999	999	0.1	20	-
2	0.1	20	999	999	1
3	0.1	20	999	999	2
4	0.1	20	999	999	3
5	0.1	20	999	999	4
6	0.1	20	999	999	5
7	999	999	0.1	20	1
8	999	999	0.1	20	6

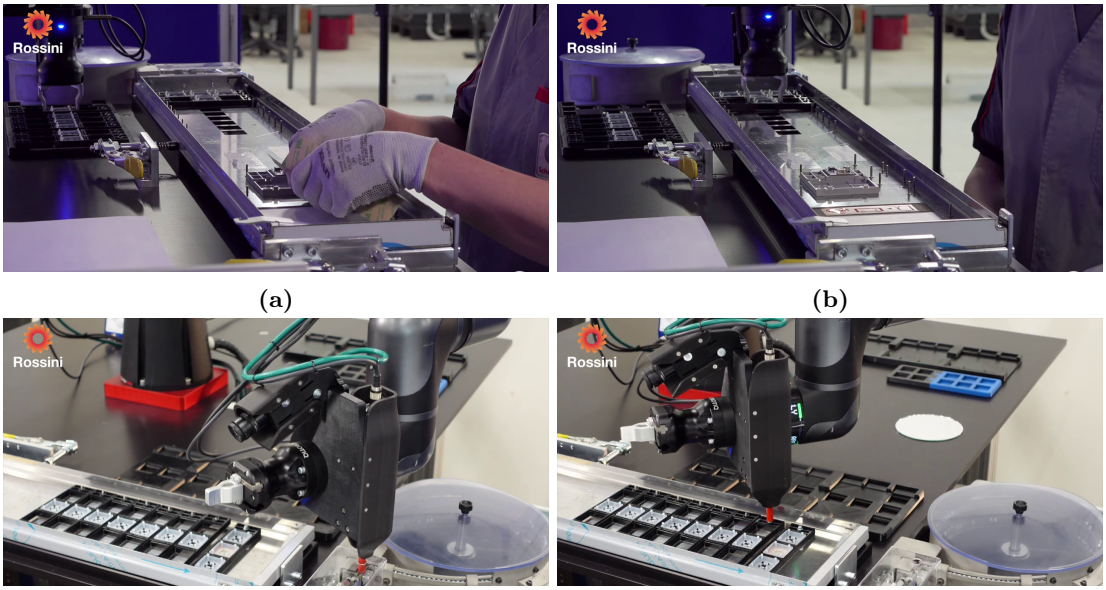


Figure 4.16: Snapshots of the Schindler use case.

- $S_H = \{T_1, T_7, T_8\}$
- $S_R = \{T_2, T_3, T_4, T_5, T_6\}$

At this point, the dynamic scheduler starts to allocate the tasks to each agent, synchronizing the tasks execution. A complete and detailed video of the use case can be found on the ROSSINI Youtube channel[4]. When the job starts, the robot waits for the human operator to place the elevator panel over the table, i.e. T_1. Once tha panel is placed, the human operator and the robot starts to collaborate, each one working on its own task. Focusing on the robot, it firstly scan the panel to detect the model and the number of the buttons, which is the execution of T_2. Then, it starts performing T_3. In this task, it picks the buttons from the container[5], pass them over a mirror, exploit the 2D camera to detect the number or symbol of the buttons, and finally place them on the correct position inside the casing. After correctly placing all the buttons on the casing, the robot picks and places the frames over the buttons, i.e. T_4. When all the frames are placed correctly, the robot can start the execution of T_5, which is the pick and place of the frames the panel. This is illustrated in Figures 4.16a – 4.16b. Lastly, the robot execute T_6 which is the screwing of all the frames, fixing them to the panel, as shown in Figures 4.16c – 4.16d. During these operations, the human operator executes the pick and place the display in the panel, which is T_1, and, after the execution of T_6, proceeds with the wiring of all the components.

During the overall collaboration, the safety layer is active and the flexible execution layer scales the robot velocity based on the minimum human-robot distance. This is shown in Figure 4.17.

[4] `https://www.youtube.com/watch?v=eXB4mfTmV-k`

[5] For simplicity, the buttons are placed inside a 3D-printed container, but their order is unknown. In the final case, this will be replaced by a dispenser that will release the buttons in the same position in a random order.

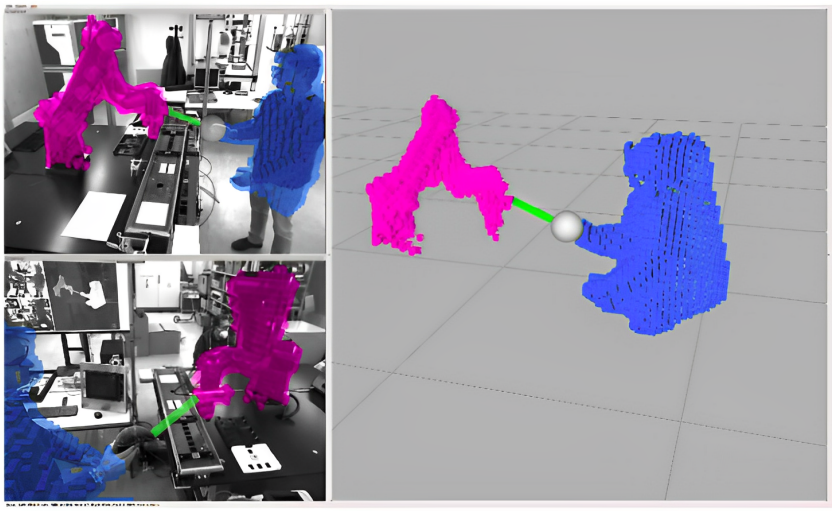


Figure 4.17: Monitoring of the human operator.

Table 4.10: IMA Tasks Description

Task Index	Description
1	Change gripper fingers.
2	Load reel.
3	Fix reel to the machine.

4.3.4 IMA Use Case

The IMA use case consists of of the autonomous replacement of raw-material reels to an automatic machine. Currently, the operation is performed entirely manually by the operator. The new reel is taken from the warehouse, loaded onto the machine and then fixed.

However, very often the operator is busy performing other tasks and is not always able to change the reel promptly. This leads to unwanted machine downtime. Furthermore, due to the high weight of the reel, this loading and unloading operation may cause back problems for the operator. The goal of the ROSSINI project is to partially automatise this operation, creating a flexible collaborative scenario. In particular, a mobile collaborative robot composed by a MIR500 and two UR10e work together in a large workshop. The robot is in charge of changing the reel, while the human operator is responsible of fixing it. The robot is equipped with a 2D camera in order to detect the real position of the reels and the machine with respect to the robot. Moreover, the two manipulators are equipped with two different grippers: a Robotiq 3F, used to remove the empty reel from the machine, and a Zimmer pneumatic gripper, exploited for loading the reel. In particular, the fingers of the Zimmer grip must be changed based on the reel dimensions. The setup is shown in Figure 4.18.

The detailed list of all the tasks composing the job can be found in Table 4.10, while the data required by the task assignment are listed in Table 4.11.

When the collaboration starts, the task assignment computes the following nominal

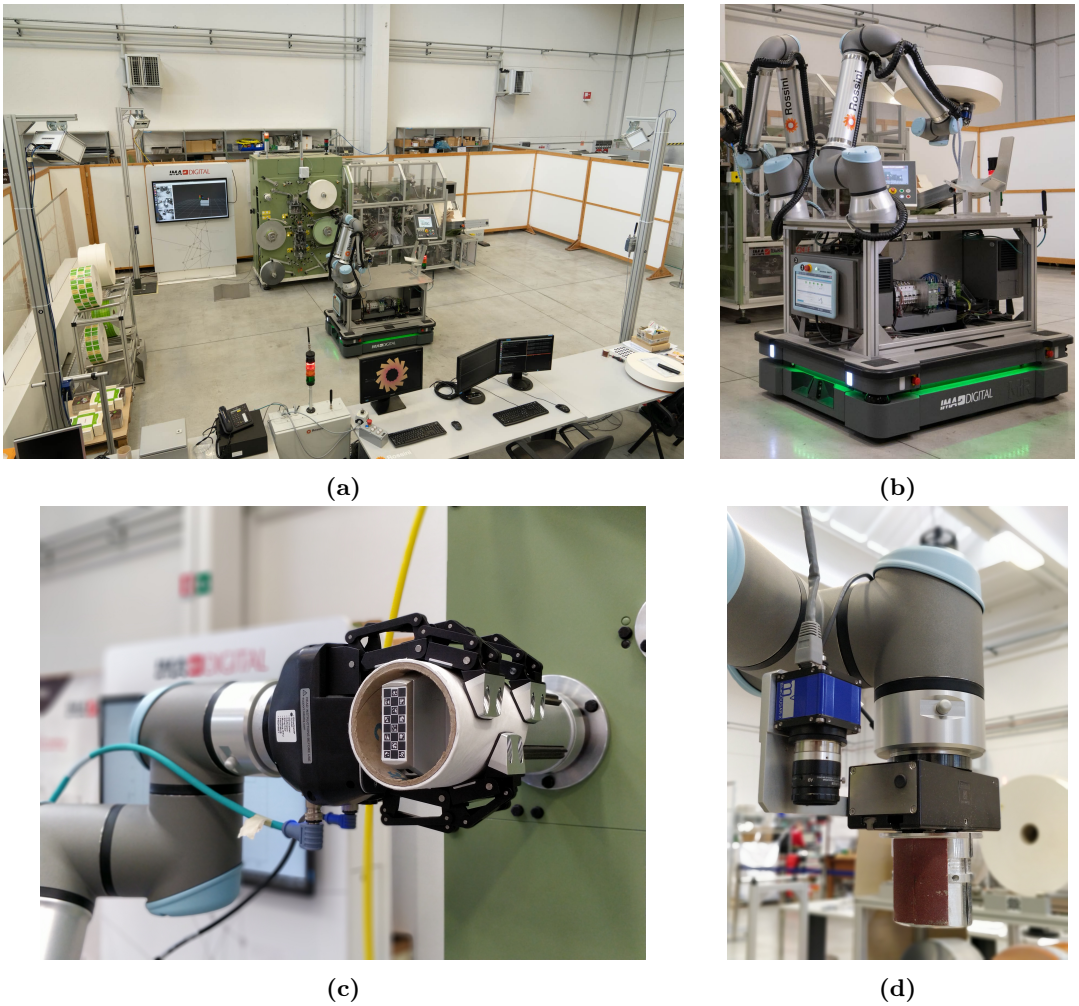


Figure 4.18: Setup of the IMA use case.

Table 4.11: IMA Task Assignment Data

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	$i \rightarrow j$
1	999	999	0.1	20	-
2	0.1	20	999	999	1
3	999	999	0.1	20	2

schedules:

- $S_H = \{T_1, T_3\}$
- $S_R = \{T_2\}$

At this point, the dynamic scheduler sends the first task and the two agents start to perform the job. A complete and detailed video of the use case can be found on the ROSSINI YouTube channel[6], while a series of snapshots is illustrated in Figure 4.19.

Initially, the human operator checks and, if necessary, changes the fingers of the Zimmer gripper, while the robot waits as T_1 is a priority. This situation is shown in Figure 4.19a. Once the change of the fingers is complete, the operator confirms that it has been executed and the dynamic scheduler assigns the reel load to the robot, which starts to approach

[6] https://www.youtube.com/watch?v=nCTmw5rmxcc

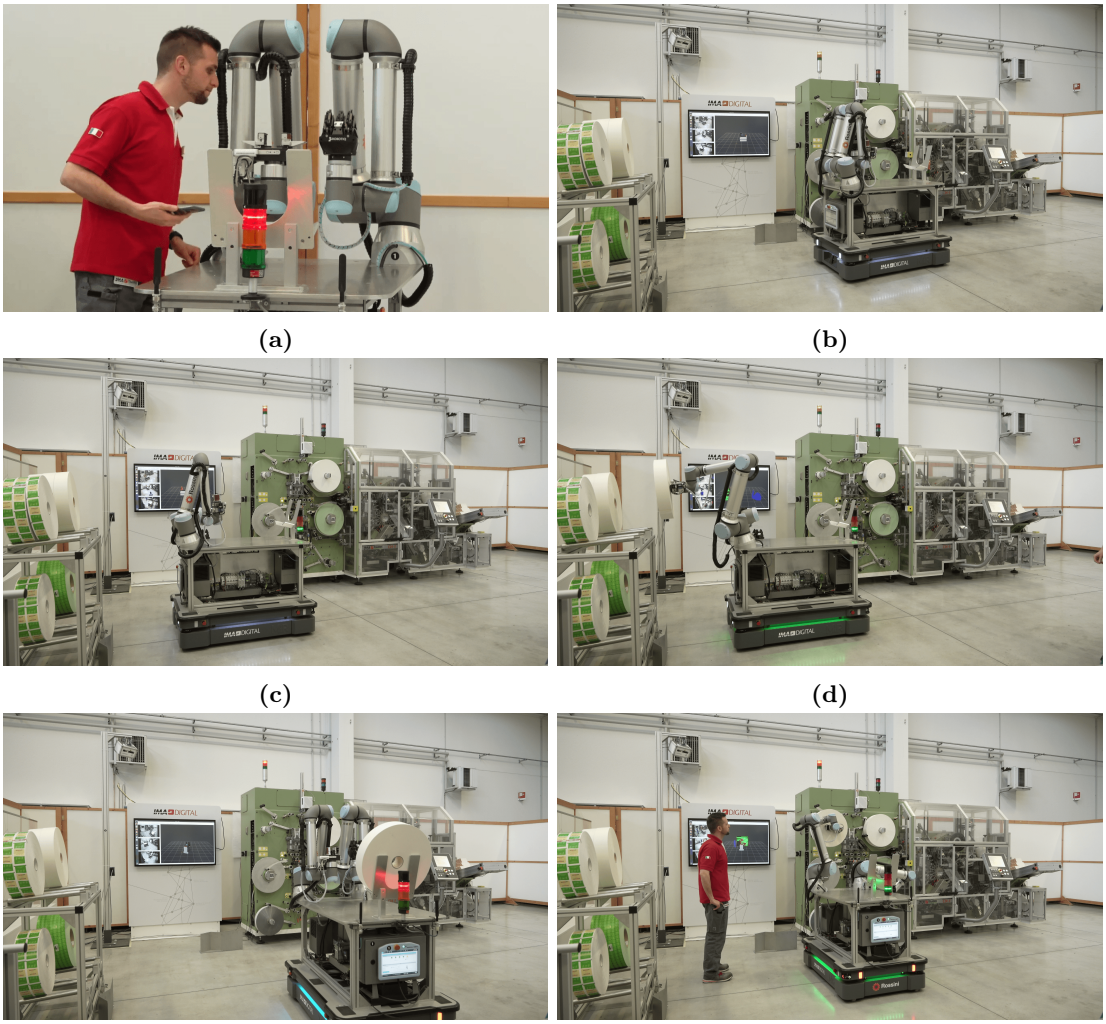


Figure 4.19: Snapshots of the IMA use case.

the warehouse. This can be seen in Figure 4.19b.

Upon reaching the warehouse, the mobile base stops and the manipulator performs the

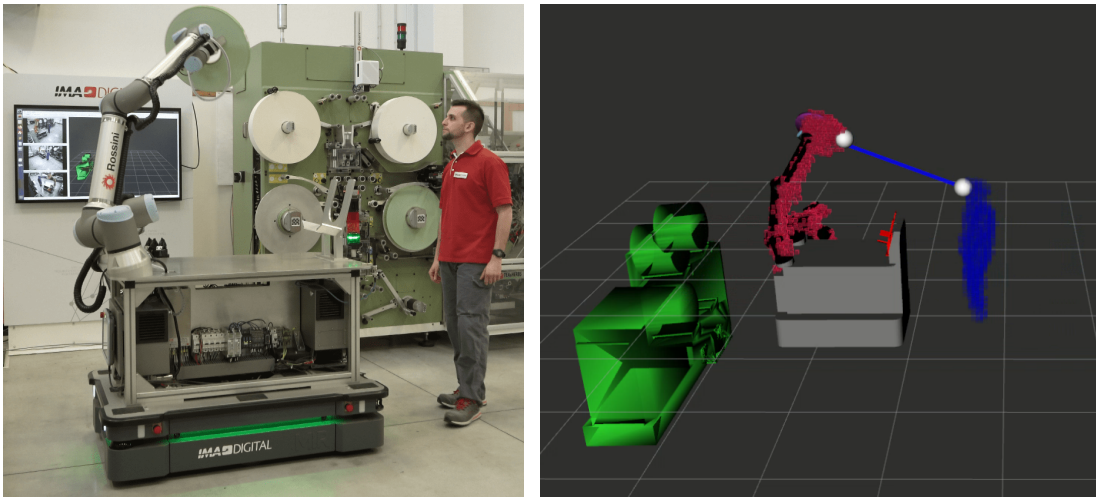


Figure 4.20: Voxel map representation.

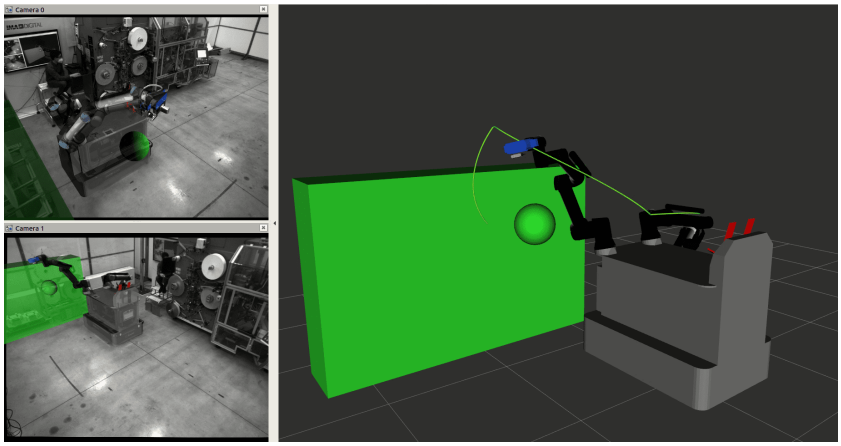


Figure 4.21: Dynamic replanning.

pick of the reel, shown in Figures 4.19c – 4.19d. In order to cope with any discrepancies between the theoretical and actual position of the warehouse, in this phase the 2D camera is exploited to identify the position of the reel with respect to the robot.

Once the pick is complete, the moving platform moves to the machine and proceeds with the loading of the reel, Figures 4.19e – 4.19f. Once the reel is loaded, the task is performed and the fastening request is sent to the operator, who proceeds to execute it.

During the experiments, the safety system is always active and it monitors the overall workspace in order to both avoid collisions and to calculate the maximum robot speed. In particular, Figure 4.20 shows the voxel maps of both the human operator and the robot and the two closest points. Figure 4.21, instead, shows the replanning of the trajectory in case an obstacles hinders the path.

Chapter 5

A Resilient and Effective Task Scheduling Approach For Industrial Human-Robot Collaboration

The cognitive layer presented in Chpater 2 is capable of adapting at runtime the schedule of the two agents based on the actual execution times and can delegate some tasks to the human operator if a failure occurs. However, to achieve a more natural and realistic collaboration, the scheduling strategy should be able to recover from errors when possible. Furthermore, it should be also able to adapt to the human operator. This chapter presents a resilient and effective task scheduling framework that addressed these issues, improving the collaboration. The experimental validation shows that the proposed framework is suitable for real industrial scenarios.

