Enhancing Performance in Human–Robot Collaboration: A Modular Architecture for Task Scheduling and Safe Trajectory Planning

Enhancing Performance in Human-Robot Collaboration: A Modular Architecture for Task Scheduling and Safe Trajectory Planning

Andrea Pupa[1], Simone Comari[2], Mohammad Arrfou[3], Gildo Andreoni[3], Alessandro Carapia[4], Marco Carricato[2], and Cristian Secchi[1]

***Abstract*—The integration of robots into shared workspaces alongside humans is the basis of Human-Robot Collaboration (HRC). This field of research has changed the paradigm of the industrial context, making HRC of pivotal importance for both researchers and the industry. In this context, a suitable task scheduling and trajectory planning strategy are crucial to achieve good performances and create a synergy between the two actors. Indeed, the task scheduling should be able to optimally distribute the tasks between the actors and recover from possible failures, i.e. by rescheduling the tasks. The trajectory planning strategy must comply with the safety standards that impose a reduction of velocity based on human behaviour. To this end, the monitoring system must also be safe-certified; otherwise, safety cannot be guaranteed. This paper proposes a novel architecture that integrates a dynamic task scheduling module with a dynamic trajectory planning module that explicitly considers ISO/TS 15066. For this purpose, the framework exploits a secure and certified monitoring system capable of tracking the human operator even in case of occlusions. The overall platform has been extensively validated both in a real and complex industrial scenario within the context of the ROSSINI EU project, where a dual-arm mobile robot collaborates with a human operator in an automatic machine-tending operation, and in a mock-up scenario.**

***Note to Practitioners*—This paper focuses on the implementation of a modular and integrated architecture to handle the challenges associated with effective task scheduling and trajectory planning, with the aim of improving the performances of collaborative robots in real HRC scenarios. The proposed methodology is structured into three distinct layers. Firstly, the safety layer monitors in real-time the human operator and provides the other layers with the maximum admissible robot speed. Secondly, the scheduling layer distributes tasks between humans and robots efficiently, recovering from possible failures and rescheduling the tasks, if necessary. Lastly, the flexible execution layer plans collision-free paths and maximizes the robot speed while explicitly considering ISO/TS 15066 guidelines. Thus, it ensures that the robot behaviour is compliant with the safety standards. Although the overall architecture is fully integrated, each layer is designed to function independently. Practitioners have the flexibility to utilize each layer as a standalone module if desired. Moreover, the architecture has been validated in a real complex industrial scenario, proving its practical applicability, robustness, and effectiveness.**

***Index Terms*—Human-robot collaboration, safety, industrial application, task scheduling, trajectory planning**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 818087 (ROSSINI).

[1] Andrea Pupa and Cristian Secchi belong to the Department of Science and Methods of Engineering, University of Modena and Reggio Emilia, Italy. E-mail: {andrea.pupa, cristian.secchi}@unimore.it

[2] Simone Comari and Marco Carricato belong to IRMA L@B, a research group under the Department of Industrial Engineering, University of Bologna, Italy. E-mail:{simone.comari, marco.carricato}@unibo.it

[3] Mohammad Arrfou and Gildo Andreoni belong to DATASENSING s.r.l., Italy. E-mail: {mohammad.arrfou, gildo.andreoni}@datasensing.com

[4] Alessandro Carapia belongs to I.M.A. Industria Macchine Automatiche S.p.A., Italy. E-mail: alessandro.carapia@ima.it

I. Introduction

Industrial applications where collaborative robots, also called *cobots*, are used in close proximity to human operators are exponentially increasing [1]. The growing interest is supported by a market that is gradually but substantially shifting resources from the production of traditional industrial robots to collaborative robots [2]. In these applications, humans and robots work in close proximity, sharing the same workspace and interacting with each other without the need for a physical barrier. This has led to the emergence of industrial applications of mobile collaborative robotics. Thanks to the inherent versatility of having a manipulator on top of a mobile platform, many research groups focused their effort on industry-oriented tasks that may benefit from this combination. First, relevant works about this kind of integrated mobile robots may be found in [3], [4], but the topic quickly resonated at the European level, leading to large projects such as the ROBO-PARTNER project [5] in the automotive field, the VALERI project [6] in the aerospace field or the EuRoC competition [7]. More recently, [8], [9] presented a mobile manipulation system for automated logistic applications, where the authors changed the picking strategy switching from lifting to dragging items on board, with the main benefit of reduced payload requirements on the robotic arm. A modular design was proposed in [10], in which the mobile base and the cobot arm can freely couple and decouple, allowing them to operate either together or independently. A comprehensive review of similar system architectures and real-case applications is given in [11], whereas the main challenges and methods related to safety assurance are discussed in [12]. The lack of a physical barrier between humans and robots makes it necessary to pay special attention to how to assess and ensure safety. In fact, the presence of humans in the workspace can lead to unexpected situations that can compromise the safety of the human operators. To this aim, several safety standards have been developed to ensure safety in these

collaborative scenarios [13]. In particular, ISO 10218-1 [14], and ISO 10218-2 [15] address this issue by defining a set of requirements that a robot must satisfy to be considered safe for Human-Robot Collaboration (HRC). In addition, the technical specification ISO/TS 15066 [16] provides detailed guidelines to assess these requirements. However, the mere application of these regulations drastically reduces the performance of the robot. To date, indeed, the potential of collaborative robotics is not fully exploited, making it very often not competitive with respect to traditional solutions. As a consequence, researchers have focused their attention on reducing this difference in performance, trying both to create new safety sensors and to introduce new control algorithms.

In this paper, we propose a novel approach to improve the performance of collaborative robots. The idea is to mutually integrate the task allocation and the motion planning layers with a suitable safe tracking system. The data coming from the monitoring system is exploited in real-time to ensure that the robot's behaviour is compliant with the safety standards. Moreover, the proposed scheduling approach is able to adapt to unexpected events, e.g., errors in task execution, by rescheduling the tasks.

This work is framed in the ROSSINI[1] (RObot enhanced SenSing Intelligence and actuation to Improve job quality in manufacturing) EU project led by DATASENSING s.r.l. [2] together with the University of Modena and Reggio Emilia and IMA S.p.A. [3], supported by the University of Bologna. The broader scope of the ROSSINI project is developing an inherently safe hardware-software platform for the design and deployment of HRC applications in manufacturing. To this end, three different industrial scenarios are proposed, which employ the architecture proposed in this paper as a common ground. Among the three scenarios, the one proposed by IMA is selected and described here due to its higher complexity and more significant impact. The target application is analogous to the one presented in the MaXima project [17], namely the autonomous replacement of raw-material reels to an automatic tea-packaging machine by means of a combination of serial manipulators and a mobile platform. The robotic system developed in MaXima suffered from stops and failures due to the poor management of environmental uncertainties and changes. Moreover, strong restrictions over motions and power were applied to guarantee safety in case of unexpected collisions. The solution proposed in this paper, on the contrary, yields a higher level of flexibility, adaptability and enhanced safety management, thus resulting in a more resilient system better suited to industrial needs.

In particular, the main contributions of this paper are:

- a novel dynamic scheduling algorithm capable of optimizing collaboration even in the event of errors or unforeseen events, such as the request for new tasks;
- a novel adaptive framework that mutually integrates the task scheduling strategy, a suitable kino-dynamic motion-planning procedure, and an innovative safe-certified monitoring system, which makes the framework suitable for real HRC scenarios;
- the validation of the approach both in a real and complex industrial scenario and in a mock-up one,thereby demonstrating the robustness and modularity of the framework under practical conditions.

The paper is organized as follows. Section II presents a review of the literature, while Section III details the task scheduling and the task execution problem in a real HRC scenario, focusing on safety issues. In Section IV, the proposed overall architecture is presented. The architecture is composed of two different layers: the safety layer, detailed in Section V, and the control layer, detailed in Section VI. Lastly, Section VII provides details regarding the validation process, encompassing both a real and complex industrial scenario offered by IMA, nd a mock-up scenario, accompanied by a comparison with a state-of-the-art algorithm. Finally, Section VIII addresses conclusions and future works.

It is worth underlining that the safety layer is composed of an innovative, safe sensing system, which constitutes a side result achieved while carrying out the main activity. Due to the confidential nature of the information about the sensing system, Section V will not delve into all technical details related to it.

II. Related Works

To comply with safety regulations, it is crucial to have a monitoring strategy that can identify and track the human operator within the scene in a safe way. Conventionally, safety in work cells is ensured by enclosing the robot with either a physical or virtual barrier [18] and establishing a sequence of roles for the robot and the human [19]. However, this approach limits the benefits introduced by HRC.

To ensure close proximity and continuous collaboration, several technologies have been utilized to improve safety in HRC scenarios, including laser scanners [20] and RGB-D sensors, which have shown an excellent potential in monitoring human operators.

Despite the large amount of research on 3D-reconstruction techniques based on RGB-D data [21], to the best of the authors' knowledge, only one safety-rated solution was available on the European market at the beginning of the ROSSINI project: *SafetyEYE* by *Pilz*. The system employs a single camera without merging data from multiple sensors. In contrast to safety laser scanners that cover a planar 2D surface, *SafetyEYE* is able to monitor a user-defined 3D volume where it can detect the presence of a human and adapt the robot behaviour accordingly. The main limitation of such a solution is the inability to properly distinguish a human from other objects present in the monitored area and to fuse information coming from multiple sensors to minimize occlusions. For the sake of completeness, we must acknowledge the existence of a new technology that was released later on in 2021 by *Veo Robotics*, which employs 3D time-of-flight sensors and holds a safety

[1]For more details, please visit https://www.rossini-project.com/.

[2]DATASENSING s.r.l. is part of DATALOGIC group. For more details, please visit https://www.datasensing.com/.

[3]IMA S.p.A. is a world leader in the manufacturing of automatic machines for the packaging of pharmaceuticals, cosmetics, food, tea, and coffee. For more details, please visit https://ima.it/en/.

certification, however, valid only in the North American area. Their *FreeMove*® system uses 3D data to identify the work cell, including the robot, work-piece, human and occlusions, in order to implement dynamic Speed and Separation Monitoring.

In the literature, safety is addressed not only with the sensors but also at the control level. In [22], an approach to estimate the future human occupancy of the workspace is proposed. The idea is to exploit the data coming from a 3D camera to evaluate the speed of the human and build an occupancy map assuming a kinematic model of the operator. The resulting occupancy map is then used by a suitable control strategy to reduce the speed of the robot in order to avoid collisions. In [23], a safety framework for multiple collaborative robots is presented. The authors propose to scale the robot's speed to prevent a safety index from falling below a threshold. The safety index is computed by considering the distance between the robot and the human and the speed of the robot. If the scaling procedure is not enough, an emergency stop is triggered, thus ensuring the safety of the human.

Although these approaches ensure safety, they may fall short in guaranteeing optimal robot behavior, potentially diminishing the competitiveness of cobots compared to industrial solutions. Indeed, modifying the path online might offer a more effective strategy to enhance robot performance and maintain competitiveness in collaborative robotics. To this aim, other works focused on developing optimization-based solutions. In [24], the authors exploit the solution of an optimization problem in real-time to constrain the robot inside a safe set, evaluating the variation of a safety index. The concept of safe set and Control Barrier Functions (CBFs) is also used in [25]. In this work, the ISO/TS 15066 constraint is approximated with a set of CBFs, and an optimization problem guarantees that the robot's behaviour is compliant with the safety standards. Despite being a promising method for ensuring safety, CBFs have some limitations. One limitation is that they require a relative degree of at least one with respect to the controlled system. This requirement is not met when dealing with a velocity-controlled robot and a constraint at the velocity level. Another limitation is that in complex environments, such as industrial settings, a CBF is needed for each object in the scene, becoming computational expensive. Moreover, they usually require an initial trajectory that the robot has to follow.

