Ref. Ares(2023)6622060 - 30/09/2023

This project is funded by the European Union
G.A No 101016906

Project Number: 101016906
Start Date of Project: 2021/01/01
Duration: 48 months

Type of document D5.2 – V1
Mobile arms safety functionalities

Dissemination level	PUBLIC
Submission Date	2023-09-30
Work Package	WP5
Task	T5.2
Type	Report
Version	1
Author	UNICLAM
Approved by	DTI, UNIROMA1

DISCLAIMER:
The sole responsibility for the content of this deliverable rests with the authors. It does not necessarily reflect the opinion of the European Union. Neither the Directorate-General for Communications Networks, Content and Technology, Artificial Intelligence and Digital Industry nor the European Commission are responsible for any use that may be made of the information contained therein.

This project is funded by the European Union
G.A No 101016906

Executive Summary

This document describes design and testing of the farming robot safety functionalities. In particular, the developed control architecture allows modifying a nominal reference trajectory planned for agronomic purposes in order to account for the operators safety. In detail, the nominal trajectory is modified in terms of both path to follow and time law, while accounting for mobile arms constraints as well. To this aim, a safety field is defined the aim of which is to quantify the level of human safety and, based on its value, a Safety Planner computes virtual inputs that allows such trajectory modifications by solving a dedicated QP problem. The safety planner algorithm is integrated with the low-level controller described in the Deliverable 5.1.

Table of Content

1 Introduction 5
2 Preliminaries and problem formulation ..... **Errore. Il segnalibro non è definito.**
2.1 Control Barrier Functions 5
2.2 Robot Kinematics 6
2.3 Hierarchical Quadratic Programming 7
2.4 Considered tasks 8
2.4.1 Joint limits tasks 8
2.4.2 Operational tasks 8
2.5 Human Safety Field 9
2.6 Problem formulation 11
3 Proposed solution 11
3.1 Trajectory modification 12
3.2 Low-level controller for safe trajectory tracking 13
3.3 Time derivative of the safety field 14
3.4 Safety planner design 16
4 Validation 19
4.1 Laboratory experiments 19
4.2 Outdoor experiments 23
5 Reference to Web Content 26
6 References 26
7 Bibliography 27

This project is funded by the European Union G.A No 101016906

Abbreviations and Acronyms

RGB-D camera	Red, green, blue and depth camera
ROS	Robot Operating System
(H)QP	(Hierarchical)-Quadratic-Programming
CLIK	Closed-Loop-Inverse-Kinematics
DoFs	Degrees of Freedom
CBF(s)	Control Barrier Function(s)

1 Introduction

One of the challenges of the CANOPIES project is to enable effective human-robot interaction and collaboration in order to exploit their different capabilities to achieve a common goal. When dealing with such settings, the safety of the human operators is a key aspect that needs to be properly handled by dedicated algorithms. The Farming Robot must be capable of detecting the presence of human operators that share its own workspace, and of modifying its autonomous behaviour while performing the desired operation. In this deliverable, we provide a description of the safety algorithm we designed for this purpose, together with its experimental validations both in indoor environment and in real field.

More in detail, the proposed method utilizes an appropriate metric to measure safety levels and adjust 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 further enhance safety levels. The approach is incorporated into the Hierarchical Quadratic Programming control framework described in Deliverable D5.1, allowing the robot to simultaneously address other control objectives, such as handling joint limits.

2 Basic concepts and problem formulation

The proposed approach relies on some basic tools which are Control Barrier Functions (CBF), robot modelling, and Hierarchical-Quadratic-Programming (HQP) that are, first, briefly recalled. Then, the safety field concept designed to assess and control the safety in human-robot cooperation settings is presented. Finally, we state the formulation of the problem addressed. Note that certain concepts, such as robot modelling and HQP formulation, have been extensively covered in Deliverable D5.1. These concepts are summarized here in order to ensure the self-contained nature of this document and align with the newly adopted notation that is consistently used throughout the document.

2.1 Control Barrier Functions

CBFs allows formally handling inequality constraints (Ames, et al., 2019), (Basso & Pettersen, 2020) and are here used to ensure the human safety is always above a given threshold as detailed in Section 2.5.

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 k-th generic constraint be expressed in the 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 $\boldsymbol{x}$ 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 (Ames, et al., 2019):

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 $\frac{\partial h_k}{\partial \xi}(\xi) \neq 0 \forall \xi \in \partial \mathcal{C}_k$, then, any Lipschitz continuous controller $u(\xi)$ for the system in Eq.(1) satisfying Eq.(3) renders the set $\mathcal{C}_k$ (where the constraint $h_k(\boldsymbol{\xi}) \geq 0$ is satisfied) 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 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. This property will be exploited to set-up an HQP problem, as detailed in Section 2.3.

2.2 Robot Kinematic Modelling

Let us consider a mobile robot equipped with a movable torso and a dual arm system, in which each end effector configuration is denoted by $\boldsymbol{x}_y = [\boldsymbol{p}_y^{\mathrm{T}} \ \boldsymbol{o}_y^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^7$, ($y = L, R$, Left and Right, respectively), where $\boldsymbol{p}_y \in \mathbb{R}^3$ is the position part, and $\boldsymbol{o}_y = [o_{y,1} \quad o_{y,2} \quad o_{y,3} \quad o_{y,4}]^{\mathrm{T}} = \left[\kappa_{o_y} \quad \boldsymbol{\rho}_{o_y}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^4$ 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 denote the product of two quaternions. Vector $\boldsymbol{o}_0 = [1 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$ represents 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 both 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 $\boldsymbol{q}_B = \left[\boldsymbol{p}_B^{\mathrm{T}}, \ \boldsymbol{o}_B^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^b$ is the vector describing the mobile base configuration in terms of position and orientation expressed as quaternion, $\boldsymbol{q}_T \in \mathbb{R}^t$ gathers the joint variables of the torso, while $\boldsymbol{q}_L, \boldsymbol{q}_R \in \mathbb{R}^p$ are the vectors of joint variables relative to arms, and $n = b + t + 2p$ is the total number of degrees of freedom (DoFs).

The collective vector of end effector linear/angular velocities is described by $\boldsymbol{v} = \left[\boldsymbol{v}_L^{\mathrm{T}} \quad \boldsymbol{v}_R^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{12}$ and its relationship with the system 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:

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

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.

2.3 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 fulfil in order 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 recommendable to set priority orders among them obtaining a task hierarchy. This is usually implemented by projecting the joint velocity related to the fulfilment of a lower-priority task onto the null space of the higher-priority tasks, leading to different kinds of Task-Priority (TP) control frameworks (Chiaverini, 1997), (Di Lillo, Arrichiello, Antonelli, & Chiaverini, 2018).

A popular alternative is to exploit the Hierarchical Quadratic Programming (HQP) control framework (Escande, Mandard, & Wieber, 2014), (Kim, et al., 2019) in which the task hierarchy is implemented by solving a cascade of 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 h arbitrary tasks by solving the cascade of h QP problems, leading to a sequence of optimal variables $\{\boldsymbol{w}_1^\star, \boldsymbol{w}_2^\star, \cdots, \boldsymbol{w}_h^\star\}$ 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.

2.4 Considered tasks

In the following the tasks considered in the HQP hierarchy are detailed.

2.4.1 Joint limits tasks

Let us consider the first constraint in Eq. (7), it can be expressed by exploiting the CBF framework by introducing the following functions:

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

that lead to:

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

with $\bar{\phi}_i,\ \underline{\phi}_i > 0$. Let us define vector functions $\underline{\boldsymbol{h}} = [\underline{h}_1, \underline{h}_2 \cdots, \underline{h}_n]$ and $\bar{\boldsymbol{h}} = [\bar{h}_1,\ \bar{h}_2 \cdots, \bar{h}_n]$, and matrices $\underline{\boldsymbol{\Phi}} = \mathrm{diag}\{\underline{\phi}_1, \underline{\phi}_2, \cdots, \underline{\phi}_n\}$ and $\bar{\boldsymbol{\Phi}} = \mathrm{diag}\{\bar{\phi}_1, \bar{\phi}_2, \cdots, \bar{\phi}_n\}$; then, it is straightforward to verify that the constraint can be more conveniently expressed as:

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

Then, it holds:

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

where $\boldsymbol{J}_c = [\boldsymbol{I}_n \quad \boldsymbol{I}_n]^{\mathrm{T}}$ is the constraints Jacobian.