The work presented in this chapter is published in [3, 9].

5.1 Introduction

Industrial applications where human and robot work closely together are becoming the new paradigm of industrial settings. Collaborative robots can take over repetitive, challenging or dangerous tasks improving the well-being of the operators. There are multiple challenges that emerge from this close collaboration. On one hand, the absence of barriers makes it necessary to pay close attention to how to guarantee the safety of the operator [82, 83]. On the other hand, it becomes necessary to understand how to create a synergy between humans and robot that is as natural as possible, making the most out of the collaboration [84]. Therefore, a strategy on how to allocate and schedule tasks between human and robot is crucial to improve the human-robot team.

The basis of this strategy is an investigation of how the different tasks that make up the work can be distributed best among the actors in the nominal situation, the task allocation problem. The individual tasks are subject to constraints that prescribe how, when and by whom the tasks can be executed.

The characteristics of a good task allocation are captured in the fluency concept that relates to how well the operator and the robot are adapted to each other. Fluent collaboration benefits the work execution and the job-quality of the human operator. These aspects are not captured by optimizing for task efficiency alone. There are subjective and objective metrics for fluency available of which the latter ones can directly be used as an optimization criterion. The objective metrics include the relative portion of functional and non-functional delays of the actors, and the amount of parallel work [30].

The task allocation problem can be solved during the design phase and results in the nominal schedule. This procedure has widely been investigated in other works. [85–88] focused on heterogeneous multi-agent task allocation in industrial setting. While other authors, e.g. [34–36], propose to model the human-robot collaboration (HRC) problem as a nonlinear optimization problem. These strategies allow to find the best nominal schedule.

Even if optimal, a nominal task schedule cannot guarantee a real improvement of the collaboration. In the task execution phase, many factors come into play that cannot be anticipated within the nominal schedule. This requires an efficient and resilient team that can anticipate and adequately respond to these abnormalities. The requirements for an efficient team of human and automated agents, i.e., robots have been formulated in [89] and contributed to a design method in [90]. The requirements and the design method were targeted at general automation challenges and applied in open ended scenarios. The industrial practice is much more constrained and the irregularities that exist have several common causes. Identifying these causes can help the design of resilient solutions for task allocation.

Firstly, it may happen that not all the actors are always available to carry out the collaboration, e.g., the robot has been dispatched to another work station. In this case it would be highly inefficient to design again the job considering only the remaining actors. It would be more convenient to consider this actor availability problem from

the beginning and adapt the schedule accordingly. Secondly, the tasks may depend on each other, i.e., there could be precedence constraints, and this interdependence must be considered while ensuring the parallelism between the actors. Thirdly, the operators, as human beings, are inherently different from each other, each with their own skills, capabilities, or individual preferences. Some of them may need or prefer to be guided more during the work, e.g., newly hired workers. For others, on the other hand, an excess of information could be annoying and counterproductive from the job quality perspective. Lastly, there is always the possibility that failures will occur that prevent the correct execution. Therefore, a fallback scenario must be considered when designing a scheduling strategy for HRC scenarios.

This chapter firstly presents how an industrial HRC process, with its irregularities during execution time, can be formalized into a set of interdependent tasks. Secondly, it proposes a novel adaptive task scheduling framework for collaborative cells. As a start, the framework uses a database to understand how the collaborative job is composed out of multiple interdependent tasks. Subsequently, at runtime, it monitors the task execution to understand the human operator skills and the task result, i.e., failure or success. This information is exploited to adapt the schedule online, making the framework flexible and able to face most of the situations that may arise in a real HRC scenario.

The main contributions of this chapter are:

- A formulation of four primitive situations that encompass most of the scenarios that could occur in a real HRC.
- A novel adaptive task scheduling framework that is effective and applicable to the formalized situations, i.e., suitable for a real industrial application.
- The validation of the proposed framework in several variants, one for each situation, of the same experimental scenario, proving the effectiveness of the framework.

5.2 Related Works

Different approaches were presented in the literature to deal with the problem of multi-agent task scheduling. In [91] the author proposes a heuristic method in order to allocate and schedule the tasks between multiple processors. The approach is based on the communication time between the processors and the number of successor tasks, making it suitable for the problem of handling precedence constraints. In [92] a branch-and-bound procedure and a climbing discrepancy search heuristic for the parallel machine scheduling problem with precedence constraints and sequence dependent setup times is proposed. This algorithm can minimize the sum of the completion time and maximum lateness. In [93] the authors propose a Load-Balance Scheduling Algorithm, which allows to allocate and schedule the tasks in a multiprocessor system. The idea is to use an Earliest Deadline First (EDF) heuristic to first create a n ordered tasks list. Then, based on the actual workload, to allocate the task to one processor. In general, these solutions cannot be directly applied in a HRC application, as they consider the presence of homogeneous actors.

In [94] the authors model the HRC working process as a chessboard setting, where the decision of each actor is described by the chess piece move and formulated it as a Markov game model. To optimize it, they propose a decentralized Deep-Q-network based MARL (DQN-MARL) algorithm. In [95] the task scheduling is formulated as a Mixed-Integer Linear Programming Problem (MILP) inspired by the Multimode Multiprocessor Task Scheduling Problem. The cost function aims at reducing the total makespan and the solution is obtained with a constraint programming model and the use of a Genetic Algorithm (GA). In [96], instead, the authors propose the use of a Simulated Annealing (SA) algorithm to find the optimal solution.

These works, however, do not consider the differences between individual operators. As the scheduling procedure adapts to the robot that is available, e.g., considering the robot workspace, it should also be able to modify the schedule based on the human operator that is currently going to perform the collaborative job. In [38] an integrated task allocation and task scheduling strategy for HRC is proposed. The task allocation is solved offline exploiting a two-level feedforward optimization. Furthermore, this strategy is enriched with a feedback procedure based on mutual trust to re-allocate the tasks online. In [39] the authors propose a multi-criteria decision-making framework for task allocation which generates a solution that best matches the criteria you want to optimize. Moreover, in case of unexpected events, the algorithm can be exploited for re-scheduling the remaining tasks. In [97] the authors propose a genetic algorithm that exploits human and robot data, e.g., ergonomics or capabilities, to optimally schedule the tasks in a HRC scenario. The actor characteristics are given offline as input by the user. In [40] a two-level of abstraction and allocation for HRC scenario is presented. The first layer exploits the use of the A* algorithm to optimize a cost function. The second layer, instead, handles the task execution and the respective failures. If the system detects some errors, it is possible to reschedule the tasks recomputing the optimal solution. In Chapter 2 a two-layer dynamic rescheduling framework is presented. The first layer builds the nominal schedule solving offline a MILP problem, while the second layer exploits the real human execution time to reschedule online the tasks. In this work the job quality is considered inside the MILP problem both as data to be optimized and as constraints. This work has been further extended in Chapter 4 to integrate the scheduling strategy with the safety required by the robot trajectory planner.

Even if they adapt online based on what is currently happening, none of the proposed works is so general to handle all the considered primitive situations at the same time:

1. Scarce resources
2. Parallelism between the actors
3. Different human operator skills
4. Errors during the execution

The proposed approach focuses on developing a resilient scheduling framework, that is general and applicable to real industrial scenarios. Differently from other works, e.g.

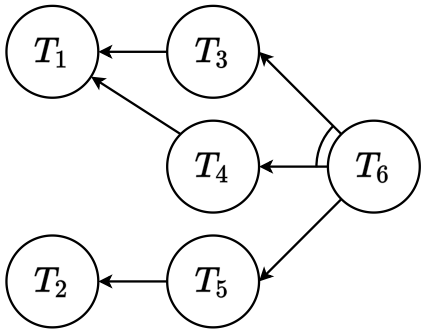


Figure 5.1: AND/OR graph representation.

[98], that are more oriented on the optimization and reduction of the total makespan. The framework does not generate the nominal schedule, it is therefore to a large extend supplementary to existing methods.

5.3 Problem Statement

Industrial Human-Robot Collaboration is characterized by multiple agents that work toward a common goal. In this work, a situation where a human operator H and a robot R collaborate in a shared workspace, namely the collaborative cell, is considered. The human and the robot have a pre-defined task distribution which is defined by a set of nominal task schedules. The agents must perform their respective tasks in order to complete the collaborative job. A job typically represents an industrial process such as the assembly of a product. In this work it is assumed that the nominal task schedules have already been computed, e.g., solving the optimization problem in (2.3) or exploiting other works like [99].

An effective representation of a general collaborative job can be achieved with an AND/OR graph $\mathcal{G} = (T, E)$, as shown in Figure 5.1 [100]. Each node represents the task T_i while each directed edge E_{ij} means that the parent task T_i must be executed after child task T_j. Multiple unlinked edges sharing the same parent represent an OR constraint. This constraint imposes that the parent task can be executed only if at least one of the children has been completed, e.g., T_4 and T_5 in Figure 5.1. Thanks to the OR constraint it is possible to define multiple paths that lead from the same starting point to the same end point, namely *equivalent paths*, e.g., $T_2 + T_5 \equiv T_1 + T_3 + T_4$ in Figure 5.1. Where the unlinked edges represent the OR relations, while the ones connected with an arc represent the AND relations. For example in the Figure T_6 can be executed after both T_3 and T_4 or after only T_5. In this work, the first tasks that belong to equivalent paths are defined as *equivalent tasks*, e.g., T_1 is equivalent to T_2. When multiple edges share the same parent and they are connected with an arc, they model an AND constraint. This constraint imposes that the parent task can be executed only if all the children have been completed, e.g., T_3 and T_4. The job is finished when the final task T_6 is completed, this does not require that all tasks in the job are completed.

Thanks to their structure, the AND/OR graphs intrinsically model both the parallelism between the actors, which is required to reduce the waiting times, and the precedence

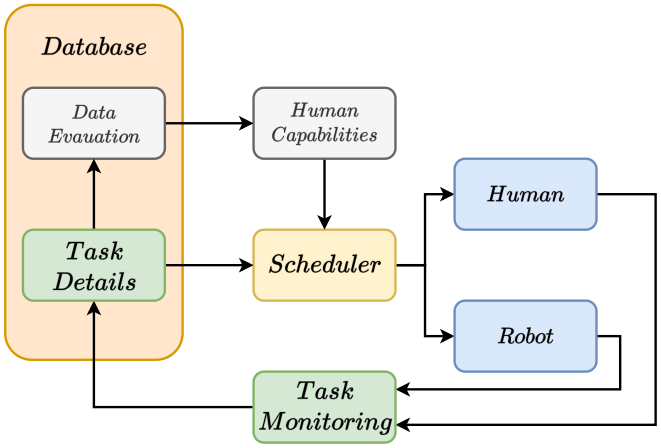


Figure 5.2: The Resilient and Effective Scheduling architecture.

constraints between the tasks. Furthermore, through the OR constraints it is possible to represent already in the design phase all the different situations that may arise due to the available actor capabilities. This might be the temporary absence of an actor such as a robot that is in maintenance, or an inexperienced operator that needs additions task instructions. In this work only human operators are considered to have different skill levels. To handle this, each human task is defined with the minimum expertise level that is required from the operator to be able to perform that task. The robot is assumed to have a fixed set of capabilities that are known during design time, which is common in industrial settings.

Scheduling the task does not ensure that the actor will always perform it correctly. For this reason, each task T_i is associated with recovery actions that allow to restore the nominal behavior of the human-robot team. This set of actions can be very complex and in turn represented with a AND/OR graph. For ease of reading, this set of actions will be represented in the chapter as a single task. Moreover, the collaborative cell is enriched with a task monitoring strategy that allows to check if the task execution has been successful and, in case of failure, it provides information about the error. To achieve this, many algorithms are already available in literature [101, 102].

The goal of this work is to design a scheduling framework that:

- Taking as input the skills of the human operator and its knowledge in the collaborative job, automatically schedules online the tasks between the different actors choosing the most suitable path for the situation on the AND/OR tree.
- Automatically handles most situations and the failures that may occur within an industrial scenario, making it suitable for HRC in industrial settings.

5.4 Architecture

The proposed framework is shown in Figure 5.2, where different components may be distinguished:

- The **Database** block is responsible of storing all the information regarding the tasks composing the collaborative job, through the *Task Details* block, and analyzing all

the data to evaluate the desired metrics, this is achieved in the *Data Evaluation* block.

- The **Scheduler** block, which is the core of the framework, takes care of choosing the most suitable task for each actor online, considering the human operator capabilities and the parallelism in the collaborative job.
- The **Task Monitoring** block oversees the task execution and it communicates to the Database if the task has been completed or not. Moreover, in case of failure, it gives information about the error so that proper recovery actions can be scheduled.
- The **Human Capabilities** block takes the result of the data analysis to estimate the human operator capabilities and knowledge.

It is worth noting that in Figure 5.2 the overall framework has been presented. However, the grey blocks are out of the scope of this work and have been included for completeness.

The overall procedure starts offline with the design of the AND/OR graph and inserting all the tasks inside the database. Each task T_i is associated with its description and requirements, i.e., the actor that must perform it, the precedence constraints as AND/OR constraints and the minimal required expertise level. For example, T_6 in Figure 5.1 can be defined as following:

$$
\begin{aligned}
T_6 : \{&description : \ ''Final'', \\
&requirements : \\
&\qquad \{actor : H, \\
&\qquad precedence : \ ''(T_3 \ \wedge \ T_4) \ \vee \ T_5'', \\
&\qquad level : 1\}\}
\end{aligned}
$$

Subsequently all these task definitions are passed to the scheduler, which searches along the AND/OR tree to choose, for each actor, the tasks that best suit the current scenario, e.g., operator experience, actor availability. This is achieved exploiting the data coming from the others blocks. Once a task has been chosen, it is forwarded to the respective actor who must perform it.

At this point the task monitoring block continuously checks which tasks have been concluded and what the final task result is, i.e. success or failure. This information is also stored inside the database and used for two different purpose. Firstly, in case of error, it is exploited to choose a proper set of recovery tasks that may performed in order to continue the collaboration. Secondly, it used by the data evaluation strategy in order to increase or reduce the human operator expertise level. Finally, the scheduler is triggered again to assign another task, until the collaborative job is concluded.

5.5 Scheduler

The scheduler is the core of the proposed framework and has the goal of distributing the tasks among the available actors, i.e., humans and robots. It requires as input both the AND/OR graph of the collaborative job, already defined in a previous design phase, and

the availability of actors along with their skills and capabilities. These two inputs are exploited to choose at runtime the best path for current scenario, tailoring the specific needs of the human operator during the collaboration. Thus, the scheduler aims at improving the synergy between the actors with a consequent improvement of the HRC.

The entire scheduling pipeline is implemented according to the pseudo-code reported in Algorithm 7. The scheduler needs as input the AND/OR graph $\mathcal{G}$ coming from the

Algorithm 7 Scheduler()

```
1: Require: G, A, H_L
2: End_J ← false
3: for a ∈ A do
4:     Free_a ← true
5:     F_A ← pushback(Free_a)
6: end for
7: while End_J = false do
8:     for T ∈ G do
9:         if checkRequirements(T, F_A) then
10:            if checkLevel(T, G, H_L) then
11:                F_A ← setBusy(T, F_A)
12:                T_A ← pushback(T)
13:                G ← discardEqTasks(T)
14:            end if
15:        end if
16:    end for
17:    for T ∈ T_A do
18:        R, E ← taskMonitor(T)
19:        if R = ∅ then
20:            continue
21:        else if R = Executed then
22:            if isFinal(T) then End_J ← true
23:            end if
24:        else if R = Failed then
25:            G ← applyRecovery(T, E, G)
26:        end if
27:        T_A ← remove(T, T_A)
28:        F_A ← setFree(T, F_A)
29:    end for
30: end while
```

database, the set of available actors $\mathcal{A}$, and the actual human operator expertise level H_L (Line 1). It immediately sets to false the variable End_J, which is used to identify when the collaborative job is concluded and instantiate to true the list of Boolean variables $\mathcal{F}_\mathcal{A}$, which indicates if each actor is free in order to start a new task (Lines 2 - 5).

Then, the algorithm enters a while loop where it continuously executes the overall pipeline until the collaborative job has been finished. Firstly, the scheduler checks for each task T if the requirements have been satisfied (Line 9). This is translated in checking both if the respective agent is available to accept a new task and if the precedence tasks have been executed. If this is the case, the algorithm checks if the level of the human operator is sufficient to execute the task (Line 10). This condition allows the scheduler

to handle the *"Different Operator Skills"* situation detailed in Section It is worth noting that if the actor of the task T is not the human operator, the function **checkLevel**() always returns true. This is because the robot has always the required skills to perform the tasks assigned to it. If the level check is also successful, the task T is scheduled to the respective actor, who is immediately marked as unavailable, and it is added to the list of the tasks to be monitored $\mathcal{T}_{\mathcal{A}}$ (Lines 11 - 12). At this point, all the equivalent tasks of T are discharged and removed from the graph. Since the scheduler works online, this procedure is necessary because otherwise it could happen that the algorithm runs through two parallel paths, which is unnecessary to reach the goal.

Then the scheduler uses the database to check, for all the active tasks, the information regarding the task monitoring (Line 18). The task monitoring strategy allows to detect critical deviations in the collaboration. In turn, critical deviations from the nominal behavior are translated into a failure which is stored inside the database. If no results are available, the task is not concluded, and the scheduler continues to inspect the other tasks (Line 20). In the other cases, the behavior of the scheduler depends on the type of the result. If the task has been concluded, the algorithm only checks if this is a final task, and the job is finished (Line 22). If the actor has failed, the scheduler locally adapts the graph in order to generate a recovery procedure (Line 22). This local adaptation depends on the type of error E coming from the monitoring and it is further detailed in Section Lastly, the concluded task is removed from $\mathcal{T}_{\mathcal{A}}$ and the actor is marked as free (Lines 27 - 28).

5.6 Different Operator Skill

By definition, a HRC application is characterized by the presence of both humans and robots. The differences between these two actors are quite intuitive. Human operators are capable of very complex tasks, improving execution every time, but they can hardly reach or work in hazardous environments. Robots, on the other hand, are less affected by hazards in the surrounding environment, but they are not able to intrinsically learn from their last task execution.

Unlike robots, the human is never a constant factor, human operators have different skills, which can be improved or acquired. When starting on a new type of job, it is very likely that a human operator needs detailed instructions on the tasks that need to be performed. Therefore, it becomes useful to divide the work into several tasks that are easy to comprehend, allowing the operator to acquire the necessary skills and knowledge. At some point, the operator will acquire much more experience in the collaborative work, and it may be more convenient to combine some tasks into one, avoiding unnecessary fragmentation. An example can be a complex wiring activity: a new human operator may require wire-by-wire instructions, while for expert operator a single instruction with an overview of all the wires is sufficient.