An alternative approach to address the planning problem consists in exploiting kino-dynamic planning techniques [26], [27], i.e. planning algorithms that take into account also the dynamics of the robot. However such algorithms are not able to handle kino-dynamic constraints that change in real-time, which is the case of the safety standards (e.g. varying operator positions). Thus the use of kino-dynamic planning algorithms may lead to inefficient robot behavior.

In this context, in [28], the authors address the problem by developing an integrated framework that combines dynamic planning and a control layer. The planner leverages the information coming from a suitable monitoring system to generate a collision-free trajectory for the robot. The control layer, instead, regulates the robot speed explicitly considering the constraint imposed by the ISO/TS 15066. Moreover, the two layers are mutually integrated: if the trajectory becomes unfeasible or the speed is reduced too much, the planner dynamically adapts the trajectory.

All proposed strategies, however, rely on the utilization of precise tracking systems, such as motion-capture sensors. Furthermore, these works assume that such systems are inherently reliable, which is not true. In the context of safety, it is mandatory that the tracking system itself holds a safety certification to ensure its reliability and adherence to safety standards. Thus, all the aforementioned approaches can not be applied in a real industrial scenario.

To improve the performance of collaborative robotics, safety is not the only aspect that needs to be focused on. One of the advantages of collaboration is the possibility of creating a synergy between humans and robots, allowing the execution of complex jobs that are not feasible for a single actor. Therefore, a key role is played by how the tasks are assigned and scheduled to the human and the robot [29].

A lot of research has been done in this direction, proposing different approaches. Some works model the problem as a non-linear optimization problem, see, e.g., [30]–[32]. However, due to the high complexity of the problem, these approaches are unsuitable for real-time applications. Indeed, in real collaborative scenarios, both the human and the robot are required to be reactive to unexpected events.

In [33], a method for efficiently handling Temporal Constraint Satisfaction Problems (TCSP) is proposed. The main idea is to achieve a dispatchable representation of the TCSP, allowing for dynamic fast scheduling. The approach, however, does not consider the possibility of switching the tasks online between the agents, which may be desirable in HRC scenarios. In [34] the authors introduce "Tercio", an algorithm for task scheduling that handles intercoupled temporal and spatial constraints. However, the obtained solution is suboptimal and the system was not validated in a real industrial scenario. In [35], a two-level feed-forward optimization strategy for offline sub-task allocation for HRC is proposed. Moreover, the strategy is enriched with an online feedback strategy which leverages mutual trust to reallocate the sub-tasks. In [36], the authors present a multi-criteria decision-making framework for task allocation. The algorithm is capable of handling unexpected events by rescheduling the remaining tasks. An unexpected event may also be caused by the human, e.g., by a different execution time. For this reason, other works like [37], [38] embed a monitoring system to detect possible deviations from the expected execution time and reschedule the tasks accordingly.

To improve task allocation, it would be better to exploit also the information coming from the planning layer. To this aim, researchers focused on the Task Allocation and Motion Planning (TAMP) problem [39]. In [40] the authors present "GRSTAPS", an algorithm for efficiently addressing the TAMP problem. This strategy, however, assumes that task requirements do not change during the execution, making it unsuitable for HRC scenarios. In [41], a hierarchical framework composed of two layers is proposed. The top layer allocates high-level operations without taking into account their motion primitive, while the bottom layer optimizes the

execution based on the current scenario. Moreover, to top layer exploits the information about the feasibility of each task to adapt the task allocation. In [42], the authors propose an integrated framework that combines task allocation and motion planning procedures while fulfilling safety regulations. The framework leverages the results of motion planning to reschedule the tasks online. These solutions, however, are hardly applicable to real HRC scenarios. In fact, they do not consider many of the limitations or problems that may arise in a real industrial setting. Above all, since the tracking system must be safe-certified, the tracking information is not as detailed as that provided by a motion capture system. Moreover, to the best of the author's knowledge, the TAMP literature lacks integration and validation in real and complex industrial experiments, such as the one in [17], where a mobile manipulator is considered.

III. Problem Statement

Consider an industrial collaborative scenario where a human operator H and a n-DoF velocity-controlled collaborative robot R have to collaborate in order to perform a job. The job is composed of a predefined set of tasks $(T_1, \dots, T_N)$, each of which is characterized by a nominal execution time t_{ai} and an intrinsic cost w_{ai}, where $a \in A = \{H, R\}$ represents the actor that executes the task i. The intrinsic cost w_{ai} may be used to encode the actor capabilities or the estimation of the expenses that should be incurred if the respective actor executes the task.

In order to accomplish each task, the robot 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 $\mathbf{q}_{des}(t_f) = \mathbf{q}_f \in \mathbb{R}^n$. To be considered admissible, the trajectory $\mathbf{q}_{des}(t)$ must satisfy the following two conditions:

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

$$d(\sigma_{Ri}(\mathbf{q}(\bar{t})), \sigma_{Hj}(\bar{t})) \geq d_{adm} \qquad \begin{array}{l} \forall i \in \{1, \dots, n\}, \\ \forall j \in \{1, \dots, m\}, \\ \forall \bar{t} \in [t_i, t_f], \end{array} \quad (1)$$

where $\sigma_{Ri}(\mathbf{q}(\bar{t}))$ and $\sigma_{Hj}(\bar{t})$ are functions that encompass the description and position of the i-th robot link and the j-th human limb at time $\bar{t}$, $d(\sigma_{Ri}(\mathbf{q}(\bar{t})), \sigma_{Hj}(\bar{t}))$ is a function that compute the distance between the i-th link and the j-th limb, i.e. the distance between the two closest points, and d_{adm} is the minimum admissible distance.

To be compliant with the safety standards, the robot's velocity must be lower than an upper bound that depends on the human-robot distance. Further information can be found in Section V-A2. To this purpose, it is possible to explicitly isolate the magnitude of the velocity along the trajectory by applying a path-velocity decomposition:

$$\mathbf{q}_{des}(t) = \mathbf{q}_{des}(s(t)) \qquad t \in [t_i, t_f], \quad (2)$$

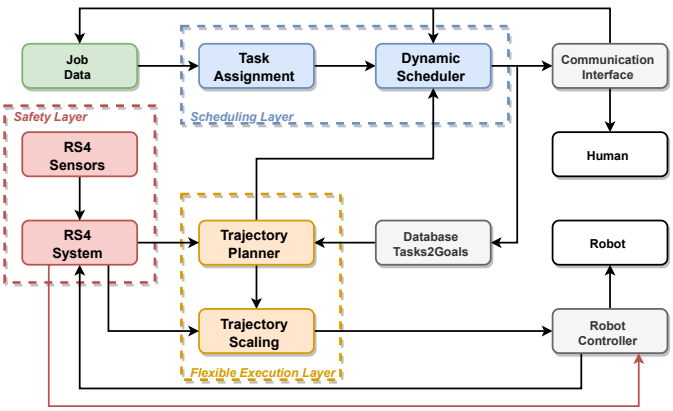


Fig. 1: Proposed overall framework.

where s is the curvilinear abscissa that parameterizes 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 (2) with respect to time, the velocity profile can be isolated:

$$\dot{\mathbf{q}}_{des}(t) = \mathbf{q}'_{des}(s(t))\dot{s} \qquad t \in [t_i, t_f], \quad (3)$$

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 compliance with the safety standards without modifying the overall path.

In this work, we aim to design a modular, fully integrated and safe dynamic framework that:

- optimally schedules online the task between the robot and the human, generating two ordered task schedules that take into account the priority of the tasks, the nominal execution time and the actor capabilities to minimize the makespan;
- for each robot task, finds a trajectory that is always collision-free and compliant with the safety standards by ensuring the respect of (1) and modulating the magnitude of the velocity in (3) at run-time;
- integrates these two procedures with a safe-certifiable RGB-D based 3D vision monitoring system.

IV. Architecture

The proposed architecture is represented in Fig. 1, where three components may be distinguished.

1) **Safety Layer**: it is responsible for tracking the human operator and the robot inside the collaborative setting in a *safe* way. It takes all data from the safety sensors as input, 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) **Scheduling Layer**: it is responsible for scheduling the tasks between the two actors. It firstly builds the optimal nominal schedule taking as input the job data to be performed. Subsequently, it adapts the online schedule to face the possible deviations that may arise during real collaboration/interaction.

3) **Flexible Execution Layer**: it is responsible for planning the collision-free trajectories for the robot at run-time. It first exploits the scene and human zone knowledge to compute the trajectory. Subsequently, the robot speed is adapted along the path according to the velocity limit calculated by the safety layer, and, if necessary, the overall trajectory is re-planned.

The entire procedure starts with defining the data composing the job, i.e. tasks t_{ai} and weights w_{ai}. The task-assignment block initially solves an optimization problem to build a nominal schedule that minimizes the overall cost while complying with the task precedences and job quality constraints. Subsequently, the task scheduling assigns at run-time the tasks to each actor and, if necessary, locally adapts the nominal schedule considering the change in the job, e.g. failures, changes in the execution time. The human tasks are sent to a proper communication interface that allows the user to interact with the framework. Instead, the robot task is first stored into a database to convert the single job into a set of intermediate goals that the robot has to execute. Once the goals are defined, the planners plan a collision-free trajectory that the robot could ideally run at maximum speed. Lastly, the trajectory scaling adapts the velocity along the path at run-time, ensuring both collision avoidance, i.e. the collision-free path is not changed, and compliance with the safety regulations to finally send the desired velocity to the low-level robot controller. If necessary, e.g., the trajectory becomes infeasible, the planner may change the trajectory on the fly.

During the overall procedure, the safety layer is permanently active. It merges the data from different safe sensors to create two voxel maps, one for the human and one for the robot. The voxel maps can be used to compute the maximum allowed robot speed that is exploited by the trajectory scaling to adapt the robot speed. Moreover, when it is necessary to stop the robot, the safety layer implements two strategies: sending a safe stop to the robot controller and a maximum allowed robot speed equal to 0. In this way, safety in the stop phase of the robot is in any case guaranteed by the use of the safety interface of the low-level control, while the correspondence between the input sent by the trajectory scaling and the real behaviour of the robot is maintained.

V. Safety Layer

A. Smart Safe Sensing System - RS^4

The safety controller RS^4, namely *ROSSINI Smart Safe Sensing System* depicted in Fig. 2, is developed as part of a modular and scalable platform for integrating human-centered robotic technologies in industrial environments. The RS^4 system reconstructs the monitored environment using sensory data to detect whether the human operator and robot are too close in a collaborative zone or if an object is in a restricted zone. If a collision risk is detected, the RS^4 delivers a stop or slowdown request to the robot. The safety controller integrates sensory data from devices like 3D safety cameras, laser scanners, radar, and application-specific safety inputs. It is compliant with safety standards for Electro-Sensitive Protective Equipment (ESPE, per IEC 61496-1) and video-based protective devices

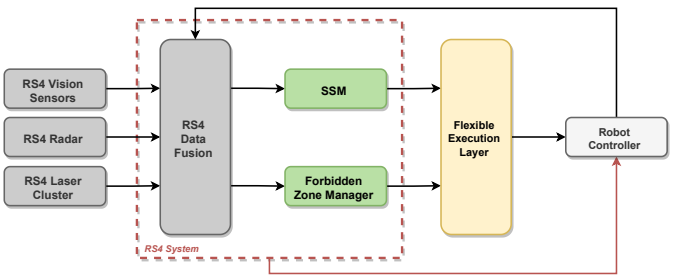


Fig. 2: Overview of the RS^4 functional blocks

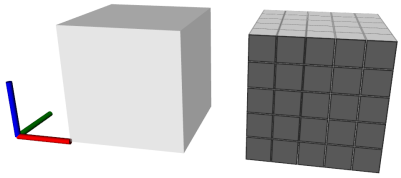


Fig. 3: A 3D area is represented as a voxel map discretized as a set of cubes that divides the 3D area to monitor. The voxel map resolution is adjusted by the voxel side length.

using stereo techniques (VBPDST, per IEC/TS 61496-4-3), through its integration with vision safety sensors.