2.4.2 Operational tasks

As in (Park & Lee), the robot operational task is specified by means of absolute and relative task variables:

$$\boldsymbol{\sigma} = \begin{bmatrix} \boldsymbol{\sigma}_a \\ \boldsymbol{\sigma}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma}_{a_p} \\ \boldsymbol{\sigma}_{a_o} \\ \boldsymbol{\sigma}_{r_p} \\ \boldsymbol{\sigma}_{r_o} \end{bmatrix} = \begin{bmatrix} \beta \boldsymbol{p}_L + (1-\beta)\boldsymbol{p}_R \\ \boldsymbol{o}_a \\ \boldsymbol{p}_R - \gamma \boldsymbol{p}_L \\ \boldsymbol{o}_r \end{bmatrix} \tag{11}$$

with $\beta \in [0\ 1]$ and $\gamma \in \{0, 1\}$, whose meaning will be detailed in the following. Furthermore, $\boldsymbol{o}_a$ is the quaternion related to the 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. In addition, $\boldsymbol{o}_r$ is the quaternion extracted from the 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. 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:

$$\boldsymbol{\nu} = \begin{bmatrix} \beta \boldsymbol{I}_6 & (1-\beta)\boldsymbol{I}_6 \\ -\gamma \boldsymbol{I}_6 & \boldsymbol{I}_6 \end{bmatrix} \boldsymbol{v} = \boldsymbol{J}_t \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{12}$$

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 (Basile, Caccavale, Chiacchio, Coppola, & Marino, 2013).

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; in particular, $\beta \in \{0, 1\}$ results 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.
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:

$$\boldsymbol{\sigma}^d(t) = \begin{bmatrix} \boldsymbol{\sigma}^d_{a_p} \\ \boldsymbol{\sigma}^d_{a_o} \\ \boldsymbol{\sigma}^d_{r_p} \\ \boldsymbol{\sigma}^d_{r_o} \end{bmatrix} \in \mathbb{R}^{14}, \qquad \boldsymbol{\nu}^d(t) = \begin{bmatrix} \dot{\boldsymbol{\sigma}}^d_{a_p} \\ \boldsymbol{\omega}^d_a \\ \dot{\boldsymbol{\sigma}}^d_{r_p} \\ \boldsymbol{\omega}^d_r \end{bmatrix} \in \mathbb{R}^{12} \tag{13}$$

coherent with the cooperation mode, is made available.
The operational task introduces the following equality constraint in the QP problem:

$$\boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{b}_\sigma \tag{14}$$

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. In order to create a solution that can be applied to off-the-shelf robots that typically feature a joint position/velocity interface, 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., $\boldsymbol{q}_r \approx \boldsymbol{q}$ $(\dot{\boldsymbol{q}}_r \approx \dot{\boldsymbol{q}})$.
Based on the above assumption and on the kinematic model in Eq. (5), the following virtual model is considered:

$$\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}_r = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{15}$$

This assumption holds for the CANOPIES-Farming Robot (as well as for many off-the-shelf robotic platforms) and allows to decouple the control design for the fulfilment of high-level tasks from the low-level trajectory tracking.

2.5 Human Safety Field

A safety field to account for the human-robot relative position is defined in this section. The field definition is based on the work in (Lippi & Marino, 2020) and takes into consideration the overall robotic structure as well as an arbitrary number n_o of significant human points, e.g., the endpoints of the links composing the human skeleton representation. 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 a single point P_o of the human operator is first analysed. In detail, let $\boldsymbol{p} \in \mathbb{R}^3$ be the position vector of the robot point P, $\boldsymbol{p}_o \in \mathbb{R}^3$ the position vector of the human point P_o, and $d = \|\boldsymbol{p} - \boldsymbol{p}_o\|$ the distance between the points P and P_o. The following local safety index is defined:

$$f(\boldsymbol{p}, \boldsymbol{p}_o) = \chi(d) \tag{16}$$

whose a representation is shown in Figure 1, and where χ represents any non-negative continuous

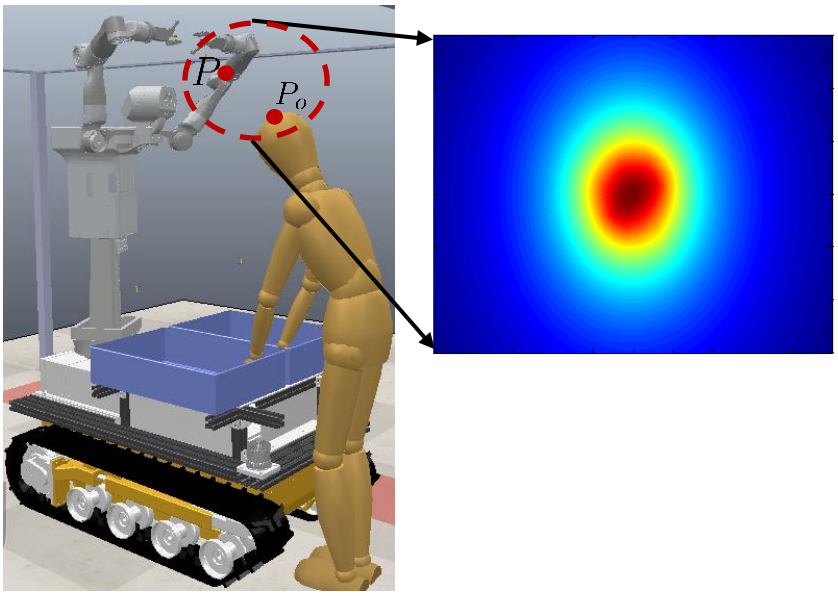

Figure 1: The safety field in the case of two points P and P_O belonging to the robot and to the human, respectively.

monotonically increasing Lipschitz function.

Based on the local index in Eq.(16), the overall safety field can be derived by integrating it along the robot links and by then evaluating it for each human point. We approximated each robot link l as a segment starting at $\boldsymbol{p}_l^0$ and ending at $\boldsymbol{p}_l^1$. In detail, 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, that is 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:

$$F_l(\boldsymbol{p}_l^0, \boldsymbol{p}_l^1, \boldsymbol{p}_o) = \int_0^1 f(\boldsymbol{p}_l^s, \boldsymbol{p}_o)ds \tag{17}$$

In order to generalize the approach, we consider all the links of the robotic platform and the j-th human point $\boldsymbol{p}_{o,j}$. The safety field taking into account the single j-th human point and the entire robot structure is obtained as:

$$F^j(\boldsymbol{q}, \boldsymbol{p}_{o,j}) = \sum_{l=1}^{n} F_l(\boldsymbol{p}_l^0, \boldsymbol{p}_l^1, \boldsymbol{p}_{o,j}) \tag{18}$$

that is summing up the field related to all links of the robot. Finally, by considering all the n_o human points, the overall safety field is computed as:

$$F = \frac{1}{n_o} \sum_{j=1}^{n_o} F^j(\boldsymbol{q}, \boldsymbol{p}_{o,j}) \tag{19}$$

The above formulation allows accounting for multiple human operators by simply adding the respective points in Eq. (19). In addition, to compute the safety field we assume that the robot is able to detect and localize the human operator within the shared workspace. To this aim, several approaches existing in the literature and made available to the community can be exploited; more details on this aspect will be provided in Section 4.

3 Problem formulation and proposed solution

After the definition of the safety field which allows to assess how safe is the collaboration between humans and robot, it is necessary to devise a control strategy able to ensure that the safety does not fall below a given threshold. In detail, we formulated the following problem tailored for the farming robot case.

3.1 Problem formulation

Problem: Let us consider a dual-arm robotic system performing an agronomic cooperative task encoded by a task function $\boldsymbol{\sigma}$ as per Eq. (11), 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 and that the level of human safety is assessed by the index $F(t)$ in Eq. (19), for which a time-varying minimum value $\underline{\mathrm{F}}(t)$ is assigned. The objective is to design a Safety Planner capable of:

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

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

$$F(t) \geq \underline{\mathrm{F}}(t) \tag{20}$$

with $\underline{\mathrm{F}}(t) > 0$ a time-varying threshold, while considering the kinematic constraints of the robot.
The above formulation requires the definition of the safety time-varying threshold $\underline{\mathrm{F}}(t)$. In (Lippi & Marino, 2020), it is proven that $\underline{\mathrm{F}}(t)$ 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 experimentally set by assessing, for instance, how stressed the human operator is and how much risk he/she perceives, that is according to biometric and/or subjective parameters.