The framework handles this situation in two different phases. Firstly, during the design of the AND/OR graph and the insertion of the tasks inside the database. Secondly, adding

the check level step inside the scheduler, see Algorithm 7 in Section This constraint is implemented according to the pseudo-code in Algorithm 8. The level checking needs as

Algorithm 8 checkLevel()

```
1: Require: T, G, H_L
2: a ← getActor(T)
3: if a ≠ H then
4:     return true
5: else
6:     T_E ← getEqTasks(T, G)
7:     for t ∈ T_E do
8:         if t.level > T.level and t.level ≤ H_L then
9:             return false
10:        end if
11:    end for
12:    if T.level ≤ H_L then
13:        return true
14:    end if
15:    return false
16: end if
```

input the task to be analyzed T, the AND/OR graph $\mathcal{G}$ and the actual human operator expertise level H_L (Line 1). It immediately gets the actor a that must perform the task and, if it is not a human operator, it returns *true* allowing to schedule the task (Lines 2 - 4). If the actor is the human it is necessary to investigate more to decide if T is the most suitable task. The algorithm firstly builds a list of all the equivalent tasks $\mathcal{T}_E$ analyzing the AND/OR graph (Line 6). Then, for each equivalent task, it checks if there is a task that requires a greater knowledge than the one that is required by the current analyzed task and if this knowledge is still admissible for the current human operator. If this is the case, the algorithm returns $false$ (Line 9). It is worth noting that this part of the code allows to always choose the task that best suits the human knowledge level, improving the collaboration and the job quality for the human operator. If no better tasks are available, the algorithm checks if this task can be performed by the human operator before allowing its execution (Line 13).

5.7 Error Representation

During the collaboration, it is very unlikely everything happens exactly as planned. Human operators and robots will make errors during the task execution, preventing the nominal behavior. The framework must be able to handle these errors and to adapt the behavior of the actors accordingly.

Errors can be classified into two categories based on the way the error is handled:

- **Restorable Error**, which is an error that does not preclude the task and, after some checking and restoring actions, it is possible to retry the execution. This may happen when the robot accidentally hits something and, after the human operator confirms that there are no damages or safety problems, the task is assigned again to the robot. i.e, the repair task brings the product to the state *before* the erroneous

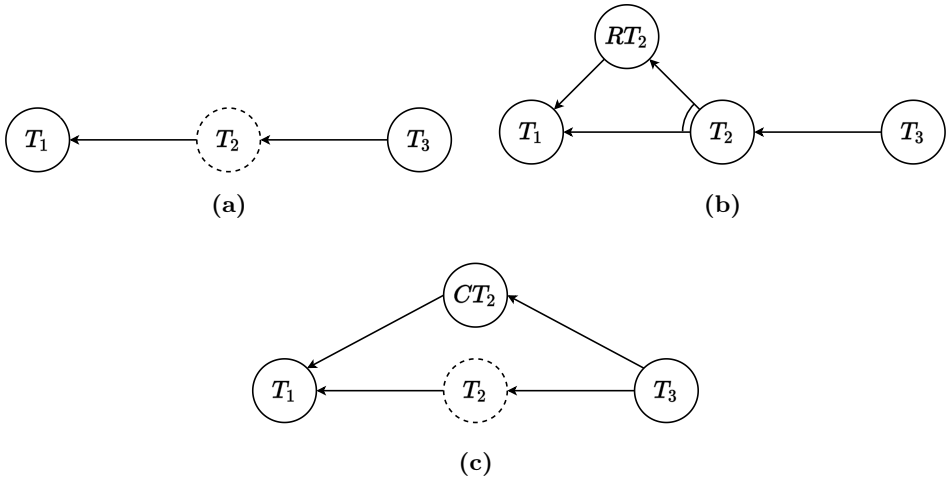


Figure 5.3: Representation of the errors and the adaptation of the AND/OR graph.

task.

- **Non-Restorable Error**, which is an error that precludes the task execution, and it is necessary to execute another task to continue the collaborative job. This may happen when the robot places an object with a wrong orientation and the scheduler assigns a new task to replace the object in the correct pose. i.e., the repair task brings the product to the state *after* the erroneous task.

In the proposed framework, both categories of errors are handled by locally adapting the AND/OR graph. This allows the scheduler to continue working without losing its generality. It is worth noting that some errors are of a type that do not allow the collaboration to continue, e.g., a safety violation. Since these cases require more severe intervention of an operator and a consequent reset of the collaborative job, they are not covered by the proposed framework.

5.7.1 Restorable Error

After a restorable error occurs, the nominal behavior of the actors could be ideally restored since the execution of the task is not precluded. The way the framework handles this situation is illustrated in Figure 5.3b, where the dashed line in Figure 5.3a means that T_2 is failed.

A new graph path that goes from the previous task to the failed one is generated. This path contains all the tasks that must be executed to restore the nominal behavior, e.g., ask to the human if it is safe, and is attached with the original AND/OR graph with an AND constraint. Then, the task that has previously failed is reset and marked as a task that must be still scheduled. According to Algorithm 7 in Section with this strategy the scheduler is forced to go through this new restoring path before scheduling again the task that had previously failed.

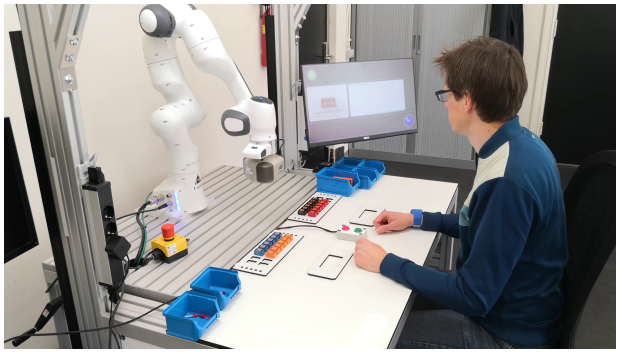


Figure 5.4: Setup of the Resilient and Effective Scheduling experiments.

5.7.2 Non-Restorable Error

When the error compromises the correct execution of the collaborative job, the restoring procedure may not be enough. In this case it is necessary to ask to one of the actors to execute a set of tasks to recover from the failure and continue the collaboration with the next tasks. The procedure implemented in the framework is illustrated in Figure 5.3c.

A new graph path is inserted that goes from the previous task to the task that follow the failed one. This path is composed by all the tasks necessary to correct from the failure, e.g., adjust the orientation, and it is attached with an OR constraint. According to Algorithm 7 in Section this new path allows the scheduler to continue the collaborative job, omitting the previously failed task.

5.8 Validation

The proposed framework has been experimentally validated in a human-robot collaborative scenario, where the human operator works together with a Franka Emika Panda, a 7-DoF collaborative robot. Several experiments have been carried out, focusing on four situations:

- Different human operator skills
- Substitution of human actions with robot action
- Error handling
- Parallelism of the actors with resource sharing

During the experiments, the human operator has been guided exploiting the Arkite's Human Interface Mate (HIM), which also enables the interaction. The complete setup for the experiment is shown in Figure 5.4. All the software components were developed in Python 3.8 and exploiting Apache Kafka, while the Franka Emika Panda is position controlled using the MoveIt Motion Planning Framework and the standard ROS libraries.

The first three situations can be represented with the AND/OR tree shown in Figure 5.5a, where different paths can be distinguished. The yellow one requires only the human operator to perform the job, while on the blue path the robot performs some of the tasks. In the last part, the tree divides based on the expertise of the human operator. A detailed description of all the tasks composing the collaborative job is shown in Table 5.1. It is worth noting that the tasks related to the pick and place of both the casing and

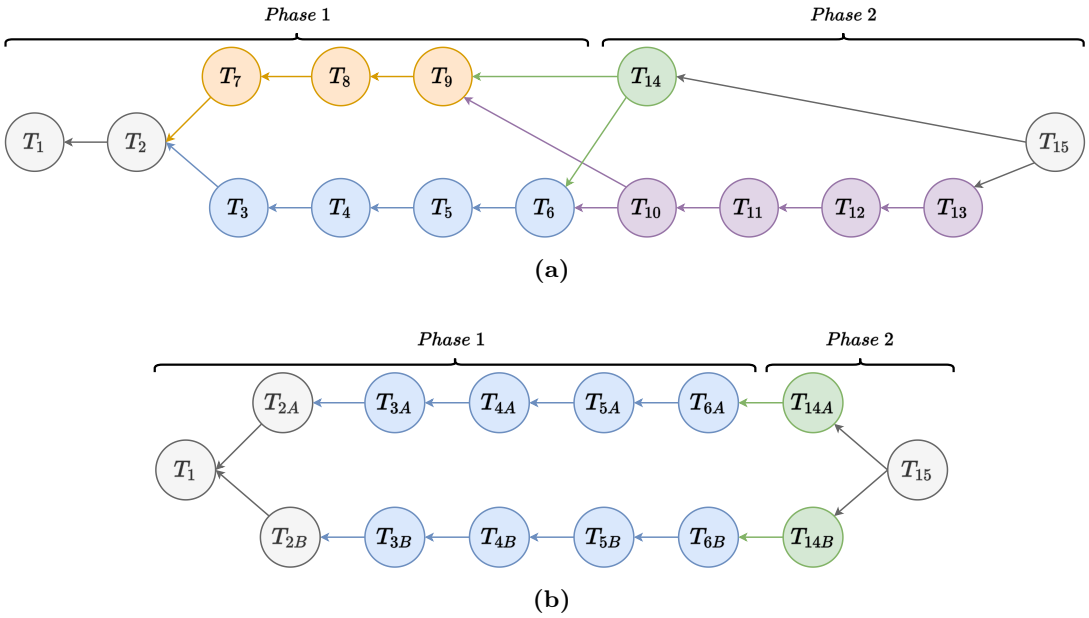


Figure 5.5: AND/OR graphs representing all the experiments performed.

the connectors are part of the first phase. While the ones related to the wiring are part of the second phase. The parallel work situation, instead, is schematized in Figure 5.5b. This represents a double assembly job where both a robot and an expert human operator are available. For ease of reading, the tasks that have been doubled are denoted with the subscripts A and B.

5.8.1 Different Skills

These experiments mainly focus on the second phase of the collaborative scenario. Initially, the operator is inexperienced in the collaborative work to be pursuit. The collaboration starts when the operator confirms to be ready and the scheduler immediately asks to place the casing in the correct spot, i.e., T_1 and T_2. At this point, the robot starts to pick and place all the connectors, while the human operator is waiting. This is because no parallel tasks are available, see the blue path in Figure 5.5a. Once the robot has completed the tasks, the collaborative job continues with the second phase. Since

Table 5.1: Tasks description of AND/OR graph in Figure 5.5a

Task Index	Description	Agent	Phase
1	Start the job.	H	Phase 1
2	Pick&Place casing.	H	
3	Move in Home.	R	
4-6	Robot Pick&Place 3 connectors.	R	
7-9	Human Pick&Place 3 connectors.	H	
10-13	Connect 4 wires one by one - not expert user.	H	Phase 2
14	Connect all the wires - expert user.	H	
15	Confirm completion.	H	

the operator is learning, the operator receives detailed step-by-step instructions about the wiring, represented by the purple path in Figure 5.5a. Exploiting the input coming from the human capabilities block, the scheduler is aware of the human expertise level.

After multiple executions of the collaborative job the human operator became more expert, and the *Human Capabilities* block detects an upgrade of the user level. In this context, displaying the wiring instructions step by step may be tedious and annoying, with a consequent reduction of the job quality. The scheduler can adapt and chooses the green path in Figure 5.5a, which merges $T_{10} - T_{13}$ in one single task T_{14} so the operator only receives high level instructions.

This experiment validates the functionality of the **checkLevel**() constraint presented in Section demonstrating that the overall framework can adapt to different human operators.

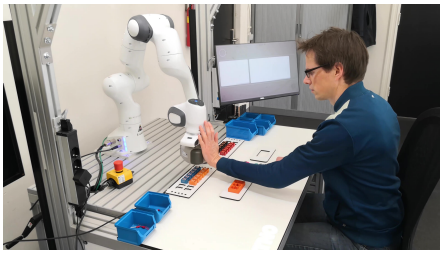
5.8.2 Actor Substitution

This experiment simulates the situation where, for whatever reason, the robot is unavailable. This may happen when the tool of the robot is under maintenance, and a fallback strategy is required. In the analyzed scenario this applies to the robot tasks during the first phase. To simulate such unavailability, the robot actor is removed from the actors list and, without changing the tree, the job is started. As before, the human operator confirms and places the casing in the correct position. At this point, the scheduler can only go through the only human path, the yellow one in Figure 5.5a, asking to the human operator to pick and place the connectors.

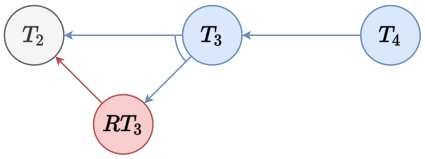
5.8.3 Error Handling

In this experiment, the human operator intentionally triggers robot errors which are subsequently handled by the framework. The framework uses the task monitoring block of Section to detect the correct execution of the task. The first time a task fails, a restorable error is triggered. If the same task fails another time, a non-restoring action is required.

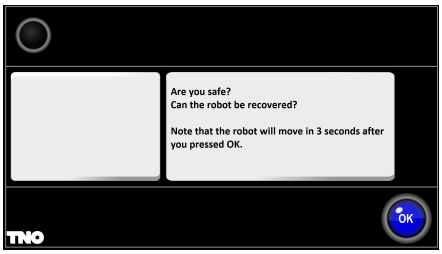
During the execution of T_3 the human operator hinders the robot which immediately stops for safety reason. At this point the tree is locally adapted inserting a restoring task. Thanks to this task, the scheduler firstly asks the human operator if he is safe and, if possible, moves the robot to the home position ready to retry the execution. This is shown in Figure 5.6b and Figure 5.6c. Subsequently, the human operator hinders the robot again, causing another failure. Since the task failed twice, the task monitoring generates a not restorable error. This is because it would be better to ask the human operator to execute this task. For this reason, the AND/OR graph is modified again adding a new path to reach T_4. As before, the human operator must confirm that everything is fine and the robot goes back in a home position, but this time the scheduler asks to the human to pick and place the connector in the correct place. All the steps are illustrated in Figure 5.6d and Figure 5.6e. From this point the robot can resume its work.



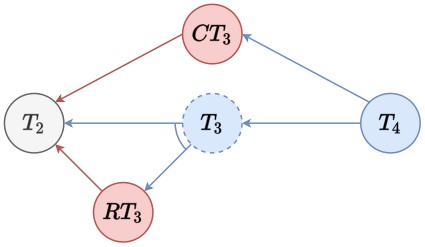
(a)



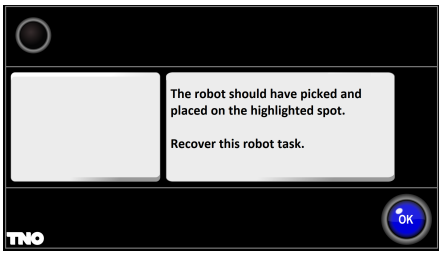
(b)



(c)



(d)



(e)

Figure 5.6: Failure of T_3 and adaptation of the framework.

5.8.4 Parallel Work

The previous situations validate the effectiveness of the framework and its features in a sequential scenario. The framework is also capable of scheduling parallel tasks. In this case the robot works in parallel sharing resources with the human operator. This is shown in the last scenario, where the human and the robot work on two products at the same time while sharing a workspace. Without losing generality, the scenario has been simplified avoiding the possibility of different paths.

When the collaborative job starts, the scheduler immediately asks to the human operator to pick and place the first casing. Subsequently, the robot starts to insert the three connectors while the human operator places the second casing. Once the robot has finished working on the first casing, the scheduler asks the human to make the wiring. It is worth noting that, since the human operator is an expert user, the wiring is composed by a single task. In the meantime, the robot finishes to put the connectors inside the second casing, allowing the human to conclude the job with the second wiring.

Chapter 6

Safe Trajectory Planning and Optimal Control

This chapter reports the research activities carried out following the work on the flexible execution layer presented in Chapter 3. In particular, two architectures are proposed. The first is a natural extension of the architecture validated in Rossini. In particular, the dynamic replanning envisaged by the Flexible execution layer does not take into account the operator's reaction. The aim was therefore to make the robot's behavior more predictable and to adapt to the characteristics of the operator. The other architecture, on the other hand, functions as a support for what Rossini's work was. Starting from a reference trajectory, the purpose of this architecture is to give the operator direct control over the collaboration.

The work presented in this chapter is published in [8, 10].

6.1 A Dynamic Planner for Safe and Predictable Human Robot Collaboration

6.1.1 Introduction

Industrial applications where collaborative robots are used in close proximity to human operators are growing rapidly. Despite the increase of the flexibility in the production, this new paradigm requires great prudence in guaranteeing the safety of operators. The problem of ensuring safety and implementing collision avoidance behavior HRC applications has been extensively addressed in the literature, as already detailed in Section 3.1. Most of the approaches, however, only focused on the optimal robot behaviour in term of execution time. In order to improve the collaboration, it is very important to also analyse how the human operators are affected by online replanning strategies. In [103] the authors embed the human factors inside the control problem proposing the Expectable Motion Unit (EMU). This unit ensures the reduction of the involuntary motions of the human operator, which are usually caused by a startle or surprise, imposing a velocity constraint. In [104] the EMU is used inside an MPC control framework to generate deviations for improving the human robot collaboration. A deviation from the path, however, may affect negatively the human operators. Indeed, according to [105], the human operator feels safer when the robots follow exactly the expected path, since they have a predictable behaviour.

This work proposes a novel framework for trajectory planning that takes inspiration from these two approaches to enhance the HRC while ensuring safety. Given a desired final configuration, the framework firstly plans a collision-free trajectory for the robot. Subsequently, it computes at runtime the optimal input that allows the robot to follow the desired trajectory with a maximum admissible deviation from the original path. This maximum deviation depends on the preferences of the human operator performing the collaborative job, with a consequent improvement of the collaboration. Furthermore, the proposed architecture can be widely applied in all industrial environments, regardless of their complexity.

The main contributions of this work are:

- A framework for trajectory planning that considers the high dynamism of the environment and the human preferences to generate collision free-trajectory in a HRC scenario.
- A novel optimization problem that explicitly considers the human operators preferences to improve the collaboration, while ensuring safety in collaborative industrial scenarios.

6.1.2 Problem Statement

Consider an industrial application where a human operator and a n-DOFs velocity-controlled collaborative robot have to work together to perform a job. The robot can be

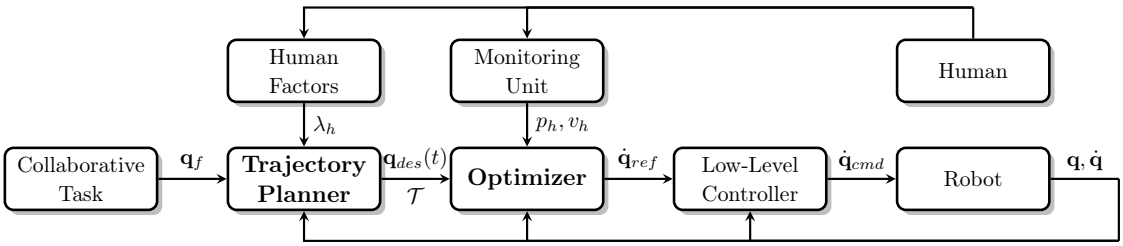


Figure 6.1: The proposed overall framework.

modeled as:

$$\dot{\mathbf{q}} = \mathbf{u}, \tag{6.1}$$

where $\dot{\mathbf{q}} \in \mathbb{R}^n$, and $\mathbf{u} \in \mathbb{R}^n$ are the joints velocities and the controller input, respectively.