1) RS^4 Data Fusion: Fusing point clouds from different sensors requires defining sensor poses relative to a reference frame, a preliminary step called *calibration*. Depth data fusion may involve either multiple homogeneous sensors with the same type and intrinsic parameters or heterogeneous sensors like laser scanners, RGB-D cameras, or stereo cameras. Depth sensor data must also be synchronized with precise timestamps to match the same scene from different perspectives. In a multi-camera setup, like the one in the experimental campaign described in Section VII, calibration is often addressed as *camera registration*. The pose of each camera is retrieved with respect to a reference frame during device installation, leveraging the method in [43]. Additionally, the system uses a calibration verification strategy [44]. Inertial Measurement Unit (IMU) devices monitor any camera pose changes, with periodic checks against known points in industrial scenes. If a change is detected, manual recalibration is required. This verification could also extend to automatic calibration methods, such as in [45], [46].

The heterogeneous data acquired and processed by the RS^4 system controller can be divided into three categories:

- Boolean data, that is, simple data coming from physical buttons, robot controller, RS^4 skin;
- 2D data, acquired by RS^4 Laser sensors;
- 3D data, acquired by RS^4 Vision sensors and RS^4 Radar.

While the first category typically does not need a powerful processing unit, the latter two require sufficient computing power to handle large data volumes in real-time while minimizing system latency.

Data from vision sensors, further detailed in Section V-B, is represented in a spatial format called a *voxel map*, a 3D space discretization into unit elements known as *voxels*, this is illustrated in Fig. 3. Following [47], each voxel represents a unit volume in a 3D space, similar to pixels in a 2D image, and

can store occupancy status as *free*, *occupied*, or *occluded*. To guarantee safety, a conservative approach is always applied to define the robot's surrounding space that might cause a collision. Indeed, both *occupied* and *occluded* voxels are processed by modules detailed in Sections V-A2 and V-A3, ensuring that even spaces not directly monitored are treated as potential collision zones.

Once occupancy status is assigned, a *meaning label* is applied to each voxel using a human-robot detection algorithm, labeling each voxel as either human or robot. The minimum distance is then calculated as the Euclidean distance between the closest voxels in the clusters representing the human and the robot, with the robot-human direction given by the unit vector $\mathbf{n}_{RH}$ and computed as:

$$\mathbf{n}_{RH} = \frac{\mathbf{x}_H - \mathbf{x}_R}{\|\mathbf{x}_H - \mathbf{x}_R\|}, \tag{4}$$

where $\mathbf{x}_H$ and $\mathbf{x}_R$ are the position of the closest human and robot voxel, respectively.

2) Speed and Separation Monitoring: In the Speed and Separation Monitoring (SSM) operational mode, the robotic system and operator may move concurrently in a shared workspace. Risk reduction is always realized by preserving at least the Protective Separation Distance (PSD) between the operator and the robot. The PSD is calculated through Eq. (5), taken from the ISO/TS 15066:

$$S_p(t_0) = S_h + S_r + S_s + C + Z_d + Z_r, \tag{5}$$

where $S_p(t_0)$ is the PSD at current time t_0. S_h is the contribution to the protective separation distance attributable to the human's change in location, S_r is the contribution to the protective separation distance attributable to the robot's system reaction time, and S_s is the one due to the robot's system stopping distance. 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.

Analyzing Eq. (5), it is possible to expand the first three terms as:

$$S_h = \int_{t_0}^{t_0+T_s+T_r} v_h(t)\, dt, \tag{6}$$

$$S_r = \int_{t_0}^{t_0+T_r} v_r(t)\, dt, \tag{7}$$

$$S_s = \int_{t_0+T_r}^{t_0+T_s+T_r} v_s(t)\, dt, \tag{8}$$

where T_s represents the robot stopping time and T_r is the robot's reaction time. v_h and v_r are the directed speed of the human operator towards the robot and the one of the robot towards the human operator, respectively. v_s is the speed of the robot during the stopping phase.

Let us assume 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 are slower than the robot dynamics, which is usually the case in a generic HRC application. Under these assumptions, the Eq. (6) – (8) can be approximated as follow:

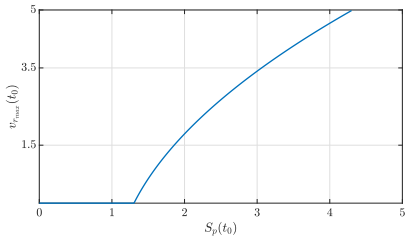


Fig. 4: Maximum allowed robot velocity towards the human operator in case of $v_h(t_0) = 2\,m/s$, $a_{max} = 7.5\,m/s^2$, $T_r = 0.002\,s$, $C+Z_d+Z_r = 1.3\,m$.

$$S_h = v_h(t_0)(T_s + T_r), \tag{9}$$

$$S_r = v_r(t_0)T_r, \tag{10}$$

$$S_s = v_r(t_0)T_s - a_{max}\frac{T_s^2}{2}, \tag{11}$$

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

Recalling 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 Eq. (9) – (11) as:

$$S_h = v_h(t_0)(\frac{v_r(t_0)}{a_{max}} + T_r), \tag{12}$$

$$S_r = v_r(t_0)T_r, \tag{13}$$

$$S_s = \frac{v_r(t_0)^2}{2a_{max}}. \tag{14}$$

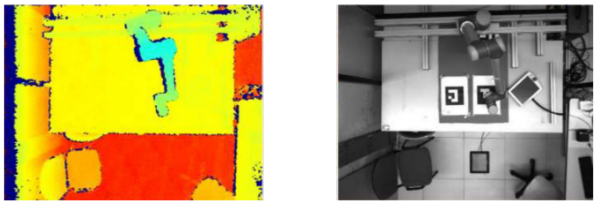
By substituting Eq. (12) – (14) into Eq. (5), an upper bound for the robot velocity can be obtained [48]:

$$v_{r_{max}}(t_0) = \sqrt{v_h(t_0)^2 + (a_{max}T_r)^2 - 2a_{max}K(t_0)} + \\ -a_{max}T_r - v_h(t_0), \tag{15}$$

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

Using Eq. (15), one can derive the safety limit imposed by the ISO/TS 15066, i.e. the maximum velocity that the robot can reach in the direction of a human operator. Fig. 4 shows a realistic trend of the velocity limit $v_{r_{max}}(t_0)$ as a function of the actual separation distance $S_p(t_0)$ between the human operator and the robot. In this example, v_h is chosen according to ISO standards, T_r is an arbitrary robot reaction time purely chosen for illustration purposes. Whereas the values for the remaining parameters are taken from the technical specifications of the devices (i.e. cobot + position tracking system) employed in the experimental setup described in Section VII .

3) Forbidden Zone Manager: The Safety-rated Monitored Stop (SMS) robot feature is used to halt the robot's motion when an operator enters a specific predefined workspace, as described by the ISO/TS 15066. In general, a predefined area around the robot is called a *forbidden zone* when any intrusion



(a) (b)

Fig. 5: Depth and grayscale images from the RS^4 Vision sensor

must trigger a robot stop or a slow down in case the robot supports PFL modality. This is ensured by the RS^4 controller, which is responsible for conveying safety information to the low-level controller of the robot. Indeed, the RS^4 System controller is configured to monitor an area that meets the requirements of ISO 13855 for the realization of the forbidden zone monitoring and can detect the presence of an operator therein following the data flow as seen in Fig. 2. Sensory data from the vision sensor or RS^4 laser scanner cluster are used in this case to execute zone violation verification.

B. RS^4 Vision Sensor

The RS^4 vision sensor, conceived and developed within the ROSSINI project, is a prototype of a 3D safety-compliant camera that monitors an area shared by humans and robots. Its main objective is to exploit 3D vision sensors for SSM implementation by integrating multiple safety-rated vision devices in an overall system with a safety controller. This is achieved by tracking the motion of objects inside the monitored area where occlusions are minimized thanks to a suitable set of camera perspectives that guarantee redundancy of the measurements and avoid blind spots. The design focus has been centered on human detection capability and the latency of the image acquisition and elaboration pipeline in order to reduce the reaction time of the overall system.

The main high-level features implemented in the RS^4 vision sensor are described in the following.

- **Onboard depth map creation**: the creation of a depth map of the monitored environment is computationally expensive, so this step is performed directly on the RS^4 vision sensor. This approach eliminates the need for further processing on the device receiving images, as 3D information is already available. It also enables the distribution of processed data to multiple devices when using more than one camera. An example of a RS^4 depth image is shown in Fig. 5a.
- **High-speed industrial communication interface**: the device features a fast communication interface that is able to transmit with low latencies the images coming from the sensor and generated onboard to multiple receivers.
- **Grayscale image**: on top of the depth map, the camera can provide a standard grayscale image at a wavelength within the 400 nm to 1500 nm range (visible light spectrum). This feature can simplify future tasks, such as marker detection and object identification, which are not yet implemented. An example of a RS^4 grayscale image is shown in Fig. 5b;
- **Synchronized triggering**: multiple cameras could be linked to a central controller for synchronous triggering in order to be able to fuse the retrieved images related to the same instant. A typical RS^4 system is composed of a set of cameras that are connected to a controller which was developed during the ROSSINI project. For an exhaustive understanding of the system, please refer to the associated patent application [49]. Synchronous triggering of sensors is a common technique that is typically used in sensors to ensure a synchronous working of sensors, and can be achieved via hardware or software. Though it is not a novel feature it is worth mentioning for its importance, particularly in critical safety systems. The benefit of having synchronous cameras lies in their ability to assess the scene concurrently at the same timestamp of data images taken from different perspectives. Otherwise, Without this synchronization, data could be associated with disparate timestamps, potentially leading to erroneous measurements.

The RS^4 vision sensor is developed to have arm or body detection capabilities. Note that the terms “human arm” and “body” denote detection capabilities, not distance range, as outlined in IEC 61496-4-3 standards. This is a standardized way to describe vision-based protective devices (VBPD) in the industry. According to IEC 61496-4-3, a VBPD can be described with finger, hand, arm, and body detection capabilities. For instance, using a VBPD with body detection capability requires to consider an additional intrusion distance C of 850 mm; according to ISO 13855. Two camera models with distinct features were developed. The first model offers higher detection capability up to the size of a human arm but covers a smaller volumetric zone coverage due to sensor and processing limitations. This model is suited for scenarios requiring high detection in close human-robot interaction areas. The second model covers a larger volume but with reduced detection capability, focusing on detecting the human body. In the proposed experimental setting (described in Section VII), the need to monitor a large area led to the use of cameras with body detection capability. In this case, also wide coverage is mandatory, and thus multiple cameras with body detection capabilities are employed, positioned to partially overlap and maximize the monitored volume.

VI. Control Layer

A. Scheduling Layer

The goal of the scheduling layer is to allocate and schedule the tasks among the two actors, minimizing the total makespan and improving collaboration. This is achieved in two different steps. The first one is performed by the task assignment block, which exploits the job data , e.g. the task execution time of each actor, to optimally assign the tasks between the two actors. The output of this step is a first nominal schedule that is compliant with the desired job quality constraint and with the precedence constraint. The second step, instead, is performed by the dynamic scheduler, which exploits the real execution times and the changes in the job to reschedule the tasks. It is worth pointing out that this part is inspired by our previous

work in [50] and presents an innovative dynamic scheduling strategy. This strategy takes into account the occurrence of unforeseen tasks caused by various factors, such as unexpected machine stoppages or issues. Indeed, in these cases it may be convenient to change the job data, e.g. task execution time, and the reallocation of the tasks between the actors. The methodology proposed in [50] necessitates the interruption of the framework and reevaluation of the optimal schedule, leading to potential time loss in real industrial scenarios. In this work, instead, this procedure is directly incorporated into the dynamic scheduler, ensuring the adaptability of the overall framework.