3.2 Proposed solution

Figure 2 shows the proposed architecture for solving the problem defined in the previous section. 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, 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 based on the safety of the interaction. Then, we describe the low-level controller and the derivative of the safety field. Finally, we detail the proposed safety planner.
Regarding the Perception System, an off-the-shelf solution were used, namely OpenPose, as mentioned in Section 4; then, it will not be detailed further in the following.

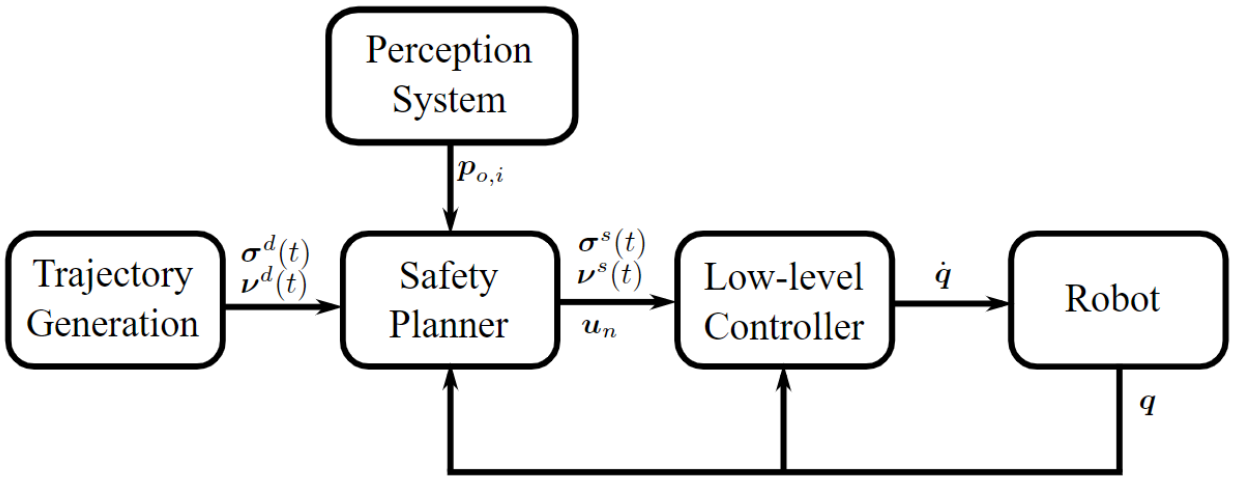

Figure 2: Overall control architecture. A Trajectory Generation module generates a nominal trajectory, that is then modified by a Safety Planner in function of the Perception System outputs in order to obtain a reference trajectory. Finally, the latter is given in input to a controller that computes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

3.2.1 Trajectory parametrization

Let us assume that the desired nominal trajectory $\boldsymbol{\sigma}^d(t)$ in Eq. (13) (see Figure 2) is parameterized with respect to a time parameter $c(t)$, i.e., $\boldsymbol{\sigma}^d(c(t))$, such that $c(t) : [t_0, t_f] \in \mathbb{R} \rightarrow [t_0, t_f] \in \mathbb{R}$, 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 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:

$$\boldsymbol{\nu}^d(c(t), \dot{c}(t)) = \boldsymbol{\eta}\left(c(t)\right) \dot{c}(t) \tag{21}$$

with

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

and

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

Additionally, we introduce the time-varying term $\Delta\boldsymbol{\sigma} = \begin{bmatrix} \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{a_p} & \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{a_o} & \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{r_p} & \Delta\boldsymbol{\sigma}^{\mathrm{T}}_{r_o} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{14}$ 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:

$$\Delta\boldsymbol{\nu}=\begin{bmatrix}\dot{\Delta}\boldsymbol{\sigma}_{a_p}^{\mathrm{T}} & \Delta\boldsymbol{\omega}_{a}^{\mathrm{T}} & \dot{\Delta}\boldsymbol{\sigma}_{r_p}^{\mathrm{T}} & \Delta\boldsymbol{\omega}_{r}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \tag{22}$$

the safe trajectory is computed as follows:

$$\boldsymbol{\sigma}^s(t)=\begin{bmatrix}\boldsymbol{\sigma}_{a_p}^s\\ \boldsymbol{\sigma}_{a_o}^s\\ \boldsymbol{\sigma}_{r_p}^s\\ \boldsymbol{\sigma}_{r_o}^s\end{bmatrix}=\begin{bmatrix}\boldsymbol{\sigma}_{a_p}^d+\Delta\boldsymbol{\sigma}_{a_p}\\ \boldsymbol{\sigma}_{a_o}^d*\Delta\boldsymbol{\sigma}_{a_o}^{-1}\\ \boldsymbol{\sigma}_{r_p}^d+\Delta\boldsymbol{\sigma}_{r,p}\\ \boldsymbol{\sigma}_{r_o}^d*\Delta\boldsymbol{\sigma}_{r_o}^{-1}\end{bmatrix},\quad \boldsymbol{\nu}^s(t)=\begin{bmatrix}\dot{\boldsymbol{\sigma}}_{a_p}^s\\ \boldsymbol{\omega}_{a}^s\\ \dot{\boldsymbol{\sigma}}_{r_p}^s\\ \boldsymbol{\omega}_{r}^s\end{bmatrix}=\boldsymbol{\nu}^d(c(t),\dot{c}(t))+\Delta\boldsymbol{\nu}, \tag{23}$$

which is the actual trajectory to be tracked by the robot.
Let $\Delta\boldsymbol{\varepsilon}\in\mathbb{R}^{12}$ be the vector of position and orientation deviations defined as:

$$\Delta\boldsymbol{\varepsilon}=\begin{bmatrix}\Delta\boldsymbol{\sigma}_{a_p}^{\mathrm{T}} & \boldsymbol{\rho}_{\Delta\boldsymbol{\sigma}_{a_o}}^{\mathrm{T}} & \Delta\boldsymbol{\sigma}_{a_p}^{\mathrm{T}} & \boldsymbol{\rho}_{\Delta\boldsymbol{\sigma}_{r_o}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \tag{24}$$

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:

$$\Delta\boldsymbol{\nu}=-\boldsymbol{T}\Delta\boldsymbol{\varepsilon}+\boldsymbol{\zeta},\quad \Delta\boldsymbol{\varepsilon}(t_0)=\mathbf{0}_{12} \tag{25}$$

where $\mathbf{0}_m$ 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 $\boldsymbol{\zeta}=\begin{bmatrix}\boldsymbol{\zeta}_{a_p}^{\mathrm{T}} & \boldsymbol{\zeta}_{a_o}^{\mathrm{T}} & \boldsymbol{\zeta}_{r_p}^{\mathrm{T}} & \boldsymbol{\zeta}_{r_o}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}\in\mathbb{R}^{12}$, with $\boldsymbol{\zeta}_{(\cdot)}\in\mathbb{R}^3$ input vectors to be designed. 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:

$$\boldsymbol{\zeta}=\begin{bmatrix}\boldsymbol{\psi}_{a_p} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \boldsymbol{\psi}_{a_o} & \mathbf{0}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\psi}_{r_p} & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{\psi}_{r_o}\end{bmatrix}\begin{bmatrix}u_{\alpha_{a_p}}\\ u_{\alpha_{a_o}}\\ u_{\alpha_{r_p}}\\ u_{\alpha_{r_o}}\end{bmatrix}=\boldsymbol{\Psi}(t)\boldsymbol{u}_\alpha \tag{26}$$

where $\boldsymbol{\Psi}\in\mathbb{R}^{12\times4}$ 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.
Notably, the case $c(t)=t$, $\Delta\boldsymbol{\varepsilon}=\mathbf{0}_{12}$ 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 can be online scaled while, by modifying $\Delta\boldsymbol{\sigma}$, the path is modified.

