A Control Architecture for Safe Trajectory Generation in Human–Robot Collaborative Settings

Jozsef Palmieri, *Graduate Student Member, IEEE*, Paolo Di Lillo, *Associate Member, IEEE*, Martina Lippi, *Associate Member, IEEE*, Stefano Chiaverini, *Fellow, IEEE*, and Alessandro Marino, *Senior Member, IEEE*

***Abstract*— This paper introduces a control architecture that enables a robotic system to ensure the safety of human operators entering its workspace. The proposed method utilizes an appropriate metric to measure safety levels and adjusts the robot's motion to maintain this metric above a minimum threshold. To guarantee safety, the robot scales down and deviates from its intended path. For redundant robots, internal motion is exploited to enhance safety levels further. The approach is incorporated into a Hierarchical Quadratic Programming control framework, allowing the robot to address other control objectives simultaneously, such as handling joint limits. Experimental results with a dual-arm mobile robot developed as part of the EU-funded CANOPIES project demonstrate the effectiveness of the proposed method.**

***Note to Practitioners*—This paper was motivated by the problem of ensuring human safety in unstructured environments shared with human operators. We propose a control architecture that allows complex dual-arm robotic systems to operate effectively in such scenarios. The devised architecture gives the robot the capability to slow down a trajectory to follow as well as to deviate from a nominal path to keep a human operator safe. We tested the devised approach in a precision farming setting; however, it can be adopted in any human-robot interaction scenario.**

***Index Terms*— Human–robot interaction, human safety, dual-arm system, trajectory scaling, path modification, precision agriculture.**

I. Introduction

WITH the advancement of technology, robots are becoming more and more prevalent in our daily lives. From factories [1] to operating rooms [2], from agricultural fields [3] to homes [4], robots are increasingly used to perform various tasks and make our lives easier. This means that robots can perform tasks that are dangerous or tedious for human workers while allowing humans to focus on more complex tasks. This potentially leads to increased efficiency, productivity, and cost savings for companies adopting collaborative robots. However, as robots become more integrated into our society, there is an important question to consider: how can we coexist in a workspace shared with robots in a way that is safe and "low-invasive" for everyone involved?

Regarding safety, current collaborative robots have numerous integrated safety features, e.g., lightweight or emergency stop procedures. However, additional safety strategies still need to be designed to guarantee that robots do not pose any risks to human workers in the shared workspace, especially in unstructured, dynamic environments. In particular, it is necessary to ensure that, in any operational condition of the human operator and the robots, the human level of safety can be identified and appropriate intervention is carried out if human safety may be compromised. Such safety aspects have also been addressed in several standards for industrial contexts. Among these, the technical specification ISO/TS 15066:2016 "Robots and robotic devices - Collaborative Robots" specifically focuses on the safety requirements for industrial robot systems.

Although ensuring safety is of utmost importance, we also recognize the value of allowing the robot to successfully carry out the assigned tasks while adhering to its constraints. This means that a balance should be found between preserving the robot's activity and implementing a human safety strategy, i.e., the human safety strategy cannot be overly conservative;

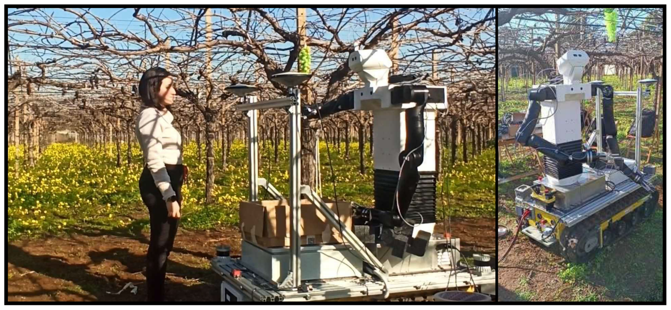
Fig. 1. Example of robot sharing its workspace with a human operator in an agriculture context.

Manuscript received 4 November 2023; accepted 2 January 2024. Date of publication 24 January 2024; date of current version 15 January 2025. This article was recommended for publication by Associate Editor H. Wang and Editor J. Yi upon evaluation of the reviewers' comments. This work was supported in part by the Horizon 2020-Information and Communication Technologies (H2020 ICT) Project CANOPIES under Grant 101016906, in part by the Programma Operativo Regionale (POR) Fondo Sociale Europeo (FSE) LAZIO (2014–2020) under Project DE G06374/2021, and in part by the Project "Ecosistema dell'innovazione—Rome Technopole" financed by European Union (EU) in NextGenerationEU Plan through Ministero dell'Università e della Ricerca (MUR) under Grant 1051 (23.06.2022). *(Corresponding author: Paolo Di Lillo.)*

Jozsef Palmieri, Paolo Di Lillo, Stefano Chiaverini, and Alessandro Marino are with the Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, 03043 Cassino, Italy (e-mail: pa.dilillo@unicas.it).

Martina Lippi is with the Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, 00146 Roma, Italy.

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2024.3350976, provided by the authors.

Digital Object Identifier 10.1109/TASE.2024.3350976

otherwise, it may prove unnecessarily invasive and detrimental to the robot's mission.

Motivated by the above considerations, in this work, we propose a comprehensive architecture for safe human-robot coexistence that modifies the robotic task by taking into account human safety features as well as constraints or secondary objectives of the robotic platform. Figure 1 shows an illustrative example of the considered setup, where a human operator shares the workspace with a mobile dual-arm robot in a precision agriculture setting. Specifically, we design a safety planner that, based on an optimal formulation, allows modulating the velocity and deviating from the planned nominal path to ensure human safety. The proposed solution relies on Control Barrier Functions (CBFs) and is integrated within a Hierarchical Quadratic Programming (HQP) framework to take into account possible constraints or general objectives of the robotic platform. We validate the proposed solution in an agricultural setting inspired by the H2020 European project CANOPIES, where the robotic platform is involved in a harvesting task in a table-grape vineyard shared with human operators. This work is built on [5], where a CBF-based approach was first presented to guarantee human safety in human-multi-robot settings by modulating the velocity of the robot team. Here, the approach is significantly extended by *i)* introducing the possibility of altering the assigned robot path, *ii)* integrating the strategy within an HQP framework to handle several control objectives, *iii)* providing a formulation for dual-arm systems, and *iv)* validating the approach within a challenging real-world agricultural setting. The main contributions of the paper can be summarized as follows:

- design of a comprehensive control architecture to define the robot inputs aimed at ensuring that the human level of safety is always above a certain threshold while satisfying robot constraints;
- formulation for dual-arm systems exploiting the redundancy to improve safety;
- extensive validation in both laboratory and real outdoor agricultural settings;
- comparison with additional baselines for ensuring human safety.

The remainder of the manuscript is organized as follows. Relevant works are discussed in Section II. Preliminary notions for the proposed methodology, including the robot kinematic model and the human safety index, are presented in Section III. Based on these, the main problem addressed in this work is formally stated and a solution is designed in Section IV. Experimental results, both in laboratory conditions and in real outdoor settings, are reported in V, while conclusive remarks are provided in VI.

II. Related Works

As mentioned in the Introduction, the main challenge in a human-robot co-existence scenario lies in formulating a strategy that not only guarantees continuous human safety throughout all robotic operations but also optimizes the robot's efficiency in executing tasks. One of the most commonly adopted methods to ensure safe coexistence is through the implementation of reactive evasive strategies: when the human is in the proximity of the robot, the latter moves away to increase safety. In this context, the study in [6] has defined a danger index based on distance, velocity, and inertia measures and then produced a virtual force moving the robot away from the human and with an intensity proportional to the danger index. The human prediction based on neural networks has also been introduced in [7] for the reactive control safety strategy, while a fast method to compute the distance between the robot and moving obstacles using RGB-D sensors has been proposed in [8], and exploited to generate repulsive forces. More recently, a method combining a 3D depth sensor and robot position sensors has been developed in [9] to estimate the human-robot distance more reliably. As for the previous work, this has been used to generate the repulsive force driving the robot away from the human. The problem of computing repulsive forces for collision avoidance actions even if obstacles are moving faster than the robot has been addressed in [10], where a quadratic programming optimization problem has been formulated to minimize the deviation from the reference trajectory. Repulsive and attractive forces have also been exploited in [11], where an architecture based on Explicit Reference Governor has been proposed. This generates the robot trajectory, driving the robot towards a desired pose while taking into account the input and state constraint of the nonlinear robot dynamics.

A further approach to human safety relies on modulating the robot's velocity without deviating from the desired path while guaranteeing speed and separation monitoring (SSM) [12]. This strategy has been pursued, for instance, in [13] where robot links are represented as beams and an optimization problem is formulated which provides the velocity scaling factor and the joint velocity commands while guaranteeing a minimum human-robot distance and the fulfillment of physical robot constraints. This work has been recently extended in [14] to include an efficient method to compute danger zones in real-time, i.e., the 3D volumes corresponding to links of the robot where safety constraints are violated. Computing optimal smooth stop trajectories for collision avoidance has been addressed in [15], where the dynamics and torque limits of the robot are taken into account. A proactive optimization framework has been proposed in [16] to find the path that minimizes the expected overall execution time, obtained by considering worst-case scenarios for human safety. In order to determine the expected execution time, the human state and possible slowing down of the trajectory for safety reasons have been taken into consideration. A further human-safe solution minimizing the overall execution time and modulating the robot velocity, can be found in [17], where an online replanning module has been included to optimally adapt the trajectory in case of violations of safety constraints. A multi-modal system integrating both depth and thermal cameras has been developed instead in [18], where a fuzzy inference method is presented to modulate the robot velocity to achieve human safety. Additionally, the case of multiple mobile manipulators has been addressed in [19], where a trajectory scaling approach has been devised by taking into account the relative distance and velocity information of the robotic team with respect to the human operators.

To conclude, we can observe that the reactive evasive method on the one hand allows for a rapid reaction to immediate proximity, potentially minimizing the risk of collision; on the other hand, frequent reactive movements may disrupt the robot task execution, leading to inefficiencies. As far as the trajectory scaling approach is concerned, on the one hand, it enables the robot to maintain a consistent path and continue its task execution, contributing to operational efficiency; on the other hand, depending on the velocity modulation, there may be a risk of closer proximity between the robot and the human, which requires careful calibration. Furthermore, relying solely on velocity modulation could potentially result in situations where the robot comes to a halt and experiences stalls if the person stops directly along the assigned path. In light of these considerations, we propose a novel hybrid approach where we aim to find a combination of velocity modulation and path modification to guarantee human safety. Such a combination is modulated through the selection of weights in the formulated optimization problem. In this way, the system can benefit from the strengths of both strategies, i.e., the robot can effectively avoid collisions with the human by modulating the velocity and modifying the path as required to continue the robot task operation. Moreover, we propose a comprehensive framework that also takes into account the robot's physical limitations and exploits the redundancy of the system to maximize human safety.

III. Mathematical Background and Problem Formulation

In this section, the basic theory of Control Barrier Functions, robot modeling, and Hierarchical Quadratic Programming are first presented. Then, the safety field concept employed to assess the human safety is presented. Finally, we state the formulation of the problem addressed in this paper.

A. Control Barrier Functions

In this section, we provide the basic theory of Control Barrier Functions, which allows the formal handling of inequality constraints [20], [21]. Let us consider a system with dynamics:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(t, \boldsymbol{\xi}) + \boldsymbol{g}(\boldsymbol{\xi})\boldsymbol{u}, \tag{1}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are Lipschitz-continuous vector fields, $\boldsymbol{\xi} \in \mathcal{D} \subset \mathbb{R}^l$ and $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^q$ are state and input of the system, respectively. Let the kth generic constraint be expressed in the following general form: $h_k(\boldsymbol{\xi}) \geq 0$, where $h_k(\cdot)$ is a continuous differentiable function in the domain $\mathcal{D}$. According to the CBF framework, let $\mathcal{C}_k \subset \mathcal{D}$ be defined as:

$$\begin{aligned} \mathcal{C}_k &= \{\boldsymbol{\xi} \in \mathbb{R}^l : h_k(\boldsymbol{\xi}) \geq 0\}, \\ \partial\mathcal{C}_k &= \{\boldsymbol{\xi} \in \mathbb{R}^l : h_k(\boldsymbol{\xi}) = 0\}, \\ \mathrm{Int}(\mathcal{C}_k) &= \{\boldsymbol{\xi} \in \mathbb{R}^l : h_k(\boldsymbol{\xi}) > 0\}, \end{aligned} \tag{2}$$

implying that the state $\boldsymbol{\xi}$ is required to belong to the set $\mathcal{C}_k$ in order to satisfy constraint k. Function h_k is a CBF if an extended class $\mathcal{K}_\infty$ function α_k exists such that, for a dynamic system represented as in Eq. (1), it holds:

$$\sup_{\boldsymbol{u}\in\mathcal{U}} \left[L_f h_k(\boldsymbol{\xi}) + L_g h_k(\boldsymbol{\xi})\boldsymbol{u}\right] \geq -\phi_k \alpha_k(h_k(\boldsymbol{\xi})), \tag{3}$$

where $\phi_k > 0$, and $L_f h_k$ and $L_g h_k$ are the Lie derivatives of function h_k with respect to f and g, respectively. Then, the following theorem holds [20].

Theorem 1: Let function $h_k : \mathcal{D} \subset \mathbb{R}^l \to \mathbb{R}$ be a continuously differentiable function and the corresponding set $\mathcal{C}_k$ defined as in Eq. (2). If h_k is a CBF on $\mathcal{D}$ and $\frac{\partial h_k}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \neq 0$ $\forall \boldsymbol{\xi} \in \partial\mathcal{C}_k$, then any Lipschitz continuous controller $u(\boldsymbol{\xi})$ for system in Eq. (1) satisfying Eq. (3) renders the set $\mathcal{C}_k$ asymptotically stable.

Since Eq. (3) is affine in the control input $\boldsymbol{u}$, the latter can be computed as the result of a convex optimization problem subject to the constraint:

$$\begin{aligned} \boldsymbol{u}^\star &= \arg\min_{\boldsymbol{u}} \frac{1}{2}\left(\boldsymbol{u} - \boldsymbol{u}_{(\cdot)}\right)^{\mathrm{T}} \boldsymbol{Q}\left(\boldsymbol{u} - \boldsymbol{u}_{(\cdot)}\right) \\ \text{s.t.} &\quad \sup_{\boldsymbol{u}\in\mathcal{U}} \left[L_f h_k(\boldsymbol{\xi}) + L_g h_k(\boldsymbol{\xi})\boldsymbol{u}\right] \geq -\phi_k h_k(\boldsymbol{\xi}), \quad \forall k \end{aligned} \tag{4}$$

where $\boldsymbol{u}_{(\cdot)}$ is any nominal input for the system and $\boldsymbol{Q} \in \mathbb{R}^{q\times q}$ is a positive definite weight matrix.