1) Task Assignment: Focusing on the task assignment procedure, it takes as input the data of the collaborative job to be performed. The job is represented as a directed acyclic graph $\mathcal{G} = (T, E)$ where each vertex represents the task $T_i \in T$ and each directed edge $E_{ij} \in E$ represents the precedence constraint, i.e. T_i must be executed before T_j. This is illustrated in Fig. 6a. Intuitively, some tasks may be executed in parallel in order to improve performance. To this aim, the overall graph $\mathcal{G}$ can be redrawn, grouping together all the parallel tasks into several sets called levels L_l, as shown in Fig. 6b. This allows representing merely the precedence relations among the overall tasks. Some tasks, however, may be independent of others, meaning that their execution order is irrelevant and can be assigned at all levels. This is the case of T_1 and T_3 in the illustrated graph since there is no path that goes from T_1 to T_3 and vice-versa. Therefore, the task assignment problem is translated into defining which actor should execute each task and how the tasks should be distributed among the levels. This can be obtained by solving the following multi-objective Mixed Integer Linear Programming problem:

$$
\begin{aligned}
\min_{x, c_l} \quad & \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{a \in A} w_{ai} x_{ail} + \frac{1}{t_{max}} \sum_{l=1}^{L} c_l \\
\text{s.t.} \quad & \\
& \sum_{l=1}^{L} (x_{Ril} + x_{Hil}) = 1 && \forall i \in \{1, \dots, N\}, \\
& \sum_{i=1}^{N} t_{ai} x_{ail} \leq c_l && \forall l \in \{1, \dots, L\}, \\
& && \forall a \in A, \\
& \sum_{a \in A} \sum_{l=1}^{L} l \cdot x_{ail} < \sum_{a \in A} \sum_{l=1}^{L} l \cdot x_{ajl} && \forall i \to j, \\
& K_m \leq K_{m,max} && \forall m \in \{1, \dots, M\}, \\
& K_{m,av} \leq K_{m,av,max} && \forall m \in \{1, \dots, M\},
\end{aligned}
\tag{16}
$$

where $w_{ai} > 0$ represents the cost required by the actor a, i.e. human or robot, to execute T_i. These costs are used to embed all the relevant factors that the user what to consider in the task assignment procedure, e.g. electrical cost, tool wear, and actor capability. Thus, the calculation method of these costs is a design parameter, e.g., [51], [52]. Moreover, the cost of the human w_{Hi} may be a combination of both the process costs and the human job quality parameters. This allows embedding the job quality in the scheduling procedure, trying to optimize it. 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. Further details on the definition of these parameters can be found in [50].

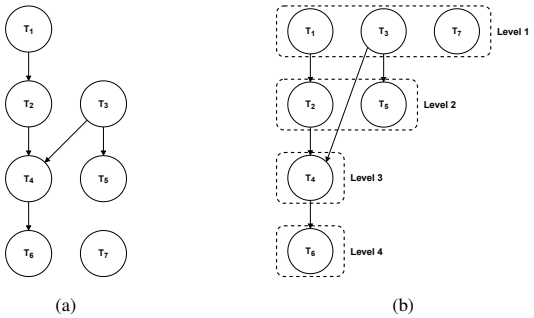


Fig. 6: Fig. 6a shows the directed acyclic graph of a job composed by seven tasks, while Fig. 6b shows the division into four levels.

The first constraint in Eq. (16) guarantees that each task is executed only once and by only one actor. The second constraint imposes the maximum parallelism criterion for each level. It ensures that, at each level, the execution time of each actor is less than or equal to the cycle time of the level c_l. However, since c_l is an optimization variable and the cost function requires to minimize the sum of c_l, the optimization problem will choose a schedule such that the two actors will work in parallel as much as possible, i.e. penalizing unnecessary idle times. The third constraint guarantees compliance with the precedence relation by imposing that the precedence tasks are assigned to a higher level. By definition of the graph $\mathcal{G}$, this means that the precedence tasks are executed before the successor tasks. The last constraints impose that the job quality metrics for the human operator will not violate the upper bounds.

In the MILP problem (16), a great role is played by the selection of the maximum number of levels L. A small number of levels may cause the infeasibility of the problem, i.e. the number of levels can not ensure all precedences. A very large number of levels, however, may require a higher unnecessary computational cost. For this purpose, it is convenient to choose L slightly higher than the number of nodes that belong to the longest path of the graph representing the job. In other words terms, utilizing the SoA algorithm, i.e. topological sorting [53], on the graph $\mathcal{G}$ allows us to determine the longest path LP, where each edge is assigned a unit weight. Consequently, L can be selected as $L = LP + 1$.

The result of the optimization problem (16) is the two nominal schedules S_H and S_R, i.e. the ordered set of tasks that must be executed by each actor.

2) Dynamic Scheduler: Starting from the result of the task assignment, the dynamic scheduler schedules the tasks at run-

time for each actor. To this aim, it exploits both the data coming from the communication interface and the flexible execution layer to understand when each actor has completed the scheduled task, i.e. the actor is free to execute a new job. However, during the collaboration, some deviations may arise from the expected behaviour. In fact, it may happen that some tasks are not performed correctly. Or still, some machine could face a micro stoppage, i.e. a short-latency stops due to a temporary problem, and the help of one of the two actors is required. In these cases, a rescheduling procedure is needed that tries to maintain the optimality of the schedule.

The dynamic scheduler procedure is implemented according to the pseudo-code reported in Alg. 1. The algorithm takes as