During the collaboration, the robot has to perform a set of predefined tasks. Each task is associated with a trajectory $\mathbf{q}_{des}(t) \in \mathbb{R}^n$ that allows the robot to reach a desired final configuration $\mathbf{q}_{des}(t_f) = \mathbf{q}_f \in \mathbb{R}^n$ from an initial configuration $\mathbf{q}_{des}(t_i) = \mathbf{q}_i \in \mathbb{R}^n$. As detailed in Section 3.2, the trajectory is considered admissible if it these two conditions hold:

1. It does not collide with the human operator, i.e. equation (3.4) is true.
2. It is compliant with the safety limits imposed by the ISO/TS 15066, i.e. equation (3.14) is respected.

However, satisfying (3.14) may lead to significant and somehow unpredictable deviations from the nominal trajectory. Thus, the safe behavior of the robot may be perceived unsafe by the user. Indeed, as shown in [105], the human operator should be able to predict the robot behavior during the collaboration task.

In this work, the trajectory of the robot is considered predictable when the trajectory of each link is sufficiently close to the desired one, namely:

$$||\mathbf{x}_i - \mathbf{x}_{des,i}|| \leq \lambda_h \quad \forall i \in \{1, \dots, n\}, \tag{6.2}$$

where $\mathbf{x}_i \in \mathbf{R}^3$ and $\mathbf{x}_{des,i} \in \mathbf{R}^3$ represent the real Cartesian position of the i-th link and the desired one, respectively, and the term λ_h represents the maximum deviation that the human operator may accept. This parameter embeds the human characteristics, such as trust or confidence [38], and integrating it into the control strategy translates on constraining each link to lie inside a set of tubes, one around each nominal path.

In a first approximation, λ_h is considered constant and it can be related, e.g., to the level of experience of the operator (high for confident users and low for the unconfident ones). It is assumed that the level of confidence does not change during the task and, consequently, that λ_h stays constant during the collaboration. An online calculation of λ_h will be subject of future research.

This work aims at designing a safety kinodynamic architecture that:

- Computes a nominal trajectory that is collision-free, i.e. a trajectory that the robot could ideally execute at maximum speed and that avoids all the surrounding

obstacles.

- Starting from the nominal trajectory, it automatically builds a set of safety tubes where the robot could stay without colliding with the environment, while maintaining a predictable behaviour.
- Exploiting the tracking of the human movements, it computes the optimal safety robot inputs that aim at following the nominal trajectory while avoiding the human. The control inputs are computed online solving a linear optimization problem and ensure that the robot behaviour is compliant with the velocity limit imposed by the the ISO/TS 15066 standard.

6.1.3 Architecture

The proposed safety collaborative strategy is summarized in Figure 6.1, where two main components may be distinguished:

- The **Trajectory Planner** block, which is responsible of planning the collision-free trajectory $\mathbf{q}_{des}(t)$ that allows the robot to reach the desired final configuration $\mathbf{q}_f$.
- The **Optimizer** block solves online a MILP problem to compute the optimal input $\dot{\mathbf{q}}_{ref}$ that allows the robot to follow the planned trajectory while being compliant with the safety standards.

The overall procedure starts with the assignment of the collaborative task to be executed, which can be implemented with task scheduling procedure. The task is associated with a desired final configuration $\mathbf{q}_f$ that is exploited by the **Trajectory Planner** to compute the collision-free trajectory $\mathbf{q}_{des}(t)$. Moreover, it leverages the maximum admissible deviation λ_h to build a set of tubes around each link trajectory $\mathcal{T}$. To this purpose, the **Trajectory Planner** must be aware of all the static obstacles that are inside the collaborative workspace. In this phase, the human operator is ignored since is a very dynamic obstacle.

Then, the **Optimizer** exploits the human monitoring information to compute online the optimal input $\dot{\mathbf{q}}_{ref}$. This optimization problem ensures that: the implemented behaviour is compliant with the ISO/TS 15066, since (3.14) is added as a constraint, and that the resulting Cartesian motion lies inside the tubes $\mathcal{T}$.

Lastly, the input is forwarded to the low-level controller that takes care of implementing the desired velocity.

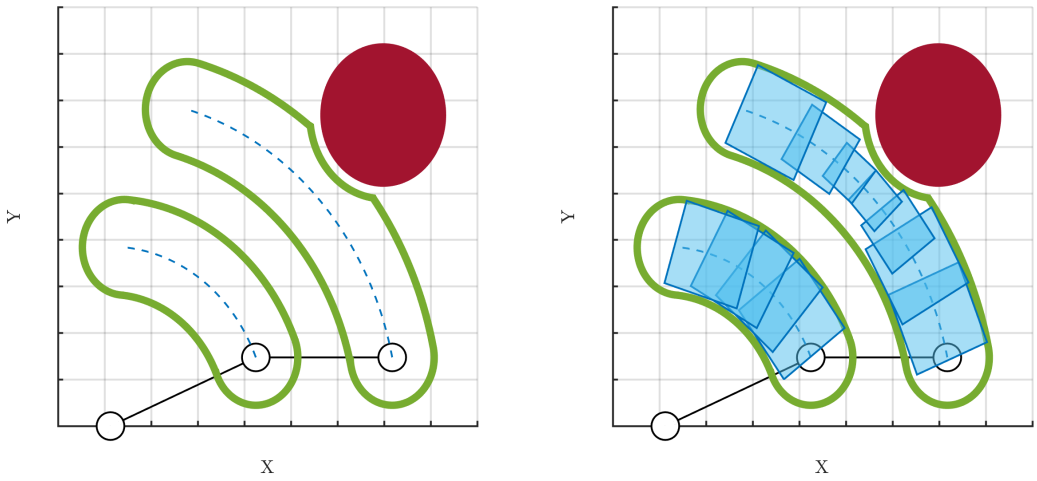
Planner

The role of this component is to find a robot trajectory $\mathbf{q}_{des}(t)$ that is collision-free and that the robot could ideally execute at maximum speed. Since the human operator is in a certain sense an unpredictable obstacle, in this first planning phase it is not considered, and the trajectory is calculated only on the static collaborative workspace.

The trajectory planning is implemented according to the pseudo-code reported in Algorithm 9. The trajectory planner needs as input the initial and the final configuration, respectively $\mathbf{q}_i$ and $\mathbf{q}_f$, and the maximum admissible path deviation λ_h (Line 1). It

Algorithm 9 TrajcetoryPlanner()

1: **Require:** $\mathbf{q}_s, \mathbf{q}_f, \lambda_h$
2: $\bar{\mathbf{q}}_{des}(t) \leftarrow$ **plan**$(\mathbf{q}_s, \mathbf{q}_f)$
3: $\mathbf{q}_{des}(t) \leftarrow$ **generateTrajectory**$(\bar{\mathbf{q}}_{des}(t))$
4: $\mathcal{T} \leftarrow$ **generateTubes**$(\mathbf{q}_{des}(\cdot), \lambda_h)$
5: **send**$(\mathbf{q}_{des}(t), \mathcal{T})$



(a) **(b)**

Figure 6.2: Visual representation of the proposed predictability constraint in 2D.

firstly plans the collision-free path $\bar{\mathbf{q}}_{des}(t)$ that the robot should perform (Line 2). The function **plan** can be implemented using the same strategies of Chapter 3. Once the path has been computed, the planner defines a trajectory that follow the desired path and which is compliant with the robot limits, i.e. maximum joint velocity, maximum torque. To this purpose it is possible to exploit the strategy presented in [106], which allows to obtain the maximum speed trajectory (Line 3). Subsequently, it computes the set of tubes $\mathcal{T}$ such that: in each tube the i-th link can freely move without colliding and whose radius is, at most, equal to λ_h. Lastly, the trajectory and the set of tubes are forwarded to the optimizer block (Line 5).

Mixed-Integer Linear Programming Problem

Once the nominal trajectory $\mathbf{q}_{des}(t)$ and the set of tubes $\mathcal{T}$ have been computed, the optimizer generates a safe and predictable reference trajectory to be set as a setpoint to track to the robot.

First, a mathematical description of the set of tubes $\mathcal{T}$ needs to be computed in order to synthesize the controller. The idea is to fit the tube with a set of simpler volumes, such as spheres or cubes, and to constrain the Cartesian links positions inside this set of volumes. To keep the computational cost low, it has been chosen to use cubes, as they allow a linear formulation and, therefore, to set up a linear optimization problem

Figure 6.2 reports a simplified example of how this step is performed on a 2-joints planar

manipulator. Considering Figure 6.2a, the Trajectory planner provides first the desired trajectory (light blue dots) and the set of tubes (green areas). The set of tube is computed as collision free considering the position of the obstacle (dark red area). Then, for each link, a set of cubes is located inside the original tube, see Figure 6.2b, ensuring a minimum of overlap to guarantee communication between the cubes, and contain the initial and final position.

This constraint, with (3.14) and the limits of the robot are finally used to summarize an optimization problem, the solution of which will be used as a reference to control the robot.

Since this component works as a dynamic planner, it considers the robot as perfectly kinematic initializing a state $\mathbf{q}^v$ as the state of the robot $\mathbf{q}$ at the first control cycle, and updating it using the model (6.1).

The procedure starts by constraining the Cartesian position of the generic i-th link to lie inside the j-th cube of its trajectory. The choice behind the use of cubes is related to possibility to express these constraints in a linear form. This can be done by constraining the position to lie in the volume of space enclosed within the 6 planes defining the faces of the cube.

Let $c_i^j = [{}^x c_i^j, {}^y c_i^j, {}^z c_i^j]^T \in \mathbb{R}^3$ and $l_i^j \in \mathbb{R}$, with $l_i^j \leq \lambda_h$, be the centre and the edge size of the cube, respectively. The i-th link position $x_i^v \in \mathbb{R}^3$ lies inside of that cube if:

$$\bar{A}_i^j x_v^i \leq \bar{b}_i^j, \tag{6.3}$$

where:

$$\bar{A}_i^j = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \quad \bar{b}_i^j = \begin{bmatrix} {}^x c - l/2 \\ -{}^x c - l/2 \\ {}^y c - l/2 \\ -{}^y c - l/2 \\ {}^z c - l/2 \\ -{}^z c - l/2 \end{bmatrix} \tag{6.4}$$

are the matrices grouping the coefficients of the 6 planes.

Since the robot is considered as perfectly kinematic, (6.1) holds and it is possible to assume that the position of the i-th link evolves as a single integrator in the discrete time domain as:

$$\mathbf{x}_i^v(k+1) = \mathbf{x}_i^v(k) + B J_i^p(\mathbf{q}^v)\mathbf{u}^v(k), \tag{6.5}$$

where $B = \text{diag}\{\Delta t_c\} \in \mathbb{R}^{3\times 3}$ is the input matrix, with Δt_c the sampling time ($t = k\Delta t_c$, $k \in \mathbb{Z}$), $J_i^p(\mathbf{q}^v) \in \mathbb{R}^{3\times n}$ is the position part of the i-th link jacobian, and $\mathbf{u}^v \in \mathbb{R}^n$ represents the joint velocity input.

Using (6.5) in (6.3), the position of the i-th link of the robot can be constrained to lie inside a cube by constraining its future position and choosing the input velocities $\mathbf{u}^v$ such that:

$$A_i^j \mathbf{u}^v(k) \leq b_i^j, \tag{6.6}$$

where:

$$A_i^j = \bar{A}_i^j B J_i^p(\mathbf{q}^v), \;\; b_i^j = \bar{b}_i^j - A_i^j x_i(k). \tag{6.7}$$

The position of each link must lie inside the set made up by the union of all cubes.

To this aim, it is necessary to expand the constraint (6.6) to all the cubes of the trajectory. This cannot be done by simply imposing (6.6) for all cubes. This in fact will turn to an AND type of constraint, requiring that the robot is at the same time in all cubes, which is clearly impossible. The overall constraint needs to be converted into an OR type constraint, i.e. requiring that the robot stays in one cube at a time. This can be done by (see [107] for more details):

1. Designing a constraint like (6.6) for each cube.
2. Adding for each cube a binary extra variable representing if the i-th link belongs to that cube.
3. Adding a constraint on the extra binary variables to force the i-th link to belong to a single cube at a time.

Formally, by setting for each link:

$$A_i^j \mathbf{u}^v(k) - M\epsilon_i^j \leq b_i^j \quad j = 1, \ldots, Np, \tag{6.8}$$

and

$$\sum_{j=1}^{N_q} \epsilon_i^j \leq N_q - 1, \tag{6.9}$$

where $M \in \mathbb{R}^6$ is a large arbitrary positive vector and $\epsilon_i^j \in \{0, 1\}$ is the extra binary variable, and $N_q \in \mathbb{N}$ the number of cubes of each trajectory. With this approach, if the j-th constraint of the i-th link is not satisfied, the corresponding binary variable ϵ_i^j is equal to 1. The constraint (6.9) ensures that at least one of the constraints (6.8) is satisfied, which represents the cube where the i-th link lies in. It is worth noting that the formulation of the constraint is still linear.

Finally, the input is computed solving the following optimization problem:

$$\begin{aligned}
\min_{u^v, \epsilon} \quad & ||\dot{\mathbf{q}}_{des}^v - \mathbf{u}^v||^2 \\
\text{s.t.} \quad & \\
& A_i^j \mathbf{u}^v - M\epsilon_i^j \leq b_i^j \quad j = \{1, \ldots, Np\}, \\
& \sum_{j=1}^{N_q} \epsilon_i^j \leq N_q - 1 \qquad i = \{1, \ldots, n\}, \\
& \mathbf{u}^v < \dot{\mathbf{q}}_{max}, \\
& \mathbf{u}^v > \dot{\mathbf{q}}_{min}, \\
& J_{rh}^i \mathbf{u}^v < \beta_{rh}^i \qquad i = \{1, \ldots, n\},
\end{aligned} \tag{6.10}$$

where $\dot{\mathbf{q}}_{max} \in \mathbb{R}^n$ and $\dot{\mathbf{q}}_{min} \in \mathbb{R}^n$ represents the maximum and minimum joint velocity of the robot, respectively, $J^i_{rh} \in \mathbb{R}^{1\times n}$ is the jacobian of the i-th link towards the human, β^i_{rh} is the velocity limit of the i-th link computed as in (3.14) and:

$$\dot{\mathbf{q}}^v_{des} = \dot{\mathbf{q}}_{des} + K^v(\mathbf{q}^v - \mathbf{q}_{des}) + D^v(\dot{\mathbf{q}}^v - \dot{\mathbf{q}}_{des}), \tag{6.11}$$

where $K^v \in \mathbb{R}^{n\times n}$ and $D^v \in \mathbb{R}^{n\times n}$ are the proportional and derivative gains, respectively, introduced to allow the system to be attracted to the desired trajectory after a deviation induced by the optimizer, e.g. to avoid the human.

At each control cycle, the state of the robot $\mathbf{q}^v$ is updated using the model (6.1), and the reference speed $\dot{\mathbf{q}}_{ref}$ is set equal to $\mathbf{u}^v$ returned by the optimizer.

Low-Level Controller

Once the reference $\dot{\mathbf{q}}_{ref}$ has been computed from the optimizer, it is necessary that the real robot follows exactly this reference in order to behave like the model represented in equation (6.1). Indeed, if the robotic system is composed only of a manipulator, the approximation as a first order system is valid. However, if the system is also composed of a mobile base, this approximation is very strong and most likely the robot will not be able to follow $\dot{\mathbf{q}}_{ref}$ exactly. Thus the Low-Level Controller is added, which allows to track the corresponding trajectory. This block is implemented as a simple PD controller by setting:

$$\dot{\mathbf{q}}_{cmd} = \dot{\mathbf{q}}_{ref} + K_l(\mathbf{q} - \mathbf{q}_{ref}) + D_l(\dot{\mathbf{q}} - \dot{\mathbf{q}}_{ref}), \tag{6.12}$$

where $K_l \in \mathbb{R}^{n\times n}$ and $D_l \in \mathbb{R}^{n\times n}$ are the proportional and derivative gains, respectively.

6.1.4 Validation

The proposed framework has been experimentally validated on a mobile manipulator composed by a UR10e, a 6-DoF collaborative manipulator, and a Neobotix MPO-500, a mobile base for industrial application. The tracking of the human and the overall environment has been achieved exploiting seven OptiTrack Prime^x cameras with the OptiTrack Motive software.

A complete setup of the experiments is shown in Figure 6.3, where it is possible to note the UR10e mounted on the Neobotix MPO-500, two of the seven OptiTrack Prime^x cameras, two obstacles tracked with the OptiTrack markers (yellow circles) and a wrist band used to track the right arm of the human operator (red circle). All the software components were developed using ROS Kinetic Kame meta-operating system and they ran on a Intel(R) Core(TM) i7-7700HQ with Ubuntu 16.04. The trajectory planner layer is based on the RRT-Connect algorithm [75] and it is implemented exploiting OMPL [108] and FCL [109] libraries. The MILP problem in (6.10) is solved online exploiting the Gurobi solver [52].

Concerning the frequencies, the communication with the UR10e works at 500 Hz, while the one with the Neobotix MPO-500 works at 50 Hz. The OptiTrack, instead, works

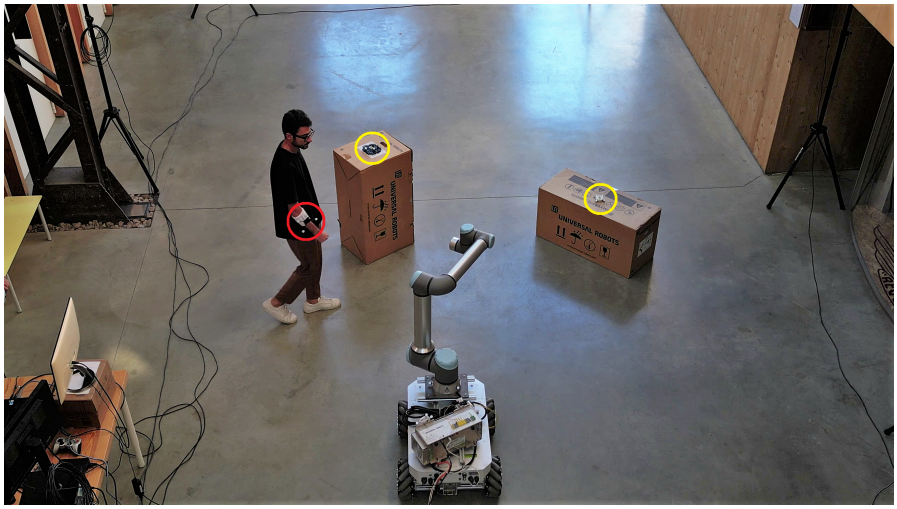


Figure 6.3: Setup of the experiments.