3.2.2 Low-level controller for safe trajectory tracking

The low-level controller considers three levels of priority among the tasks: 1) joint position/velocity limits; 2) operational task; 3) null-space joint velocity.
The first QP problem to solve to fulfil the joint limits tasks is:

$$\begin{aligned}\min_{\boldsymbol{w}_1,\dot{\boldsymbol{q}}}\quad & \frac{1}{2}\boldsymbol{w}_1^T\boldsymbol{Q}_{w,1}\boldsymbol{w}_1\\ \text{s.t.}\quad & \underline{\boldsymbol{b}}_l\le\boldsymbol{J}_l(\boldsymbol{q})\dot{\boldsymbol{q}}+\boldsymbol{w}_1\le\bar{\boldsymbol{b}}_l,\end{aligned} \tag{27}$$

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:

$$\begin{aligned} \min_{\boldsymbol{w}_2, \dot{\boldsymbol{q}}} \quad & \frac{1}{2} \boldsymbol{w}_2^T \boldsymbol{Q}_{w,2} \boldsymbol{w}_2 \\ \text{s.t.} \quad & \boldsymbol{J}_\sigma(\boldsymbol{q}) \dot{\boldsymbol{q}} + \boldsymbol{w}_2 = \boldsymbol{b}_\sigma \\ & \underline{\boldsymbol{b}}_l \leq \boldsymbol{J}_l(\boldsymbol{q}) \dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \leq \bar{\boldsymbol{b}}_l, \end{aligned} \tag{28}$$

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:

$$\tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \boldsymbol{\sigma}_{a_p}^s - \boldsymbol{\sigma}_{a_p} \\ \boldsymbol{\rho}_{\tilde{\sigma}_{a_o}} \\ \boldsymbol{\sigma}_{r_p}^s - \boldsymbol{\sigma}_{r_p} \\ \boldsymbol{\rho}_{\tilde{\sigma}_{r_o}} \end{bmatrix}$$

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

$$\begin{aligned} \min_{\boldsymbol{w}_3, \dot{\boldsymbol{q}}} \quad & \frac{1}{2} \boldsymbol{w}_3^T \boldsymbol{Q}_{w,3} \boldsymbol{w}_3 \\ \text{s.t.} \quad & \dot{\boldsymbol{q}} + \boldsymbol{w}_3 = \dot{\boldsymbol{q}}_n \\ & \boldsymbol{J}_\sigma(\boldsymbol{q}) \dot{\boldsymbol{q}} + \boldsymbol{w}_2^\star = \boldsymbol{b}_\sigma \\ & \underline{\boldsymbol{b}}_l \leq \boldsymbol{J}_l(\boldsymbol{q}) \dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \leq \bar{\boldsymbol{b}}_l \end{aligned} \tag{29}$$

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 defined in Section above3.1. This term is computed by the Safety Planner in order to increase the safety field value without affecting the operational tasks. Furthermore, by exploiting the same control framework, other control objectives might be introduced, including keeping a certain minimum distance from obstacles in the scene or defining virtual walls.

In case none of the kinematic constraints are 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 (Siciliano & Slotine, 1991):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_\sigma^\dagger (\boldsymbol{\nu}^s + \boldsymbol{K}_\sigma \tilde{\boldsymbol{\sigma}}) + \boldsymbol{N}_\sigma \dot{\boldsymbol{q}}_n \tag{30}$$

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

3.2.3 Time derivative of the safety field

To design the safety planner, we first derive the time derivative of the safety field $F(t)$ in Eq.(19) Specifically, in the following we prove it to be linear with respect to the derivative of the scaling parameter, that is $\dot{c}(t)$, the path modification input $\boldsymbol{u}_\alpha$, and the input $\dot{\boldsymbol{q}}_n$ in Eq.(30). This property is exploited to implement the scaling and path modification procedure.

The time derivative $\dot{F}$ is computed starting from the derivative of f in Eq. (16). In particular, $\dot{f}$ can be computed as follows:

$$\dot{f} = \frac{\partial \chi(d^s_{i,l})}{\partial d^s_{i,l}} \dot{d}^s_{i,l}, \tag{31}$$

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

$$\dot{d}^s_{i,l} = \frac{(\boldsymbol{p}^s_l - \boldsymbol{p}_o)^{\mathrm{T}}(\dot{\boldsymbol{p}}^s_l - \dot{\boldsymbol{p}}_o)}{d^s_{i,l}} \tag{32}$$

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 joints variables of the robot is:

$$\dot{\boldsymbol{p}}^s_l = \boldsymbol{J}^s_l(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{33}$$

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 taking into account the input in Eq. (30) and omitting the dependency of the Jacobian matrices from $\boldsymbol{q}$ for sake of readability, Eq. (33) can be rewritten as:

$$\dot{\boldsymbol{p}}^s_l = \boldsymbol{J}^s_l \boldsymbol{J}^{\dagger}_{\sigma}(\boldsymbol{\nu}^s + \boldsymbol{K}_{\sigma}\tilde{\boldsymbol{\sigma}}) + \boldsymbol{J}^s_l \boldsymbol{N}_{\sigma}\dot{\boldsymbol{q}}_n. \tag{34}$$

By folding Eqs. (21)-(26) it holds:

$$\dot{\boldsymbol{p}}^s_l = \boldsymbol{\gamma}_1\dot{c} + \boldsymbol{\Gamma}_2\boldsymbol{u}_{\alpha} + \boldsymbol{\Gamma}_3\dot{\boldsymbol{q}}_n + \boldsymbol{\gamma}_4 \tag{35}$$

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

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

By virtue of Eq. (31) and Eq. (35), the derivative of the safety index can be rewritten as:

$$\dot{f} = \lambda_1\dot{c} + \boldsymbol{\lambda}^{\mathrm{T}}_2\boldsymbol{u}_{\alpha} + \boldsymbol{\lambda}^{\mathrm{T}}_3\dot{\boldsymbol{q}}_n + \lambda_4, \tag{36}$$

where the expressions of $\lambda_1,\ \lambda_4 \in \mathbb{R}, \boldsymbol{\lambda}_2 \in \mathbb{R}^4, \boldsymbol{\lambda}_3 \in \mathbb{R}^3$ are:

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

In light of Eqs .(17)-(19), the derivative of the local safety index is extended to the entire structure of the dual-arm robot and to all the human relevant points as:

$$\dot{F}(t) = \mu_1(t)\,\dot{c}(t) + \boldsymbol{\mu}_2(t)^{\mathrm{T}}\boldsymbol{u}_\alpha + \boldsymbol{\mu}_3(t)^{\mathrm{T}}\dot{\boldsymbol{q}}_n + \mu_4(t) \tag{37}$$

with $\mu_1, \mu_4 \in \mathbb{R}$, $\boldsymbol{\mu}_2 \in \mathbb{R}^4$, $\boldsymbol{\mu}_3 \in \mathbb{R}^3$ defined as:

$$\begin{cases} \mu_1 = \dfrac{1}{n_o}\displaystyle\sum_{j=1}^{n_o}\sum_{l=1}^{n}\int_0^1 \lambda_1(\boldsymbol{p}_l^s, \boldsymbol{p}_{o,j}, \boldsymbol{q}_i, c)\; ds \\ \boldsymbol{\mu}_2 = \dfrac{1}{n_o}\displaystyle\sum_{j=1}^{n_o}\sum_{l=1}^{n}\int_0^1 \boldsymbol{\lambda}_2(\boldsymbol{p}_l^s, \boldsymbol{p}_{o,j}, \boldsymbol{q}_i, c)\; ds \\ \boldsymbol{\mu}_3 = \dfrac{1}{n_o}\displaystyle\sum_{j=1}^{n_o}\sum_{l=1}^{n}\int_0^1 \boldsymbol{\lambda}_3(\boldsymbol{p}_l^s, \boldsymbol{p}_{o,j}, \boldsymbol{q}_i, c)\; ds \\ \mu_4 = \dfrac{1}{n_o}\displaystyle\sum_{j=1}^{n_o}\sum_{l=1}^{n}\int_0^1 \lambda_4(\boldsymbol{p}_l^s, \boldsymbol{p}_{o,j}, \dot{\boldsymbol{p}}_{o,j}, \boldsymbol{q}_i, c)\; ds. \end{cases} \tag{38}$$

that, as $\dot{f}$ in Eq. (36) is linear with respect to $\dot{c}, \boldsymbol{u}_\alpha, \dot{\boldsymbol{q}}_n$.