Algorithm 1 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:      for a ∈ A do
7:          if End_a then T_a ← next(S_a)
8:          end if
9:          (End_a, Fail) ← checkEnd_a()
10:         if Fail then
11:             Δ_JD ← setChangeJobData()
12:         end if
13:     end for
14:     Δ_JD ← checkChangeJobData()
15:     if Δ_JD then
16:         (S_H, S_R) ← reschedule()
17:     end if
18:     if T_H = ∅ and T_R = ∅ then
19:         l ← l + 1
20:     end if
21: end while
```

an input the two nominal task schedules S_H and S_R (Line 1). Firstly, it sets to true the two variables End_H and End_R, meaning that the two actors are ready to start a new task, and set to false the variable $Fail$, which is used to identify when one task is failed (Lines 2 - 3). At this point, the scheduling procedure is initialized at the first level, and the algorithm starts a loop to iterate through all the levels of the schedule (Lines 4 - 5). Inside the loop, the dynamic scheduler checks for each actor if they are free to execute a task and if this is the case, it assigns the next task in their schedule (Line 7). If no tasks are available in the level, the function $\mathbf{next}(S_a)$ returns $\emptyset$. Then, it checks the end of the tasks that the actor is performing, and, in case of failure, a variation in the job data is triggered (Lines 9-11). This variation is used to encompass the best method to recover from the task. For instance, if an error occurs for which a recovery procedure has been established, the recovery task can be added to the list with the highest priority. Alternatively, if the robot encounters an unknown error, the robot's weight and nominal execution time values for the task may be greatly increased, reallocating the task to a human operator. The method to recover from the task depends on the error, and it is possible to exploit a strategy already proposed in literature [54].

Subsequently, the dynamic scheduler analyzes if there have been any other changes in the job to be performed (Line 14). For example, this could involve an unexpected need for assistance from a machine that was not previously planned, or, if an external procedure identifies that an actor is consistently late, an adjustment to their nominal execution times. If a change is detected, a rescheduling procedure is applied (Line 16). Since the number of tasks is generally small for the scenario analyzed in this work, the procedure is carried out by solving again the optimization problem (16). In case the number of tasks increases, e.g. an assembly HRC scenario, Alg. 1 can be extended with the solution proposed in [42]. Lastly, the algorithm checks if both actors have performed all the tasks in the level to move on with the schedule until the last level.

B. Flexible Execution Layer

Once the dynamic scheduler computes which is the best task that the robot may execute at the moment, the task is sent to the database, which returns the final configuration $\mathbf{q}_f$ that the robot has to reach. At this point, the flexible execution layer takes care of both planning and executing the trajectory. The goal of the flexible execution layer is to ensure the correct execution of the tasks, i.e. generating collision-free trajectories that comply with the safety regulations. Since the environment of an HRC scenario is highly dynamic, and the safety regulations take into account the real position of the human operator in the scene, the overall procedure is divided into two steps. Initially, the trajectory planning plans a collision-free trajectory taking into account only the robot joint limits, i.e. a maximum speed trajectory. In this initial phase, the human operator is not considered because the goal is to understand if an admissible trajectory exists. Subsequently, the trajectory scaling regulates at run-time the velocity along the path, resulting in a trajectory that is compliant with the safety regulations. Moreover, during the execution, the trajectory planner remains active and replans the trajectory if necessary.

In particular, the flexible execution layer is an extension of our previous work in [28]. The trajectory planner has been enhanced to handle cases where the trajectory is ideally feasible, but temporarily obstructed by a dynamic obstacle, i.e. in this case the human operator, or if the goal is not reachable in the current setting. Additionally, trajectory scaling has been integrated with safety signals to address real-time issues. Indeed, thanks to the use of a safe certifiable monitoring system, i.e. the RS^4, it is possible to enrich the flexible execution layer with the relevant safe information, e.g. human-robot distance or robot status. This is further detailed in Section VI-B2.

1) Trajectory Planner: The dynamic trajectory planning procedure is implemented according to the pseudo-code reported in Alg. 2. The algorithm starts from the actual initial configuration $\mathbf{q}_i$ and the final desired configuration $\mathbf{q}_f$, which comes from the custom database (Line 1). It immediately plans a collision-free trajectory $\mathbf{q}_{des}(t)$ considering only the joint limits of the robot and the static obstacles inside the

Algorithm 2 TrajcetoryPlanner()

1: **Require:** $\mathbf{q}_i, \mathbf{q}_f$
2: $(\mathbf{q}_{des}(t), Admissible) \leftarrow \mathbf{plan}(\mathbf{q}_i, \mathbf{q}_f, static)$
3: $\mathbf{send}(\mathbf{q}_{des}(t))$
4: $\mathbf{q}_c \leftarrow \mathbf{q}_i$
5: **while** $\mathbf{q}_c \neq \mathbf{q}_f$ **and** $Admissible$ **do**
6: $valid \leftarrow \mathbf{checkTraj}(\mathbf{q}_c, h_P)$
7: **if not** $valid$ **then**
8: $(\mathbf{q}_{des}(t), tmpFail) \leftarrow \mathbf{replan}(\mathbf{q}_{des}(t), \mathbf{q}_{free}, \mathbf{q}_f)$
9: $\mathbf{send}(\mathbf{q}_{des}(t))$
10: **if** $tmpFail$ **then**
11: $Timer \leftarrow 0$
12: **end if**
13: **end if**
14: **if** $Admissible$ **and** $tmpFail$ **and** $Timer \geq T$ **then**
15: $(\mathbf{q}_{des}(t), tmpFail) \leftarrow \mathbf{plan}(\mathbf{q}_c, \mathbf{q}_f, dynamic)$
16: $\mathbf{send}(\mathbf{q}_{des}(t))$
17: $Timer \leftarrow 0$
18: **end if**
19: $(\mathbf{q}_c, Admissible) \leftarrow \mathbf{update}()$
20: **end while**
21: **if** $Admissible$ **then**
22: $(End_R, Fail) \leftarrow (true, false)$
23: **else**
24: $(End_R, Fail) \leftarrow (false, true)$
25: **end if**
26: $\mathbf{send}(End_R, Fail)$

scene, i.e. a trajectory that the robot could ideally execute at maximum speed (Line 2). The function $\mathbf{plan}()$ is implemented exploiting planning algorithms already available in the literature, e.g., [55]–[57][4], and requires as input an argument defining which maps of the environment should be considered, $static$ or $dynamic$. If no trajectories are found, the returned trajectory $\mathbf{q}_{des}(t)$ is empty, and the variable $Admissible$ is set to $false$. On the contrary, if a feasible trajectory is found, it means that at least one solution exists, and the variable $Admissible$ is set to $true$. Unlike previous work in [28], distinguishing between the $static$ and $dynamic$ maps helps determine whether a planning failure is recoverable, e.g. waiting for the human operator to move, or whether the goal is in general unreachable, triggering a failure. Subsequently, the trajectory is forwarded to the trajectory scaling block, and the actual robot state $\mathbf{q}_c$ becomes equal to the initial configuration (Lines 3 - 4). At this point, the algorithm starts a loop until the trajectory is completed or becomes unfeasible, e.g. the robot transitions to a safety stop. Inside the loop, the planner starts to take into account also the zone occupied by the human operator and continuously checks if the trajectory is still collision-free (Line 6). Since, in general, an HRC scenario is highly dynamic, it may happen that a collision-free trajectory suddenly becomes a colliding one and vice-versa. To this aim, the checking procedure is performed only on a predefined horizon of the trajectory h_P, starting from the actual robot configuration $\mathbf{q}_c$. When a colliding configuration is found, $\mathbf{q}_{des}(t)$ is updated through the function $\mathbf{replan}()$ (Line 8).

[4]In this paper, the RRT-Connect algorithm was exploited for its simplicity.

The $\mathbf{replan}()$ function takes care of planning a new collision-free trajectory that goes from a desired configuration (in this case, the last collision-free robot configuration $\mathbf{q}_{free}$) to the desired final configuration. Furthermore, the new trajectory is merged with the previous one, thus ensuring continuity for the trajectory scaling block, which is blind to these replannings. Such a continuity is guaranteed for both path, velocity, and acceleration. If the $\mathbf{replan}()$ function fails, it means that the human operator is actually hindering the robot, e.g. the human is exactly over the object that the robot has to pick. However, the robot is still able to perform and complete the task, i.e. the variable $Admissible$ is still $true$. In this case, the algorithm sets to true the variable $tmpFail$, which indicates that the failure of the replanning is temporary, and starts a $Timer$ and, after a desired amount of time T, a new planning procedure is tried (Lines 11 - 15). It is worth pointing out that, in this case, the function $\mathbf{plan}()$ uses the dynamic map, i.e. the map with the human operator. At this point, the actual robot configuration $\mathbf{q}_c$ and the variable $Admissible$ are updated, exploiting the information coming from the trajectory scaling. When the trajectory has been completed, or no feasible trajectories have been found, the algorithm exits the while loop and sets the variable End_R accordingly.

2) Trajectory Scaling: After receiving an admissible trajectory to be performed, the trajectory scaling block takes care of generating the control inputs such that the robot behaviour is compliant with the safety constraint imposed by the ISO/TS 15066 standard, as shown in Eq. (15). This is achieved by regulating only the magnitude of the velocity $\dot{s}$ of the trajectory without modifying the planned path that is always collision-free. The procedure is implemented according to the pseudo-code in Alg. 3. The algorithm takes as input the desired

Algorithm 3 TrajectoryScaling()

1: **Require:** $\mathbf{q}_{des}(t)$
2: $(\mathbf{q}_{des}(s(t)), \dot{s}(t)) \leftarrow \mathbf{pvDec}(\mathbf{q}_{des}(t))$
3: **while** $\mathbf{q} \neq \mathbf{q}_f$ **do**
4: $(v_{max}, \mathbf{n}_{RH}, \delta_s) \leftarrow \mathbf{receiveSSM}()$
5: $v_{max} \leftarrow \mathbf{min}(v_{max}, \mathbf{filter}(v_{max}))$
6: $(\mathbf{q}, \dot{\mathbf{q}}) \leftarrow \mathbf{readRobotState}$
7: $\dot{\mathbf{q}}_{cmd} \leftarrow \mathbf{scale}(v_{max}, \mathbf{n}_{RH}, \delta_s, \mathbf{q}, \dot{\mathbf{q}})$
8: $\mathbf{send}(\dot{\mathbf{q}}_{cmd})$
9: $\mathbf{send}(\mathbf{q}_c)$
10: **end while**

trajectory $\mathbf{q}_{des}(t)$ to be executed (Line 1). It firstly applies the path-velocity decomposition as shown in (2)–(3). Subsequently, it obtains from the safety layer the maximum allowed robot speed towards the human operator v_{max}, the unit vector $\mathbf{n}_{RH}$ representing the robot-human direction and δ_s which is a safety signal that is equal to 0 when a safety stop is required and 1 otherwise (Line 4). Later, it is applied a low-pass filter to v_{max} and the final value is overwritten with the minimum between the actual value and the filtered one. This procedure allows to have a profile of v_{max} with fewer discontinuities but still compliant with the safety regulations. Afterwards, the algorithm reads the state of the joints and computes the optimal scaled velocity $\dot{\mathbf{q}}_{cmd}$ to be commanded to the robot (Lines 6