B. Robot Kinematics

Let us consider a mobile robot equipped with a movable torso and a dual arm system (see Figure 1), in which each end effector configuration is denoted by $\boldsymbol{x}_y = [\boldsymbol{p}_y^{\mathrm{T}}\ \boldsymbol{o}_y^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^7$, ($y = L, R$, *L*eft and *R*ight, respectively) where $\boldsymbol{p}_y \in \mathbb{R}^3$ is the position part, and $\boldsymbol{o}_y = \left[o_{y,1}\ o_{y,2}\ o_{y,3}\ o_{y,4}\right]^{\mathrm{T}} = \left[\kappa_{o_y}\ \boldsymbol{\rho}_{o_y}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^4$ represents the unit quaternion expressing the orientation, being κ_{o_y} the scalar component and $\boldsymbol{\rho}_{o_y} \in \mathbb{R}^3$ the vector component. We use the symbol $*$ to indicate the product of two quaternions and the vector $\boldsymbol{o}_0 = \left[1\ 0\ 0\ 0\right]^{\mathrm{T}}$ to represent the quaternion relative to $\boldsymbol{R} = \boldsymbol{I}_3$ (corresponding to a null rotation), where $\boldsymbol{I}_m$ denotes the $m \times m$ identity matrix. We denote by $\boldsymbol{x} = [\boldsymbol{x}_L^{\mathrm{T}}\ \boldsymbol{x}_R^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{14}$ the vector collecting all end effector configurations. Moreover, let the joint vector $\boldsymbol{q} \in \mathbb{R}^n$ defined as:

$$\boldsymbol{q} = \left[\boldsymbol{q}_B^{\mathrm{T}} \quad \boldsymbol{q}_T^{\mathrm{T}} \quad \boldsymbol{q}_L^{\mathrm{T}} \quad \boldsymbol{q}_R^{\mathrm{T}}\right]^{\mathrm{T}},$$

where $\boldsymbol{q}_B = \left[\boldsymbol{p}_B^{\mathrm{T}},\ \boldsymbol{o}_B^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{n_b}$ is the vector describing the mobile base configuration in terms of position and orientation expressed as a quaternion, $\boldsymbol{q}_T \in \mathbb{R}^{n_t}$ gathers the joint variables of the torso, while $\boldsymbol{q}_L, \boldsymbol{q}_R \in \mathbb{R}^{n_m}$ are the vectors of joint variables relative to the manipulator arms, and $n = n_b + n_t + 2n_m$ is the total number of degrees of freedom (DoFs). The collective vector of end effector linear/angular velocities is described by $\boldsymbol{v} = \left[\boldsymbol{v}_L^{\mathrm{T}}\ \boldsymbol{v}_R^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{12}$ and its relationship with the velocity vector can be expressed as:

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_L \\ \boldsymbol{v}_R \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}\ , \tag{5}$$

with the Jacobian $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{12\times n}$ partitioned as follows:

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{J}_{B,L}(\boldsymbol{q}_B) & \boldsymbol{J}_{T,L}(\boldsymbol{q}_T) & \boldsymbol{J}_L(\boldsymbol{q}_L) & \boldsymbol{O}_{6\times n_m} \\ \boldsymbol{J}_{B,R}(\boldsymbol{q}_B) & \boldsymbol{J}_{T,R}(\boldsymbol{q}_T) & \boldsymbol{O}_{6\times n_m} & \boldsymbol{J}_R(\boldsymbol{q}_R) \end{bmatrix}, \tag{6}$$

where $\boldsymbol{O}_{x\times y}$ represents the $x \times y$ null matrix. The structure of Eq. (6) highlights that the joint velocities of the common base and torso contribute to the velocities of both end effectors, while $\dot{\boldsymbol{q}}_L$ ($\dot{\boldsymbol{q}}_R$) only contributes to the velocity of the left (right) end effector. Furthermore, we assume the robot is subject to the following joint-space kinematic constraints:

$$\begin{cases} \underline{\boldsymbol{q}} \leq \boldsymbol{q} \leq \bar{\boldsymbol{q}}, \\ \underline{\dot{\boldsymbol{q}}} \leq \dot{\boldsymbol{q}} \leq \bar{\dot{\boldsymbol{q}}}, \end{cases} \tag{7}$$

where $\underline{\boldsymbol{q}}$ ($\underline{\dot{\boldsymbol{q}}}$) and $\bar{\boldsymbol{q}}$ ($\bar{\dot{\boldsymbol{q}}}$) are the minimum and maximum joint configuration (velocity) values, respectively.

C. Hierarchical Quadratic Programming

In addition to the control of the end-effectors position and orientation, there might be several other tasks that any robot has to fulfill to guarantee safety and efficiency during the operations in complex and dynamic environments, e.g., joint position/velocity limits, self-collisions, and obstacle avoidance. In order to handle possible conflicts that can arise among tasks, it is recommended to set priority orders among them, obtaining a hierarchy. This is usually implemented by projecting the joint velocity related to the fulfillment of a lower-priority task onto the null space of the higher-priority tasks, leading to different kinds of Task-Priority (TP) control frameworks [22], [23].

A popular alternative is to exploit the Hierarchical Quadratic Programming (HQP) control framework [24], [25], in which the task hierarchy is implemented by solving a cascade of Quadratic Programming (QP) problems in which the solution of a task with a given priority is obtained by considering the solutions of all higher-priority tasks as additional constraints.

More in detail, let us consider a task $\boldsymbol{\sigma}_1 \in \mathbb{R}^{m_1}$ with associated Jacobian $\boldsymbol{J}_1(\boldsymbol{q}) \in \mathbb{R}^{m_1 \times n}$, and minimum and maximum desired task velocities $\underline{\boldsymbol{b}}_1 \in \mathbb{R}^{m_1}$ and $\bar{\boldsymbol{b}}_1 \in \mathbb{R}^{m_1}$, respectively. The respective QP problem can be formulated as:

$$\begin{aligned} \min_{\boldsymbol{w}_1, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_1^T \boldsymbol{Q}_{w,1} \boldsymbol{w}_1 \\ \text{s.t.} \quad & \underline{\boldsymbol{b}}_1 \leq \boldsymbol{J}_1(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_1 \leq \bar{\boldsymbol{b}}_1, \end{aligned} \tag{8}$$

where $\boldsymbol{Q}_{w,1} \in \mathbb{R}^{m_1 \times m_1}$ is a weighting matrix, and $\boldsymbol{w}_1 \in \mathbb{R}^{m_1}$ a *slack* variable associated with task $\boldsymbol{\sigma}_1$, that is used to relax the constraint in case of a non-feasible task. It is worth noticing that in case $\underline{\boldsymbol{b}}_1 = \bar{\boldsymbol{b}}_1 = \boldsymbol{b}_1$, the bilateral constraint in Eq. (8) is equivalent to an equality constraint. In the case of a second task $\boldsymbol{\sigma}_2 \in \mathbb{R}^{m_2}$ with Jacobian matrix $\boldsymbol{J}_2(\boldsymbol{q}) \in \mathbb{R}^{m_2 \times n}$ and minimum and maximum task velocities $\underline{\boldsymbol{b}}_2 \in \mathbb{R}^{m_2}$ and $\bar{\boldsymbol{b}}_2 \in \mathbb{R}^{m_2}$, respectively, to be performed with strict lower priority with respect to task 1, the solution can be computed by solving a second QP problem:

$$\begin{aligned} \min_{\boldsymbol{w}_2, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_2^T \boldsymbol{Q}_{w,2} \boldsymbol{w}_2 \\ \text{s.t.} \quad & \underline{\boldsymbol{b}}_1 \leq \boldsymbol{J}_1(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \leq \bar{\boldsymbol{b}}_1, \\ & \underline{\boldsymbol{b}}_2 \leq \boldsymbol{J}_2(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_2 \leq \bar{\boldsymbol{b}}_2, \end{aligned} \tag{9}$$

where, as before, $\boldsymbol{Q}_{w,2} \in \mathbb{R}^{m_2 \times m_2}$ is a weighting matrix and $\boldsymbol{w}_1^\star$ is the solution of the QP problem reported in Eq. (8). The minimization of the slack variable $\boldsymbol{w}_2 \in \mathbb{R}^{m_2}$ allows finding a solution even if the two tasks are in conflict, minimizing the secondary task error without affecting the primary one.

The approach can be generalized for a hierarchy composed of l arbitrary tasks by solving the cascade of l QP problems, leading to a sequence of optimal variables $\{\boldsymbol{w}_1^\star, \boldsymbol{w}_2^\star, \cdots, \boldsymbol{w}_l^\star\}$ that is minimal with respect to a lexicographic order. In detail, the i-th QP problem has the following structure:

$$\begin{aligned} \min_{\boldsymbol{w}_i, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_i^T \boldsymbol{Q}_{w,i} \boldsymbol{w}_i \\ \text{s.t.} \quad & \underline{\boldsymbol{b}}_k \leq \boldsymbol{J}_k(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_k^\star \leq \bar{\boldsymbol{b}}_k, \qquad \forall k \in 1, \ldots, i-1 \\ & \underline{\boldsymbol{b}}_i \leq \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_i \leq \bar{\boldsymbol{b}}_i, \end{aligned} \tag{10}$$

where $\boldsymbol{Q}_{w,i} \in \mathbb{R}^{m_i \times m_i}$ and $\boldsymbol{w}_i \in \mathbb{R}^{m_i}$ are the weighting matrix and slack variables relative to the i th task, respectively, while $\underline{\boldsymbol{b}}_i \in \mathbb{R}^{m_i}$ and $\bar{\boldsymbol{b}}_i \in \mathbb{R}^{m_i}$ are the minimum and maximum desired task velocities.

D. Considered Tasks

In this section, we describe the tasks adopted in the present work and the respective constraints in the HQP formulation.

1) Joint Limits Tasks: Let us consider the first constraint in Eq. (7). It can be expressed by exploiting the CBF framework described in Section III-A by introducing the following functions:

$$\begin{cases} \underline{h}_i(q_i) = q_i - \underline{q}_i \geq 0, & i = 1, \cdots, n \\ \bar{h}_i(q_i) = \bar{q}_i - q_i \geq 0, & i = 1, \cdots, n \end{cases} \tag{11}$$

that, based on Eq. (3), lead to:

$$\begin{cases} \dot{q}_i \geq -\underline{\phi}_i \underline{h}_i(q_i) & i = 1, \cdots, n \\ -\dot{q}_i \geq -\bar{\phi}_i \bar{h}_i(q_i) & i = 1, \cdots, n. \end{cases} \tag{12}$$

with $\bar{\phi}_i, \underline{\phi}_i > 0$. Let us define vector functions $\underline{\boldsymbol{h}} = [\underline{h}_1, \underline{h}_2 \cdots, \underline{h}_n]^T$ and $\bar{\boldsymbol{h}} = [\bar{h}_1, \bar{h}_2 \cdots, \bar{h}_n]^T$, and matrices $\underline{\boldsymbol{\Phi}} = \mathrm{diag}\{\underline{\phi}_1, \underline{\phi}_2, \cdots, \underline{\phi}_n\}$ and $\bar{\boldsymbol{\Phi}} = \mathrm{diag}\{\bar{\phi}_1, \bar{\phi}_2, \cdots, \bar{\phi}_n\}$. It is straightforward to verify that Eq. (12) can be more conveniently expressed as:

$$\underline{\boldsymbol{\Phi}}\underline{\boldsymbol{h}}(\boldsymbol{q}) \leq \dot{\boldsymbol{q}} \leq \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{h}}(\boldsymbol{q}). \tag{13}$$

By combining Eq. (13) with the second of constraints in Eq. (7), it holds:

$$\underline{\boldsymbol{b}}_c = \begin{bmatrix} -\underline{\boldsymbol{\Phi}}\underline{\boldsymbol{h}}(\boldsymbol{q}) \\ \underline{\dot{\boldsymbol{q}}} \end{bmatrix} \leq \boldsymbol{J}_c \dot{\boldsymbol{q}} \leq \begin{bmatrix} \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{h}}(\boldsymbol{q}) \\ \bar{\dot{\boldsymbol{q}}} \end{bmatrix} = \bar{\boldsymbol{b}}_c, \tag{14}$$

where $\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{I}_n \end{bmatrix}^{\mathrm{T}}$ is the constraints Jacobian.

2) Operational Task: As in [26], the robot operational task is specified by means of *a*bsolute and *r*elative task variables:

$$\boldsymbol{\sigma} = \begin{bmatrix} \boldsymbol{\sigma}_a \\ \boldsymbol{\sigma}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma}_{a_p} \\ \boldsymbol{\sigma}_{a_o} \\ \boldsymbol{\sigma}_{r_p} \\ \boldsymbol{\sigma}_{r_o} \end{bmatrix} = \begin{bmatrix} \beta \boldsymbol{p}_L + (1-\beta)\boldsymbol{p}_R \\ \boldsymbol{o}_a \\ \boldsymbol{p}_R - \gamma \boldsymbol{p}_L \\ \boldsymbol{o}_r \end{bmatrix}, \tag{15}$$

where $\boldsymbol{o}_a$ is the quaternion related to the absolute rotation matrix $\boldsymbol{R}_a = \boldsymbol{R}_L \boldsymbol{R}_R^L(\boldsymbol{r}_{RL}^L, (1-\beta)\vartheta_{LR})$, with $\boldsymbol{r}_{LR}^L$ and ϑ_{LR} expressing the axis-angle representation of the relative orientation matrix $\boldsymbol{R}_R^L$ between the two end effectors and $\boldsymbol{o}_r$ is the quaternion extracted from the relative rotation matrix

$\boldsymbol{R}_r = \boldsymbol{R}_L^{\mathrm{T}}(\boldsymbol{r}_L, \gamma\vartheta_L)\boldsymbol{R}_R$, with $\boldsymbol{r}_L$, ϑ_L being the axis-angle representation expressing the orientation of the left end effector. Regarding the variables β and γ, the first is a balance parameter in the range [0 1] expressing the level of symmetry in the arms cooperative motion, while the latter is a binary coordination parameter to select the arms coordination level. In detail, for $\gamma = 1$, the parameter $\beta \in [0, 1]$ provides the possibility to modulate the degree of symmetry between the two end effectors during cooperative operations; namely, $\beta = 0$ or $\beta = 1$ (with $\gamma = 1$) result in a leader-follower approach where either the right end effector is the leader, and the left end effector is the follower ($\beta = 0$) or vice-versa ($\beta = 1$), while $\beta = 0.5$ results in a fully symmetric approach. For $\beta = 1$ and $\gamma = 0$, the arms result in mutually independent movements. In virtue of Eq. (5), the task velocity $\boldsymbol{\nu} \in \mathbb{R}^{12}$ depends on the collective end effector velocities $\boldsymbol{v}$ as follows [26]:

$$\boldsymbol{\nu} = \begin{bmatrix} \beta \boldsymbol{I}_6 & (1-\beta)\boldsymbol{I}_6 \\ -\gamma \boldsymbol{I}_6 & \boldsymbol{I}_6 \end{bmatrix} \boldsymbol{v} = \boldsymbol{J}_t \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad (16)$$

where $\boldsymbol{J}_\sigma = \boldsymbol{J}_t\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{12\times n}$ is the task Jacobian matrix that can be used to formalize a variety of multi-robot tasks of general interest, like a cooperative transportation task [27].