3.2.4 Safety Planner

We are now ready to present the proposed control solution to problem defined in Section 3.1. By setting $\dot{c} = u_c$ and $\dot{\boldsymbol{q}}_n = \boldsymbol{u}_n$, the overall system dynamics is:

$$\begin{bmatrix} \dot{c} \\ \Delta\boldsymbol{\nu} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} 0 \\ -\boldsymbol{T}\Delta\boldsymbol{\varepsilon} \\ \mu_4 \end{bmatrix} + \begin{bmatrix} 1 & \mathbf{0}_4^{\mathrm{T}} & \mathbf{0}_n^{\mathrm{T}} \\ \mathbf{0}_{12} & \boldsymbol{\Psi} & \boldsymbol{O}_{12\times n} \\ \mu_1 & \boldsymbol{\mu}_2^{\mathrm{T}} & \boldsymbol{\mu}_3^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} u_c \\ \boldsymbol{u}_\alpha \\ \boldsymbol{u}_n \end{bmatrix}, \tag{39}$$

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:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(t,\boldsymbol{\xi}) + \boldsymbol{g}(t,\boldsymbol{\xi}) \begin{bmatrix} u_c \\ \boldsymbol{u}_\alpha \\ \boldsymbol{u}_n \end{bmatrix} = \boldsymbol{f}(t,\boldsymbol{\xi}) + \boldsymbol{g}(t,\boldsymbol{\xi})\boldsymbol{u}_\xi, \tag{40}$$

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 $\boldsymbol{\sigma}_{ap}^d, \boldsymbol{\sigma}_{ao}^d, \boldsymbol{\sigma}_{rp}^d, \boldsymbol{\sigma}_{ro}^d$;
- $\boldsymbol{u}_n$ 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 $\underline{\mathrm{F}}$ 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:

$$\underline{h}_f = F - \underline{F} \tag{41}$$

the constraint on the decision variables that implements Eq. (20) can be formulated as:

$$\boldsymbol{J}_f \boldsymbol{u}_\xi \geq -k_f \underline{h}_f - \mu_4 \tag{42}$$

and, in virtue of Eq.(37), it holds:

$$\boldsymbol{J}_f = \mathrm{diag}\{\mu_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3\}$$

and k_f is a scalar gain.

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

$$\begin{aligned} \min_{\boldsymbol{u}_\xi} \quad & \frac{1}{2}(\boldsymbol{u}_\xi - \boldsymbol{u}_{\xi,n})^{\mathrm{T}} \boldsymbol{Q}_\xi (\boldsymbol{u}_\xi - \boldsymbol{u}_{\xi,n}) + \frac{1}{2} q_w w_f^2 && (1) \\ s.t. \quad & 0 \leq u_c \leq 1, && (2) \\ & \underline{\boldsymbol{u}}_\alpha \leq \boldsymbol{u}_\alpha \leq \bar{\boldsymbol{u}}_\alpha, && (3) \\ & \boldsymbol{J}_f \boldsymbol{u}_\xi + w_f \geq -k_f \underline{\mathrm{h}}_f - \mu_4. && (4) \end{aligned} \tag{43}$$

Each constraint is detailed in the following:

- Eq. (43.2) 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 backward; the condition $\dot{c} \leq 1$ guarantees that the trajectory is not travelled with a velocity higher than the nominal one;
- Eq. (43.3) constraints the minimum and maximum 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. (43.4) handles the human-robot safety according to problem in Section 3.1. The presence of 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. (43.3).

Concerning the objective function in Eq. (43.1), it aims at finding inputs $\boldsymbol{u}_\xi = \begin{bmatrix} u_c & \boldsymbol{u}_\alpha^{\mathrm{T}} & \boldsymbol{u}_n^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ closest to $\boldsymbol{u}_\xi^d = \begin{bmatrix} u_c & \boldsymbol{u}_\alpha^{d\,\mathrm{T}} & \boldsymbol{u}_n^{d\,\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, where:

$$\begin{cases} u_c^d & = 1, \\ \boldsymbol{u}_\alpha^d & = \boldsymbol{0}_4, \\ \boldsymbol{u}_n^d & = k_n \dfrac{\partial F}{\partial \boldsymbol{q}}, \end{cases} \tag{44}$$

with $k_n > 0$. Having $u_c = u_c^d$ implies $\dot{c} = 1$, i.e., the cooperative task executed at nominal speed, having $\boldsymbol{u}_\alpha = \boldsymbol{u}_\alpha^d$ implies $\Delta\boldsymbol{\varepsilon} = \boldsymbol{0}_{12}$, 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 $\boldsymbol{Q}_\xi \in \mathbb{R}^{(n+5)\times(n+5)}$, it 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 its scaling.

After having solved the QP problem in Eq. (43), the Safety Planner makes use of its solutions in Eq.(23) to build the reference trajectory $\boldsymbol{\sigma}^s(t)(\boldsymbol{\nu}^s(t))$. Figure 3 shows the block scheme of the proposed Safety Planner, while Algorithm 1 describes the required algorithmic steps.

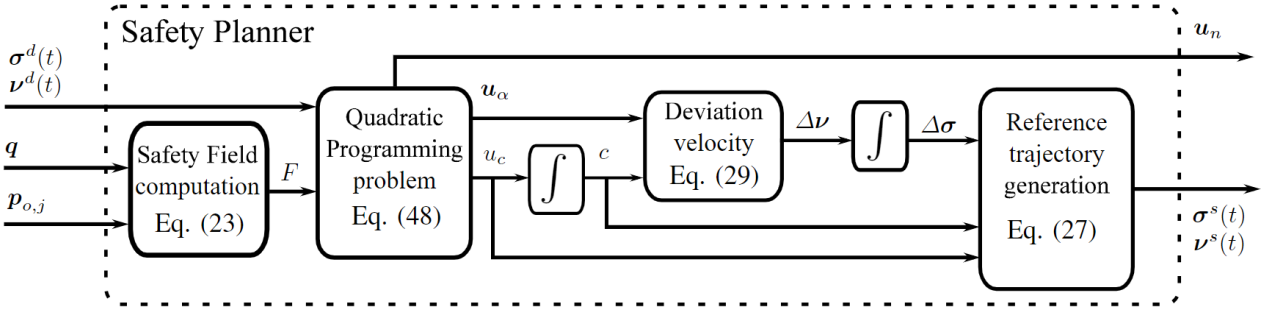

Figure 3: Safety Planner functional architecture. The inputs are the nominal trajectory, the robot configuration and the position of the human. After the computation of the current Safety Field value, the QP problem is solved, and its output is employed for modifying the nominal trajectory in order to obtain the reference trajectory that is finally sent to the controller.

Algorithm 1: Safety Planner algorithm

Data:
- Nominal trajectory $\sigma^d(\nu^d)$
- Joint position vector q
- Human point positions $p_{o,1}, \cdots, p_{o,n_o}$

Result:
- Safe trajectory $\sigma^s(\nu^s)$
- Null-space velocity vector u_n

begin

$F = \texttt{SafetyField}\ (q, p_{o,1}, \cdots, p_{o,n_o})$

$\begin{bmatrix} u_c \\ u_\alpha \\ u_n \end{bmatrix} = \texttt{SolveQP}\ (\bar{F}, \sigma^d, \nu^d)$

$c = \texttt{Integration}\ (u_c)$

$\Delta\nu = \texttt{DeviationVelocity}\ (u_\alpha, c, u_c)$

$\Delta\sigma = \texttt{Integration}\ (\Delta\nu)$

$\begin{bmatrix} \sigma^s \\ \nu^s \end{bmatrix} = \texttt{SafeTrajectory}\ (\Delta\sigma, \Delta\nu, c)$

return σ^s, ν^s, u_n

end

Algorithm 1: Overall algorithm describing the trajectory modification for ensuring human safety.

In detail, at first, the Safety field value is computed by resorting to Eq.(19) ; then, the QP problem in Eq.(43) is solved having in input the computed safety field value 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 u_α are used in Eq. (25) to compute the deviation velocity $\Delta\nu$, which is then integrated over time to obtain $\Delta\sigma$;
- Finally, $\Delta\sigma$, c and u_c are used in Eq. (23) to compute the reference trajectory $\sigma^s(\nu^s)$ that is sent to the specific employed controller, together with the vector $u_n = \dot{q}_n$.

At this point, the output of the Safety Planner is included in the HQP low-level controller in order to take into account also the higher-priority tasks in the hierarchy. In detail, the reference trajectory tracking is performed by properly using ν^s and σ^s in the expression of b_σ and by specifying the internal motion as $\dot{q}_n$. 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 $\dot{q}_n$, since other constraints relative to higher-priority tasks can be activated.

4 Validation of the proposed approach

In this section, we show experimental results obtained both within an indoor 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 performing a harvesting operation with only one end effector. In all the experiments the inputs for the path deviation in terms of orientations are set to zero ($\boldsymbol{\psi}_{a_o} = \mathbf{0}_3$ and $\boldsymbol{\psi}_{r_o} = \mathbf{0}_3$ in Eq. (26)), meaning that the orientations will not be modified by the Safety Planner, while the deviations in terms of positions will be specified in the descriptions of the specific experiments.

Regarding the human points position estimation, we made use of OpenPose framework (Sheikh, 2019), which is a popular software that allows tracking the human skeleton on 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 inside the head of the Farming Robot. Then, regarding the safety field value computation, we consider only the point representing the chest of the human to decrease the computational burden, and the function χ in Eq. (16) is set as d. Figure 4 shows, as an example, the robot and the human skeleton recognized by OpenPose in Rviz environment, highlighting with a blue sphere the human point P_O taken into consideration for the experiments presented in this section.