- 8). The $\mathbf{scale}()$ procedure is implemented according to our previous work [28], with some differences that are detailed below. Lastly, the trajectory scaling block forwards the robot configuration $\mathbf{q}_c$ to the dynamic planner.

The function $\mathbf{scale}()$ solves online the following optimization problem:

$$
\begin{aligned}
\min \quad & -\alpha \\
\text{s.t.} \quad & \\
& J_{r_i}(\mathbf{q})\mathbf{q}'(s)\dot{s}\alpha \leq v_{max} \qquad \forall i \in \{1,\dots,n\}, \\
& \dot{\mathbf{q}}_{min} \leq \mathbf{q}'(s)\dot{s}\alpha \leq \dot{\mathbf{q}}_{max}, \\
& \ddot{\mathbf{q}}_{min} \leq \frac{\mathbf{q}'(s)\dot{s}\alpha - \dot{\mathbf{q}}}{T_r} \leq \ddot{\mathbf{q}}_{max}, \\
& 0 \leq \alpha \leq \delta_s,
\end{aligned}
\tag{17}
$$

where $\alpha \in [0, \delta s]$ is the optimization variable, and it is used to represent the scaling factor. $J_{r_i}(\mathbf{q}) \in \mathbb{R}^{1\times n}$ is a *modified Jacobian* that maps, for each link, the joint velocities to the scalar velocity towards the human operator, i.e. the velocity along the robot-human direction $\mathbf{n}_{RH}$. Further details can be found in [28]. $v_{max} \in \mathbb{R}$ is the velocity limit imposed by the ISO/TS 15066. $\dot{\mathbf{q}}_{min} \in \mathbb{R}^n$ 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 control time.

The optimization problem (17) is a convex problem and, therefore, computationally cheap. The only factor that affects the convergence is the problem dimension, i.e. the number of joints and links, which is low for a general HRC application. Furthermore, the minimum found by the solver is always a global minimum, i.e. the scaling factor minimizes the speed reduction. Moreover, the problem has always a feasible solution. When the human-robot distance is high, the robot is allowed to follow exactly the desired trajectory, i.e. $\alpha = 1$. When the distance decreases, the safety regulations impose a reduction of the velocity up to, in the worst-case scenario, a full stop of the robot. This is guaranteed by the solution $\alpha = 0$.

Then, the optimal safe velocities are commanded to the robot:

$$\dot{\mathbf{q}}_{cmd} = \mathbf{q}'(s)\dot{s}\alpha, \tag{18}$$

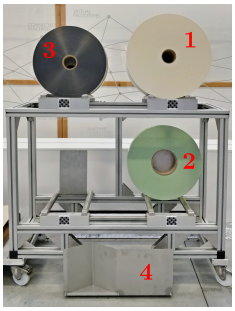
and the curvilinear abscissa s is updated accordingly:

$$s_{new} = s + \dot{s}\alpha T_r, \tag{19}$$

Lastly, the new robot configuration that is given to the dynamic planner for the replanning procedure (see Alg.2, Line 19) is equal to:

$$\mathbf{q}_c = \mathbf{q}'(s_{new}). \tag{20}$$

It is worth noting that, as explained in Sec. V-B, the safety layer computes only one robot-human direction $\mathbf{n}_{HR}$. This is due to the fact that the low-level controller of the robot does not provide the joint states with a safety communication interface, meaning that the actual robot configuration can not be directly exploited by the safety layer. Thus, differently from the approach in [28], in this work, all modified Jacobians rely on the same robot-human directions, i.e. the direction between the two closest voxels.



(a) The local storage wagon. 1 is the filter-paper reel, 2 is the tag reel, 3 is the outer-envelope reel, and 4 is the VL-shaped marker.

(b) The ROSSINI AMR.

(c) The C24-E automatic machine.

Fig. 7: Elements of the work cell.

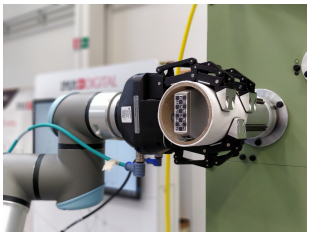
VII. Experiments

A. IMA Industrial Scenario

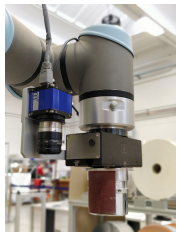
To prove the efficacy of the outlined system architecture, the work is deployed in the real-case industrial scenario offered by IMA within the ROSSINI project. The work cell includes a *C24-E automatic machine* (Fig. 7c), a *local storage wagon*[5] (or simply *wagon* in the following) (Fig. 7a) and an *Autonomous Mobile Robot* (AMR) (Fig. 7b). Specifically, the AMR features the following components:

1) two *UR10e* by *Universal Robots*, a 6DoF serial arm, light-weighted (33.5 kg), with 10Kg payload and reduced dimension (1.3m reach), and classified as a collaborative robot; one UR10 (i.e. #1) mounts a 3-finger adaptive gripper by *Robotiq* (Fig. 8a) to take care of light and sophisticated manipulations, whereas the other UR10 (i.e. #2) is equipped with an industrial monochrome 2D camera by *Matrix Vision* and a GEH6060IL gripper by *Zimmer* (Fig. 8b), and is in charge of inspection/detection tasks and heavy-reel manipulation;
2) a *MiR500* by *MiR*, an Autonomous Guided Vehicle (AGV) with 500 kg of payload and an advanced navigation system that allows the vehicle to orient itself

[5]An intermediate storing station between the warehouse and the automatic machine where enough reels for a whole shift can be stocked.



(a) UR10e #1: adaptive finger.



(b) UR10e #2: industrial camera + electrical gripper with custom fingers.

Fig. 8: AMR tools detail.

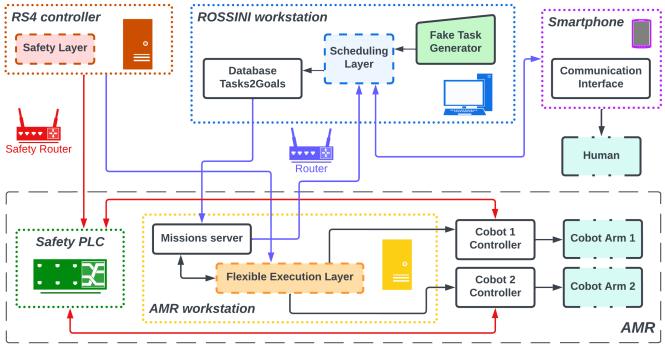


Fig. 9: ROSSINI network architecture.

in a mapped environment and safely halt whenever an unexpected object appears too close;

3) an Industrial PC by *B&R*, running Linux Ubuntu 18.04 OS and in charge of all vision-related computations as well as the management of the camera signals and advanced communication with both the URs and the MiR platform;
4) a group of safety Programmable Logic Controller (PLCs), antennas and relays that take care of parsing and distributing emergency signals among all system components.

Both the wagon and the automatic machine were equipped with *3D VL-shaped markers* (Fig. 7a) to help the AGV navigation system precisely locate the objects in the map and perform an accurate docking (± 5 mm with respect to horizontal axes, ± 1 degree around the vertical axis).

On top of these components, three safety cameras (i.e. RS^4 sensors) were placed at three corners of the shop floor, and two ground stations were used to deploy the RS^4 system and the scheduling layer. They are named, respectively, the *RS^4 controller* and the *ROSSINI workstation*, which also behaves as the central unit of the overall system. With reference to Fig. 1, Fig. 9 illustrates how the different blocks are practically deployed in the work cell. It is important to notice that the safety layer communicates on two separate levels that, in turn, use two dedicated and suitable channels. The safety signals, in fact, are directly forwarded from the RS^4 controller to the AMR safety PLCs via a safety Wi-Fi network supported by Siemens. The safety PLCs are then responsible for distributing the safety signal to all the AMR devices according to the situation. In parallel, a standard LAN, comprehensive of a dedicated Wi-Fi to connect the AMR with the ground stations, is used to transfer high-level commands, feed-backs and general-purpose data among all the workstations involved. ROS and ROS2 are employed as a common communication protocol to transfer all information and to develop different software packages.

The macro-tasks that the AMR can perform are organized into *missions*. They are executed by a mission server running on the onboard workstation alongside the cobots' drivers (e.g. robot controller that allows sending commands to the actual robot's low-level controller in a dedicated control box) and the flexible execution layer. A simple task generator running on the ROSSINI workstation is manually launched to emulate the requests that may come from one (or more) automatic machine(s). The tasks accounted for by the scheduling layer in

TABLE I: List of tasks. w stands for *intrinsic cost*, t for *time*, h for *human* and r for *robot*.

ID	Name	w_H	t_H [s]	w_R	t_R [s]
0	Micro-stoppage	0.4	120	9999	9999
1	Change Left Filter Paper Reel	0.6	120	0.6	120
2	Change Left Tag Reel	0.4	120	0.4	120
3	Change Left Outer Envelop Reel	0.8	120	9999	9999
4	Change Right Filter Paper Reel	0.6	120	9999	9999
5	Change Right Tag Reel	0.4	120	9999	9999
6	Change Right Outer Envelop Reel	0.8	120	9999	9999
7	Mission For Operator	0.1	100	9999	9999
8	Mission Assistance	0.1	100	9999	9999
9	System Shutdown	0.1	0.1	9999	9999

the experiments and their relative costs and expected execution times are listed in Table I. The costs were defined heuristically. Alternatively, one can use automatic methods available in the literature [29], [58]. Missions with ID ≥ 7 are artificial tasks that are triggered by the task manager either according to some internal logic or due to an anomaly that occurred and was raised by the AMR while performing an action."Mission for operator" (ID=7), in particular, is used to send a human operator to either check if the UR10e #2 arm mounts the right set of fingers or to complete the *manual splicing* operation required after a successful reel loading. Setting a high cost for the robot practically prevents the task manager from assigning that task to the AMR. As a consequence, the only tasks which the AMR may handle are the ones with ID equal to 1 or 2.

Upon reception of either of the two tasks, the AMR will leave the charging station and execute the task as illustrated by the flowchart in Fig. 10. In this scheme, only the procedure for a paper-reel substitution is given (ID=1). The procedure for a tag-reel substitution (ID=2) is similar except for the reel-core disposal step, which is, instead, handled by the UR10e #2 while the UR10e #1 will move away to leave more manoeuvring space for the other arm. Thanks to the eye-on-hand camera, it is possible to include either the wagon or the C24-E machine in the planning scene considered for obstacle-free trajectory planning. The accurate spawning of these objects with respect to the AMR is done resorting to visual ChAruko boards[6] placed in a known position with respect to the stationary elements (e.g. wagon and machine) and exploiting their 3D CAD model. The observed reels instead

[6]A *ChAruko board* is a combination of a *chessboard pattern* and an *ArUco board*. The latter contains many *ArUco markers*, synthetic square markers composed of a wide black border and an inner binary matrix which determines its identifier. This allows a swift board detection even if some markers are occluded.

are added to the scene thanks to a vision-based detection and pose estimation discussed in [59].

For the sake of simplicity, the scheme does not show the fallback procedures[7] in detail. In general, if an erroneous behaviour is controllable and reversible, such as the missed detection of a marker or an unexpected self-collision when manipulating the reel, the robot will:

1) in case the reel is attached to the gripper, autonomously try to place it back to the picking-up point (wagon or on-board buffer);
2) move back to idle pose (both arms safely retracted, grippers closed);
3) move to the charging station;
4) request a "Mission Assistance" task (ID=8) for the operator to restore any anomaly in the scene (e.g. reel on board, reel core wrongly placed, etc.) and/or complete the task.

If any stage fails, the system will raise an error and stop completely. At this moment, the manual intervention of an experienced user is the only way to revert and reset this state, and it is notified by requesting a "System Shutdown" task (ID=9). However, as explained in Sec. VI-A, the overall collaborative job is not stopped; ideally, it could continue by assigning all the remaining tasks to the human operator, see Alg. 1 Line 11.

When the AGV moves, the safety layer and the flexible execution layer remain idle because the arms are retracted and blocked, and the safety handling is passed to the MiR500 technology. Thanks to the laser scanners on opposite corners of the platform and the stereo camera at its front, in fact, the presence of a human can be promptly detected. This will trigger either a replanning of the path that reaches the target work location or a safety-monitored stop in case the obstacle is too close. In all other cases, i.e. when the AGV is not moving and the robotic arms are in motion, the safety layer ensures that the AMR halts when a human is too close to the robot. At the same time, the flexible execution layer ensures that the scene's current obstacle is avoided while modulating the operative speed as described in Sec. V-A2.