It is worth remarking that parameters β and γ might be time-varying, which means that the robot might change in real-time the cooperation mode. Moreover, we assume that a continuous and differentiable desired trajectory:

$$\boldsymbol{\sigma}^d(t) = \begin{bmatrix} \boldsymbol{\sigma}^d_{a_p} \\ \boldsymbol{\sigma}^d_{a_o} \\ \boldsymbol{\sigma}^d_{r_p} \\ \boldsymbol{\sigma}^d_{r_o} \end{bmatrix} \in \mathbb{R}^{14}, \quad \boldsymbol{\nu}^d(t) = \begin{bmatrix} \dot{\boldsymbol{\sigma}}^d_{a_p} \\ \boldsymbol{\omega}^d_{a} \\ \dot{\boldsymbol{\sigma}}^d_{r_p} \\ \boldsymbol{\omega}^d_{r} \end{bmatrix} \in \mathbb{R}^{12}, \quad (17)$$

coherent with the cooperation mode is made available.

The operational task introduces the following equality constraint in the QP problem:

$$\boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{b}_\sigma, \quad (18)$$

with $\boldsymbol{b}_\sigma$ the desired task velocity.

Finally, let $\boldsymbol{q}_r(t) \in \mathbb{R}^n$ ($\dot{\boldsymbol{q}}_r(t)$) be the reference joint position (velocity) of the robot controller. We make the following assumption.

Assumption 1: The robot is provided with an inner motion control loop ensuring the tracking of a reference joint trajectory, i.e., $\boldsymbol{q}_r \approx \boldsymbol{q}$ ($\dot{\boldsymbol{q}}_r \approx \dot{\boldsymbol{q}}$).

The above assumption states that the robot is equipped with an inner motion control loop ensuring the tracking of reference joint trajectories generated by the devised safety strategy. Such an assumption increases the applicability of the approach since almost all commercial robots offer the possibility to track externally generated joint reference trajectories, while they usually do not provide a torque control interface. Based on the above assumption and the kinematic model in Eq. (5), the following virtual model is considered:

$$\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}_r = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}. \quad (19)$$

E. Human Safety Field

In [19], a safety field to account for the human-robot relative position is defined. The field takes into consideration the overall robotic structure as well as an arbitrary number n_h of significant human points, e.g., the endpoints of the links composing the human skeleton, as in [28]. To derive the overall safety field, a local index to assess the human safety in the case of single point P of the robot structure and single point P_h of the human operator is first analyzed. Let $\boldsymbol{p} \in \mathbb{R}^3$ be the position vector of the robot point P, $\boldsymbol{p}_h \in \mathbb{R}^3$ the position vector of the human point P_h, and $d = \|\boldsymbol{p} - \boldsymbol{p}_h\|$ the Euclidean distance between the points P and P_h. The following local safety index is defined:

$$f(\boldsymbol{p}, \boldsymbol{p}_h) = \chi(d), \quad (20)$$

where χ represents any non-negative continuous monotonically increasing Lipschitz function. Based on this local index, the overall safety field can be derived by integrating the local index in Eq. (20) along the robot links and then evaluating it for each human point. As in [19], we approximate each robot link l as a segment starting at $\boldsymbol{p}_l^0$ and ending at $\boldsymbol{p}_l^1$. Thus, let $s \in [0, 1]$ represent the curvilinear abscissa variable along the segment l and $\boldsymbol{p}_l^s \in \mathbb{R}^3$ the segment point corresponding to the abscissa s, i.e., computed as $\boldsymbol{p}_l^s = \boldsymbol{p}_l^0 + (\boldsymbol{p}_l^1 - \boldsymbol{p}_l^0)s$. The respective link safety index is defined as:

$$F_l(\boldsymbol{p}_l^0, \boldsymbol{p}_l^1, \boldsymbol{p}_o) = \int_0^1 f(\boldsymbol{p}_l^s, \boldsymbol{p}_h)ds. \quad (21)$$

At this point, let us consider all the links of the robotic platform and the j-th human point $\boldsymbol{p}_{h,j}$. The safety field taking into account the j-th human point and the entire robot structure is obtained as:

$$F^j(\boldsymbol{q}, \boldsymbol{p}_{h,j}) = \sum_{l=1}^{n} F_l(\boldsymbol{p}_l^0, \boldsymbol{p}_l^1, \boldsymbol{p}_{h,j}). \quad (22)$$

By considering all the n_h human points, the overall safety field is computed as

$$F = \frac{1}{n_h}\sum_{j=1}^{n_h} F^j(\boldsymbol{q}, \boldsymbol{p}_{h,j}). \quad (23)$$

The above formulation allows accounting for multiple human operators by simply adding the respective points in Eq. (23). Note that to compute the safety field, we assume the robot can detect and localize the human operator within the shared workspace. To this aim, several approaches existing in the literature can be exploited, such as [29] and [30].

F. Problem Formulation

We are now ready to present the main problem addressed in this work:

Problem 1: Let us consider a dual-arm robotic system performing a cooperative task encoded by a task function $\boldsymbol{\sigma}$ *as in Eq. (15), for which a nominal desired trajectory* $\boldsymbol{\sigma}^d(t)$ *is assigned by a Trajectory Generation module. Let us also consider a human operator that shares the same workspace as the robot, that is monitored through a Perception System,*

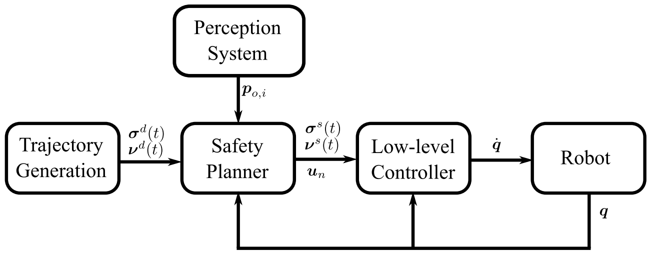

Fig. 2. Overall control architecture. A Trajectory Generation module generates a nominal trajectory that is then modified by a Safety Planner in function of the Perception System outputs in order to obtain a reference trajectory. Finally, the latter is given in input to a controller that computes the control input to be sent to the robot.

and that the level of human safety is assessed by the index $F(t)$ in Eq. (23), for which a time-varying minimum value $\underline{F}(t)$ is assigned. The objective is to design a Safety Planner capable of:

1) *scaling down the nominal trajectory $\boldsymbol{\sigma}^d(t)$;*
2) *modifying the nominal path of $\boldsymbol{\sigma}^d(t)$;*
3) *exploiting the redundancy of the system generating an internal joint motion;*

so as to generate a safe trajectory $\boldsymbol{\sigma}^s(t)$ as in Eq. (27) which maximizes the fulfilment of the safety condition:

$$F(t) \geq \underline{F}(t), \tag{24}$$

with $\underline{F}(t) > 0$ a time-varying threshold while taking into account the kinematic constraints of the robot.

The above formulation requires the definition of the safety time-varying threshold $\underline{F}(t)$. In [19], it is proven that $\underline{F}(t)$ can be chosen in such a way as to ensure a minimum safety distance between the human operator and the robot. However, this threshold could also be experimentally set by assessing, for instance, how stressed the human operator is and how much risk he/she perceives. To this aim, several human biological reactions could be exploited as reported, for instance, in [31].

IV. Proposed Solution

Figure 2 shows the proposed architecture for solving Problem 1. In particular, this is composed of four modules: (i) the Perception System; (ii) the Trajectory Generation module; (iii) the Safety Planner; and (iv) the Low-level Controller. The Perception System is responsible for detecting humans as further detailed in Section V, while the Trajectory Generation module defines the desired task trajectory $\boldsymbol{\sigma}^d(t)$ $(\boldsymbol{\nu}^d(t))$. Based on the outputs of these modules, the Safety Planner generates a safe trajectory, which is finally sent as a reference to the Low-level Controller module. In the following, we focus on the Safety Planner and Low-level Controller modules. Specifically, we first present the parameterization of the nominal trajectory and the variables for its modification. Then, we describe the low-level controller and the derivative of the safety field. Finally, we detail the proposed safety planner.

A. Trajectory Modification

Let us assume that the desired *nominal* trajectory $\boldsymbol{\sigma}^d(t)$ in Eq. (17) is parameterized with respect to a time parameter $c(t)$, i.e., $\boldsymbol{\sigma}^d(c(t))$, such that $c : [t_0, t_f] \in \mathbb{R} \rightarrow [t_0, t_f] \in \mathbb{R}$, with t_0 and t_f being the starting and final time instants of the nominal trajectory, respectively. In the following, we omit the time dependency of the variables if it is not strictly necessary. Note that by modifying the time parameter $c(t)$, the time law of the nominal trajectory can be modulated without modifying the nominal path. The time derivative of the scaled trajectory $\boldsymbol{\sigma}^d(c(t))$ can be expressed as:

$$\boldsymbol{\nu}^d(c(t), \dot{c}(t)) = \boldsymbol{\eta}(c(t))\,\dot{c}(t), \tag{25}$$

with

$$\boldsymbol{\eta}(c) = \begin{bmatrix} \dfrac{\partial \boldsymbol{\sigma}^d_{a_p}}{\partial c} \\ 2\boldsymbol{E}(\boldsymbol{o}_a)\dfrac{\partial \boldsymbol{\sigma}^d_{a_o}}{\partial c} \\ \dfrac{\partial \boldsymbol{\sigma}^d_{r_p}}{\partial c} \\ 2\boldsymbol{E}(\boldsymbol{o}_r)\dfrac{\partial \boldsymbol{\sigma}^d_{r_o}}{\partial c} \end{bmatrix},$$

$$\boldsymbol{E}(\boldsymbol{o}_y) = \begin{bmatrix} -o_{y,2} & -o_{y,1} & -o_{y,4} & o_{y,3} \\ -o_{y,3} & o_{y,4} & o_{y,1} & -o_{y,2} \\ -o_{y,4} & -o_{y,3} & o_{y,2} & o_{y,1} \end{bmatrix}.$$

Additionally, we introduce the time-varying term $\Delta\boldsymbol{\sigma} = \left[\Delta\boldsymbol{\sigma}^{\mathrm{T}}_{a_p}\ \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{a_o}\ \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{r_p}\ \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{r_o}\right]^{\mathrm{T}} \in \mathbb{R}^{14}$ that allows deviating from the nominal path such that a *safe* trajectory $\boldsymbol{\sigma}^s(t)$ $(\boldsymbol{\nu}^s(t))$ can be generated. Specifically, $\Delta\boldsymbol{\sigma}_{a_p}$ and $\Delta\boldsymbol{\sigma}_{a_o}$ represent the variations applied to the absolute position and orientation variables, respectively, while $\Delta\boldsymbol{\sigma}_{a_r}$ and $\Delta\boldsymbol{\sigma}_{a_r}$ represent the variations applied to the relative position and orientation variables, respectively. By denoting with $\Delta\boldsymbol{\nu} \in \mathbb{R}^{12}$ the vector of the deviation velocities defined as:

$$\Delta\boldsymbol{\nu} = \begin{bmatrix} \Delta\dot{\boldsymbol{\sigma}}^{\mathrm{T}}_{a_p} & \Delta\boldsymbol{\omega}^{\mathrm{T}}_{a} & \Delta\dot{\boldsymbol{\sigma}}^{\mathrm{T}}_{r_p} & \Delta\boldsymbol{\omega}^{\mathrm{T}}_{r} \end{bmatrix}^{\mathrm{T}}, \tag{26}$$

the safe trajectory is computed as follows:

$$\boldsymbol{\sigma}^s(t) = \begin{bmatrix} \boldsymbol{\sigma}^s_{a_p} \\ \boldsymbol{\sigma}^s_{a_o} \\ \boldsymbol{\sigma}^s_{r_p} \\ \boldsymbol{\sigma}^s_{r_o} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma}^d_{a_p} + \Delta\boldsymbol{\sigma}_{a_p} \\ \boldsymbol{\sigma}^d_{a_o} * \Delta\boldsymbol{\sigma}^{-1}_{a_o} \\ \boldsymbol{\sigma}^d_{r_p} + \Delta\boldsymbol{\sigma}_{r,p} \\ \boldsymbol{\sigma}^d_{r_o} * \Delta\boldsymbol{\sigma}^{-1}_{r_o} \end{bmatrix},$$

$$\boldsymbol{\nu}^s(t) = \begin{bmatrix} \dot{\boldsymbol{\sigma}}^s_{a_p} \\ \boldsymbol{\omega}^s_{a} \\ \dot{\boldsymbol{\sigma}}^s_{r_p} \\ \boldsymbol{\omega}^s_{r} \end{bmatrix} = \boldsymbol{\nu}^d(c(t), \dot{c}(t)) + \Delta\boldsymbol{\nu}, \tag{27}$$

which is the actual trajectory to be tracked by the robot.