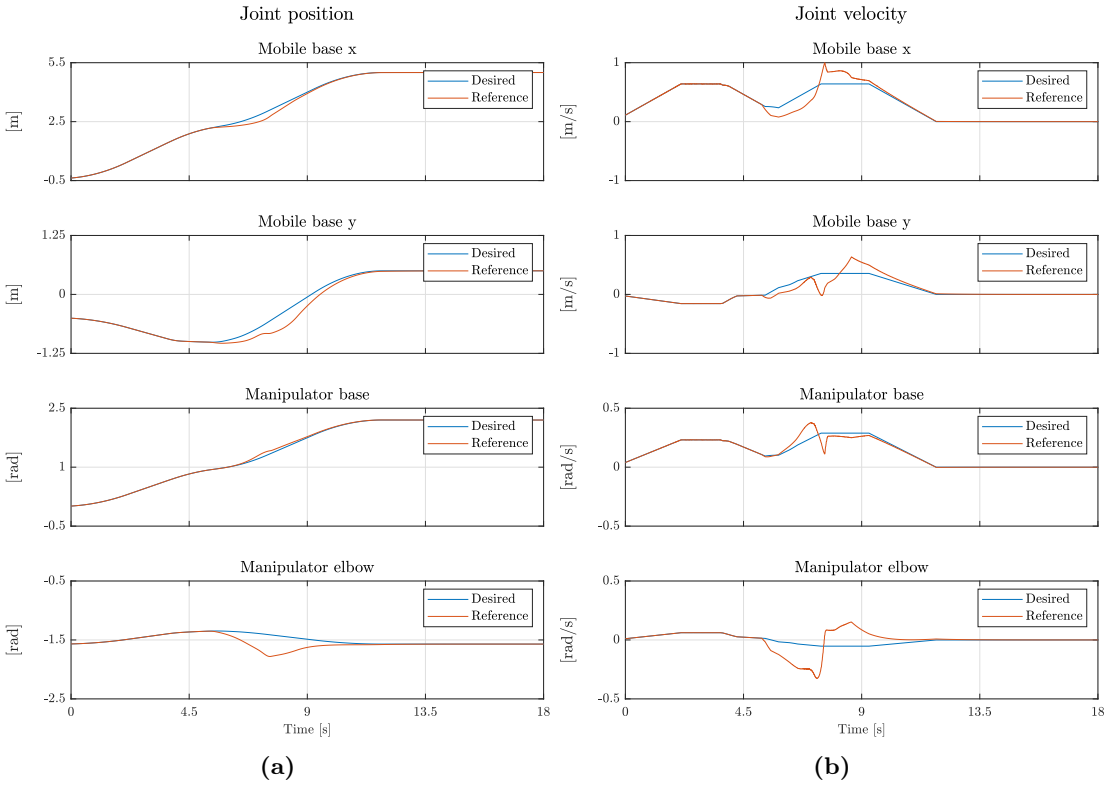


Figure 6.4: Joint positions and velocities in a scenario with a confident human operator.

at a frequency of 240 Hz. The optimization problem convergence is strictly related to the number of of control points used for the robot. With 4 control points, it is solved in approximately 2.5 ms. Lastly, the gain matrices K^v is equal to identity matrix, while D^v is a zero matrix.

To demonstrate the effectiveness of the framework several experiments have been carried out, considering both a *confident* human operator, i.e. an operator that would allow a great deviation from the planned path, and a *unconfident* human operator, i.e. an operator that does not trust the robot and does not allow a great deviation.

In the first experiment, the robot has to reach a desired final configuration avoiding the obstacles and a confident human operator, i.e. large cubes size. The planner successfully

Table 6.1: Results Comparison

	Length $[m]$	Time $[s]$
Flexible Execution and Safety Layers	6.63	19.79
Proposed Work	3.83	12.66
Improvement	57.82%	63.89%

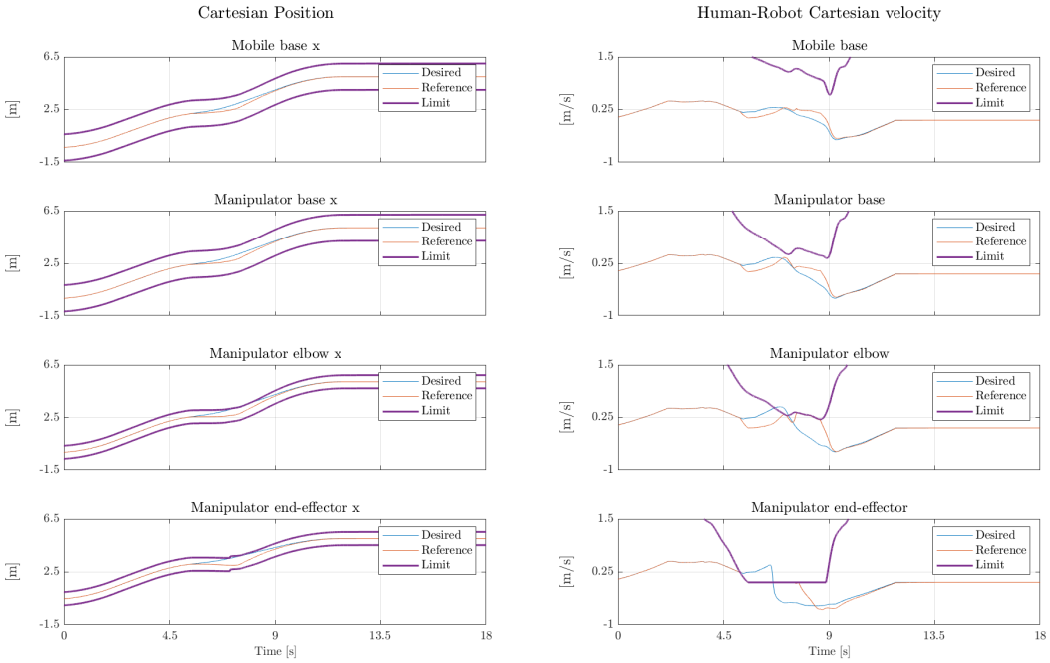


Figure 6.5: Constraints in a scenario with a confident human operator.

plans a trajectory and the robot starts to follow it. At $4.5 \leq t \leq 9.5$, the human operator approaches the robot causing a drop of the velocity, see (3.14), and making the planned path inefficient. Since the operator is classified as confident, λ_h is big and the optimizer is capable of computing a set of inputs that deviate from the desired path to obtain a better trajectory, while ensuring the safety.

Figure 6.4 shows the joint positions and velocities of the first two joints of the mobile robot and the base and the elbow of the manipulator. In particular, the blue lines represent the desired values while the red lines represent the reference values. Figure 6.5, instead, shows the constraints. In particular, in Figure 6.5a it is possible to see the links Cartesian position and their respective upper and lower bounds due to the cubes. While in Figure 6.5b, the links velocity towards the human operator with the velocity limit imposed by the ISO/TS 15066 are reported.

The same experiment has been performed with an unconfident operator, constraining the robot to stay very close to the planned path. Differently from before, as the human gets close to the robot, the only admissible solution is to command a zero velocity and completely stop the robot. The details are reported in Figure 6.6.

Lastly, a comparison with the flexible execution and safety layers architecture developed

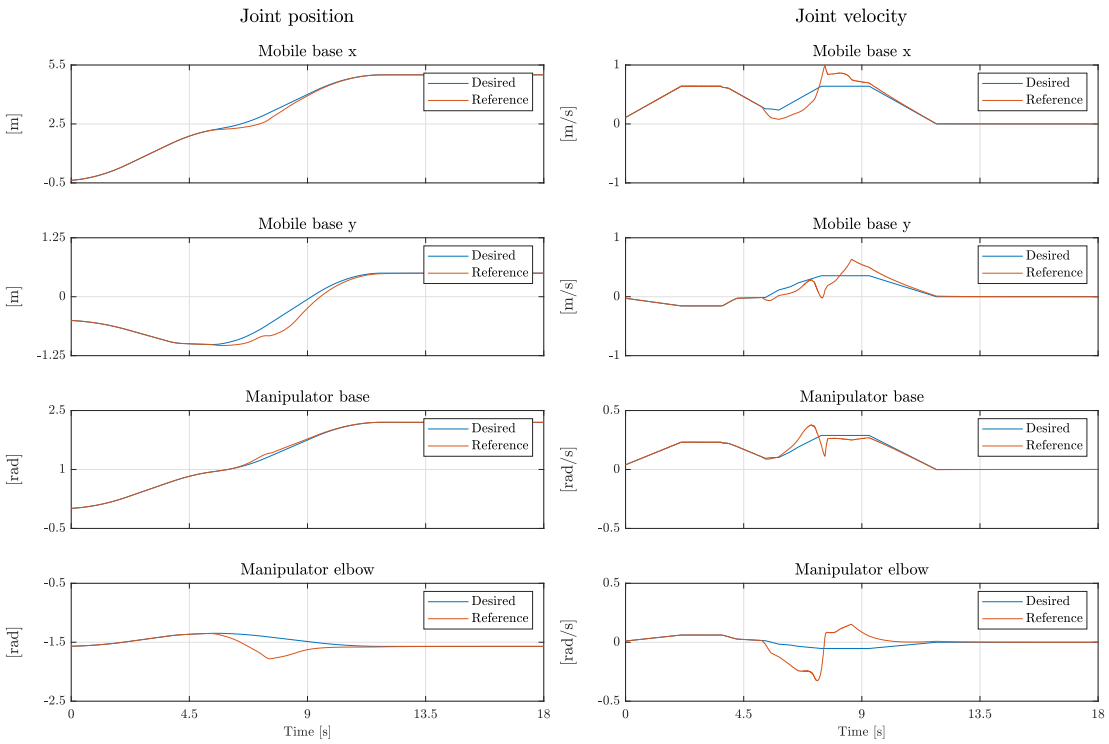


Figure 6.6: Joint positions and velocities in a scenario with an unconfident human operator.

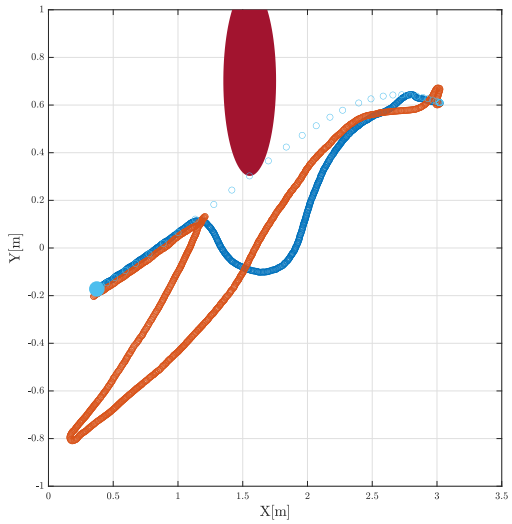


Figure 6.7: Comparison between the cartesian position of the end-effector with the two approaches.

in Chapter 3 has been performed. To have a more reliable comparison, the obstacles have been removed and, since a change in the path is allowed, the human operator is confident. The comparison between the two trajectories is shown in Figure 6.7, where the cyan dots represent the desired path while the cyan marker at the left represents q_i. The blue line represents the path followed with the proposed architecture, while the red one represents the one with the approach proposed in [1]. The dark red ellipse represents the area occupied by the human during the experiment. The z component has not been plotted, since it is approximately constant during the experiment. In the graph it is possible to note that the proposed framework outperform the one in Chapter 3 achieving both a shortest path, $L_{prop} = 3.83\ m$ against $L_{comp} = 6.63\ m$, and a lower execution time, $T_{prop} = 12.66\ s$ against $T_{prop} = 19.79\ s$. These results are listed in Table 6.1.

It is worth noting that the proposed approach is a generalization of the Flexible Execution and Safety Layers proposed in Chapter 3. In fact, if $\lambda_h = 0$ the proposed approach reduces to the one in Chapter 3 where no replanning is present. Using the architecture proposed in this work, it is possible to keep on achieving the same safe behavior with a higher degree of flexibility, while considering human confidence.

6.2 Towards an Optimal Human-Based Control Approach for Mobile Human-Robot Collaboration

6.2.1 Introduction

Collaborative robotics is becoming more and more important within industrial environments, changing the paradigm. The new industrial scenarios, in fact, are characterized by shared workspaces in which humans and robots collaborate to achieve a common goal. This possibility of interaction has made it possible to solve many manufacturing challenges and to remove physical barriers inside the work cells, increasing the flexibility.

As deeply discussed in previous chapters, the lack of physical barriers require to pay special attention in how guaranteeing safety. Furthermore, it is very it is important to have a strategy that optimal assigns the tasks, making the most of the human–robot collaboration (HRC). For this reason, researchers focused on developing architectures for safe and optimal dynamic scheduling between humans and robots. However, tasks between humans and robots in advance is not always the best choice. Sometimes it may be better to let the operator drive the collaboration, leveraging his experience and knowledge of the job.

In [38] the authors present a two-level feedforward optimization strategy for offline subtask allocation for HRC. Their strategy is integrated with a mutual trust feedback procedure that is exploited to re-allocate online the subtasks. In [110] a weighted probabilistic state machines based algorithm that recognize the human intention is presented. The tasks are divided into two different categories, namely the communication of explicit and implicit intention, depending on whether the operator engages the robot or not. Depending on the type of estimated intention, the robot acts accordingly. In [111] the authors implement a Neural Network to estimate the desired trajectory of the human operator. This estimation is integrated into impedance control such that the robot arm follows the human operator. However, these solutions do not consider the probability of collision between humans and robots.

To increase the robot workspace, giving it more flexibility, HRC research has focused on the study of mobile manipulators [112]. Several works have been done in the field of trajectory planning and control of these collaborative platforms [113–116]. In [117] the authors present a two-layers framework for task assignment and motion planning for a bi-manual mobile manipulator. The task assignment exploits search trees in temporal windows to determine feasible task assignments of agents using task and spatial constraint-based heuristics. The tree branches are then used as input to plan the motion

for each agent. In [118] a strategy that exploits concurrent and sequential task representation for task allocation in multi human-robot collaboration in industrial environments is presented.

Human intention-based strategies have been widely applied also in the field of collaborative mobile manipulators. In [119] the authors present a framework for human-robot collaboration with mobile manipulators. Their strategy allows to enhance the physical human-robot interaction while satisfying the imposed constrained. In [120] a hierarchical architecture that allows the user to generate commands via a P300-based brain computer interface is proposed. In [60] the authors present a gesture polysemy based control strategy. Thanks to the proposed method the operator is capable of controlling both the motion of mobile robot and the posture of robot using one hand. In [121] a hierarchical finite state machine that exploits voice command is proposed. The framework analyzes user's input to obtain the human intention which is then transferred to the mission plan for executing the tasks. However, these solutions are difficult to extend and do not allow the control of the size of the collaboration area. In the case of mobile platforms, in fact, it could be useful to restrict the work area, in order to have a more efficient collaboration.

This work proposes a novel control architecture for collaborative mobile manipulators that allows the human operator to actively lead the collaboration by dynamically determining the collaborative area. Using Control Barrier Functions (CBF) [82, 122] the robotic system is constrained to move inside a desired collaborative volume that depends on the pose of the operator. The size of the collaborative volume can be dynamically determined, depending on external factors as, e.g., the job to execute. Moreover, the framework is endowed of a collision avoidance strategy to ensure safety during the collaboration. The proposed architecture provides several advantages to the control of mobile collaborative robots. First, it allows a control of the whole body of the robot, allowing to exploit all the DOFs of the system for achieving the desired performance unlike the approaches that exploit separately the mobile base and the manipulator. Second, thanks to the modularity of CBFs, it is possible to dynamically change the number of constraints that the mobile manipulator has to satisfy making it reactive to changes in the environment and/or in the task it has to satisfy. Finally, exploiting time-varying CBFs, it is possible to adapt online the collaborative volume in order to improve acceptability and performance of the human-robot collaborating team.

The main contributions of this work are:

- A control architecture that allows the human operator to lead the collaboration, forcing the robot to stay close, while ensuring safety.
- A time-varying definition of the collaborative area that gives flexibility to the overall framework.
- A control architecture that is modular and can be easily extended.

6.2.2 Problem Statement

Consider an industrial workspace where a human operator and a mobile robot manipulator have to collaborate accomplishing a job. Defining the space coordinates of the robot

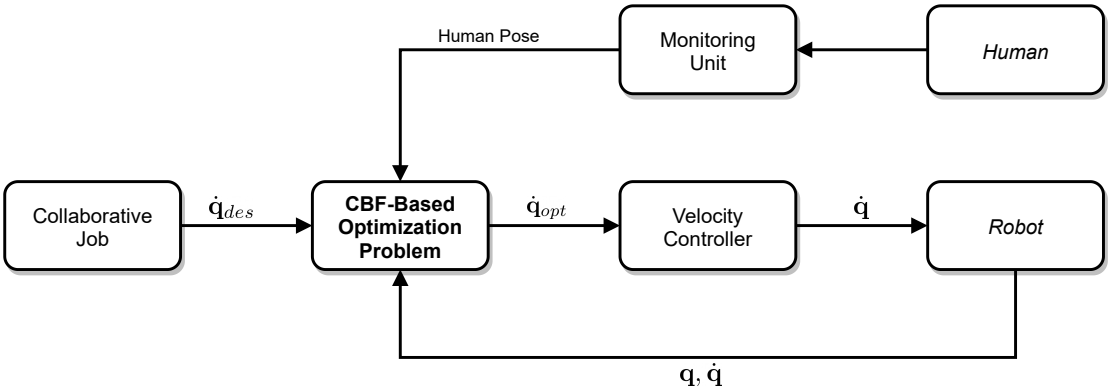


Figure 6.8: The proposed overall architecture.

as:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_b \\ \mathbf{q}_a \end{bmatrix}, \tag{6.13}$$

where $\mathbf{q}_b \in \mathbb{R}^{n_b}$ is a reduced representation of the configuration space of the differential drive mobile base, while $\mathbf{q}_a \in \mathbb{R}^{n_a}$ represents the space coordinates of the robot arm. The overall model of the robot can be described by the following equation:

$$\dot{\mathbf{x}} = J(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} J_b(\mathbf{q}_b) & J_a(\mathbf{q}_a) \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_b \\ \dot{\mathbf{q}}_a \end{bmatrix}, \tag{6.14}$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\dot{\mathbf{x}} \in \mathbb{R}^m$ represent the end-effector pose and velocity, respectively, while $J(\mathbf{q}) \in \mathbb{R}^{m \times (n_b + n_a)}$ is the augmented Jacobian[123].

To ensure safety and to allow the human operator to lead the collaborative job, the workspace is enriched with a monitoring unit that allows to track the human movements. Many strategies are already available in literature that deal with the human tracking, e.g. skeleton tracking with multiple cameras [67], markers on the human body [68], machine learning techniques [69]. In particular, the robot position is defined safe if it is compliant with equation (3.1).

This work aims at designing the overall control architecture shown in Figure 6.8 that starting from a desired robot velocity:

- Generates the **optimal** control input $\dot{\mathbf{q}}$ such that the robot stay close to the human operator while performing the job, maximizing the performances.
- Ensures safety for the human operator during the collaboration, implementing a collision avoidance behavior.
- Allows to adapt at runtime the size of the collaborative area, making it flexible and suitable to many HRC scenarios.

6.2.3 Control Barrier Functions

Theory Background

Control barrier functions (CBFs) [64, 124], are mathematical tools that can be exploited to encode the *forward invariance* of a set. In the context of collaborative robotics, they

can be used to guarantee the safety of the robotic application. Indeed, a a lot of safety-critical constraints can be defined as ensuring that the system remains inside a subset of its state, also called *safe set*. This is the case, for example, of a robot that remains at a distance which is higher than a desired threshold w.r.t. the human operator [73]. Thus, the problem of ensuring safety may be expressed as enforcing the forward invariance of the safe set.

Consider the following nonlinear system:

$$\Sigma = \begin{cases} \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + g(\boldsymbol{x}(t))\boldsymbol{u} \\ \boldsymbol{y}(t) = k(\boldsymbol{x}), \end{cases} \tag{6.15}$$

where $\boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^n$ and $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}^m$ are the state, the input and the output of the system. The vector field $f : \mathbb{R}^n \to \mathbb{R}^n$ represents the internal dynamics of the system, $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is a state dependent input matrix and $k : \mathbb{R}^n \to \mathbb{R}^m$ is the output map.