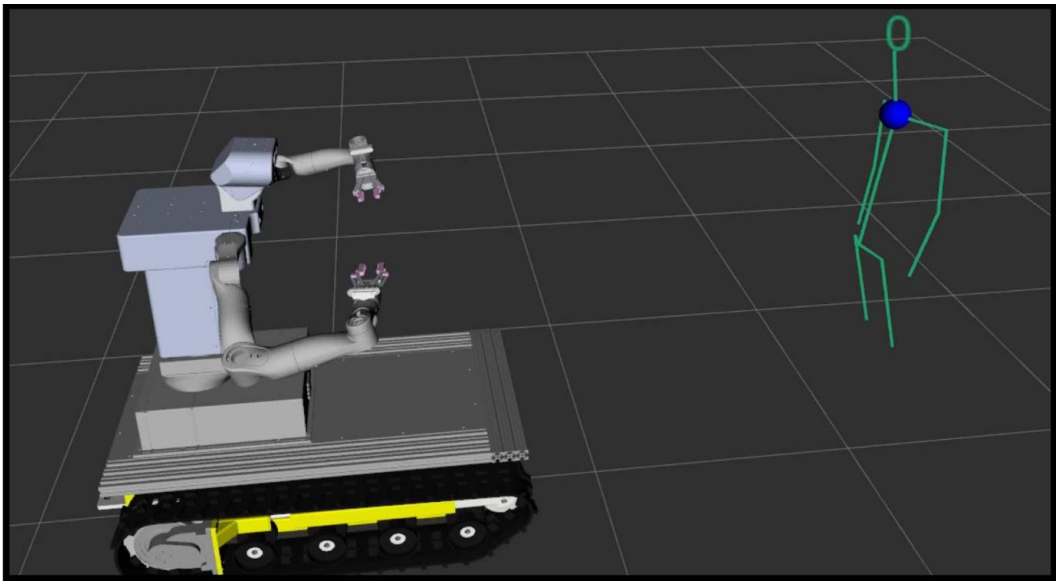

Figure 4: Representation of the robot and the human skeleton in Rviz environment. The blue sphere represents the human point taken into consideration for the safety field value computation.

All the experiments described in this Section are included in the video provided in Section 5.

4.1 Laboratory experiments

Here we show two indoor experiments in which the two end effectors are asked to perform a cooperative motion in order to track a nominal circular trajectory with the absolute frame. Figure 5 shows the laboratory conditions adopted for these experiments. In both of them the motion is chosen in order to be performed in a symmetrical manner, thus setting $\beta = 0.5$ and $\gamma = 1$ in Eq.(11). 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 $\boldsymbol{u}_\alpha$, in order to highlight both the effects in the nominal trajectory scaling and modification separately. The desired nominal absolute and relative poses have a first initialization phase in which the absolute position is asked to reach:

$$\boldsymbol{\sigma}_{a_p}^d = \boldsymbol{\sigma}_{a_p,0} + [0.2 \quad 0 \quad 0]^{\mathrm{T}}$$

with a trapezoidal velocity profile, where $\boldsymbol{\sigma}_{a_p,0}$ is the initial position of the absolute frame. The absolute frame orientation is kept at the initial value, while the relative position between the two end effectors is asked to reach:

$$\boldsymbol{\sigma}_{r_p}^{d} = [0 \quad -0.15 \quad 0]^{\mathrm{T}}$$

keeping the relative orientation constant at its initial value. After this initialization phase, the relative pose is asked to be kept constant, while the absolute position is asked to follow a nominal circular trajectory with a radius of 0.16m on the xy plane to be followed with a trapezoidal velocity profile in a periodic manner. In both the 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 the experiments the input $\boldsymbol{\zeta}_{r_p}$ is set to $\mathbf{0}_3$, meaning that the relative position between the two end effectors is not modified by the Safety Planner, while the input related to the absolute position deviation is set to:

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

where $\hat{\boldsymbol{n}}_o^a$ is the unit vector connecting the considered human point and the absolute frame position vectors.



Figure 5: Two snapshots of the Farming Robot performing a cooperative motion in laboratory environment. On the left, the operator is far enough from the robot such as the safety field is above the considered threshold. On the right, the operator gets close to the robot making the robot modify the nominal trajectory.

In the first experiment, we adjusted the value in the Q_ξ matrix in order to give more importance to the virtual input $u_{\alpha_{a_p}}$ with respect to u_c, i.e., asking to deviate more from the nominal trajectory rather than scale it down for keeping the safety field value above the minimum threshold. Figure 6 shows the obtained results.

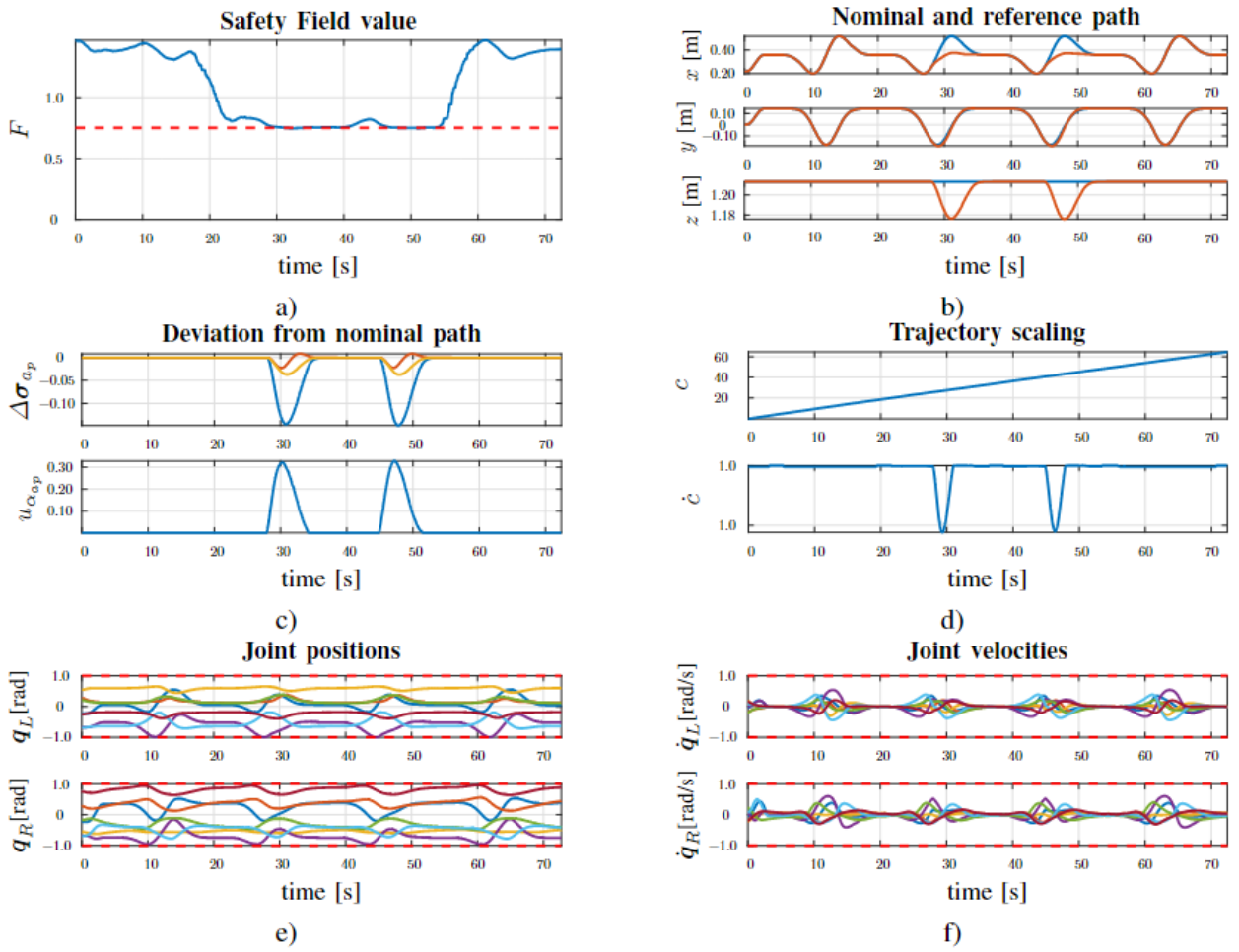

Figure 6: First experiment, deviating more than scaling. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line). The safety field value stays above the minimum threshold during the experiment; b) Nominal (blue) and safe (red) trajectories for the absolute frame. The 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 path deviation input. The value resulting from the QP problem is used to compute the deviation from the nominal trajectory. d) Parameter c and virtual scaling input. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds. The value of the scaling input resulting from the QP problem is used to scale down the nominal trajectory. In this case this virtual input is weighted more than the path deviation input, and consequently, the nominal trajectory is not scaled down much.