Let $\Delta\boldsymbol{\epsilon} \in \mathbb{R}^{12}$ be the vector of position and orientation deviations defined as:

$$\Delta\boldsymbol{\epsilon} = \left[\Delta\boldsymbol{\sigma}^{\mathrm{T}}_{a_p}\ \boldsymbol{\rho}^{\mathrm{T}}_{\Delta\boldsymbol{\sigma}_{a_o}}\ \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{r_p}\ \boldsymbol{\rho}^{\mathrm{T}}_{\Delta\boldsymbol{\sigma}_{r_o}}\right]^{\mathrm{T}}, \tag{28}$$

where $\boldsymbol{\rho}_{\Delta\boldsymbol{\sigma}_{a_o}}$ and $\boldsymbol{\rho}_{\Delta\boldsymbol{\sigma}_{r_o}}$ are the vector parts of $\Delta\boldsymbol{\sigma}_{a_o}$ and $\Delta\boldsymbol{\sigma}_{r_o}$, respectively, the path modification obeys the following law:

$$\Delta\boldsymbol{\nu} = -\boldsymbol{T}\Delta\boldsymbol{\epsilon} + \boldsymbol{\zeta}, \quad \Delta\boldsymbol{\epsilon}(t_0) = \boldsymbol{0}_{12} \tag{29}$$

where $\boldsymbol{0}_m$ denotes the m dimensional zero vector, $\boldsymbol{T} = \mathrm{diag}\{\tau_{a_p}\boldsymbol{I}_3, \tau_{a_o}\boldsymbol{I}_3, \tau_{r_p}\boldsymbol{I}_3, \tau_{r_o}\boldsymbol{I}_3\}$, with $\tau_{(\cdot)} > 0$, and $\boldsymbol{\zeta} = \left[\boldsymbol{\zeta}^{\mathrm{T}}_{a_p}\ \boldsymbol{\zeta}^{\mathrm{T}}_{a_o}\ \boldsymbol{\zeta}^{\mathrm{T}}_{r_p}\ \boldsymbol{\zeta}^{\mathrm{T}}_{r_o}\right]^{\mathrm{T}} \in \mathbb{R}^{12}$, with $\boldsymbol{\zeta}_{(\cdot)} \in \mathbb{R}^3$ input vectors to be designed. Specifically, the positive gains $\tau_{(\cdot)}$ can be interpreted as stiffness parameters of a virtual spring pushing the safe trajectory towards the nominal one. It is worth noticing that if all the input vectors $\boldsymbol{\zeta}_{(\cdot)}$ are zero-vectors, i.e., $\boldsymbol{\zeta}_{(\cdot)} = \mathbf{0}_3$, the safe trajectory $\boldsymbol{\sigma}^s$ asymptotically converges to $\boldsymbol{\sigma}^d$. These inputs are designed as:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\psi}_{a_p} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \boldsymbol{\psi}_{a_o} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\psi}_{r_p} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\psi}_{r_o} \end{bmatrix} \begin{bmatrix} u_{\alpha_{a_p}} \\ u_{\alpha_{a_o}} \\ u_{\alpha_{r_p}} \\ u_{\alpha_{r_o}} \end{bmatrix} = \boldsymbol{\Psi}(t)\boldsymbol{u}_\alpha, \quad (30)$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{12\times 4}$ is any matrix that allows deviating the nominal trajectory along proper directions $\boldsymbol{\psi}_{(\cdot)} \in \mathbb{R}^3$ which can be chosen depending on the specific operation to accomplish, and the components of $\boldsymbol{u}_\alpha$ ($u_{\alpha_{a_p}}, u_{\alpha_{a_o}}, u_{\alpha_{r_p}}$ and $u_{\alpha_{r_o}}$) are scalar inputs that modulate the amount of deviation. A possible choice for matrix $\boldsymbol{\Psi}(t)$ will be presented in Sec. V.

It is worth noticing that the case $c(t) = t$, $\Delta\boldsymbol{\epsilon} = \mathbf{0}_{12}$ implies $\boldsymbol{\sigma}^d(c(t)) = \boldsymbol{\sigma}^d(t)$ and that $\dot{c} = 1$ implies $\boldsymbol{\nu}^s(t) = \boldsymbol{\nu}^s(c(t)) = \boldsymbol{\nu}^d(c(t))$, meaning that we are tracking the nominal reference velocity. The rationale behind this design choice is that, by modulating the scaling parameter $c(t)$, the nominal trajectory $\boldsymbol{\sigma}^d(t)$ can be online scaled while, by modifying $\Delta\boldsymbol{\sigma}$, the path is modified.

B. Low-Level Controller for Safe Trajectory Tracking

In this work, the low-level controller exploits the HQP formulation described in Sec. III-C and includes the tasks listed in Sec.III-D. More specifically, it considers three priority levels among the tasks: 1) joint position/velocity limits, 2) operational task, and 3) null-space joint velocity. According to Eq. (14), the first QP problem to solve in order to fulfil the joint limits tasks is:

$$\begin{aligned} \min_{\boldsymbol{w}_1, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_1^T \boldsymbol{Q}_{w,1}\boldsymbol{w}_1 \\ \text{s.t.} \quad & \underline{\boldsymbol{b}}_l \le \boldsymbol{J}_l(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_1 \le \bar{\boldsymbol{b}}_l, \end{aligned} \quad (31)$$

with $\boldsymbol{Q}_{w,1} \in \mathbb{R}^{n\times n}$ a positive definite weight matrix and $\boldsymbol{w}_1 \in \mathbb{R}^n$ a slack variable. The operational task is executed in the null space of the joint limit tasks by solving the second QP problem:

$$\begin{aligned} \min_{\boldsymbol{w}_2, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_2^T \boldsymbol{Q}_{w,2}\boldsymbol{w}_2 \\ \text{s.t.} \quad & \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_2 = \boldsymbol{b}_\sigma \\ & \underline{\boldsymbol{b}}_l \le \boldsymbol{J}_l(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \le \bar{\boldsymbol{b}}_l, \end{aligned} \quad (32)$$

with $\boldsymbol{Q}_{w,2} \in \mathbb{R}^{12\times 12}$ a positive definite weight matrix, $\boldsymbol{w}_2 \in \mathbb{R}^{12}$ a slack variable, and $\boldsymbol{b}_\sigma = \boldsymbol{\nu}^s + \boldsymbol{K}_\sigma\tilde{\boldsymbol{\sigma}}$, where $\boldsymbol{K}_\sigma$ is a positive-definite matrix of gains and:

$$\tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \boldsymbol{\sigma}^s_{a_p} - \boldsymbol{\sigma}_{a_p} \\ \boldsymbol{\rho}_{\tilde{\sigma}_{a_o}} \\ \boldsymbol{\sigma}^s_{r_p} - \boldsymbol{\sigma}_{r_p} \\ \boldsymbol{\rho}_{\tilde{\sigma}_{r_o}} \end{bmatrix}, \quad (33)$$

where $\boldsymbol{\rho}_{\tilde{\sigma}_{a_o}}$ is the vector part of the quaternion $\boldsymbol{\sigma}^s_{a_o} * \boldsymbol{\sigma}^{-1}_{a_o}$, $\boldsymbol{\rho}_{\tilde{\sigma}_{r_o}}$ is the vector part of the quaternion $\boldsymbol{\sigma}^s_{r_o} * \boldsymbol{\sigma}^{-1}_{r_o}$, and it allows to track the safe trajectory $\boldsymbol{\sigma}^s$ ($\boldsymbol{\nu}^s$) obtained as output of the Safety Planner. Finally, a null-space motion is considered by solving the following final QP problem:

$$\begin{aligned} \min_{\boldsymbol{w}_3, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\boldsymbol{w}_3^T \boldsymbol{Q}_{w,3}\boldsymbol{w}_3 \\ \text{s.t.} \quad & \dot{\boldsymbol{q}} + \boldsymbol{w}_3 = \dot{\boldsymbol{q}}_n \\ & \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_2^\star = \boldsymbol{b}_\sigma \\ & \underline{\boldsymbol{b}}_l \le \boldsymbol{J}_l(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \le \bar{\boldsymbol{b}}_l, \end{aligned} \quad (34)$$

where $\boldsymbol{Q}_{w,3} \in \mathbb{R}^{n\times n}$ is a positive definite weight matrix, $\boldsymbol{w}_3 \in \mathbb{R}^n$ is a slack variable, and $\dot{\boldsymbol{q}}_n$ is the joint velocity vector that refers to point 3) of Problem 1. This term will be computed by the Safety Planner to increase the safety field value without affecting the operational tasks. It is worth noticing that other control objectives might be introduced in the same control framework, such as requiring minimum distance from obstacles in the scene or defining virtual walls.

In case none of the kinematic constraints is active, i.e., the joint positions and velocities are within the specified limits, and the matrix $\boldsymbol{J}_\sigma$ is full-rank, the input $\dot{\boldsymbol{q}}$ can be computed in closed-form by resorting to a standard closed-loop inverse kinematic law [32]:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_\sigma^\dagger\left(\boldsymbol{\nu}^s + \boldsymbol{K}_\sigma\tilde{\boldsymbol{\sigma}}\right) + \boldsymbol{N}_\sigma\dot{\boldsymbol{q}}_n, \quad (35)$$

where $\boldsymbol{N}_\sigma = \boldsymbol{I} - \boldsymbol{J}_\sigma^\dagger\boldsymbol{J}_\sigma$ is a null space projector of $\boldsymbol{J}_\sigma$.

C. Time Derivative of the Safety Field F

To design the safety planner, we first derive the time derivative of the safety field F in Eq. (23). Specifically, in the remainder of the section, we prove it to be linear with respect to the derivative of the scaling parameter, i.e., $\dot{c}(t)$, the path modification input $\boldsymbol{u}_\alpha$, and the input $\dot{\boldsymbol{q}}_n$ in Eq. (35). This property is exploited to implement the scaling and path modification procedure presented in Section IV-D.

The time derivative $\dot{F}$ is computed from the derivative of f in Eq. (20). In particular, $\dot{f}$ can be computed as:

$$\dot{f} = \frac{\partial\chi(d^s_{i,l})}{\partial d^s_{i,l}}\dot{d}^s_{i,l}, \quad (36)$$

being $d^s_{i,l} = \|\boldsymbol{p}^s_l - \boldsymbol{p}_h\|$, i.e., the distance between the point $\boldsymbol{p}^s_l$ and the human operator $\boldsymbol{p}_h$. The time derivative of the distance is computed as follows:

$$\dot{d}^s_{i,l} = \frac{(\boldsymbol{p}^s_l - \boldsymbol{p}_h)^{\mathrm{T}}(\dot{\boldsymbol{p}}^s_l - \dot{\boldsymbol{p}}_h)}{d^s_{i,l}}, \quad (37)$$

with $d^s_{i,l} \neq 0$. At this point, the relationship between the linear velocity of the point $\boldsymbol{p}^s_l$ and the joint variables of the robot is taken into account, that is:

$$\dot{\boldsymbol{p}}^s_l = \boldsymbol{J}^s_l(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad (38)$$

where $\boldsymbol{J}^s_l(\boldsymbol{q}) \in \mathbb{R}^{3\times n}$ is the positional Jacobian matrix associated with the point $\boldsymbol{p}^s_l$, i.e., the first three rows of the Jacobian matrix that relates the joint velocities to the linear/angular velocity of the point $\boldsymbol{p}^s_l$. By considering the input in Eq. (35) and omitting the dependency of the Jacobian matrices from $\boldsymbol{q}$ for sake of readability, Eq. (38) can be rewritten as:

$$\dot{\boldsymbol{p}}^s_l = \boldsymbol{J}^s_l\boldsymbol{J}_\sigma^\dagger\left(\boldsymbol{\nu}^s + \boldsymbol{K}_\sigma\tilde{\boldsymbol{\sigma}}\right) + \boldsymbol{J}^s_l\boldsymbol{N}_\sigma\dot{\boldsymbol{q}}_n. \quad (39)$$

By folding Eqs. (25)-(30) in Eq. (39), it holds:

$$\dot{\boldsymbol{p}}_l^s = \boldsymbol{\gamma}_1 \dot{c} + \boldsymbol{\Gamma}_2 \boldsymbol{u}_\alpha + \boldsymbol{\Gamma}_3 \dot{\boldsymbol{q}}_n + \boldsymbol{\gamma}_4, \tag{40}$$

that is linear with respect to $\dot{c}$, $\boldsymbol{u}_\alpha$, and $\dot{\boldsymbol{q}}_n$ and where the coefficients $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_4 \in \mathbb{R}^3$ and $\boldsymbol{\Gamma}_2 \in \mathbb{R}^{3\times 4}$, $\boldsymbol{\Gamma}_3 \in \mathbb{R}^{3\times n}$ are defined as:

$$\begin{cases} \boldsymbol{\gamma}_1 = \boldsymbol{J}_l^s \boldsymbol{J}_\sigma^\dagger \boldsymbol{\eta} \\ \boldsymbol{\Gamma}_2 = \boldsymbol{J}_l^s \boldsymbol{J}_\sigma^\dagger \boldsymbol{\Psi} \\ \boldsymbol{\Gamma}_3 = \boldsymbol{J}_l^s \boldsymbol{N}_\sigma \\ \boldsymbol{\gamma}_4 = \boldsymbol{J}_l^s \boldsymbol{J}_\sigma^\dagger \left[\boldsymbol{K}_\sigma \tilde{\boldsymbol{\sigma}} - \boldsymbol{T} \Delta \boldsymbol{\epsilon} \right]. \end{cases}$$

By virtue of Eq. (36) and Eq. (40), the derivative of the safety index f can be rewritten as:

$$\dot{f} = \lambda_1 \dot{c} + \boldsymbol{\lambda}_2^{\mathrm{T}} \boldsymbol{u}_\alpha + \boldsymbol{\lambda}_3^{\mathrm{T}} \dot{\boldsymbol{q}}_n + \lambda_4, \tag{41}$$

where λ_1, $\lambda_4 \in \mathbb{R}$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^4$, $\boldsymbol{\lambda}_3 \in \mathbb{R}^3$ are obtained as:

$$\begin{cases} \lambda_1 = \dfrac{\partial \chi}{\partial d_{i,l}^s} \dfrac{(\boldsymbol{p}_l^s - \boldsymbol{p}_h)^{\mathrm{T}}}{d_{i,l}^s} \boldsymbol{\gamma}_1 \\ \boldsymbol{\lambda}_2^{\mathrm{T}} = \dfrac{\partial \chi}{\partial d_{i,l}^s} \dfrac{(\boldsymbol{p}_l^s - \boldsymbol{p}_h)^{\mathrm{T}}}{d_{i,l}^s} \boldsymbol{\Gamma}_2 \\ \boldsymbol{\lambda}_3^{\mathrm{T}} = \dfrac{\partial \chi}{\partial d_{i,l}^s} \dfrac{(\boldsymbol{p}_l^s - \boldsymbol{p}_h)^{\mathrm{T}}}{d_{i,l}^s} \boldsymbol{\Gamma}_3 \\ \lambda_4 = \dfrac{\partial \chi}{\partial d_{i,l}^s} \dfrac{(\boldsymbol{p}_l^s - \boldsymbol{p}_h)^{\mathrm{T}}}{d_{i,l}^s} \boldsymbol{\gamma}_4 - \dfrac{\partial \chi}{\partial d_{i,l}^s} \dfrac{(\boldsymbol{p}_l^s - \boldsymbol{p}_h)^{\mathrm{T}}}{d_{i,l}^s} \dot{\boldsymbol{p}}_h. \end{cases}$$