Video 1 shows a portion of a collaborative job composed by four tasks: the reel-change task of a filter-paper reel T_1 and of a tag reel T_2, both on the right-side mandrels of the automatic machine, identified by the mission ID 1 and 2, and the respective *manual splicing* tasks T_3 and T_4, both identified by mission ID 7. According to the costs in Tab. I, the Task Assignment assigns to the robot T_1 and T_2, while the human operator must perform the task T_3 and T_4. In Fig. 11, a few snapshots of the former video are displayed and capture the most salient moments of the full cycle. It is worth emphasize that, in the video, the human operator deliberately hinders and disturbs the robot during execution, instead of waiting for the robot to finish its task. This allowed to show the full potentiality of the safety features to be shown. Furthermore, T_3 and T_4 should be executed after the robot finishes its task. This is why they are not shown in the video, which would depict the human performing the task while the robot is stopped.

[7]An operation that reverts a failed change of state and brings the system back to the original one.

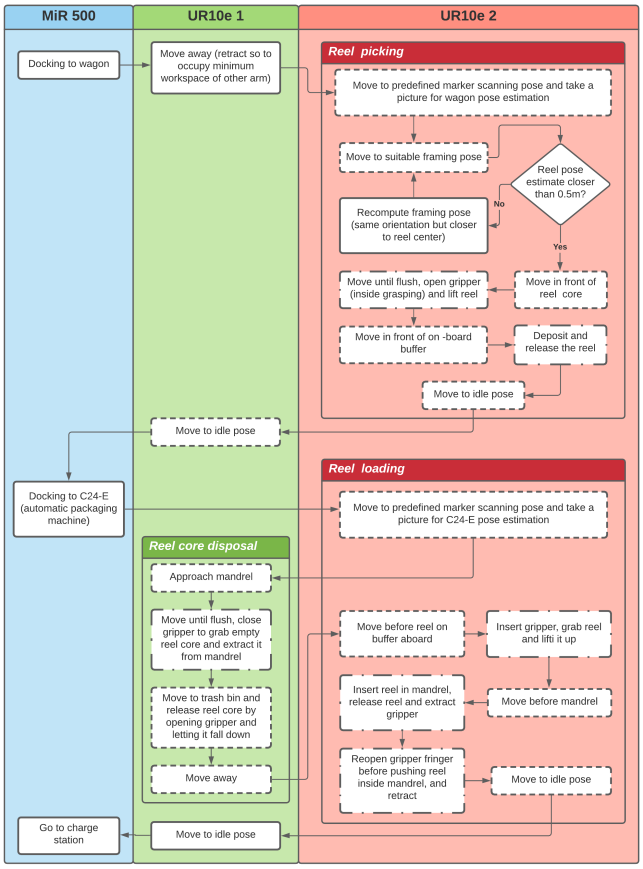
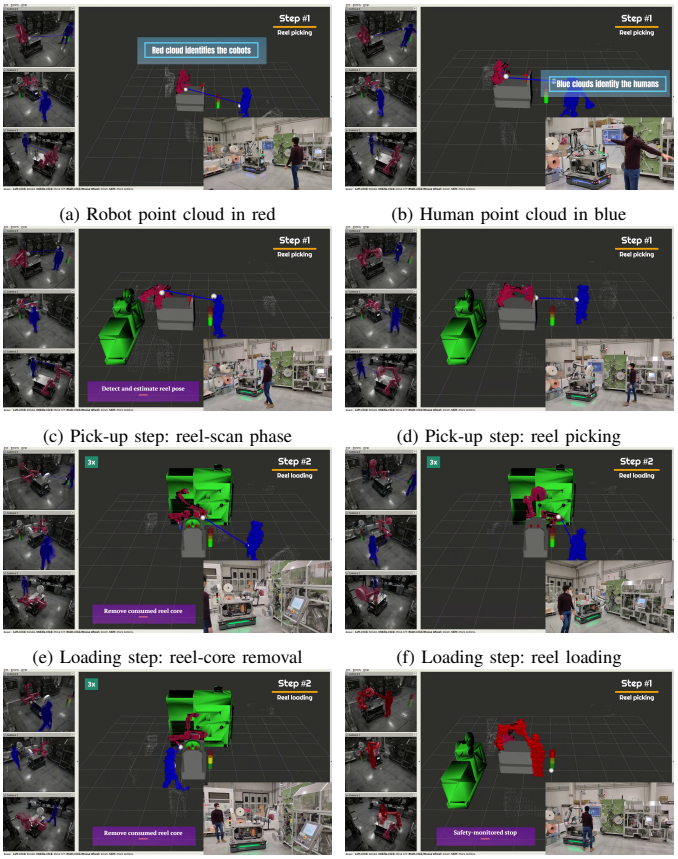


Fig. 10: Paper-reel substitution procedure. Dashed blocks define obstacle-free trajectory motions computed on-the-fly using the flexible execution layer. Dot-dashed blocks identify simple linear trajectory motions executed up to stroke or the desired distance is reached without considering the planning scene.

The video 2, instead, illustrates an example of dynamic replanning where the non-safety information coming from the RS[4] sensors is used to generate a dynamic obstacle (in green) representing the human agent. In this video, it is also possible to better appreciate the flexible execution layer at work. The replanning occurs every time the collision object is updated, in this case, at an average rate of 15 Hz, if a collision is detected within 20 steps ahead of the current trajectory. Regarding computing time, replanning takes 20 ms on average, like any trajectory planning in this framework. A few snapshots of this video are proposed in Fig. 12. Fig. 13 proves that the proposed scaling strategy always guarantees that the robot speed is always lower then the imposed upper bound. Furthermore, when the human operator and the robot are really close it is better for the safety to stop the robot in any case, i.e. imposing $\delta_s = 0$. This can be noted from $t = 50$ s to $t = 75$ s, and from $t = 120$ s to $t = 150$ s.

It is worth to emphasize that the trajectory scaling works at the same frequency as the robot, i.e. 500 Hz. Indeed, thanks to its convexity, the optimization problem in (17) can be easily solved in less than 1 ms. Furthermore, this optimization problem is the only one that must be solved in real-time [60], [61], while the planner can work at slower frequencies. Indeed,



(a) Robot point cloud in red (b) Human point cloud in blue

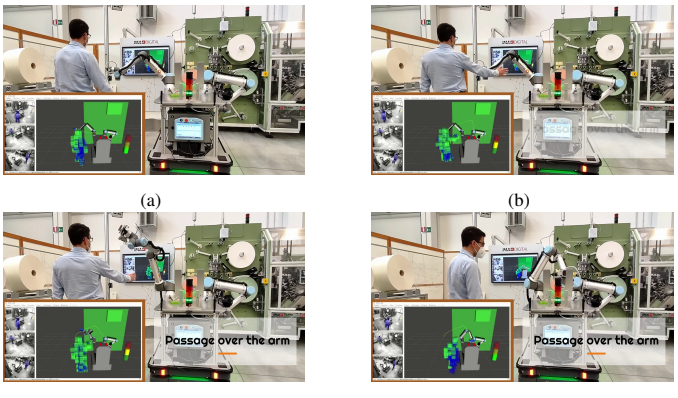
(c) Pick-up step: reel-scan phase (d) Pick-up step: reel picking

(e) Loading step: reel-core removal (f) Loading step: reel loading

(g) Example of reduced speed (h) Example of safety-monitored stop

Fig. 11: Snapshot from video 1 of a filter-paper reel change task. The safety-cameras stream is displayed on the left side of each image. In the bottom right corner, the live view from an external camera. In the background, the planning scene with augmented information, such as the human/robot point clouds.

as mentioned in Section VI-B1, when a replanning occurs, the new trajectory is merged with the previous one to ensure continuity. Consequently, the scaling block remains unaffected by the trajectory change and continues to solve the optimization, incrementing the curvilinear abscissa s according to eq. (19).



(a) (b)

(c) (d)

Fig. 12: Snapshot from video 2 of a replanning example.

B. Mock-up Scenario

In the industrial scenario provided by the IMA company, the scheduling layer has not shown its full potential. This

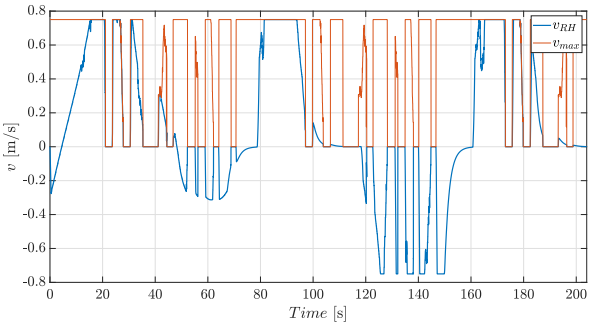


Fig. 13: Real robot speed towards the human operator, v_{RH}, with the respective upper bound, v_{max}.

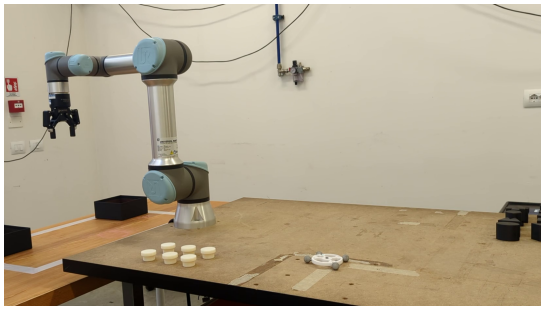


Fig. 14: Setup of the mock-up scenario.

is because the number of tasks to be scheduled during the experiments is limited, i.e. 4 tasks. As the number of tasks increases, it becomes crucial to have a scheduling strategy that can ensure optimality. Thus, for this purpose, a mock-up scenario has been set up to stress the scheduling layer and compare it with a state-of-the-art method available in the literature. In this scenario, the human operator has to collaborate with a UR5e robot in order to fill two boxes with 3D printed shapes. In the end, each box must contain 3 white shapes and 6 black shapes. The entire setup is illustrated in Fig. 14, while a detailed description of the tasks is provided in Tab. II. Fig. 15, instead, shows the directed acyclic graph of the job, highlighting the precedences constraints, i.e. the arrows as detailed in VI-A. It is worth noting that for this specific scenario, the Flexible Execution Layer has been deactivated[8], proving also the modularity of the overall architecture.

As the collaborative job starts, the Task Assignment solves online the optimization problem defined in eq. (16). The resulting schedule is the following:

- $S_H = \{[], [11-16], [], [], [17-22], []\}$,
- $S_R = \{[1], [5-7], [3], [2], [8-10], [4]\}$.

In particular, the human operator has three empty levels, this means that while the robot is performing T_2, T_3, and T_4, there are no other tasks that can be executed in parallel. The overall schedule is also represented by the Gantt Chart in Fig. 16a.

[8]The human operator is not detected inside the scene, i.e. the robot behaviour is not always safe

TABLE II: List of tasks. P&P stands for *Pick and Place*.

T_i	**Description**	w_H	t_H [s]	w_R	t_R [s]
1-2	P&P of empty box	0.8	20.0	0.2	15.0
3-4	P&P of full box	0.8	20.0	0.2	15.0
5-10	P&P of white 3D shape	9999	9999	0.5	16.0
11-22	P&P of black 3D shape	0.5	8.0	9999	9999

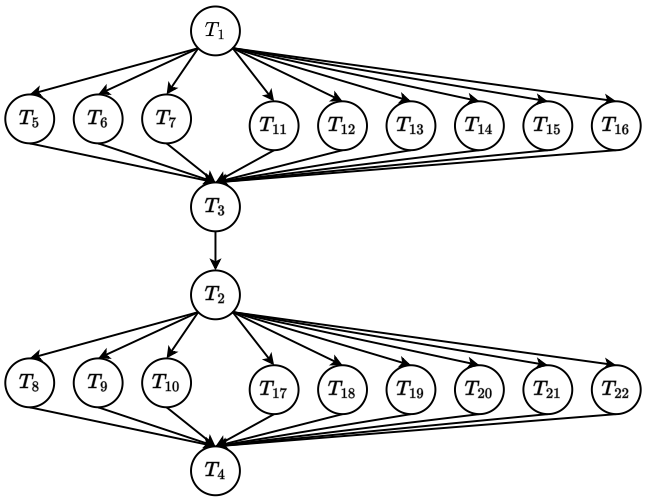
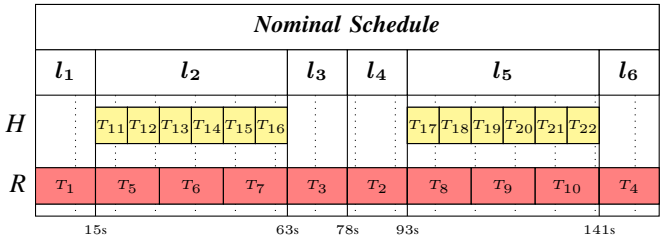
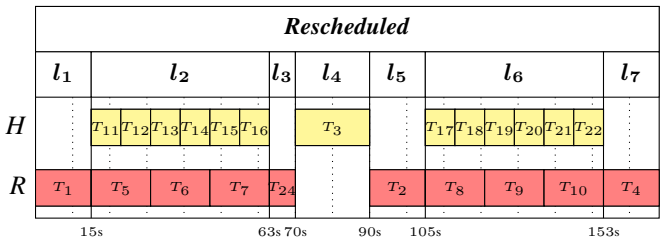


Fig. 15: Directed acyclic graph of the collaborative job performed in the mock-up scenario.



(a)



(b)

Fig. 16: Gantt Charts of the mock-up scenario. l_i stands for the i-th level.

At this point, the robot starts the job by picking and placing the box, i.e. T_1. Subsequently, the two actors fill the box terminating all the tasks in the second level. Then, the robot starts executing T_3, but it fails in the picking phase and the gripper reaches the maximum closure. This triggers a change in the job data Δ_{JQ}: $w_{3R} = 9999$ and a new task with maximum priority is added T_{23}. This new task can only be executed by the robot, i.e. high weights for the human operator, and allows to move the robot in a safe home position. According to Alg. 1 this causes a reschedule, see Line 16. The updated schedule is obtained in 0.03 seconds and it is equal to:

- $S_H = \{[], [3], [], [17-22], []\}$,
- $S_R = \{[24], [], [2], [8-10], [4]\}$,

which is also represented by the Gantt Chart in Fig. 16b. As a consequence, the robot immediately moves in the home position and the task T_3 is executed by the human operator. From this point the two actors conclude the programmed schedule.

The entire collaboration, the online rescheduling, and the results can also be appreciated in the accompanying video 3.

To prove the novelty of the proposed work, the results have been compared with the task scheduling algorithm proposed in [62], where, to have a more faithful comparison, the rescheduling trigger has been changed with the one proposed in this paper. The obtained schedule is the same as the one obtained with the proposed algorithm, i.e. the one in Fig. 16a. As before, the two actors start collaborating, but during the execution of T_3 the robots makes an error and the reschedule procedure starts. Also after the reschedule the result is the same, i.e. the new schedule is the one obtained in Fig. 16b. However, the algorithm proposed in [62] requires 31.6 seconds to obtain the solution. This is shown in the second part of video 3. It is worth noting that both optimization problems have been solved by exploiting an open-source solver available in the literature, i.e. Coin-or branch and cut (CBC) [63].

VIII. Conclusion and Future Works

In this paper, we presented a dynamic architecture that addresses the Task Allocation and Motion Planning problem for real and complex industrial Human-Robot Collaboration scenarios, explicitly handling the safety issues and possible errors that may arise at run-time. Firstly, the architecture assigns tasks between the human operator and the robot, maximizing the parallelism between the actors and reducing collaboration costs. Subsequently, during the execution, the proposed framework leverages a safe-certifiable monitoring system to detect the human operator in the working area. This information is exploited to dynamically adapt the robot's velocity along the planned path or replan a new one. The architecture has been validated in a use case provided by IMA s.p.a., where the human operator and the robot have to change and load the reels of an industrial machine. The experimental part focuses on the tasks performed by the robot, i.e. validation of the recovery from the errors and the safety trajectory scaling implementation.

Future works will aim at improving all presented strategies. The safety monitoring system can be enriched with non safe certified sensors and algorithm that allow to track the human operator inside the working area, e.g. AI techniques for skeleton tracking. The information coming from these sensors can then be exploited for non safe features, such as rescheduling tasks based on the human execution time [50]. The dynamic scheduling strategy can be improved by including other important data that impact the collaboration like the human fatigue [64] or the transition time between tasks [65]. Furthermore, there is potential to deeply integrate the task scheduling procedure with trajectory execution. This integration would enable the robot to optimize for tasks with minimized collision risks, and it would open the door to extending the study to multi-robot systems. Regarding the trajectory execution strategy, it is mainly based on the Speed and Separation Monitoring imposed by the ISO/TS 15066. Still, the performances can be improved considering also the limit imposed by the PFL collaborative mode. However, this is not trivial. As explained in [66], [67], the PFL limit depends

on the closest human-body part: detecting and identifying such a part in a safe-certified way is challenging, and it is not currently possible with the proposed monitoring system. This work could be further enhanced by conducting a statistical analysis on a meaningful real-world use case, comparing it with other works in the literature, such as [33], [34], [40]. This would also provide the industry with a more comprehensive evaluation of safety.

REFERENCES

[1] N. Pedrocchi, F. Vicentini, M. Matteo, and L. M. Tosatti, "Safe Human-Robot Cooperation in an Industrial Environment," *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 27, 2013. [Online]. Available: https://doi.org/10.5772/53939