Figure 6.a) reports the evolution of the safety field value over time (solid blue line) during the experiment, compared to the imposed minimum threshold $\underline{\mathrm{F}}$ (horizontal red-dashed line). In the first 22 seconds of the experiment, the human operator stays far from the robot while it performs the initialization movement and the first circular trajectory; then, he gets closer to the robot and the safety field value decreases. When the robot performs for the second time the nominal circular trajectory, the safety field value gets closer to the imposed threshold, triggering the activation of the constraints in the QP problem. 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 6.b) and Figure 6.c). Specifically, the former shows

both the nominal (in blue) and safe (in red) trajectories, while the latter shows the computed virtual input $u_{\alpha_{ap}}$ (bottom part) together with the corresponding $\Delta\boldsymbol{\sigma}$ (top part) that quantifies the deviation from the nominal path. At the same time, Figure 6.d) shows the evolution of the virtual input u_c (bottom part) and the corresponding $c(t)$ 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 6.e) and Figure 6.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. The robot performs twice the circular movement while the human operator stays still; then he steps back increasing the safety field value and the robot is free to follow the nominal trajectory. During the entire experiment (the same applies to the following ones), the Safety Planner computes the virtual input $\dot{q}_n$ in order to generate internal motions aimed at increasing the safety field value. This input is not graphically reported here, but it is evident in the accompanying video and in Figure 7, in which the robot lowers the elbow while following the safe trajectory with the end effector to increase the distance from the human operator.

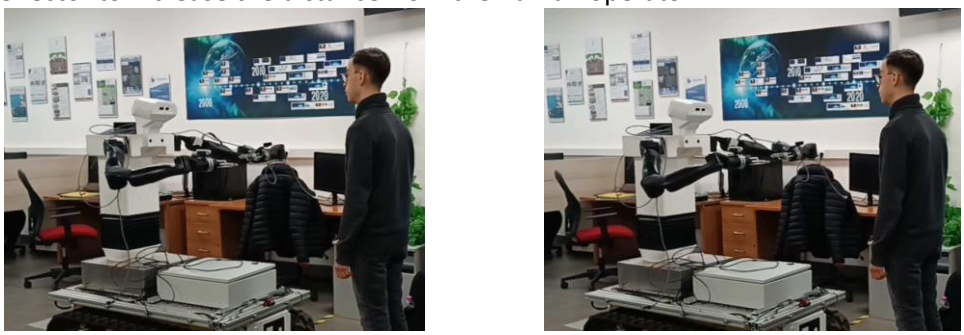

Figure 7: two snapshots of the robot motion showing the elbow adjustment due to the human presence.

In the second experiment, we chose to give more importance to the input u_c with respect to $u_{\alpha_{a_p}}$, in order to validate the scaling procedure. The results are reported in Figure 8.

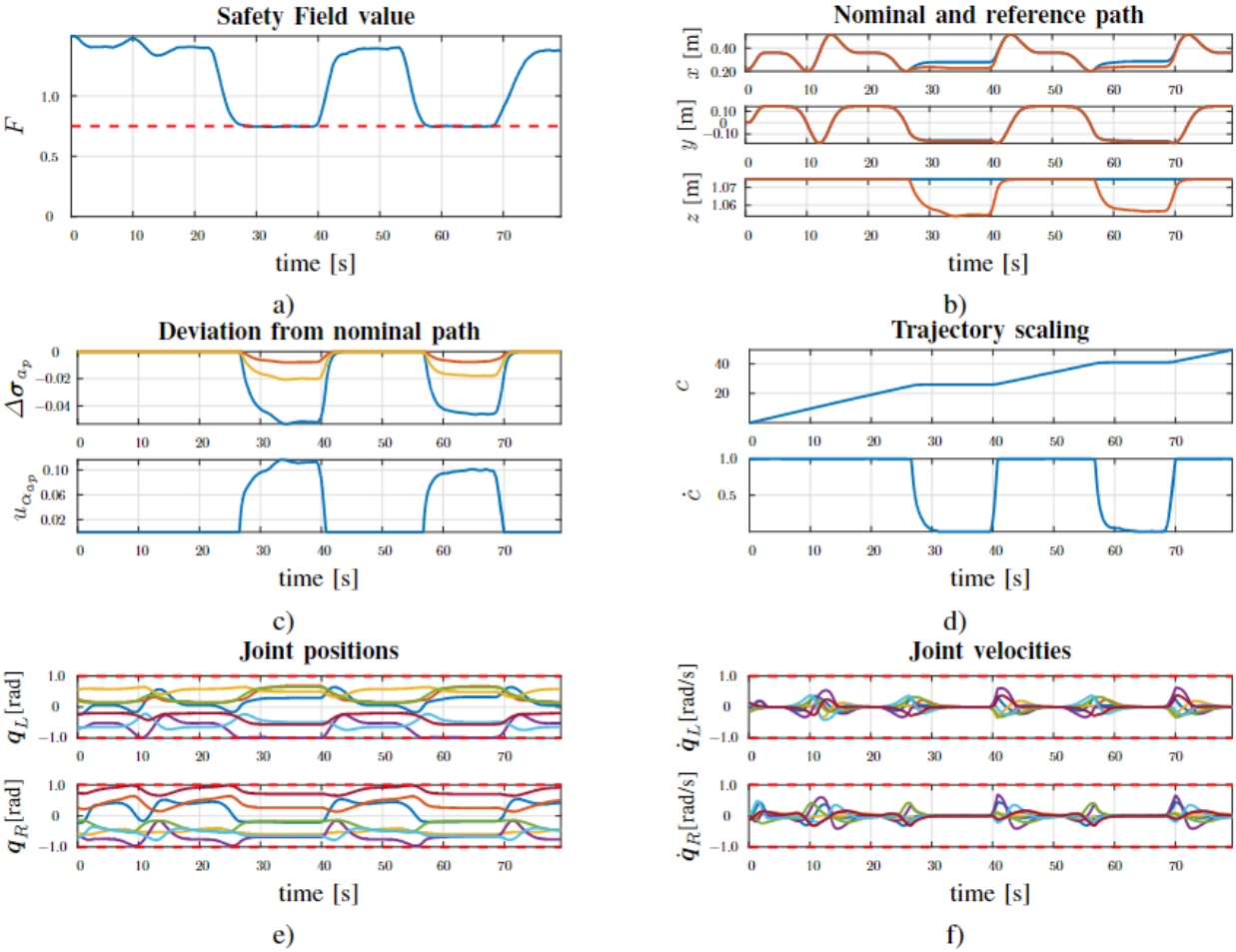

Figure 8: Second experiment, scaling input weighted less that path deviation input. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line). The safety field value stays above the minimum threshold during the experiment; b)

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 path deviation input. The value of the path deviation input is smaller with respect to the previous experiment given the chosen relative weights. d) Parameter c and virtual scaling input. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds. The value of the scaling input is used to scale down the nominal trajectory. In this case, this virtual input is weighted less than the path deviation input, thus, the trajectory is scaled down until completely stopping it.

Even in this case, looking at Figure 8.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 8.d) where $\dot{c}$ decreases its value until completely stopping the nominal trajectory ($\dot{c} = 0$ at $t = 27$s). Figure 8.b) and Figure 8.c) show that, in this case, the deviation from the nominal path is smaller with respect to the previous experiment, given the chosen relative weights, while Figure 8.e) and Figure 8.f) show that the joint kinematic constraints are still respected. After about 10s, the human operator steps back in order to increase the safety field value, allowing the robot to resume the tracking of the nominal trajectory ($\dot{c} = 1$ at $t = 40$ s). The same happens while the robot performs another iteration of the movement.

4.2 Outdoor experiments

In the following, we report the results obtained during harvesting experiments conducted in Aprilia vineyard in the February 2023, with no vegetation and fake bunches, and in September 2023 in realistic conditions. Figure 9 shows some snapshots of the experiment and the experimental conditions in the February campaign, while Figure 10 shows some snapshots of one of the experiments performed in the September 2023 campaign. In particular, we provide results relative to harvesting operations performed with a single arm. It is worth noticing that harvesting operations have been also demonstrated in dual-arm mode, as can be seen in the accompanying video in Section 5.