In light of Eqs. (21)-(23), the derivative of the local safety index is extended to the entire structure of the dual-arm robot and all the relevant human points as:

$$\dot{F}(t) = \mu_1(t)\,\dot{c}(t) + \boldsymbol{\mu}_2(t)^{\mathrm{T}} \boldsymbol{u}_\alpha + \boldsymbol{\mu}_3(t)^{\mathrm{T}} \dot{\boldsymbol{q}}_n + \mu_4(t) \tag{42}$$

with μ_1, $\mu_4 \in \mathbb{R}$ and $\boldsymbol{\mu}_2 \in \mathbb{R}^4$, $\boldsymbol{\mu}_3 \in \mathbb{R}^3$ defined as:

$$\begin{cases} \mu_1 = \dfrac{1}{n_h} \displaystyle\sum_{j=1}^{n_h} \sum_{l=1}^{n} \int_0^1 \lambda_1(\boldsymbol{p}_l^s, \boldsymbol{p}_{h,j}, \boldsymbol{q}_i, c)\, ds \\ \boldsymbol{\mu}_2 = \dfrac{1}{n_h} \displaystyle\sum_{j=1}^{n_h} \sum_{l=1}^{n} \int_0^1 \boldsymbol{\lambda}_2(\boldsymbol{p}_l^s, \boldsymbol{p}_{h,j}, \boldsymbol{q}_i, c)\, ds \\ \boldsymbol{\mu}_3 = \dfrac{1}{n_h} \displaystyle\sum_{j=1}^{n_h} \sum_{l=1}^{n} \int_0^1 \boldsymbol{\lambda}_3(\boldsymbol{p}_l^s, \boldsymbol{p}_{h,j}, \boldsymbol{q}_i, c)\, ds \\ \mu_4 = \dfrac{1}{n_h} \displaystyle\sum_{j=1}^{n_h} \sum_{l=1}^{n} \int_0^1 \lambda_4(\boldsymbol{p}_l^s, \boldsymbol{p}_{h,j}, \dot{\boldsymbol{p}}_{h,j}, \boldsymbol{q}_i, c)\, ds \end{cases} \tag{43}$$

that, as $\dot{f}$ in Eq. (41), is linear with respect to $\dot{c}$, $\boldsymbol{u}_\alpha$, and $\dot{\boldsymbol{q}}_n$.

D. Safety Planner Design

We are ready to present the proposed solution to Problem 1. By setting $\dot{c} = u_c$ and $\dot{\boldsymbol{q}}_n = \boldsymbol{u}_n$, the overall system dynamics is:

$$\begin{bmatrix} \dot{c} \\ \Delta \dot{\boldsymbol{\nu}} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} 0 \\ -\boldsymbol{T}\Delta\boldsymbol{\epsilon} \\ \mu_4 \end{bmatrix} + \begin{bmatrix} 1 & \boldsymbol{0}_4^{\mathrm{T}} & \boldsymbol{0}_n^{\mathrm{T}} \\ \boldsymbol{0}_{12} & \boldsymbol{\Psi} & \boldsymbol{O}_{12\times n} \\ \mu_1 & \boldsymbol{\mu}_2^{\mathrm{T}} & \boldsymbol{\mu}_3^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} u_c \\ \boldsymbol{u}_\alpha \\ \boldsymbol{u}_n \end{bmatrix}, \tag{44}$$

that, by defining the overall state variable $\boldsymbol{\xi} = \begin{bmatrix} c & \Delta\boldsymbol{\sigma}^{\mathrm{T}} & F \end{bmatrix}^{\mathrm{T}}$, can be rewritten as:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(t, \boldsymbol{\xi}) + \boldsymbol{g}(t, \boldsymbol{\xi}) \begin{bmatrix} u_c \\ \boldsymbol{u}_\alpha \\ \boldsymbol{u}_n \end{bmatrix} = \boldsymbol{f}(t, \boldsymbol{\xi}) + \boldsymbol{g}(t, \boldsymbol{\xi}) \boldsymbol{u}_\xi, \tag{45}$$

which is in the same form as in Eq. (1).

Let us recall the effect of each component of the overall input $\boldsymbol{u}_\xi \in \mathbb{R}^{n+5}$ vector:

- u_c allows scaling down the nominal trajectory $\boldsymbol{\sigma}^d$;
- the components of $\boldsymbol{u}_\alpha = \begin{bmatrix} u_{\alpha_{ap}} & u_{\alpha_{ao}} & u_{\alpha_{rp}} & u_{\alpha_{ro}} \end{bmatrix}^{\mathrm{T}}$ allow modulating the amount of deviation from $\boldsymbol{\sigma}_{ap}^d$, $\boldsymbol{\sigma}_{ao}^d$, $\boldsymbol{\sigma}_{rp}^d$ and $\boldsymbol{\sigma}_{ro}^d$, respectively;
- $\boldsymbol{u}_n$ allows generating an internal joint motion that increases the safety field value without affecting the operational task value $\boldsymbol{\sigma}$.

The objective is to find an input $\boldsymbol{u}_\xi$ such that the safety field value is above a certain lower threshold $\underline{F}$ while satisfying other input constraints on u_c and $\boldsymbol{u}_\alpha$, which are detailed in the following and included in a dedicated QP problem to be solved. In detail, by expressing a CBF as:

$$\underline{h}_f = F - \underline{F}, \tag{46}$$

the constraint on the decision variables that implements Eq. (24) can be formulated as:

$$\boldsymbol{J}_f \boldsymbol{u}_\xi \geq -k_f \underline{h}_f - \mu_4 \tag{47}$$

where, in virtue of Eq. (42), it holds:

$$\boldsymbol{J}_f = \begin{bmatrix} \mu_1 & \boldsymbol{\mu}_2 & \boldsymbol{\mu}_3 \end{bmatrix}, \tag{48}$$

and $k_f > 0$ is a scalar gain.

At this point, the Safety Planner has to solve the following constrained optimization problem:

$$\min_{\boldsymbol{u}_\xi} \quad \frac{1}{2}(\boldsymbol{u}_\xi - \boldsymbol{u}_{\xi,n})^{\mathrm{T}} \boldsymbol{Q}_\xi (\boldsymbol{u}_\xi - \boldsymbol{u}_{\xi,n}) + \frac{1}{2} q_w w_f^2 \tag{49}$$

$$s.t. \quad 0 \leq u_c \leq 1, \tag{50}$$

$$\underline{\boldsymbol{u}}_\alpha \leq \boldsymbol{u}_\alpha \leq \bar{\boldsymbol{u}}_\alpha, \tag{51}$$

$$\boldsymbol{J}_f \boldsymbol{u}_\xi + w_f \geq -k_f \underline{h}_f - \mu_4. \tag{52}$$

Each constraint is detailed in the following:

- Eq. (50) constrains the time scaling parameter $u_c = \dot{c}$ to belong to the interval $[0, 1]$. Specifically, the condition $\dot{c} \geq 0$ results in the time parameter c never decreasing, thus ensuring that the trajectory is not travelled backwards; the condition $\dot{c} \leq 1$ guarantees that the trajectory is not travelled with a velocity higher than the nominal one.
- Eq. (51) constraints the minimum ($\underline{\boldsymbol{u}}_\alpha$) and maximum ($\bar{\boldsymbol{u}}_\alpha$) deviation velocity. Such constraints might be required in practice to avoid abrupt deviation from the nominal trajectory. Moreover, if a constraint on the maximum path deviation $\Delta\boldsymbol{\sigma}$ is needed by the application, it can be considered as well by our framework by resorting to the CBF approach;
- Eq. (52) handles the human-robot safety according to Problem 1. The slack variable w_f is needed to relax the

constraint when the input $\boldsymbol{u}_\alpha$ reaches the limits imposed by the constraints in Eq. (51).

Concerning the objective function in Eq. (49), it aims at finding inputs $\boldsymbol{u}_\xi = \left[u_c \ \boldsymbol{u}_\alpha^{\mathrm{T}} \ \boldsymbol{u}_n^{\mathrm{T}}\right]^{\mathrm{T}}$ closest to $\boldsymbol{u}_\xi^d = \left[u_c \ \boldsymbol{u}_\alpha^{d\,\mathrm{T}} \ \boldsymbol{u}_n^{d\,\mathrm{T}}\right]^{\mathrm{T}}$, where:

$$\begin{cases} u_c^d = 1, \\ \boldsymbol{u}_\alpha^d = \boldsymbol{0}_4, \\ \boldsymbol{u}_n^d = k_n \dfrac{\partial F}{\partial \boldsymbol{q}}, \end{cases} \tag{53}$$

with $k_n > 0$. Having $u_c = u_c^d$ implies $\dot{c} = 1$, i.e., the cooperative task executed at nominal speed (see Sec. III-F), having $\boldsymbol{u}_\alpha = \boldsymbol{u}_\alpha^d$ implies $\Delta\boldsymbol{\epsilon} = \boldsymbol{0}_{12}$, i.e., the nominal path is followed without modifications, and having $\boldsymbol{u}_n$ as close as possible to $\boldsymbol{u}_n^d$ proportional to the gradient of the safety field allows exploiting the possible redundancy of the system to increase the safety.

Concerning $q_w \in \mathbb{R}$, it is a positive scalar weight related to the scalar slack variable w_f, while $\boldsymbol{Q}_\xi \in \mathbb{R}^{(n+5)\times(n+5)}$ is a positive-definite weight matrix that can be tuned to modulate the relative usage of the three virtual inputs. For example, a higher value on the component corresponding to u_c means that the trajectory deviation and the internal motion have to be preferred with respect to the trajectory scaling.

After having solved the QP problem in Eq. (49), the Safety Planner makes use of its solutions in Eq. (27) to build the reference trajectory $\boldsymbol{\sigma}^s(t)(\boldsymbol{\nu}^s(t))$. Algorithm 1 describes the algorithmic steps of the proposed Safety Planner. In detail, at first, the Safety field value F is computed by resorting to Eq (23); then, the QP problem (49) is solved having in input the computed F and the nominal trajectory; the solutions of the QP problem are then exploited to compute the reference trajectory following these steps:

- u_c is integrated over time in order to obtain c,
- c and $\boldsymbol{u}_\alpha$ are used in Eq. (29) to compute the deviation velocity $\Delta\boldsymbol{\nu}$, which is then integrated over time to obtain $\Delta\boldsymbol{\sigma}$,
- finally, $\Delta\boldsymbol{\sigma}$, c, u_c are used in Eq. (27) to compute the reference trajectory $\boldsymbol{\sigma}^s(\boldsymbol{\nu}^s)$ that is sent to the specific employed controller, together with the vector $\boldsymbol{u}_n = \dot{\boldsymbol{q}}_n$ in Eq. (34), as described in Section IV-B.

At this point, the output of the Safety Planner is included in the HQP low-level controller described in Sec. IV-B in order to take into account also the higher-priority tasks in the hierarchy (joint position and velocity limits in this case). In detail, the reference trajectory tracking is performed by properly using $\boldsymbol{\nu}^s$ and $\boldsymbol{\sigma}^s$ in the expression of $\boldsymbol{b}_\sigma$ in Eq. (32) and by specifying the internal motion as $\dot{\boldsymbol{q}}_n$ in Eq. (34). It is worth noticing that in this final step, it is mandatory to adopt slack variables, as the null space of the higher-priority tasks is possibly different with respect to the one used inside the Safety Planner for computing $\dot{\boldsymbol{q}}_n$, since other constraints relative to higher-priority tasks can be activated.

Algorithm 1 Safety Planner Algorithm

```
Data:
  • Nominal trajectory $\boldsymbol{\sigma}^d(\boldsymbol{\nu}^d)$
  • Joint position vector $\boldsymbol{q}$
  • Human point positions $\boldsymbol{p}_{h,1}, \cdots, \boldsymbol{p}_{h,n_h}$
Result:
  • Safe trajectory $\boldsymbol{\sigma}^s(\boldsymbol{\nu}^s)$
  • Null-space velocity vector $\boldsymbol{u}_n$
begin
    $F =$ SafetyField $(\boldsymbol{q}, \boldsymbol{p}_{h,1}, \cdots, \boldsymbol{p}_{h,n_h}) \rightarrow$ Eq. (23)
    $\begin{bmatrix} u_c \\ \boldsymbol{u}_\alpha \\ \boldsymbol{u}_n \end{bmatrix} =$ SolveQP $(\bar{F}, \boldsymbol{\sigma}^d, \boldsymbol{\nu}^d) \rightarrow$ Eq. (49)
    $c =$ Integration $(u_c)$
    $\Delta\boldsymbol{\nu} =$ DeviationVelocity $(\boldsymbol{u}_\alpha, c, u_c) \rightarrow$ Eq. (29)
    $\Delta\boldsymbol{\sigma} =$ Integration $(\Delta\boldsymbol{\nu})$
    $\begin{bmatrix} \boldsymbol{\sigma}^s \\ \boldsymbol{\nu}^s \end{bmatrix} =$ SafeTrajectory $(\Delta\boldsymbol{\sigma}, \Delta\boldsymbol{\nu}, c) \rightarrow$ Eq. (27)
    return $\boldsymbol{\sigma}^s, \boldsymbol{\nu}^s, \boldsymbol{u}_n$
end
```

V. Approach Validation

A. Experiments

In this section, we show the experimental results obtained validating the proposed architecture. The robot taken into consideration is the one shown in Figure 1, which has been designed for the EU-funded project CANOPIES,[1] focusing on human-multi-robot collaboration paradigm for precision agriculture settings. The robotic system consists of a tracked mobile base ($n_b = 7$), a 2-DOFs torso ($n_t = 2$), and two 7-DOFs manipulators ($n_m = 7$). The torso has a rotational joint and a prismatic joint that allows its height to be changed. Additionally, as the robot is supposed to operate within table-grape vineyards, the left-end effector includes a mechanism with scissors that can be used for cutting the bunch peduncle, while the right-end effector is a standard gripper that can be used to grasp and hold the bunch to harvest. Finally, the robot incorporates a head equipped with a RealSense D435 RGB-D sensor, positioned internally to serve as the robot's "eyes". For the sake of clarity, despite our framework being able to handle all DOFs of the system, we do not move the base in the real experiments since the low-level controller of the base and the arms are not fully integrated yet. Furthermore, for a similar reason, we do use the two torso DOFs, as the robot manufacturer is still developing their low-level controllers.