The goal is to find a safe set $\mathcal{S} \subset \mathcal{D} \subset \mathbb{R}^n$ which is forward invariant for the system. This is because, thanks to the property of forward invariance, if the system starts inside the safe set it will remain inside it at all times, i.e., $\forall \boldsymbol{x}_0 \in \mathcal{S}$, $\boldsymbol{x}(t) \in \mathcal{S}\ \forall t \geq 0$. This can be obtained by exploiting CBFs, which can be defined in the following way:

Definition 1. *(**Control Barrier Function** [64]) Let $\mathcal{S}$ be the superlevel set of a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}$:*

$$\begin{aligned} \mathcal{S} &:= \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}, \\ \partial\mathcal{S} &:= \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) = 0\}, \\ \mathit{Int}(\mathcal{S}) &:= \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) > 0\}. \end{aligned}$$

The function h is a CBF if $(\partial h / \partial \boldsymbol{x})(\boldsymbol{x}) \neq 0$ for all $\boldsymbol{x} \in \partial\mathcal{S}$ and there exist an extended class $\mathcal{K}$ function [1] $\alpha(h(\boldsymbol{x}))$ such that for the system (6.15) and for all $\boldsymbol{x} \in \mathcal{S}$:

$$\sup_{\boldsymbol{u} \in \mathbb{R}^m} \underbrace{\left[\frac{\partial h}{\partial \boldsymbol{x}} f(\boldsymbol{x}) + \frac{\partial h}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \boldsymbol{u} \right]}_{\dot{h}(\boldsymbol{x})} \geq -\alpha(h(\boldsymbol{x})). \tag{6.16}$$

The CBF h can then be leveraged to regulate the input of the system such that the system (6.15) remains safe. To achieve this it is possible to set up an optimization-based control problem whose objective is to find the best approximation of a possible unsafe desired input $\boldsymbol{u}_{des}$ which satisfies the condition in (6.16). This can be formulated with the following quadratic program:

$$\begin{aligned} \min_{\boldsymbol{u} \in \mathbb{R}^m} \quad & ||\boldsymbol{u} - \boldsymbol{u}_{des}||^2, \\ \text{s.t.} \quad & \\ & \frac{\partial h}{\partial \boldsymbol{x}} f(\boldsymbol{x}) + \frac{\partial h}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \boldsymbol{u} \geq -\alpha(h(\boldsymbol{x})), \end{aligned} \tag{6.17}$$

[1] An extended class $\mathcal{K}$ is a function $\kappa : \mathbb{R} \to \mathbb{R}$ such that κ is strictly increasing and $\kappa(0) = 0$

where the function $\alpha(\cdot)$ is a design parameter which is exploited to tune the behavior of the CBF, i.e. it allows to achieve a more conservative or a more aggressive behavior. The most common choice of $\alpha(h)$ is the identity function multiplied by a scalar value $\xi > 0$, meaning that $\alpha(h) = \xi h(\boldsymbol{x})$. Higher values of ξ result in a more aggressive behavior of the CBF.

In collaborative applications, it may be also useful to constraint the output of the system, e.g., the orientation of the end-effector or the distance between the human operator and the robot end-effector. Furthermore, the safety conditions of the task may change dynamically because, e.g., the human operator is moving in the collaborative cell. Thus, it is necessary to exploit the use of the extended version of CBFs proposed in [125], which allows to encompass time-varying constraints onto the output $\boldsymbol{y}(t)$ of the system.

Definition 2. *(**Output Time-Varying Control Barrier Function** [125]) Let $\mathcal{C} \subset \mathbb{R}^m$ be the superlevel set of a continuously differentiable function $h : \mathbb{R}^m \times \mathbb{R}_{\geq 0} \to \mathbb{R}$. Then, the function h is an output time-varying control barrier function if there exist a Lipschitz continuous extended class $\mathcal{K}_\infty$ function [2] $\alpha(h(\boldsymbol{y}, t))$ such that for the system* (6.15) *and for all $\boldsymbol{y} \in \mathbb{R}^m$:*

$$\sup_{\boldsymbol{u} \in \mathbb{R}^m} \underbrace{\left[\frac{\partial h}{\partial t} + \frac{\partial h}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} f(\boldsymbol{x}) + \frac{\partial h}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} g(\boldsymbol{x}) \boldsymbol{u} \right]}_{\dot{h}(\boldsymbol{y}, t)} \geq -\alpha(h(\boldsymbol{y}, t)). \tag{6.18}$$

The control input satisfying (6.18) can then be obtained by inserting the constraint into the the optimization problem in (6.17).

CBFs-based Architecture

The proposed control architecture exploits the use of CBFs to formulate a convex optimization problem. This is possible under the assumption that the robot limits, the task that the robot has to accomplish and the safety can be represented as a time-varying constraints on the control input [126, 127]. Solving online this optimization problem, it is possible to compute the best input that satisfies all the constraints, i.e. the robot velocities that allow to execute the task in an optimal way.

Starting from the robot model defined in (6.14) it is possible to define a tasks whose execution is expressed as the minimization of a non negative, possibly time-varying, continuously-differentiable cost function $C : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ with the following optimization problem:

[2] An extended class $\mathcal{K}_\infty$ is a function $\kappa : \mathbb{R} \to \mathbb{R}$ such that it belongs to class $\mathcal{K}$ and $\lim_{r \to \infty} \kappa(r) = \infty$

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}} \quad C(\boldsymbol{\sigma}, t), \\
&\text{s.t.} \\
&\qquad \dot{\mathbf{x}} = J(\mathbf{q})\dot{\mathbf{q}}, \\
&\qquad \boldsymbol{\sigma} = k(\mathbf{q}, t),
\end{aligned}
\tag{6.19}
$$

where $\boldsymbol{\sigma} \in \mathbb{R}^n$ is the output variable and represents the time-varying task variable.

Exploiting CBFs, the optimization problem (6.19) can be adapted to be computationally cheap, i.e. a convex optimization problem that can be solved online. Defining $\mathcal{C} \in \mathbb{R}^n$ as the subset where the task is considered executed, i.e. where $C(\boldsymbol{\sigma}, t) = 0$, it is possible to formulate a CBF $h : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ defined as $h(\boldsymbol{\sigma}, t) = -C(\boldsymbol{\sigma}, t)$. Since by definition h is non negative only in the region of execution of the task, i.e. when $C(\boldsymbol{\sigma}, t) = 0$, enforcing the non negativity of h is translated in the execution of the task $\boldsymbol{\sigma}$. This can be embedded in the following convex optimization problem:

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}} \quad \tfrac{1}{2}\|\dot{\mathbf{q}}_{des} - \dot{\mathbf{q}}\|^2 \\
&\text{s.t.} \\
&\qquad \frac{\partial h}{\partial t} + \frac{\partial h}{\partial \boldsymbol{\sigma}} \frac{\partial \boldsymbol{\sigma}}{\partial x} J(\mathbf{q})\dot{\mathbf{q}} \geq -\alpha(h(\boldsymbol{\sigma}, t)),
\end{aligned}
\tag{6.20}
$$

where $\dot{\mathbf{q}}_{des} \in \mathbb{R}^{n_b+n_m}$ are the desired robot velocities $\alpha(\cdot)$ is an extended class $\mathcal{K}$ function, i.e. a function $f : \mathbb{R} \to \mathbb{R}$ such that f is strictly increasing and $f(0) = 0$.

The formulation in (6.20) can be easily extended to the execution of multiple tasks. Considering a set of M different tasks $\{T_1, \ldots, T_M\}$ that the robot has to accomplish, each of which represented by a the respective cost functions $C_1, \ldots, C_M$, the execution of these tasks can be encoded in the following optimization problem:

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}, \boldsymbol{\delta}} \quad \|\dot{\mathbf{q}}\|^2 + \beta\|\boldsymbol{\delta}\|^2 \\
&\text{s.t.} \\
&\qquad \frac{\partial h_i}{\partial t} + \frac{\partial h_i}{\partial \boldsymbol{\sigma}} \frac{\partial \boldsymbol{\sigma}}{\partial x} J(\mathbf{q})\dot{\mathbf{q}} \geq \\
&\qquad\qquad -\alpha(h_i(\boldsymbol{\sigma}, t)) - \delta_i \qquad\qquad \forall i \in \{1, \ldots, M\},
\end{aligned}
\tag{6.21}
$$

where $h_i(\boldsymbol{\sigma}, t) = -C_i(\boldsymbol{\sigma}, t)$ and $\boldsymbol{\delta} = \begin{bmatrix}\delta_1, \ldots, \delta_M\end{bmatrix}$ is the vector of slack variables corresponding to each constraint, while $\beta \geq 0$ is a scaling factor. Each δ_i indicates how much the constraint corresponding to the i-th task can be relaxed. This is needed to to guarantee the feasibility of the problem when conflicting constraints are active at the same time.

6.2.4 Architecture

In this section the proposed control framework is detailed. Starting from the general formulation of the CBFs-based architecture, it is possible to define the tasks that the robot has to accomplish during the collaboration. In this work three different tasks have been considered:

- **Human Follow**. The robot has to follow the human operator inside the workspace. The size of the collaborative area can change at runtime based on the collaborative job that the two agents has to perform.
- **Safety**. The robot has to avoid the human operator during the execution of the collaborative job.
- **Mobile Base Orientation**. The orientation of the robot has to be adjusted so that the mobile base always faces the human operator.
- **Manipulability**. The configuration of the robot has to be controlled in order to always maintain a certain level of manipulability, avoiding singularities.

These tasks are further detailed in the following sections.

Human Follow

To ensure that the operator can work within the collaboration, the first task that must be carried out by the robot is to stay close to the man, following its movements within the workspace. This can be mathematically expressed as constraining the end-effector of the manipulator to be inside a sphere centered in a specific point of the human operator, e.g. the wrist. The situation is illustrated in Figure 6.9a.

This constraint can be expressed with the following CBF:

$$h_{Hf}(\mathbf{q}, t) = R(t) - \mathcal{D}(\mathbf{x}(\mathbf{q}), \mathbf{x}_H(t)), \tag{6.22}$$

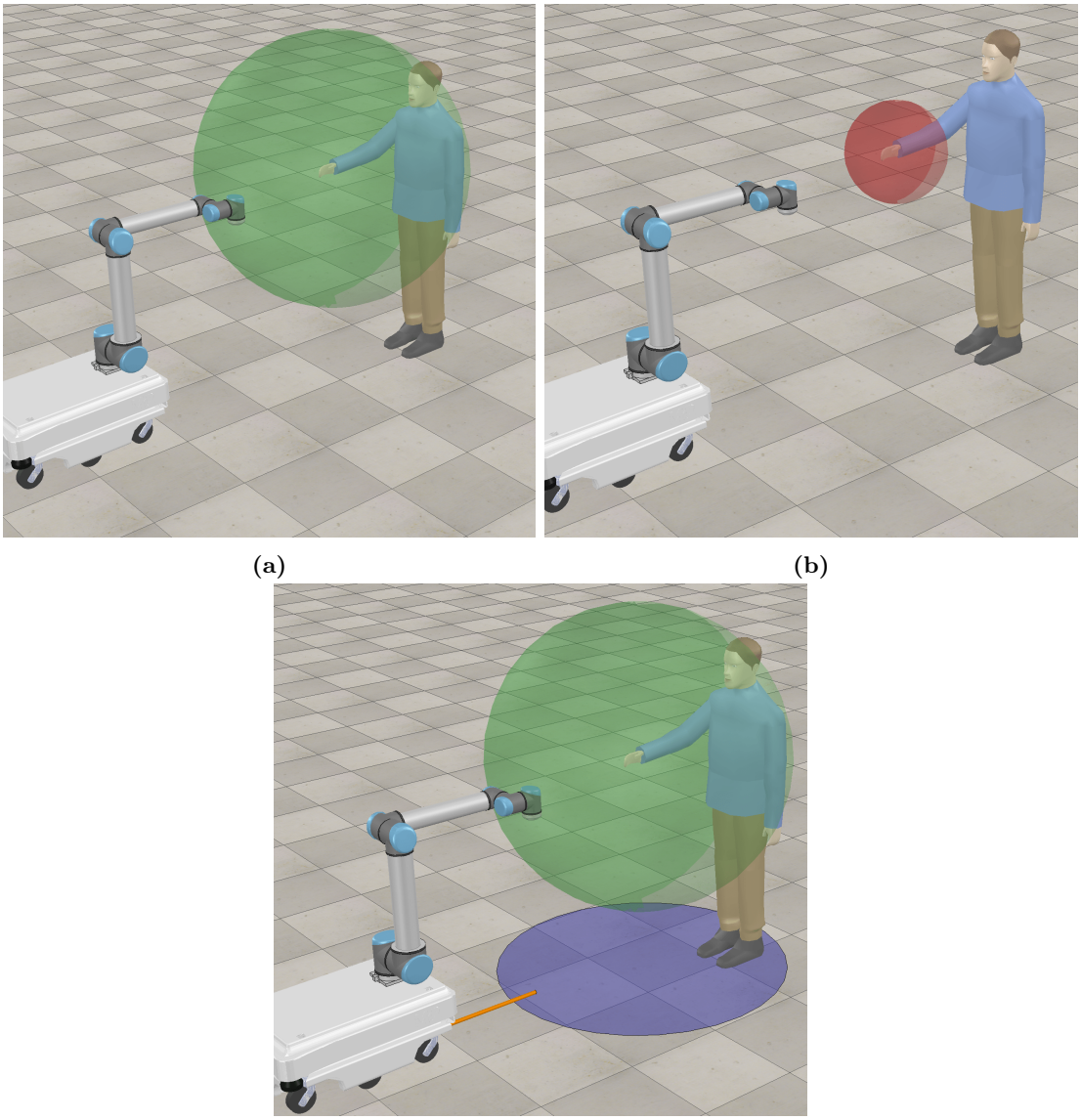
where $R(t) \in \mathbb{R}$ is the radius of the sphere, i.e. the size of the collaborative area, and can be changed at runtime depending on the task that the human-robot team has to accomplish. $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{x}_H \in \mathbb{R}^3$ are the cartesian positions of the manipulator end-effector and of the analyzed human point, respectively. Lastly, $\mathcal{D}(\mathbf{x}(\mathbf{q}), \mathbf{x}_H(t))$ is a function that computes the euclidean distance between the two positions.

In order to impose the CBF constraint it is necessary to differentiate (6.22), obtaining:

$$\dot{h}_{Hf}(\mathbf{q}, t) = \dot{R} - \frac{(\mathbf{x}(\mathbf{q}) - \mathbf{x}_H)^T}{\|\mathbf{x}(\mathbf{q}) - \mathbf{x}_H\|}(J(\mathbf{q})\dot{\mathbf{q}} - \dot{\mathbf{x}}_H). \tag{6.23}$$

Safety

During execution, it is crucial that the robot does not collide with the human operator, implementing collision avoidance behavior. As defined in Section 6.2.2, the robot configuration is considered safe if each link of the robot, both the manipulator and the mobile base, is at a safe distance from the human operator. Instead of considering each robot



(a) **(b)**

(c)

Figure 6.9: Graphical representation of the CBF-based constraints.

link, this relationship can be approximated by choosing N keypoints on the robot, i.e. the center of each link and the center of the mobile base, and imposing that the distance between these points and the operator is greater than the safety distance. Thus, it is possible to define N CBFs as:

$$h_{si}(\mathbf{q}, t) = \mathcal{D}(\mathbf{x}_i(\mathbf{q}), \mathbf{x}_H(t)) - d_{safe} \qquad \forall i \in \{1, \dots, N\}, \tag{6.24}$$

where $\mathbf{x}_i(\mathbf{q}) \in \mathbb{R}^3$ represents the cartesian position of the i-th keypoint and can be computed using partial forward kinematics [73]. A graphical representation of this constraint is shown in Figure 6.9b. It is worth noting that the graphical representation is the same for all N CBFs. This is because, unlike (6.22), the safe set of these CBFs is the entire workspace but the sphere representing the human operator.

Similarly to Section 6.2.4, it is possible to obtain the constraint computing the derivatives as:

$$\dot{h}_{si}(\mathbf{q}, t) = \frac{(\mathbf{x}_i(\mathbf{q}) - \mathbf{x}_H)^T}{\|\mathbf{x}_i(\mathbf{q}) - \mathbf{x}_H\|}(J_i(\mathbf{q})\dot{\mathbf{q}} - \dot{\mathbf{x}}_H) \qquad \forall i \in \{1, \dots, N\}. \tag{6.25}$$

Orientation

Ensuring that the manipulator end-effector stays close to the human operator is not enough to guarantee that the robot adopts optimal behavior. Since the mobile base is a differential drive robot it may happen that the robot is not capable to follow the human movements. A differential drive robot, indeed, is subject to non-holonomic constraints, i.e. constraints that forbid the lateral motion of the robot. For this reason it has been implemented a third CBF that allows to control the orientation of the mobile base, increasing the performances of the algorithm.

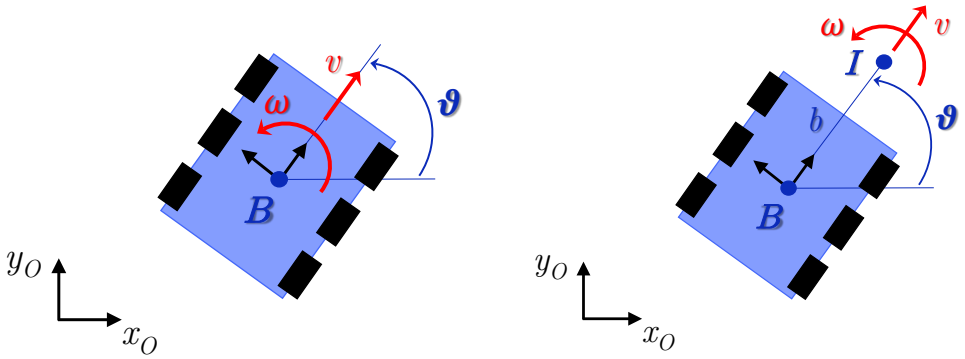
Considering only the mobile base, the differential-drive robot model can be expressed as:

$$\dot{\mathbf{x}}_B = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{6.26}$$

where $\dot{\mathbf{x}}_B = \begin{bmatrix} \dot{x}_B & \dot{y}_B & \dot{\theta} \end{bmatrix}$ is the vector of the cartesian velocity ($\dot{x}_B \in \mathbb{R}$, $\dot{y}_B \in \mathbb{R}$) and the angular velocity ($\dot{\theta} \in \mathbb{R}$) of the midpoint of the wheel axis B. $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ are the linear and angular velocity of the mobile robot and represents the control inputs, namely $\dot{\mathbf{q}}_b$ in (6.14). The robot model is shown in Figure 6.10a.

Instead of controlling the midpoint of the wheel axis, which must respect the non-holonomic constraints, it is possible to command a new reference point positioned on the normal axis of the passing wheels for B, at a distance b from it, see Figure 6.10b. This control strategy is called *Input–Output State Feedback Linearization* (I-O SFL)[128]:

$$\dot{\mathbf{x}}_I = \begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta} \end{bmatrix} = T_B^I \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} cos\theta & -bsin\theta \\ sin\theta & bcos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{6.27}$$



(a) **(b)**

Figure 6.10: Differential drive robot model.