Figure 9. Snapshots of human-robot safety experiments in the real field in the February experimental campaign with robot performing a single-arm harvesting operation. Top Left: results of the grape perception by UPC for the planning of the nominal agronomic trajectories. Top Right: the Farming Robot starts the agronomic harvesting while the operator is far away from the robot. Bottom Left: the operator get close to the robot while it grasps the harvested grape. Bottom Right: the operator moves away from the robot that releases the grape in the box.



Figure 10: Snapshots of human-robot safety experiments in the real field with robot performing a single-arm harvesting operation in the September 2023 experimental campaign. Top figures shows the robot performing the grape harvesting without the human intervention. Bottom figures show the human approaching the harvested grape (and the robot) to taste it.

The position of the peduncle to cut has been obtained in real-time by the perception software developed by UPC and UNIROMA1 (see Deliverable 4.1), 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. (11) are set as $\beta = 1$ and $\gamma = 0$ in order 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:

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

where $\hat{\boldsymbol{n}}_o^r$ is the unit vector connecting the considered human point and the right end effector position vectors.

The Trajectory Generation module in Figure 2 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 in the following, 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 of the corresponding constraint. The relative weight between the two inputs is chosen in order to make use of both of them, and the results are reported in Figure 11.

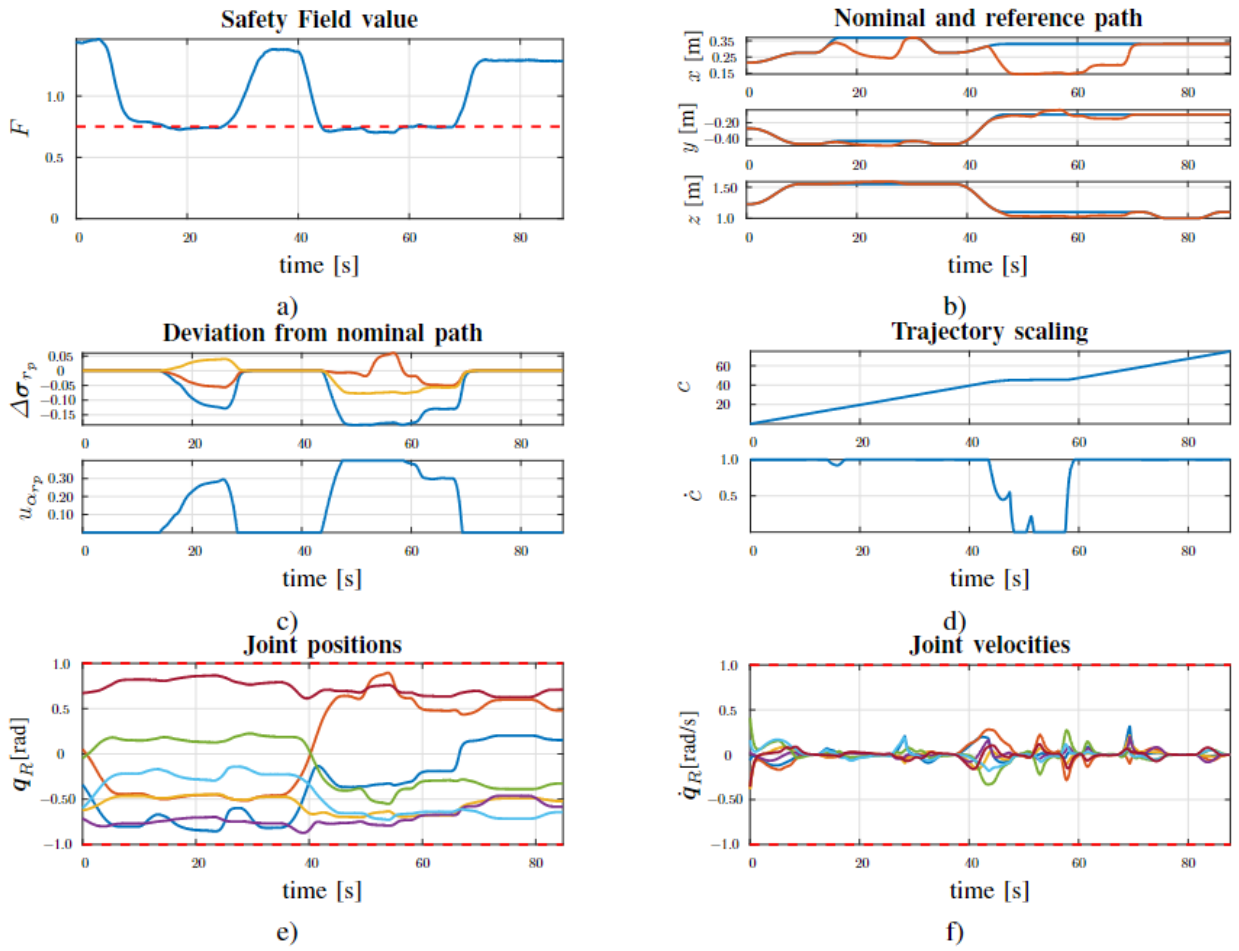

Figure 11: Field experiment. a) Safety field value over time and imposed minimum threshold (red-dashed horizontal line); b) Nominal (blue) and safe (red) trajectories for the absolute frame. The nominal trajectory is composed of a sequence of waypoints suitable for performing a harvesting operation. 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 path deviation input. d) Parameter c and virtual scaling input. e) Normalized joint position with the imposed thresholds. f) Normalized joint velocities with the imposed thresholds.

Figure 11.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 reach the proximity of the minimum threshold. This results mostly in a deviation from the nominal trajectory (Figure 11.b)-Figure 11.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 $u_{\alpha_{r_p}}$ 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. Figure 11.e) and Figure 11.f) demonstrate that the joint limits are respected during the entire experiment. Finally, at $t = 69$ s, the human operator steps back allowing the robot to finish the harvesting operation.

5 Reference to Web Content

Title	Description	URL
Experiments_Representative_Conditions.mp4	Videos of the experiments described in Section 4 in representative conditions	Link
Experiments_Realistic_Conditions.mp4	Videos of the experiments described in Section 4 in realistic conditions	Link

6 References

Id	Title	Version	Date
[D2.1]	Requirements, Specifications and Benchmarks	1.0	2021-04-30
[D2.2]	Specifications and KPIs for the two CANOPIES robot prototypes	1.0	2021-06-30
[D2.3]	Dimensioning of the Real-World Scenario for Final Demo	1.0	2021-10-29
[D2.4]	VR Farming Environment Specification	1.0	2021-10-29
[D3.2]	First Release of Farming and Logistic Robot Prototypes	1.1	2023-01-24
[D4.1]	Agronomic target modelling, detection and localization	3.0	2023-06-30
[D5.1]	Cooperative arms and mobile base functionalities	1.0	2023-02-28

This project is funded by the European Union G.A No 101016906

7 Bibliography

Ames, A., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., & Tabuada, P. (2019). Control Barrier Functions: Theory and Applications. *European Control Conference.*

Basile, Caccavale, Chiacchio, Coppola, & Marino. (2013). A decentralized kinematic control architecture for collaborative and cooperative multi-arm systems. *Mechatronics*.

Basso, E., & Pettersen, K. (2020). Task-Priority Control of Redundant Robotic Systems using Control Lyapunov and Control Barrier Function based Quadratic Programs. *IFAC World Congress.*

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*.

Di Lillo, P., Arrichiello, F., Antonelli, G., & Chiaverini, S. (2018). Safety-related tasks within the set-based task-priority inverse kinematics framework. *IEEE International Conference on Intelligent Robots and Systems (IROS).*

Escande, A., Mandard, N., & Wieber, P. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *International Journal of Robotics Research*.

Kim, S., Jang, K., Park, S., Lee, Y., Lee, S., & Park, J. (2019). Continuous task transition approach for robot controller based on hierarchical quadratic programming. *IEEE Robotics and Automation Letters*.

Lippi, M., & Marino, A. (2020). Human Multi-Robot Safe Interaction: A Trajectory Scaling Approach Based on Safety Assessment. *IEEE Trans. Control Syst. Technol.*

Park, A., & Lee, G. (s.d.). Extended cooperative task space for manipulation tasks of humanoid robots. *IEEE International Conference on Robotics and Automation.*

Sheikh, Z. C. (2019). OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 172-186.

Siciliano, B., & Slotine, J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. *Fifth International Conference on Advanced Robotics (ICAR).*