Table I reports the robot kinematic constraints in terms of maximum/minimum joint positions and velocities. The control software is designed and developed under the ROS Noetic Ninjemys[2] framework, while the QP problems described above

[1]www.canopies-project.eu
[2]http://wiki.ros.org/noetic

TABLE I
CANOPIES ROBOT JOINT POSITION LIMITS (IN [RAD]) AND JOINT VELOCITY LIMITS ([RAD/S])

LEFT ARM
$\bar{\boldsymbol{q}}_L = [1.47,\ 1.50,\ 2.44,\ 1.51,\ 2.45,\ 2.06,\ 2.41]^{\mathrm{T}}$
$\underline{\boldsymbol{q}}_L = [-0.75,\ -1.50,\ -2.44,\ -2.32,\ -2.45,\ -2.06,\ -2.41]^{\mathrm{T}}$
$\bar{\dot{\boldsymbol{q}}}_L = [1.95,\ 1.95,\ 2.35,\ 2.35,\ 1.95,\ 1.76,\ 1.76]^{\mathrm{T}}$
$\underline{\dot{\boldsymbol{q}}}_L = [-1.95,\ -1.95,\ -2.35,\ -2.35,\ -1.95,\ -1.76,\ -1.76]^{\mathrm{T}}$
RIGHT ARM
$\bar{\boldsymbol{q}}_R = [1.47,\ 1.50,\ 2.44,\ 1.51,\ 2.45,\ 2.06,\ 2.41]^{\mathrm{T}}$
$\underline{\boldsymbol{q}}_R = [-0.75,\ -1.50,\ -2.44,\ -2.32,\ -2.45,\ -2.06,\ -2.41]^{\mathrm{T}}$
$\bar{\dot{\boldsymbol{q}}}_R = [1.95,\ 1.95,\ 2.35,\ 2.35,\ 1.95,\ 1.76,\ 1.76]^{\mathrm{T}}$
$\underline{\dot{\boldsymbol{q}}}_R = [-1.95,\ -1.95,\ -2.35,\ -2.35,\ -1.95,\ -1.76,\ -1.76]^{\mathrm{T}}$

are solved by using the GUROBI solver software[3] freely available under Academic Licence. A commercial laptop with Intel Core i7 − 8750H CPU at 2.20GHz and 16GB RAM is used for all validation results. Average execution time of $\approx 1e^{-3}$ s (over 1000 tests with different system states) is obtained by considering the pipeline composed of the safety planner and low-level controller, largely meeting the real-time computation requirement (that is $1e^{-2}$ s for our robot). Note that, whenever required to improve the computational burden, more advanced optimization frameworks, such as in [33], could be employed with respect to the HQP, but they are out of the scope of this work. Moreover, the maximum RAM occupancy recorded is equal to about 100 MB, which amply adheres to the space constraints commonly encountered in commercial PCs and robot-embedded computers.

In the following, we show experimental results obtained both within an indoor laboratory environment, where the robot performs a predefined periodic movement in a cooperative and symmetric manner with the two end effectors, and within an actual vineyard, where the robot performs a harvesting operation using a single end effector. In all the experiments, the robot operates alongside a human operator in a shared workspace, each carrying out distinct tasks. For instance, in our precision agriculture setting, the human might either closely inspect the quality of harvested fruits or perform parallel harvesting activities nearby while the robot is harvesting grape bunches. Therefore, ensuring safety is a critical aspect in this co-existence scenario, particularly because the end effectors are equipped with scissors. The inputs for the path deviation in terms of orientations are set to zero ($\boldsymbol{\psi}_{a_o} = \boldsymbol{0}_3$ and $\boldsymbol{\psi}_{r_o} = \boldsymbol{0}_3$ in Eq. (30)), meaning that the Safety Planner will not modify the orientation, while the deviations in terms of positions will be specified in the descriptions of the specific experiments.

Regarding the human points position estimation by the Perception System in Figure 2, we make use of OpenPose [34], which is a popular software that allows tracking the human skeleton in the image plane. These 2D points are then projected in Cartesian 3D space by exploiting the depth information of the RealSense D435 sensor that is mounted

[3]https://www.gurobi.com

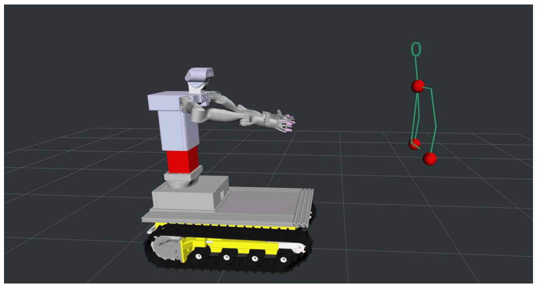
Fig. 3. Representation of the robot and the human skeleton in the Rviz environment. The red spheres represent the human points taken into consideration for the safety field value computation.

TABLE II
CONTROL GAINS AND SAFETY PLANNER PARAMETERS

CONTROL		SAFETY PLANNER	
Parameter	Equation	Parameter	Equation
$\phi_i = 2$	Eq. (3)	$\tau_{(\cdot)} = 2$	Eq. (29)
$\boldsymbol{K}_\sigma = \mathrm{diag}\{50\ \boldsymbol{I}_3,\ 20\ \boldsymbol{I}_3\}$	Eq. (32)	$k_f = 10$	Eq. (47)
		$k_n = 1$	Eq. (53)
		$\underline{u}_{\alpha_{(\cdot)}} = -0.4$	Eq. (51)
		$\bar{u}_{\alpha_{(\cdot)}} = 0.4$	Eq. (51)
		$\underline{F} = 0.75$	Eq. (46)

inside the head of the robot [35]. Then, regarding the safety field value computation, we consider three relevant points representing the chest of the human and left/right wrists, and the function χ in Eq. (20) is set as d. Figure 3 shows the robot and the human skeleton recognized by OpenPose in the Rviz environment, highlighting with red spheres the human points taken into consideration for the experiments presented in this section. Finally, the gains employed in the HQP controller and the Safety Planner parameters used in the described experiments are reported in Table II, where the variables $\underline{u}_{\alpha_{(\cdot)}}$ and $\bar{u}_{\alpha_{(\cdot)}}$ are used to denote any generic element of the corresponding vectors $\underline{\boldsymbol{u}}_\alpha$ and $\bar{\boldsymbol{u}}_\alpha$.

All the experiments described in this Section are included in the video provided as supplementary material.

1) Laboratory Experiments: Here, we show two indoor experiments in which the two end effectors perform a cooperative motion to track a nominal segment with the absolute frame. In both of them, the motion is realized in a symmetrical manner, thus setting $\beta = 0.5$ and $\gamma = 1$ in Eq. (15). In this configuration, the task function $\boldsymbol{\sigma}_a$ represents the absolute pose, while $\boldsymbol{\sigma}_r$ represents the relative pose. The difference between the two experiments lies in the relative weights set for the virtual inputs u_c and $\boldsymbol{u}_\alpha$, to highlight both the effects in the nominal trajectory scaling and modification separately. The absolute frame orientation is kept at the initial value, while the relative position between the two end effectors is required to reach:

$$\boldsymbol{\sigma}^d_{r_p} = \begin{bmatrix} 0 & -0.30 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{54}$$

keeping the relative orientation constant at its initial value. In both experiments, a human operator enters the scene and gets close to the robot while it is following the nominal trajectory. This motion leads the safety field value to decrease and results in the Safety Planner deviating and scaling down the trajectory. In particular, in the experiments, the input $\boldsymbol{\zeta}_{r_p}$ is set to $\mathbf{0}_3$, meaning that the Safety Planner does not modify the relative position between the two end effectors, while the input related to the absolute position deviation is set to:

$$\boldsymbol{\zeta}_{a_p} = \hat{\boldsymbol{n}}_o^a \, u_{\alpha_{a_p}} = \frac{\boldsymbol{p}_h - \boldsymbol{\sigma}_{a_p}}{||\boldsymbol{p}_h - \boldsymbol{\sigma}_{a_p}||} u_{\alpha_{a_p}},$$

where $\boldsymbol{p}_h$ is the average position of the considered human points and $\hat{\boldsymbol{n}}_o^a$ is the unit vector connecting this average point and the absolute frame position.

In the first experiment, we adjust the value in the $\boldsymbol{Q}_\xi$ matrix in Eq. (49) to deviate more from the nominal trajectory rather than scale it down to keep the safety field value above the minimum threshold. This is obtained by lowering the weight components in $\boldsymbol{Q}_\xi$ related to the virtual input $u_{\alpha_{a_p}}$ with respect to the ones related to u_c.

Figure 4 shows the obtained results. Figure 4.a) reports the evolution of the safety field value over time (solid blue line) during the experiment, compared to the imposed minimum threshold $\underline{F}$ (horizontal red-dashed line). In the first 25 seconds of the experiment, the human operator stays far from the robot while it performs twice the requested movement; then, he extends the right arm toward the robot, and the safety field value decreases. When the robot tries reaching the waypoint closer to the human for the third time, the safety field value gets close to the imposed threshold, triggering the activation of the constraints in the QP problem in Eq. (49). The Safety Planner computes a deviation from the nominal trajectory that keeps the safety field to a value equal to or greater than the minimum threshold. This is especially evident from Figure 4.b) and Figure 4.c). Specifically, the former shows both the nominal (in blue) and safe (in red) trajectories, while the latter shows the computed virtual input $u_{\alpha_{a_p}}$ (bottom part) together with the corresponding $\Delta\boldsymbol{\sigma}$ (top part) that quantifies the deviation from the nominal path. At the same time, Figure 4.d) shows the evolution of the virtual input u_c (bottom part) and the corresponding c value (top part), from which it is possible to understand that the scaling input, in this case, is barely used by the Safety Planner, given the relative weight that we have set to that virtual input. Finally, Figure 4.e) and Figure 4.f) show the normalized joint positions and velocities and the corresponding normalized minimum and maximum thresholds (red-dashed lines), from which it is possible to notice that the high-priority tasks in the HQP controller are successfully performed. After about 10s, the human operator retracts his arm, allowing the robot to reach the desired waypoint. Then, the same operation is repeated one more time, and the human operator extends his left arm, triggering once again the activation of the constraint on the safety field value in the QP problem. After about 10s, the human operator retracts his arm, and the robot keeps following the desired segments freely. During the entire experiment (the same applies to the following ones), the Safety Planner also computes the virtual input $\dot{\boldsymbol{q}}_n$ in Eqs. (34) and (45) to generate internal motions aimed at increasing the safety field value.

In the second experiment, we choose instead to scale down the trajectory rather than deviate from the nominal path. This is obtained by increasing the weight components in $\boldsymbol{Q}_\xi$ related to the virtual input $u_{\alpha_{a_p}}$ with respect to the ones related to u_c. The desired waypoints for the absolute position and the human operator's movements are the same as in the previous experiments, easing the comparison between the two robot's behaviors. The results are reported in Figure 5. Even in this case, looking at Figure 5.a) it is possible to observe that the safety field value never goes below the imposed minimum threshold. In this case, differently from the previous experiment, the Safety Planner makes much more use of the virtual input u_c, as can be seen from Figure 5.d) where $\dot{c}$ decreases its value until completely stopping the nominal trajectory ($\dot{c} = 0$ at $t = 27$s and $t = 47$s). Figure 5.b) and Figure 5.c) show that, in this case, the deviation from the nominal path is smaller compared to the previous experiment, given the chosen relative weights, while Figure 5.e) and Figure 5.f) show that the joint kinematic constraints are still respected.

2) Outdoor Environment: In the following, we report the results obtained during one of the harvesting experiments conducted during one of the experimental campaigns of the CANOPIES project, held in Aprilia (Italy) in February 2023. In particular, we provide results relative to harvesting operations performed with a single arm. It is worth noticing that harvesting operations have also been demonstrated in dual-arm mode, as can be seen in the accompanying video, but it is not reported here for the sake of brevity. The position of the peduncle to cut has been obtained in real-time by the perception software developed by one of the partners of the project [36], which recognizes the bunches to be harvested and gives in output the 3D coordinates of the point to be cut. The RGB-D camera mounted on the head of the robot is used for this purpose.

In this experiment, the parameters in Eq. (15) are set as $\beta = 1$ and $\gamma = 0$ to choose the right end effector to perform the harvesting operation. In this configuration, indeed, the task variable $\boldsymbol{\sigma}_r$ represents the right end effector pose. The corresponding path deviation input $\boldsymbol{\zeta}_{r_p}$ is set as:

$$\boldsymbol{\zeta}_{r_p} = \hat{\boldsymbol{n}}_o^r \, u_{\alpha_{r_p}} = \frac{\boldsymbol{p}_h - \boldsymbol{\sigma}_{r_p}}{||\boldsymbol{p}_h - \boldsymbol{\sigma}_{r_p}||} u_{\alpha_{r_p}},$$

where $\boldsymbol{p}_h$ is the average position of the considered human points and $\hat{\boldsymbol{n}}_o^r$ is the unit vector connecting this average point and the right end effector position.

The Trajectory Generation module outputs the nominal right end effector position and orientation trajectories obtained by connecting a series of proper waypoints through trajectories characterized by trapezoidal velocity profiles. The waypoints list is the following:

- a pre-grasp configuration placed at a predefined distance from the point of the peduncle to be grasped;
- a grasp configuration that allows to cut the peduncle and grasp the bunch;
- a pre-release configuration at a certain distance from the box placed on the mobile base;
- a release configuration over the box.