Once the I-O SFL has been applied, it is possible to constrain the new control point I to remain within the projection on the plane of the sphere defined in Section 6.2.4. This solution allows to guarantee that the mobile base is always oriented towards the human operator, ensuring greater freedom of movement. The situation is illustrated in Figure 6.9c.

The CBF that represents this constraint is defined as:

$$h_o(\mathbf{q}_b, t) = R(t) - \mathbf{D}(\mathbf{x}_I(\mathbf{q}_b), \mathbf{x}_{HP}(t)), \tag{6.28}$$

where $\mathbf{x}_{HP} \in \mathbb{R}^2$ are the coordinates of the human point projected on the plane while $\mathbf{D}(\mathbf{x}_I, \mathbf{x}_{HP}(t))$ is the euclidean distance on the plane.

Differentiating (6.28) it is possible to obtain:

$$\dot{h}_o(\mathbf{q}_b, t) = \dot{R} - \frac{(\mathbf{x}_I(\mathbf{q}_b) - \mathbf{x}_{HP})^T}{\|\mathbf{x}_I(\mathbf{q}_b) - \mathbf{x}_{HP}\|}(J_b(\mathbf{q}_b)T_B^I\dot{\mathbf{x}}_I - \dot{\mathbf{x}}_{HP}). \tag{6.29}$$

Manipulability

To further improve the behavior of the robot it would be better to avoid singularity configurations, which could prevent the execution of the task. In general, for a robotic manipulator, it is possible to measure its ability to perform a specific task in a given configuration through the manipulability index [129]. This index indicates how easily the robot manipulator can move in all directions of the work area. From a geometric point of view it can be represented with an ellipsoid, called manipulability ellipsoid, whose axes are related to the eigenvalues of the jacobian. The larger the axes of this ellipsoid, the greater the ability of the robot to move in its respective direction. The index of manipulability can be defined as:

$$\mu = \sqrt{det(J_a J_a^T)}, \tag{6.30}$$

which is equal to 0 when the robot is in a singularity configuration, $det(J_a) = 0$. To avoid singularities it is sufficient to guarantee that:

$$\mu \geq \mu_{min}, \tag{6.31}$$

where μ_{min} is the minimum admissible value of the manipulability index and must be designed according to the task.

Computing and constraining this index online can be computationally demanding, since it is a very complex non linear function. Thus, in [130] the authors propose an approximated index with the same local minimum as (6.30):

$$\mu_{sim} = \frac{1}{2} \sum_{i=2}^{n_a-1} sin^2(\mathbf{q}_{a,i}), \tag{6.32}$$

where it is important to underline that the first and the last joint are not used in computing the index. This is because these two joints do not affect the singularity of the manipulator.

Imposing $h_m(\mathbf{q}_a, t) = \mu_{sim}$, it is possible to obtain:

$$\dot{h}_m(\mathbf{q}, t) = \sum_{i=2}^{n_a-1} sin(\mathbf{q}_{a,i}) cos(\mathbf{q}_{a,i}) \dot{\mathbf{q}}_{a,i}. \tag{6.33}$$

Final Problem

Starting from the definition in (6.21), it is possible to use all the constraints in order to build the following optimization problem:

$$\begin{aligned}
\min_{\dot{\mathbf{q}}, \boldsymbol{\delta}} \quad & \tfrac{1}{2} \dot{\tilde{\mathbf{q}}}^T \mathbf{W} \dot{\tilde{\mathbf{q}}} + \boldsymbol{\beta}^T \|\boldsymbol{\delta}\|^2 \\
\text{s.t.} \quad & \\
& \dot{h}_{Hf}(\mathbf{q}, t) \geq -\alpha_{Hf}(h_{Hf}(\mathbf{q}, t)) + \delta_{Hf}, \\
& \dot{h}_{si}(\mathbf{q}, t) \geq -\alpha_{si}(h_{si}(\mathbf{q}, t)) - \delta_{si} \qquad \forall i \in \{1, \ldots, N\}, \\
& \dot{h}_o(\mathbf{q}_b, t) \geq -\alpha_o(h_o(\mathbf{q}_b, t)) - \delta_o, \\
& \dot{h}_m(\mathbf{q}_a, t) \geq -\alpha_m(h_m(\mathbf{q}_a, t)) - \delta_m, \\
& \mathbf{q}_{min} \leq \mathbf{q}_{act} + \dot{\mathbf{q}} T \leq \mathbf{q}_{max}, \\
& \dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{max}, \\
& \ddot{\mathbf{q}}_{min} \leq \frac{\dot{\mathbf{q}} - \dot{\mathbf{q}}_{act}}{T} \leq \ddot{\mathbf{q}}_{max},
\end{aligned} \tag{6.34}$$

where $\tilde{\mathbf{q}} = \mathbf{q}_{des} - \mathbf{q}$. $\mathbf{W} = diag(\mathbf{w}_b, \mathbf{w}_m)$, with $\mathbf{w}_b \in \mathbb{R}^{n_b}$ and $\mathbf{w}_m \in \mathbb{R}^{n_a}$, is the weighted matrix associated to the movement of the mobile base and the robot manipulator. This matrix is exploited to give different priority of movement to the different robot DOF-s. $\boldsymbol{\beta} \in \mathbb{R}^{N+3}$, instead, represents the weights associated to the slack variables, necessary to

Figure 6.11: Setup of the experiments.

prioritize the tasks. $\dot{\mathbf{q}}_{des} \in \mathbb{R}^{n_b+n_a}$ represents the desired robot velocities. $\mathbf{q}_{min} \in \mathbb{R}^{n_b+n_a}$ and $\mathbf{q}_{max} \in \mathbb{R}^{n_b+n_a}$ are the position limits and T is the cycle time. $\dot{\mathbf{q}}_{min} \in \mathbb{R}^{n_b+n_a}$ and $\dot{\mathbf{q}}_{max} \in \mathbb{R}^{n_b+n_a}$ are the joint velocity lower bounds and the joint velocity upper bounds, respectively. While $\ddot{\mathbf{q}}_{min} \in \mathbb{R}^{n_b+n_a}$ and $\ddot{\mathbf{q}}_{max} \in \mathbb{R}^{n_b+n_a}$ are the acceleration limits. Lastly, $\mathbf{q}_{act} \in \mathbb{R}^{n_b+n_a}$ and $\dot{\mathbf{q}}_{act} \in \mathbb{R}^{n_b+n_a}$ are the actual robot positions and velocity, respectively.

The first four constraints are the four CBF tasks, while the last three constraints represent the robot kinematic limits, i.e. minimum and maximum velocities and accelerations.

The optimization problem (6.34) is a convex problem and, therefore, computationally cheap. Thanks to its convexity, the solution obtained by the solver is always the global minimum of the cost function, i.e. the robot velocities that allow you to perform the job in an optimal way, maximizing the performance of the HRC. Moreover, thanks to the use of slack variables $\boldsymbol{\delta}$, the optimization problem has always a feasible solution. It is worth noting that, since it relies on the use of CBFs, the optimization problem can be easily extended to perform other tasks, making the architecture very flexible.

6.2.5 Validation

The proposed architecture has been experimentally validated in a HRC scenario. During the experiments, the human operator cooperates with a UR10e, a 6-DoF collaborative robot, which has been mounted on a MIR100, a differential-drive collaborative mobile base. To track the human position eight OptiTrack Prime^{x} cameras with the OptiTrack Motive software[74] have been used, while the size of the collaborative area is changed at runtime exploiting an Alexa virtual assistant [131]. The complete setup for the experiments is shown in Figure 6.11.

All the software components were developed using ROS Melodic Morenia. The CBF-based optimization problem has been implemented exploiting the C code generated by Matlab Coder [132]. Concerning the frequencies, the communication with the robot works at 500 Hz while the optimization problem converges in 1 ms. The OptiTrack,

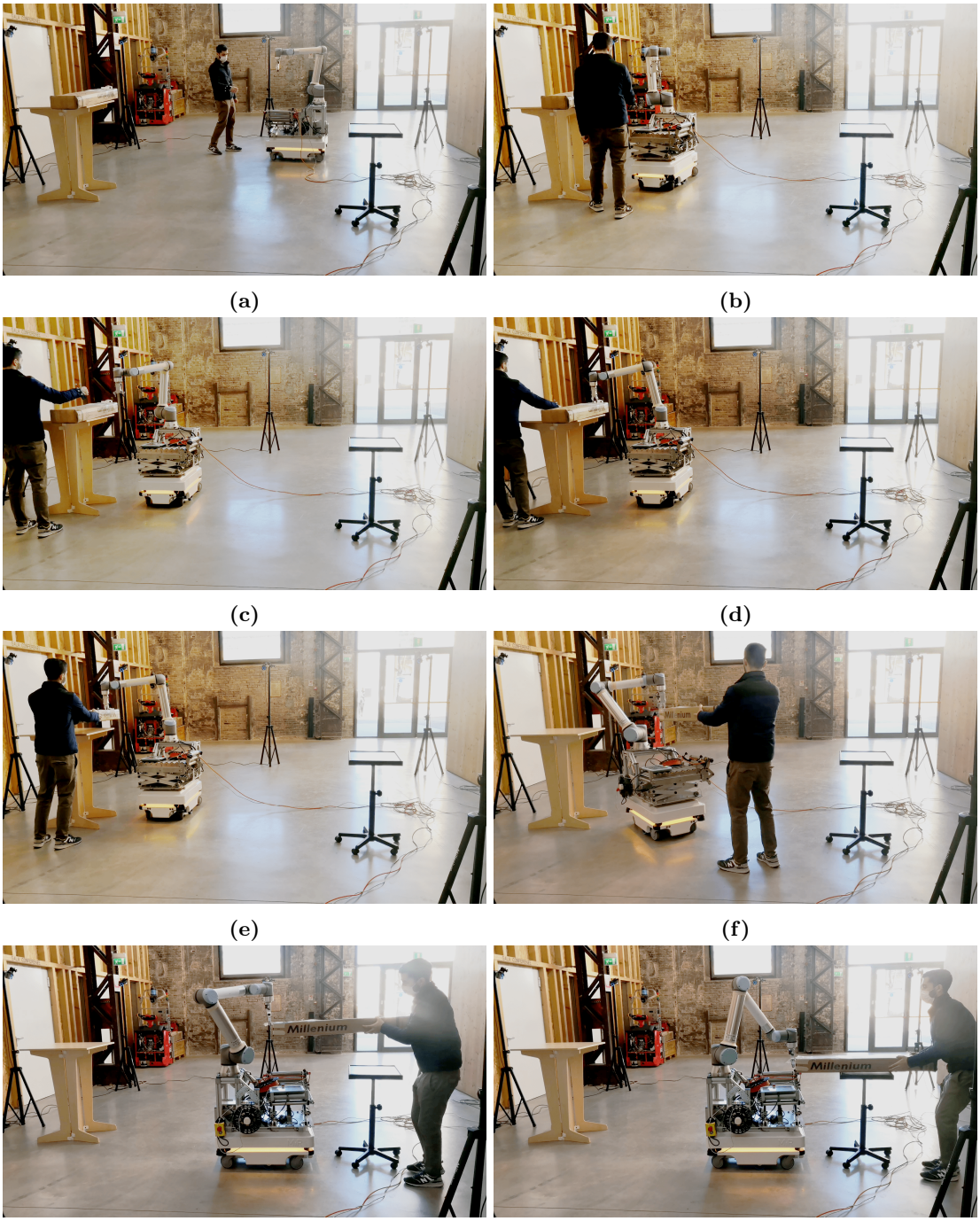


Figure 6.12: Snapshots of the experiment.

instead, works at a frequency of 240 Hz.

In the experiment the human and the robot have to perform a collaborative transportation, whose snapshots are shown in Figure 6.12. During this experiment, the optimization problem in (6.34) has been extended with an orientation constraint for the manipulator. This was necessary for the nature of the experiment, but could be ideally avoided by using a proper control strategy that generates a proper $\dot{\mathbf{q}}_{des}$. Initially, the two actors are very far from the load to be picked. For this reason, the human operator starts approaching the object, see Figure 6.12a. In this approaching phase, it is preferable to move only the mobile base, making the most out of the overall platform. This is achieved by

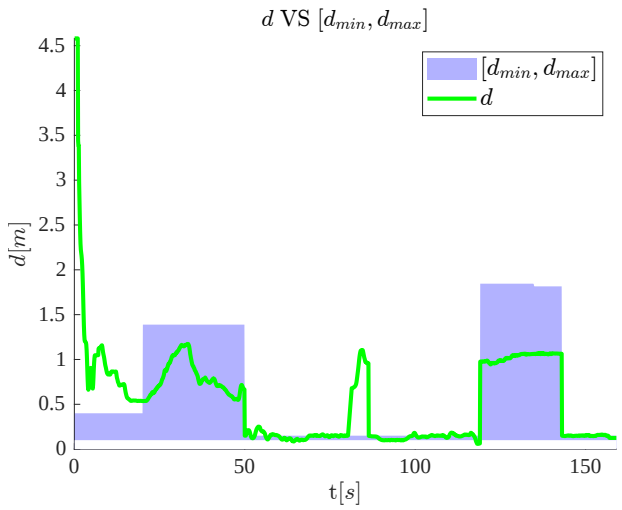


Figure 6.13: Plot of the CBF at the robot end-effector.

setting $\mathbf{w}_m \gg \mathbf{w}_b$. Subsequently, the human operator guides the robot to pick the object, see Figure 6.12b. It is worth noting that in this phase, only the mobile manipulator is moved and to achieve a better control the CBF defined in (6.22) is initially centered in the robot end-effector and follows the human operator movements. After the robot has picked up the object, the human operator begins to lift the load and, using the same control used for the picking phase, the operator gets help from the robot. This is illustrated in Figure 6.12c - 6.12d. Once the load has been completely lifted, the human operator starts approaching the placing location and the mobile base follows consequently. See Figure 6.12e After reaching the desired position, the human operator starts to place the object while the robot helps. This is shown in Figure 6.12g - 6.12h. It is worth noting that the weights associated to the slack variables, $\boldsymbol{\beta}$, are equal to 1.

A graph demonstrating that the robot end-effector stays inside the safe set is illustrated in Figure 6.13, where the blu area represent the safe set, while the green line represent the distance between the ee and the center of the two spheres, i.e. the human wrist. There are two moments in which the distance is outside the boundaries, in the first phase and around $t = 85\ s$. This is due to the fact that the control was momentarily deactivated, i.e. in the initial phase and while the human operator was picking the load.

Chapter 7

Conclusions and Future Directions

This thesis work addresses some problems that occur in the case of a human–robot collaboration scenario. In particular, it focuses on two major aspects: a more natural and flexible task planning strategy and an optimal safety-aware trajectory control strategy. These are also the problems that had been identified in the submission phase of the ROSSINI European project.

In the first part of the thesis the development and validation of the cognitive layer is reported. The aim of this layer is to enhance the HRC creating a synergy between the human operator and the robot. This is achieved firstly with the development of a new task allocation strategy that embeds job quality metrics to make the most out of the human–robot team. Subsequently, a proper dynamic scheduling algorithm takes care of compensating for deviations from the nominal behaviour.

The next part regards the development and validation of the flexible execution and safety layers. The aim of this part of the project is to improve the robot behaviour while always ensuring the safety. Initially, a collision-free path is planned and eventually replanned ensuring always a collision-free behaviour. Subsequently, a proper trajectory scaling strategy adapts the speed of the robot along the path without never violating the safety constraints imposed by the regulations.

After this layers have been completed, the thesis address the problem of the integration between them and also with the other technologies developed by the partners. This has lead to the creation of the safety aware control architecture, which allows to solve a task allocation and motion planning problem. The experimental validations have been done both in customs setups and in the ROSSINI use cases, demonstrating that the proposed strategies may be applied to most of the collaborative industrial application.

The last part focuses on research activities that have gone beyond the ROSSINI project, but which can be used both as a support to the technologies developed and as a substantial improvement of them. The problem of achieving a more resilient task scheduling strategy that is also able to adapt to the human operator is crucial for a natural collaboration. In fact, experienced users may prefer to work differently than inexperienced users. This can also be reflected in the robot's behavior, making it more predictable for

the human operator it is collaborating with.

The results of this thesis demonstrate how important and crucial it is to develop a modular and integrable architecture that allows both task allocation and execution. The results, which were also obtained in industrial scenarios, were excellent and demonstrate how the ROSSINI architecture is applicable to completely different use cases. Furthermore, the ROSSINI project was a first step towards a fully customizable architecture, whose behaviour depends on the individual operator creating a custom experience. Indeed, thanks to the use of job quality metrics within the task assignment and the rescheduling strategy based on execution time, with the ROSSINI project it is the robot that adapts to the operator in front of it and not vice versa, as is often the case. This has been further enhanced by an appropriate control strategy that is largely flexible, capable of both replanning the path to avoid collisions with the human operator and to adapt the velocity online to ensure safety.Subsequent work is further steps towards this customised experience. The resilient architecture modifies the tasks to be assigned according to the operator's capabilities, while the trajectory planning and control architectures presented in the last chapter allows predictable behaviour for the individual operator. The importance of this can also be seen from the fact that other researchers are working in this field, e.g. by taking emotions into account [103, 133]. From the experience gained in this PhD, I believe that the big step to be taken is to create a definitive architecture that can accept all these inputs and self-adapt online to the human operator, perhaps by exploiting also artificial intelligence techniques. This could see a radical change in the paradigm of collaborative robotics, which could take it to another level. In fact, if this were the case, the diffusion of collaborative robotics would not stop only at industrial scenarios, but could arrive inside our homes to help us carry out some tasks or to perform others for us.

Bibliography

[1] A. Pupa, M. Arrfou, G. Andreoni, and C. Secchi, "A safety-aware kinodynamic architecture for human-robot collaboration," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4465–4471, 2021.

[2] A. Pupa, W. Van Dijk, and C. Secchi, "A human-centered dynamic scheduling architecture for collaborative application," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4736–4743, 2021.

[3] A. Pupa, W. Van Dijk, C. Brekelmans, and C. Secchi, "A resilient and effective task scheduling approach for industrial human-robot collaboration," *Sensors*, vol. 22, no. 13, p. 4901, 2022.

[4] A. Pupa, M. Arrfou, G. Andreoni, and C. Secchi, "A human-centered dynamic task scheduling and safe task execution approach for human-robot collaboration scenarios," in *Human-Robot Collaboration: Unlocking the potential for industrial applications*, 2023.

[5] A. Pupa, C. Talignani Landi, M. Bertolani, and C. Secchi, "A dynamic architecture for task assignment and scheduling for collaborative robotic cells," in *Proceedings of the 13th International Workshop on Human-Friendly Robotics*, 2020.

[6] A. Pupa and C. Secchi, "A safety-aware architecture for task scheduling and execution for human-robot collaboration," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1895–1902.

[7] A. Pupa and C. Secchi, "A human-centered dynamic task planning approach for human-robot collaboration," in *2021 I-RIM Conference*. I-RIM, 2021, pp. 41–43.

[8] A. Pupa, F. Breveglieri, and C. Secchi, "An optimal human-based control approach for mobile human-robot collaboration," in *Proceedings of the 15th International Workshop on Human-Friendly Robotics*, 2022.