[2] C. Hinojosa and X. Potau, "Advanced industrial robotics: taking human-robot collaboration to the next level," European Foundation for the Improvement of Living and Working Conditions, Tech. Rep., Jan. 2018. [Online]. Available: https://www.eurofound.europa.eu/sites/default/files/wpfomeef18003.pdf

[3] K. Berntorp, K.-E. Arzen, and A. Robertsson, "Mobile manipulation with a kinematically redundant manipulator for a pick-and-place scenario," in *2012 IEEE International Conference on Control Applications*. Dubrovnik, Croatia: IEEE, Oct. 2012, pp. 1596–1602.

[4] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stuckler, A. Berner, J. Li, R. Klein, and S. Behnke, "Mobile bin picking with an anthropomorphic service robot," in *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, May 2013, pp. 2327–2334.

[5] G. Michalos, S. Makris, J. Spiliotopoulos, I. Misios, P. Tsarouchi, and G. Chryssolouris, "ROBO-PARTNER: Seamless Human-Robot Cooperation for Intelligent, Flexible and Safe Operations in the Assembly Factories of the Future," *Procedia CIRP*, vol. 23, pp. 71–76, 2014, 5th CATS 2014 - CIRP Conference on Assembly Technologies and Systems, Patras, Greece. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827114011366

[6] J. Saenz, F. Penzlin, C. Vogel, and M. Fritzsche, *VALERI - A Collaborative Mobile Manipulator for Aerospace Production*. World Scientific, Aug. 2016, pp. 186–195. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789813149137_0024

[7] A. Tudico, N. Lau, E. Pedrosa, F. Amaral, C. Mazzotti, and M. Carricato, "Improving and Benchmarking Motion Planning for a Mobile Manipulator Operating in Unstructured Environments," in *Progress in Artificial Intelligence*, E. Oliveira, J. Gama, Z. Vale, and H. Lopes Cardoso, Eds. Cham: Springer International Publishing, 2017, pp. 498–509. [Online]. Available: https://doi.org/10.1007/978-3-319-65340-2_41

[8] J. Aleotti, A. Baldassarri, M. Bonfè, M. Carricato, D. Chiaravalli, R. Di Leva, C. Fantuzzi, S. Farsoni, G. Innero, D. Lodi Rizzini, C. Melchiorri, R. Monica, G. Palli, J. Rizzi, L. Sabattini, G. Sampietro, and F. Zaccaria, "Toward Future Automatic Warehouses: An Autonomous Depalletizing System Based on Mobile Manipulation and 3D Perception," *Applied Sciences*, vol. 11, no. 13, 2021. [Online]. Available: https://doi.org/10.3390/app11135959

[9] F. Zaccaria, A. Baldassarri, G. Palli, and M. Carricato, "A Mobile Robotized System for Depalletizing Applications: Design and Experimentation," *IEEE Access*, vol. 9, pp. 96 682–96 691, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3092580

[10] A. Baldassarri, M. Bertelli, and M. Carricato, "Design of a reconfigurable mobile collaborative manipulator for industrial applications," *Journal of Computational and Nonlinear Dynamics*, vol. 18, no. 9, p. 091006, 06 2023. [Online]. Available: https://doi.org/10.1115/1.4062595

[11] M. Yang, E. Yang, R. C. Zante, M. Post, and X. Liu, "Collaborative mobile industrial manipulator: A review of system architecture and applications," in *2019 25th International Conference on Automation and Computing (ICAC)*. Lancaster, United Kingdom: IEEE, Sep. 2019, pp. 1–6.

[12] Z. M. Bi, C. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102022, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584520302337

[13] 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.

[14] I. 10218-1:2011(E), "Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 1: Robots," International Organization for Standardization, Geneva, Switzerland, Standard, Jul. 2011.

[15] I. 10218-2:2011(E), "Robots and Robotic Devices–Safety Requirements for Industrial Robots–Part 2: Robot systems and integration," International Organization for Standardization, Geneva, Switzerland, Standard, Jul. 2011.

[16] I. 15066:2016(E), "Robots and robotic devices–Collaborative robots," International Organization for Standardization, Geneva, Switzerland, Technical Specification, Feb. 2016.

[17] S. Comari, R. Di Leva, M. Carricato, S. Badini, A. Carapia, G. Collepalumbo, A. Gentili, C. Mazzotti, K. Staglianò, and D. Rea, "Mobile cobots for autonomous raw-material feeding of automatic packaging machines," *Journal of Manufacturing Systems*, vol. 64, pp. 211–224, 2022. [Online]. Available: https://doi.org/10.1016/j.jmsy.2022.06.007

[18] C. Vogel, M. Poggendorf, C. Walter, and N. Elkmann, "Towards safe physical human-robot collaboration: A projection-based safety system," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3355–3360.

[19] T. Bänziger, A. Kunz, and K. Wegener, "Optimizing human-robot task allocation using a simulation tool based on standardized work descriptions," *Journal of Intelligent Manufacturing*, vol. 31, pp. 1635–1648, 2020.

[20] J. Saenz, R. Behrens, E. Schulenburg, H. Petersen, O. Gibaru, P. Neto, and N. Elkmann, "Methods for considering safety in design of robotics applications featuring human-robot collaboration," *The International Journal of Advanced Manufacturing Technology*, vol. 107, pp. 2313–2331, 2020.

[21] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3D reconstruction with RGB-D cameras," *Computer Graphics Forum*, vol. 37, no. 2, pp. 625–652, 2018.

[22] 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)*. Istanbul, Turkey: IEEE, 2015, pp. 85–90.

[23] M. Lippi and A. Marino, "Human Multi-Robot Safe Interaction: A Trajectory Scaling Approach Based on Safety Assessment," *IEEE Transactions on Control Systems Technology*, 2020.

[24] 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)*. Big Island, Hawaii, United States of America: IEEE, 2017, pp. 1294–1299.

[25] 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.

[26] 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.

[27] 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*. Chicago, Illinois, United States of America: IEEE, 2014, pp. 3713–3719.

[28] 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. [Online]. Available: https://doi.org/10.1109/LRA.2021.3068634

[29] P. Tsarouchi, A.-S. Matthaiakis, S. Makris, and G. Chryssolouris, "On a human-robot collaboration in an assembly cell," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 6, pp. 580–589, 2017.

[30] 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.

[31] 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.

[32] 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.

[33] J. A. Shah and B. C. Williams, "Fast dynamic scheduling of disjunctive temporal constraint networks through incremental compilation." in *ICAPS*, 2008, pp. 322–329.

[34] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[35] S. M. 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.

[36] 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.

[37] 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.

[38] A. Pupa, C. T. Landi, M. Bertolani, and C. Secchi, "A Dynamic Architecture for Task Assignment and Scheduling for Collaborative Robotic Cells," in *Human-Friendly Robotics 2020*, M. Saveriano, E. Renaudo, A. Rodríguez-Sánchez, and J. Piater, Eds. Cham, Switzerland: Springer International Publishing, 2021, pp. 74–88. [Online]. Available: https://doi.org/10.1007/978-3-030-71356-0_6

[39] 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.

[40] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "Grstaps: Graphically recursive simultaneous task allocation, planning, and scheduling," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, 2022.

[41] M. Faroni, M. Beschi, S. Ghidini, N. Pedrocchi, A. Umbrico, A. Orlandini, and A. Cesta, "A Layered Control Approach to Human-Aware Task and Motion Planning for Human-Robot Collaboration," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1204–1210.

[42] 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)*, Prague, Czech Republic, 2021, pp. 1895–1902. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636855

[43] A. Aalerud, J. Dybedal, and G. Hovland, "Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers," *Sensors*, vol. 19, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1561

[44] G. Andreoni, M. Selvatici, and M. Arrfou, "System and method for verifying positional and spatial information using depth sensors," U.S. patentus US-20 230 196 495-A1, Jun, 2023. [Online]. Available: https://patentimages.storage.googleapis.com/94/db/6e/f54c086e2471ab/EP4205914A1.pdf

[45] M. Axelsson, M. Karlsson, and S. Rudner, "Automatic multi-camera calibration for deployable positioning systems," in *Signal Processing, Sensor Fusion, and Target Recognition XXI*, vol. 8392. SPIE, 2012, pp. 558–563.

[46] R. Itu, D. Borza, and R. Danescu, "Automatic extrinsic camera parameters calibration using convolutional neural networks," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2017, pp. 273–278.

[47] 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.

[48] A. Pupa, M. Minelli, and C. Secchi, "A dynamic planner for safe and predictable human-robot collaboration," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 507–514, 2023. [Online]. Available: https://doi.org/10.1109/LRA.2023.3334977

[49] M. Arrfou, G. Andreoni, and C. Saporetti, "Clustering and detection system and method for safety monitoring in a collaborative workspace," U.S. patentus US-20 240 012 383-A1, Jan, 2024. [Online]. Available: https://ppubs.uspto.gov/dirsearch-public/print/downloadPdf/20240012383

[50] 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. [Online]. Available: https://doi.org/10.1109/LRA.2021.3068888

[51] 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.

[52] 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.

[53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[54] 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. [Online]. Available: https://doi.org/10.3390/s22134901

[55] S. M. LaValle, "Rapidly-Exploring Random Trees : A New Tool for Path Planning," *The annual research report*, 1998.

[56] 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. Sendai, Japan: IEEE, 2004, pp. 1606–1611.

[57] 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*. Kobe, Japan: IEEE, May 2009, pp. 489–494.

[58] N. Nikolakis, K. Sipsas, P. Tsarouchi, and S. Makris, "On a shared human-robot task scheduling and online re-scheduling," *Procedia CIRP*, vol. 78, pp. 237–242, 2018.

[59] S. Comari and M. Carricato, "Vision-Based Robotic Grasping of Reels for Automatic Packaging Machines," *Applied Sciences*, vol. 12, no. 15, 2022. [Online]. Available: https://doi.org/10.3390/app12157835

[60] A. G. Pandala, Y. Ding, and H.-W. Park, "qpswift: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.

[61] K. Merckaert, B. Convens, M. M. Nicotra, and B. Vanderborght, "Real-time constraint-based planning and control of robotic manipulators for safe human–robot collaboration," *Robotics and Computer-Integrated Manufacturing*, vol. 87, p. 102711, 2024.

[62] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1017–1023.

[63] J. Forrest, T. Ralphs, S. Vigerske, H. G. Santos, J. Forrest, L. Hafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, Jan-Willem, rlougee, jpgoncal1, S. Brito, h-i gassmann, Cristina, M. Saltzman, tosttost, B. Pitrus, F. MATSUSHIMA, and to st, "coin-or/cbc: Release releases/2.10.11," oct 2023. [Online]. Available: https://doi.org/10.5281/zenodo.10041724

[64] 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.

[65] M. Lippi, P. Di Lillo, and A. Marino, "A task allocation framework for human multi-robot collaborative settings," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7614–7620.

[66] A. Pupa, M. Minelli, and C. Secchi, "A time-optimal energy planner for safe human-robot collaboration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 17 373–17 379. [Online]. Available: https://doi.org/10.1109/ICRA57147.2024.10611118

[67] F. Benzi, F. Ferraguti, and C. Secchi, "Energy tank-based control framework for satisfying the iso/ts 15066 constraint," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1288–1293, 2023.