In the experiment described below, a human operator shares the workspace with the robot while it performs the harvesting operation. Initially, the robot follows the nominal trajectory until the human operator gets closer, triggering the activation

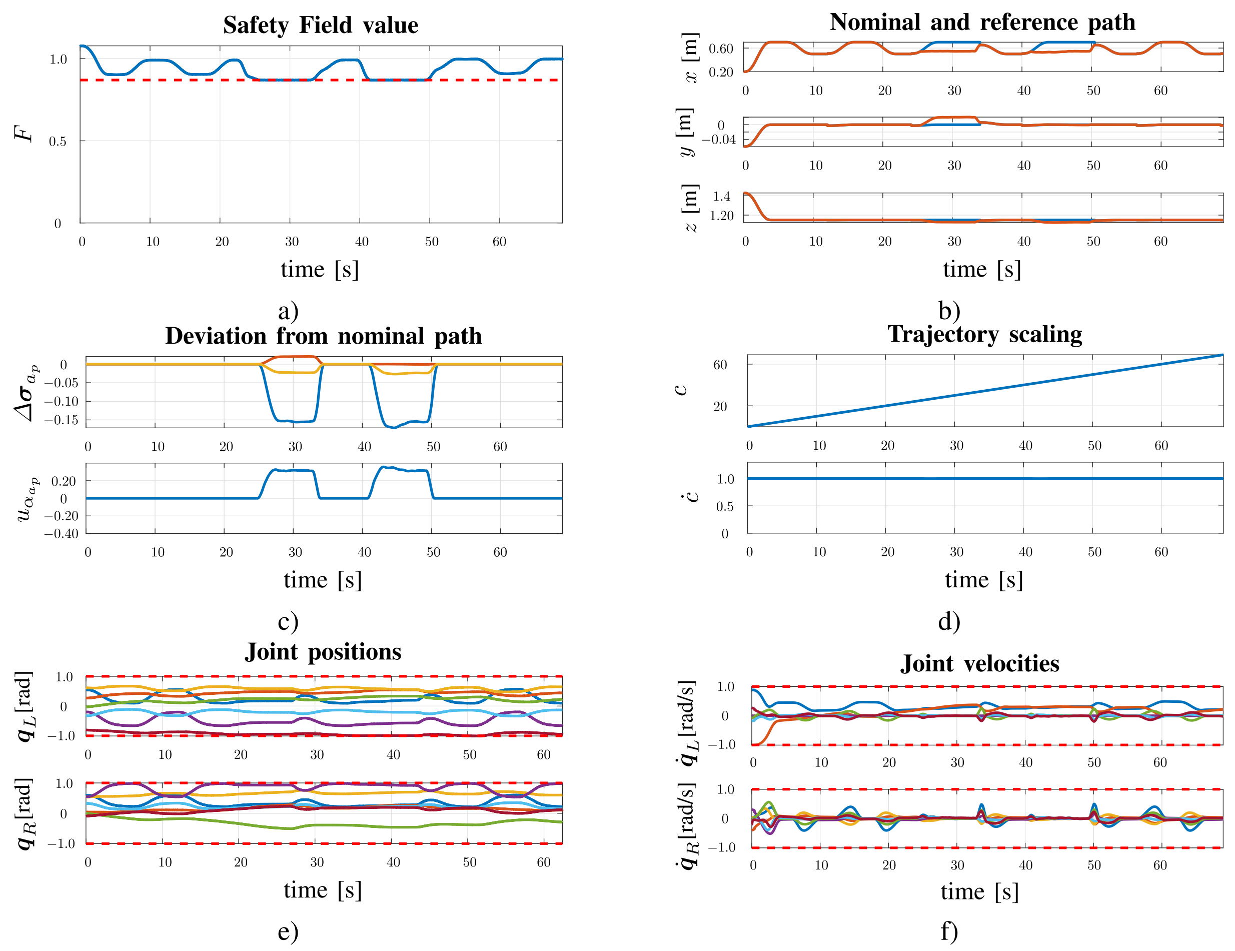

Fig. 4. First experiment, u_c weighted more than $u_{\alpha_{ap}}$. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line). The safety field value stays above the minimum threshold during the experiment; b) Nominal (blue) and safe (red) trajectories for the absolute frame. The proposed Safety Planner changes the nominal path in order to keep the safety value above the imposed minimum threshold; c) Deviation from the nominal path and virtual input u_α. The value of u_α resulting from the QP problem in Eq (49) is used to compute the deviation from the nominal trajectory. d) Parameter c and virtual input u_c. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds. The value of u_c resulting from the QP problem in Eq (49) is used to scale down the nominal trajectory. In this case, this virtual input is weighted more than u_α, and consequently, the nominal trajectory is not scaled down much.

of the constraint in Eq. (52). The relative weight between the two inputs $u_{\alpha_{rp}}$ and u_c is chosen to make use of both of them, and the results are reported in Figure 6.

Figure 6.a) shows the evolution of the safety field value over time. The human operator gets close to the robot twice during the experiment. Specifically, around $t = 10$s, the human approaches the robot and the safety field value reaches the proximity of the minimum threshold. This results mainly in a deviation from the nominal trajectory (Figures 6.b)-6.d)) for about 10s. Then, the human operator steps back, and the robot resumes the tracking of the nominal trajectory without modifications. Around $t = 50$s, the operator gets close again to the robot, leading to a further phase, with a duration of about 20s, where the path is modified. In this case, the virtual input $u_{\alpha_{rp}}$ reaches its maximum value, and the other virtual input u_c is used to scale down the trajectory. It is worth noticing that in this phase, the safety field value exceeds the minimum threshold because both the virtual inputs have reached their respective maximum and minimum values. In this condition, the algorithm maximizes the safety field value, but there is no guarantee of respecting the minimum threshold. It is worth noticing that, even in this situation, the safety of the human operator is still preserved as $\dot{c} = 0$ implies that the Safety Planner stops the nominal trajectory. Finally, at $t = 69$s, the human operator steps back, allowing the robot to finish the harvesting operation. Figures 6.e) and 6.f) demonstrate that the joint limits are respected during the entire experiment.

B. Comparison With Other Algorithms

In this section, we compare our algorithm with two baseline methods: an evasive motion baseline and an emergency stop baseline. In the following, we denote with A1 our proposed approach, A2 the evasive motion baseline and A3 the emergency stop method.

Algorithm A2 is inspired by the work in [37], where the control input is modified based on the *danger field*. Here, we use the safety field, following the same policy:

$$\dot{\boldsymbol{q}} = m\dot{\boldsymbol{q}}_\sigma + \left[\boldsymbol{I} - m\boldsymbol{J}^\dagger\boldsymbol{J}\right]\dot{\boldsymbol{q}}_0 \,, \tag{55}$$

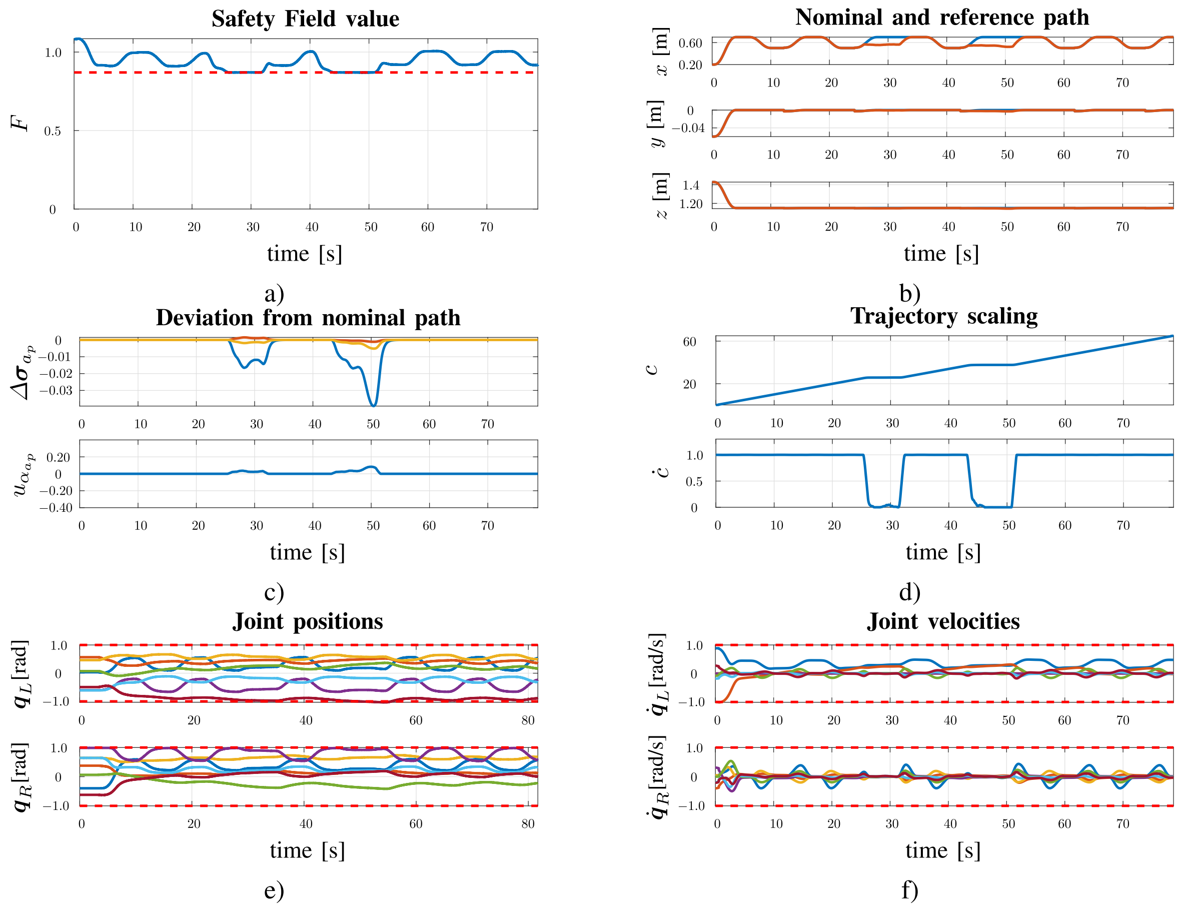

Fig. 5. Second experiment, u_c weighted less than $u_{\alpha_{a_p}}$. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line). The safety field value stays above the minimum threshold during the experiment; b) Nominal (blue) and safe (red) trajectories for the absolute frame. The proposed Safety Planner changes the nominal path in order to keep the safety value above the imposed minimum threshold; c) Deviation from the nominal path and virtual input u_α. The value of u_α is smaller with respect to the previous experiment given the chosen relative weights. d) Parameter c and virtual input u_c. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds. The value of u_c resulting from the QP problem in Eq (49) is used to scale down the nominal trajectory. In this case, this virtual input is weighted less than u_α. Thus, the trajectory is scaled down until it is completely stopped.

where:

$$m = \begin{cases} 1 & \text{if} \quad F \geq (1+\epsilon)\underline{F} \\ 0 & \text{if} \quad F \leq (1-\epsilon)\underline{F} \\ \dfrac{1}{2} + \dfrac{1}{2}\sin\dfrac{\pi(F-\underline{F})}{2\epsilon\underline{F}} & \text{otherwise,} \end{cases} \tag{56}$$

with $\dot{\boldsymbol{q}}_\sigma$ is the joint velocity input for achieving the operational task, which can be computed by resorting to the CLIK algorithm as $\dot{\boldsymbol{q}}_\sigma = \boldsymbol{J}_\sigma^\dagger(\boldsymbol{v}^s + \boldsymbol{K}_\sigma\tilde{\boldsymbol{\sigma}})$, $\dot{\boldsymbol{q}}_0$ the gradient of the safety field and $\epsilon < 1$ a design constant that enables a smooth control command.

Algorithm A3 is a simple emergency stop algorithm in which the desired trajectory for the robot is stopped when the value of the safety field goes below a certain threshold $\underline{F}$. In the formulation devised in Section IV, this is achievable by changing the value of $\boldsymbol{u}_c$ as follows:

$$\boldsymbol{u}_c = \begin{cases} 0 & \text{if } F \leq \underline{F} \\ 1 & \text{if } F > \underline{F} \end{cases} \tag{57}$$

while keeping all the components of the virtual input $\boldsymbol{u}_\alpha = \boldsymbol{0}_3$. The metrics taken into consideration for the comparison are (M1) Average safety field value, (M2) Average path error, that is the average deviation from the nominal path, and (M3) Robot idle time, that is the amount of time in which the robot task is interrupted. The three algorithms are employed in the same simulation scenario, in which the robot has to reach two waypoints with the absolute position connected by a trapezoidal velocity profile in a periodic manner while keeping the absolute orientation and the entire relative pose equal to the initial value. A human operator moves from the right side to the left side of the robot, getting close to the safety field threshold, which is set as $\underline{F} = 0.9$. It is worth noticing that in all three simulations, the mobile base DoFs are taken into account. Figure 7 shows the obtained results.

The top plot shows the safety field value over time for the three algorithms during the simulation. We can observe that Algorithm A3 reaches the lowest value compared to Algorithm A1 and A2; this is because the human operator gets closer to the robot after it is stopped, and the algorithm does not foresee any deviation from the nominal trajectory. On the other hand, Algorithm A2 drives the end effectors away from the human operator, achieving better results but still exceeding the chosen threshold. This is unavoidable since there is no formal constraint on the safety field value in the formulation. Finally, Algorithm A1 manages to keep the safety field value always over the specified threshold, given

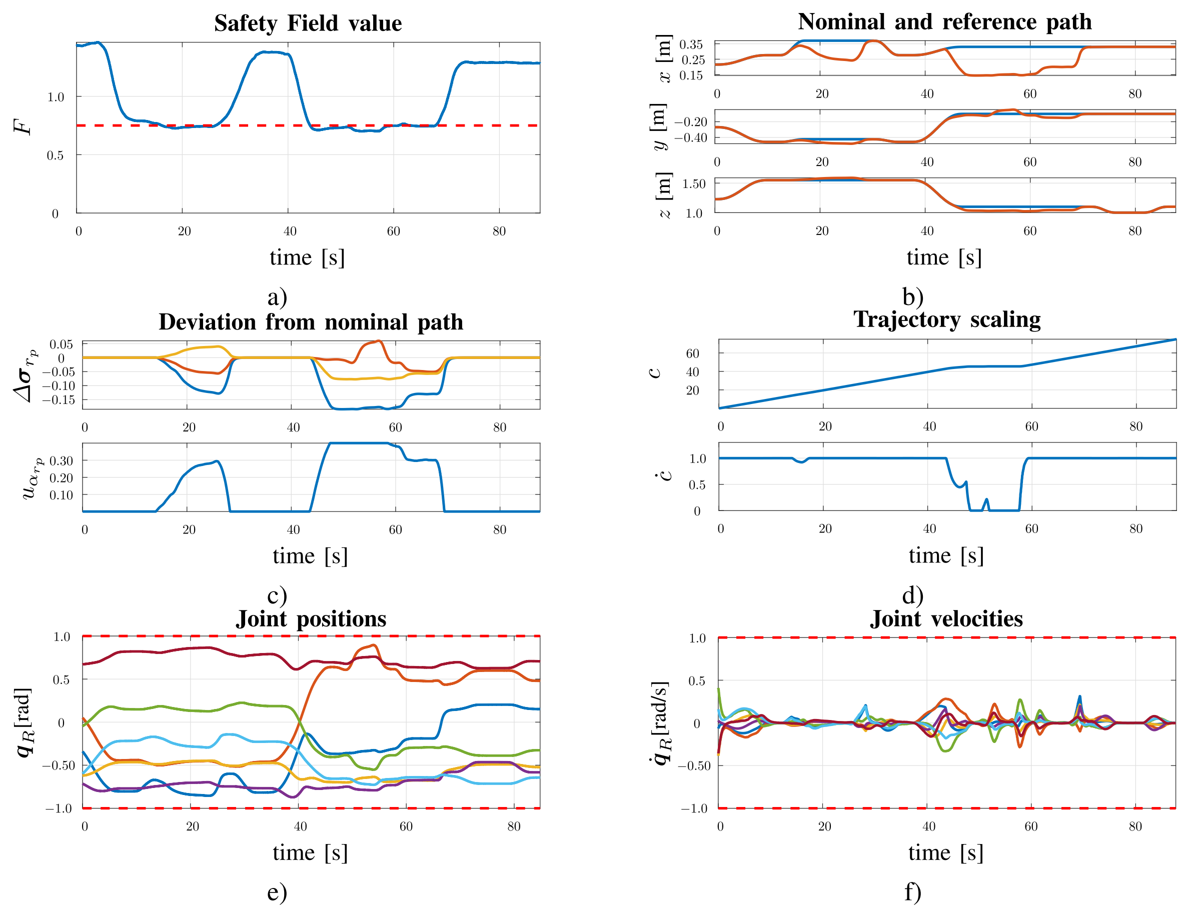

Fig. 6. Field experiment. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line); b) Nominal (blue) and safe (red) trajectories for the absolute frame. The nominal trajectory is composed of a sequence of waypoints suitable for performing a harvesting operation. The proposed Safety Planner changes the nominal path in order to keep the safety value above the imposed minimum threshold; c) Deviation from the nominal path and virtual input u_α. d) Parameter c and virtual input u_c. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds.