[9] A. Pupa, W. Van Dijk, and C. Secchi, "Towards an effective human-robot team collaboration – a resilient task scheduling approach for real industrial scenarios," in *2022 I-RIM Conference*. I-RIM, 2022.

[10] A. Pupa, M. Minelli, and C. Secchi, "A dynamic planner for safe and predictable human robot collaboration," in *Submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[11] C. James E. and P. Michael A., "Cobots," U.S. Patent 5 952 796, 1997.

[12] "A history of collaborative robots: From intelligent lift assists to cobots," https://www.engineering.com/story/a-history-of-collaborative-robots-from-intelligent-lift-assists-to-cobots, accessed: 2012-12-08.

[13] "History of the dlr lwr," https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-12464/21732_read-44586/, accessed: 2012-12-08.

[14] "More applications, the familiar simplicity: the new lbr iisy cobots," https://www.kuka.com/en-cn/company/press/news/2022/06/lbr-iisy-cobot-robot, accessed: 2012-12-08.

[15] "This made-in-taiwan robot is drawing international attention," https://english.cw.com.tw/article/article.action?id=2436, accessed: 2012-12-08.

[16] "A brief history of collaborative robots," https://www.mhlnews.com/technology-automation/article/21124077/a-brief-history-of-collaborative-robots, accessed: 2012-12-08.

[17] "Universal robots' new ur20 collaborative robot makes u.s. debut at imts 2022, expanding cobot automation in machining industry," https://www.universal-robots.com/about-universal-robots/news-centre/universal-robots-new-ur20-collaborative-robot-makes-us-debut-at-imts-2022/, accessed: 2012-12-08.

[18] "Fanuc unveils a world first collaborative robot," https://www.themanufacturer.com/articles/fanuc-unveils-a-world-first-collaborative-robot/, accessed: 2012-12-08.

[19] "Abb presents the world's first truly collaborative dual armed robot, yumi® at metalex vietnam 2015," https://new.abb.com/news/detail/28304/abb-presents-the-worlds-first-truly-collaborative-dual-armed-robot-yumir-at-metalex-vietnam-2015, accessed: 2012-12-08.

[20] "Franka emika panda in 'time's best inventions of 2018'," https://wiredworkers.io/franka-emika-panda-in-times-best-inventions-of-2018/, accessed: 2012-12-08.

[21] "Collaborative robot (cobot) market report 2022: High return on investment in collaborative robots compared to traditional industrial robotic systems expected to fuel growth," https://www.businesswire.com/news/home/20221129005915/en/Collaborative-Robot-Cobot-Market-Report-2022-High-Return-on-Investment-in-Collaborative-Robots-Compared-to-Traditional-Industrial-Robotic-Systems-Expected-to-Fuel-Growth---ResearchAndMarkets.com, accessed: 2012-12-08.

[22] L. G. Terveen, "Overview of human-computer collaboration," *Knowledge-Based Systems*, vol. 8, no. 2-3, pp. 67–81, 1995.

[23] "Rossini consortium," https://www.rossini-project.com/partners-2, accessed: 2012-12-08.

[24] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[25] Q. Pham, R. Madhavan, L. Righetti, W. Smart, and R. Chatila, "The impact of robotics and automation on working conditions and employment," *IEEE Robotics & Automation Magazine*, vol. 25, no. 2, pp. 126–128, 2018.

[26] Organisation for Economic Cooperation and Development, "Inventory for job quality," *OECD Employment and Labour Market Statistics*, 2016.

[27] National Institute for Occupational Safety and Health, *Work practices guide for manual lifting*. Cincinnati, Ohio: US Department of Health and Human Services, Public Health Service, CDC, 1981, no. 81-122.

[28] Council of European Union, "Directive 2003/10/ec of the european parliament and of the council of 6 february 2003 on the minimum health and safety requirements regarding the exposure of workers to the risks arising from physical agents (noise) (seventeenth individual directive within the meaning of article 16(1) of directive 89/391/eec)," *Official Journal of the European Union*, vol. 42, pp. 38–44, 2003.

[29] F. Wixted and L. O'Sullivan, "The effect of automated manufacturing environments on employee health," in *Proceedings of the Irish Ergonomics Society Annual Conference*, vol. 1, 2014, pp. 80–91.

[30] G. Hoffman, "Evaluating fluency in human–robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 3, pp. 209–218, 2019.

[31] C. Blum and C. Miralles, "On solving the assembly line worker assignment and balancing problem via beam search," *Computers & Operations Research*, vol. 38, no. 1, pp. 328–339, 2011.

[32] K. Xu, R. Fei, and D. He, "A tabu-search algorithm for scheduling jobs with precedence constraints on parallel machines," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2018, pp. 2774–2781.

[33] L. Sabattini, V. Digani, M. Lucchi, C. Secchi, and C. Fantuzzi, "Mission assignment for multi-vehicle systems in industrial environments," *IFAC-PapersOnLine*, vol. 48, no. 19, pp. 268–273, 2015.

[34] G. Michalos, J. Spiliotopoulos, S. Makris, and G. Chryssolouris, "A method for planning human robot shared tasks," *CIRP journal of manufacturing science and technology*, vol. 22, pp. 76–90, 2018.

[35] K. Li, Q. Liu, W. Xu, J. Liu, Z. Zhou, and H. Feng, "Sequence planning considering human fatigue for human-robot collaboration in disassembly," *Procedia CIRP*, vol. 83, pp. 95–104, 2019.

[36] A. Ayough, M. Zandieh, and F. Farhadi, "Balancing, sequencing, and job rotation scheduling of a u-shaped lean cell with dynamic operator performance," *Computers & Industrial Engineering*, p. 106363, 2020.

[37] K. Bogner, U. Pferschy, R. Unterberger, and H. Zeiner, "Optimised scheduling in human–robot collaboration–a use case in the assembly of printed circuit boards," *International Journal of Production Research*, vol. 56, no. 16, pp. 5522–5540, 2018.

[38] S. M. Rahman and Y. Wang, "Mutual trust-based subtask allocation for human–robot collaboration in flexible lightweight assembly in manufacturing," *Mechatronics*, vol. 54, pp. 94–109, 2018.

[39] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris, "Dynamic scheduling of shared human-robot manufacturing operations," *Procedia CIRP*, vol. 72, pp. 9–14, 2018.

[40] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, 2016.

[41] P. Lou, Q. Liu, Z. Zhou, H. Wang, and S. X. Sun, "Multi-agent-based proactive–reactive scheduling for a job shop," *The International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1-4, pp. 311–324, 2012.

[42] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco, "Optimal scheduling of human-robot collaborative assembly operations with time petri nets," *IEEE Transactions on Automation Science and Engineering*, 2019.

[43] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2641–2648.

[44] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human–robot collaboration," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1579–1594, 2017.

[45] R. Maderna, P. Lanfredini, A. M. Zanchettin, and P. Rocco, "Real-time monitoring of human task advancement," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)*. IEEE, 2019, pp. 433–440.

[46] S. P. Marshall, "The index of cognitive activity: Measuring cognitive workload," in *Proceedings of the IEEE 7th conference on Human Factors and Power Plants*. IEEE, 2002, pp. 7–7.

[47] E. Lamon, A. De Franco, L. Peternel, and A. Ajoudani, "A capability-aware role allocation approach to industrial assembly tasks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3378–3385, 2019.

[48] Z. Liu, X. Wang, Y. Cai, W. Xu, Q. Liu, Z. Zhou, and D. T. Pham, "Dynamic risk assessment and active response strategy for industrial human-robot collaboration," *Computers & Industrial Engineering*, vol. 141, p. 106302, 2020.

[49] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[50] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial intelligence in medicine*, vol. 45, no. 1, pp. 11–34, 2009.

[51] "Python-mip," https://www.python-mip.com/, accessed: 2012-12-08.

[52] "Gurobi," https://www.gurobi.com/, accessed: 2012-12-08.

[53] *ISO 10218-1:2011(E). Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 1: Robots*, International Organization for Standardization Std., July 2011.

[54] *ISO 10218-2:2011(E). Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 2: Robot systems and integration*, International Organization for Standardization Std., July 2011.

[55] *ISO/TS 15066:2016(E). Robots and robotic devices–Collaborative robots*, International Organization for Standardization Technical Specification, Feb. 2016.

[56] M. Ragaglia, A. M. Zanchettin, and P. Rocco, "Safety-aware trajectory scaling for human-robot collaboration with prediction of human occupancy," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 85–90.

[57] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, 2015.

[58] M. Lippi and A. Marino, "Human multi-robot safe interaction: A trajectory scaling approach based on safety assessment," *IEEE Transactions on Control Systems Technology*, pp. 1–16, 2020.

[59] A. Levratti, G. Riggio, C. Fantuzzi, A. De Vuono, and C. Secchi, "Tirebot: A collaborative robot for the tire workshop," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 129–137, 2019.

[60] J.-H. Chen and K.-T. Song, "Collision-free motion planning for human-robot collaborative safety under cartesian constraint," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[61] F. Ferraguti, N. Preda, M. Bonfe, and C. Secchi, "Bilateral teleoperation of a dual arms surgical robot with passive virtual fixtures generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4223–4228.

[62] H.-C. Lin, C. Liu, Y. Fan, and M. Tomizuka, "Real-time collision avoidance algorithm on industrial manipulators," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 1294–1299.

[63] F. Ferraguti, M. Bertuletti, C. T. Landi, M. Bonfè, C. Fantuzzi, and C. Secchi, "A control barrier function approach for maximizing performance while fulfilling to

iso/ts 15066 regulations," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5921–5928, 2020.

[64] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[65] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[66] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3713–3719.

[67] S. Moon, Y. Park, D. W. Ko, and I. H. Suh, "Multiple kinect sensor fusion for human skeleton tracking using kalman filtering," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, pp. 65–74, 2016.

[68] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE transactions on industrial electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.

[69] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1610–1623, 2010.

[70] S. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[71] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 2. IEEE, 2004, pp. 1606–1611.

[72] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[73] F. Ferraguti, C. T. Landi, S. Costi, M. Bonfè, S. Farsoni, C. Secchi, and C. Fantuzzi, "Safety barrier functions and multi-camera tracking for human–robot shared environment," *Robotics and Autonomous Systems*, vol. 124, pp. 103 388–103 406, 2020.

[74] "Optitrack - motive," https://optitrack.com/software/motive/, accessed: 2012-12-08.

[75] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[76] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *Journal of Software Engineering for Robotics*, vol. 5(1), pp. 3–16, 2014.

[77] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 12, no. 1, pp. 1–27, 2012.

[78] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 639–646.

[79] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[80] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[81] F. Fabrizio and A. De Luca, "Real-time computation of distance to dynamic obstacles with multiple depth sensors," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 56–63, 2016.

[82] F. Benzi and C. Secchi, "An optimization approach for a robust and flexible control in collaborative applications," *arXiv preprint arXiv:2103.03082*, 2021.

[83] K. Merckaert, B. Convens, C.-j. Wu, A. Roncone, M. M. Nicotra, and B. Vanderborght, "Real-time motion control of robotic manipulators for safe human–robot coexistence," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102223, 2022.

[84] T. B. Sheridan, "Human-Robot Interaction: Status and Challenges," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 58, no. 4, pp. 525–532, 2016.

[85] Y. He, Z. Shao, B. Xiao, Q. Zhuge, and E. Sha, "Reliability driven task scheduling for heterogeneous systems." in *Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems*, vol. 1, 2003, pp. 465–470.

[86] X. Qin and T. Xie, "An availability-aware task scheduling strategy for heterogeneous systems," *IEEE Transactions on Computers*, vol. 57, no. 2, pp. 188–199, 2008.

[87] I. El Makrini, K. Merckaert, J. De Winter, D. Lefeber, and B. Vanderborght, "Task allocation for improved ergonomics in human-robot collaborative assembly," *Interaction Studies*, vol. 20, no. 1, pp. 102–133, 2019.

[88] F. Pratissoli, N. Battilani, C. Fantuzzi, and L. Sabattini, "Hierarchical and flexible traffic management of multi-agv systems applied to industrial environments," in

2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 10 009–10 015.

[89] G. Klien, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich, "Ten challenges for making automation a" team player" in joint human-agent activity," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 91–95, 2004.

[90] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, "Coactive design: Designing support for interdependence in joint activity," *Journal of Human-Robot Interaction*, vol. 3, no. 1, pp. 43–69, 2014.

[91] K. Ramamritham, "Allocation and scheduling of precedence-related periodic tasks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 4, pp. 412–420, 1995.

[92] B. Gacias, C. Artigues, and P. Lopez, "Parallel machine scheduling with precedence constraints and setup times," *Computers & Operations Research*, vol. 37, no. 12, pp. 2141–2151, 2010.

[93] K. Zhang, B. Qi, Q. Jiang, and L. Tang, "Real-time periodic task scheduling considering load-balance in multiprocessor environment," in *2012 3rd IEEE international conference on network infrastructure and digital content*. IEEE, 2012, pp. 247–250.

[94] T. Yu, J. Huang, and Q. Chang, "Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning," *Journal of Manufacturing Systems*, vol. 60, pp. 487–499, 2021.

[95] C. Ferreira, G. Figueira, and P. Amorim, "Scheduling human-robot teams in collaborative working cells," *International Journal of Production Economics*, vol. 235, p. 108094, 2021.

[96] A. Nourmohammadi, M. Fathi, and A. H. Ng, "Balancing and scheduling assembly lines with human-robot collaboration tasks," *Computers & Operations Research*, vol. 140, p. 105674, 2022.

[97] A. Raatz, S. Blankemeyer, T. Recker, D. Pischke, and P. Nyhuis, "Task scheduling method for hrc workplaces based on capabilities and execution time assumptions for robots," *CIRP Annals*, vol. 69, no. 1, pp. 13–16, 2020.

[98] M. Zhang, C. Li, Y. Shang, and Z. Liu, "Cycle time and human fatigue minimization for human-robot collaborative assembly cell," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6147–6154, 2022.

[99] L. H. De Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 56–57.

[100] L. H. De Mello and A. C. Sanderson, "And/or graph representation of assembly plans," *IEEE Transactions on robotics and automation*, vol. 6, no. 2, pp. 188–199, 1990.

[101] R. Canham, A. H. Jackson, and A. Tyrrell, "Robot error detection using an artificial immune system," in *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.* IEEE, 2003, pp. 199–207.

[102] P. Trung, M. Giuliani, M. Miksch, G. Stollnberger, S. Stadler, N. Mirnig, and M. Tscheligi, "Head and shoulders: automatic error detection in human-robot interaction," in *Proceedings of the 19th ACM international conference on multimodal interaction*, 2017, pp. 181–188.

[103] R. J. Kirschner, H. Mayer, L. Burr, N. Mansfeld, S. Abdolshah, and S. Haddadin, "Expectable motion unit: Avoiding hazards from human involuntary motions in human-robot interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2993–3000, 2022.

[104] M. Eckhoff, R. J. Kirschner, E. Kern, S. Abdolshah, and S. Haddadin, "An mpc framework for planning safe & trustworthy robot motions," in *2022 International Conference on Robotics and Automation (ICRA).* IEEE, 2022, pp. 4737–4742.

[105] S. Baltrusch, F. Krause, A. de Vries, W. van Dijk, and M. de Looze, "What about the human in human robot collaboration? a literature review on hrc's effects on aspects of job quality," *Ergonomics*, vol. 65, no. 5, pp. 719–740, 2022.

[106] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robotics: Science and Systems VIII*, pp. 1–8, 2012.

[107] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *2001 European control conference (ECC).* IEEE, 2001, pp. 2603–2608.

[108] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[109] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation.* IEEE, 2012, pp. 3859–3866.

[110] M. Awais and D. Henrich, "Human-robot collaboration by intention recognition using probabilistic state machines," in *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*, 2010, pp. 75–80.

[111] Y. Li and S. S. Ge, "Human–robot collaboration based on motion intention estimation," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 1007–1014, 2013.

[112] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," in *[1992] Proceedings of the 31st IEEE Conference on Decision and Control.* IEEE, 1992, pp. 2643–2648.

[113] C. Hu, W. Chen, J. Wang, and H. Wang, "Optimal path planning for mobile manipulator based on manipulability and localizability," in *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2016, pp. 638–643.

[114] J. Varela-Aldás, V. H. Andaluz, and F. A. Chicaiza, "Modelling and control of a mobile manipulator for trajectory tracking," in *2018 International Conference on Information Systems and Computer Science (INCISCOS)*. IEEE, 2018, pp. 69–74.

[115] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constrained model predictive control for mobile robotic manipulators," *Robotica*, vol. 36, no. 1, pp. 19–38, 2018.

[116] A. H. Khan, S. Li, D. Chen, and L. Liao, "Tracking control of redundant mobile manipulator: An rnn based metaheuristic approach," *Neurocomputing*, vol. 400, pp. 272–284, 2020.

[117] S. Thakar, A. Kabir, P. M. Bhatt, R. K. Malhan, P. Rajendran, B. C. Shah, and S. K. Gupta, "Task assignment and motion planning for bi-manual mobile manipulation," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 910–915.

[118] H. Karami, K. Darvish, and F. Mastrogiovanni, "A task allocation approach for human-robot collaboration in product defects inspection scenarios," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1127–1134.

[119] B. Navarro, A. Cherubini, A. Fonte, G. Poisson, and P. Fraisse, "A framework for intuitive collaboration with a mobile manipulator," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6293–6298.

[120] G. Gillini, P. Di Lillo, F. Arrichiello, D. Di Vito, A. Marino, G. Antonelli, and S. Chiaverini, "A dual-arm mobile robot system performing assistive tasks operated via p300-based brain computer interface," *Industrial Robot: the international journal of robotics research and application*, 2020.

[121] H. Zhou, H. Min, Y. Lin, and S. Zhang, "A robot architecture of hierarchical finite state machine for autonomous mobile manipulator," in *International Conference on Intelligent Robotics and Applications*. Springer, 2017, pp. 425–436.

[122] F. Ferraguti, C. Talignani Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfe, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, pp. 2–14, 2022.

[123] M. Sorour, A. Cherubini, and P. Fraisse, "Motion control for steerable wheeled mobile manipulation," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–7.

[124] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[125] G. Notomista, S. Mayya, M. Selvaggio, M. Santos, and C. Secchi, "A set-theoretic approach to multi-task execution and prioritization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9873–9879.

[126] G. Notomista and M. Egerstedt, "Constraint-driven coordinated control of multi-robot systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1990–1996.

[127] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 2071–2076.

[128] B. d'Andréa Novel, G. Campion, and G. Bastin, "Control of nonholonomic wheeled mobile robots by state feedback linearization," *The International journal of robotics research*, vol. 14, no. 6, pp. 543–559, 1995.

[129] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[130] F. Caccavale, V. Lippiello, G. Muscio, F. Pierri, F. Ruggiero, and L. Villani, "Grasp planning and parallel control of a redundant dual-arm/hand manipulation system," *Robotica*, vol. 31, no. 7, pp. 1169–1194, 2013.

[131] D. Ferrari, F. Benzi, and C. Secchi, "Bidirectional communication control for human-robot collaboration," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7430–7436.

[132] "Matlab coder," https://mathworks.com/products/matlab-coder.html, accessed: 2012-12-08.

[133] M. Lagomarsino, M. Lorenzini, E. De Momi, and A. Ajoudani, "An online framework for cognitive load assessment in industrial tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102380, 2022.