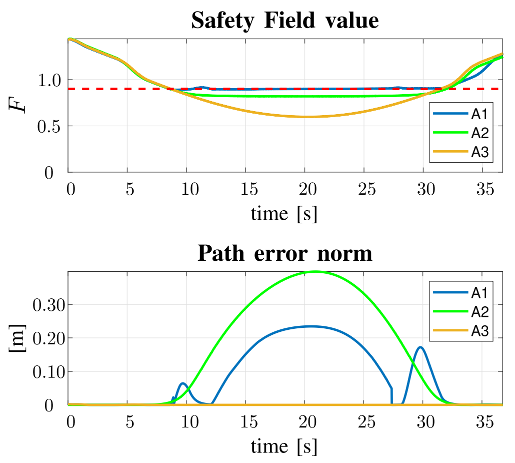

Fig. 7. Conducted comparison, A1 denotes our proposed approach, A2 the evasive motion and A3 the emergency stop. Top: safety field value over time during the simulation. Bottom: path error norm over time during the simulation.

the presence of the constraint on the minimum value of the safety field in the QP problem. The middle left plot shows the path error norm over time for the three algorithms. Algorithm A3 does not introduce any error on the path, since it only modifies the reference velocity, preserving the nominal path. Regarding the other two algorithms, A1 deviates less from the nominal path than A2. This is motivated by the fact that our algorithm has the additional degree of freedom of the trajectory scaling to exploit (as shown in the middle right) with respect to the evasive motion, which can only modify the path. Table III summarizes the metrics values obtained by the different algorithms during the simulations, showing that our approach additionally minimizes the robot's idle time. Joint limits are fulfilled in all cases as shown in the bottom plots.

TABLE III

COMPUTED METRICS FOR THE COMPARISON. A1: PROPOSED APPROACH; A2: EVASIVE MOTION; A3: EMERGENCY STOP. M1: AVERAGE SAFETY FIELD VALUE; M2: AVERAGE PATH ERROR; M3: PERCENTAGE ROBOT IDLE TIME

ALGORITHM	M1	M2	M3
A1	0.99	0.08	0
A2	0.95	0.0	0
A3	0.86	0.16	62

VI. CONCLUSION

In this paper, we have proposed an architecture for handling human safety in a human-robot interaction scenario. In detail, a Safety Planner allows modifying the nominal desired trajectory for a dual-arm robotic system in terms of velocity scaling and deviation from the nominal path for human-robot interaction settings. The level of human safety is quantified by defining a safety field, and the devised strategy finds the optimal trajectory modification parameters by solving a QP problem. The output of the Safety Planner has been integrated within an HQP control framework that allows to perform several other tasks simultaneously. Finally, the effectiveness of the proposed approach has been experimentally validated on a dual-arm robot both in indoor and realistic outdoor scenarios.

Future efforts will be devoted to i) integrating human motion and intention prediction modules in the devised strategy to increase the level of safety further and situation awareness in collaborative settings; ii) investigating human-robot task allocation strategies to properly perform complex operations by assigning elementary tasks to the available agents (human and/or robots) while guaranteeing a certain level of safety; iii) including subjective bio-metric parameters in the field assessment such as heart rate or sweating, iv) including methodologies to increase the planner robustness during periodic motions such as in [38] and [39], and v) in the case of humans interacting with multiple robots, tackling the multi-robot nature of the system. In this case, the safety of the human operators has to be assessed by taking into account the degree of cooperation of the robots and controlled by increasing the overall redundancy of the system; furthermore, a distributed setting in order to avoid the bottleneck of a central unit has been devised in such a scenario.

References

[1] A. Hentout, M. Aouache, A. Maoudj, and I. Akli, "Human–robot interaction in industrial collaborative robotics: A literature review of the decade 2008–2017," *Adv. Robot.*, vol. 33, nos. 15–16, pp. 764–799, Aug. 2019.

[2] A. E. Abdelaal, P. Mathur, and S. E. Salcudean, "Robotics in vivo: A perspective on human–robot interaction in surgical robotics," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 3, no. 1, pp. 221–242, May 2020.

[3] J. P. Vasconez, G. A. Kantor, and F. A. A. Cheein, "Human–robot interaction in agriculture: A survey and current challenges," *Biosyst. Eng.*, vol. 179, pp. 35–48, Mar. 2019.

[4] K. S. Jones and E. A. Schmidlin, "Human–robot interaction: Toward usable personal service robots," *Rev. Hum. Factors Ergonom.*, vol. 7, no. 1, pp. 100–148, Sep. 2011.

[5] M. Lippi and A. Marino, "A control barrier function approach to human-multi-robot safe interaction," in *Proc. 29th Medit. Conf. Control Autom. (MED)*, Jun. 2021, pp. 604–609.

[6] D. Kulić and E. A. Croft, "Real-time safety for human–robot interaction," *Robot. Auton. Syst.*, vol. 54, no. 1, pp. 1–12, 2006.

[7] N. Najmaei and M. R. Kermani, "Prediction-based reactive control strategy for human–robot interactions," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 3434–3439.

[8] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human–robot collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 338–345.

[9] H. Nascimento, M. Mujica, and M. Benoussaad, "Collision avoidance in human–robot interaction using Kinect vision system combined with robot's model and data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10293–10298.

[10] H. Liu et al., "Real-time and efficient collision avoidance planning approach for safe human–robot interaction," *J. Intell. Robot. Syst.*, vol. 105, no. 4, p. 93, Aug. 2022.

[11] K. Merckaert, B. Convens, C.-J. Wu, A. Roncone, M. M. Nicotra, and B. Vanderborght, "Real-time motion control of robotic manipulators for safe human–robot coexistence," *Robot. Comput.-Integr. Manuf.*, vol. 73, Feb. 2022, Art. no. 102223.

[12] J. A. Marvel, "Performance metrics of speed and separation monitoring in shared workspaces," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 2, pp. 405–414, Apr. 2013.

[13] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human–robot collaborative manufacturing environments: Metrics and control," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 882–893, Apr. 2016.

[14] B. Lacevic, A. M. Zanchettin, and P. Rocco, "Safe human–robot collaboration via collision checking and explicit representation of danger zones," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 846–861, Apr. 2023.

[15] L. Scalera, R. Vidoni, and A. Giusti, "Optimal scaling of dynamic safety zones for collaborative robotics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 3822–3828.

[16] M. Faroni, M. Beschi, and N. Pedrocchi, "Safety-aware time-optimal motion planning with uncertain human state estimation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12219–12226, Oct. 2022.

[17] A. Palleschi, M. Hamad, S. Abdolshah, M. Garabini, S. Haddadin, and L. Pallottino, "Fast and safe trajectory planning: Solving the cobot performance/safety trade-off in human–robot shared environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5445–5452, Jul. 2021.

[18] M. Costanzo, G. De Maria, G. Lettera, and C. Natale, "A multimodal approach to human safety in collaborative robotic workcells," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1202–1216, Apr. 2022.

[19] M. Lippi and A. Marino, "Human multi-robot safe interaction: A trajectory scaling approach based on safety assessment," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 4, pp. 1565–1580, Jul. 2021.

[20] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 3420–3431.

[21] E. A. Basso and K. Y. Pettersen, "Task-priority control of redundant robotic systems using control Lyapunov and control barrier function based quadratic programs," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9037–9044, 2020.

[22] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 398–410, Jun. 1997.

[23] P. D. Lillo, D. D. Vito, and G. Antonelli, "Merging global and local planners: Real-time replanning algorithm of redundant robots within a task-priority framework," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 1180–1193, Apr. 2023.

[24] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, Jun. 2014.

[25] S. Kim, K. Jang, S. Park, Y. Lee, S. Y. Lee, and J. Park, "Continuous task transition approach for robot controller based on hierarchical quadratic programming," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1603–1610, Apr. 2019.

[26] H. A. Park and C. S. G. Lee, "Extended cooperative task space for manipulation tasks of humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 6088–6093.

[27] F. Basile, F. Caccavale, P. Chiacchio, J. Coppola, and A. Marino, "A decentralized kinematic control architecture for collaborative and cooperative multi-arm systems," *Mechatronics*, vol. 23, no. 8, pp. 1100–1112, Dec. 2013.

[28] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall, *A Survey on Human Motion Analysis From Depth Data*. Berlin, Germany: Springer, 2013, pp. 149–187.

[29] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Human skeleton tracking from depth data using geodesic distances and optical flow," *Image Vis. Comput.*, vol. 30, no. 3, pp. 217–226, Mar. 2012.

[30] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 5636–5643.

[31] T. Arai, R. Kato, and M. Fujita, "Assessment of operator stress induced by robot collaboration in assembly," *CIRP Ann.*, vol. 59, no. 1, pp. 5–8, 2010.

[32] B. Siciliano, "A closed-loop inverse kinematic scheme for on-line joint-based robot control," *Robotica*, vol. 8, no. 3, pp. 231–243, Jul. 1990.

[33] Z. Zhang, Y. Lin, S. Li, Y. Li, Z. Yu, and Y. Luo, "Tricriteria optimization-coordination motion of dual-redundant-robot manipulators for complex path planning," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 4, pp. 1345–1357, Jul. 2018.

[34] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021.

[35] R. P. Joshi et al. (2019). *ROS OpenPose*. [Online]. Available: https://github.com/ravijo/ros_openpose

[36] T. A. Ciarfuglia, I. M. Motoi, L. Saraceni, M. Fawakherji, A. Sanfeliu, and D. Nardi, “Weakly and semi-supervised detection, segmentation and tracking of table grapes with limited and noisy data,” *Comput. Electron. Agricult.*, vol. 205, Feb. 2023, Art. no. 107624.

[37] B. Lacevic, P. Rocco, and A. M. Zanchettin, “Safety assessment and control of robotic manipulators using danger field,” *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1257–1270, Oct. 2013.

[38] Z. Zhang and Z. Yan, “A varying parameter recurrent neural network for solving nonrepetitive motion problems of redundant robot manipulators,” *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 6, pp. 2680–2687, Nov. 2019.

[39] Z. Zhang et al., “A varying-parameter convergent-differential neural network for solving joint-angular-drift problems of redundant robot manipulators,” *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 2, pp. 679–689, Apr. 2018.

Jozsef Palmieri (Graduate Student Member, IEEE) received the M.Sc. degree from the University of Cassino and Southern Lazio in 2022, where he is currently pursuing the Ph.D. degree. He has been involved in the H2020 CANOPIES Project. His studies and research project are particularly focused on kinematic control of mobile dual-arm systems both in and out of the contexts of human–robot interaction.

Paolo Di Lillo (Associate Member, IEEE) received the Ph.D. degree in robotics from the University of Cassino and Southern Lazio in 2018. He is currently an Assistant Professor with the University of Cassino and Southern Lazio. His research interests include dynamic and kinematic control methods for redundant base-fixed manipulators, mobile dual arm systems, human–robot collaboration, assistive robotics, and control underwater vehicle-manipulator systems. He has been involved in the H2020 project DexROV on autonomous underwater intervention with remote supervision via satellite communication and on H2020 project CANOPIES on robotic systems for precision agriculture.

Martina Lippi (Associate Member, IEEE) received the M.Sc. (cum laude) and Ph.D. degrees in information engineering from the University of Salerno, Italy, in 2017 and 2020, respectively. She has been a Visiting Scholar with the KTH Royal Institute of Technology, Sweden, in 2019. She was a Post-Doctoral Researcher with Roma Tre University, Italy, from November 2020 to June 2022, where she has been an Assistant Professor since June 2022. Her research interests include human–robot interaction, multimanipulator systems, and distributed control.

Stefano Chiaverini (Fellow, IEEE) was born in Naples, Italy, in December 1961. He received the Laurea and the Research Doctorate degrees in electronics engineering from the University of Naples, Italy, in 1986 and 1990, respectively. He is currently a Professor in automatic control with the University of Cassino and Southern Lazio, where he was the Head of the Department of Electric and Information Engineering from 2000 to 2018. His research interests include manipulator inverse kinematics techniques, redundant manipulator control, cooperative robot systems, force/position control of manipulators, underwater robotic systems, and mobile robotic systems.

Alessandro Marino (Senior Member, IEEE) received the M.Sc. degree (cum laude) in computer science engineering from the University of Naples Federico II, Italy, in 2006, and the Ph.D. degree in automation and robotics from the University of Basilicata, Italy, in 2010. Since 2018, he has been an Associate Professor with the University of Cassino and Southern Lazio. He is currently a Local Principal Investigator of the H2020-ICT Project CANOPIES and the H2020-CS2 Project LABOR. His research interests include modeling and control of robotic systems, multi-robot systems, human–robot interaction, and distributed control.

Open Access funding provided by ‘Università degli Studi di Cassino e del Lazio Meridionale’ within the CRUI CARE Agreement