Ref. Ares(2023)1460291 - 28/02/2023

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 D5.1 – V1

Cooperative arms and mobile base functionalities

Dissemination level	PUBLIC
Submission Date	2023-02-28
Work Package	WP5
Task	T5:1
Type	Report
Version	1
Author	UNICLAM
Approved by	DTI, PAL

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 (FR) manipulation functionalities. In particular, the developed manipulation algorithms represent the low-level built-in motion control functionalities of the farming robot for controlling the bimanual systems both in an uncoordinated and coordinated way to perform harvesting and pruning operations, while taking into account kinematic and obstacle constraints and system redundancy.

Simulation and experimental results, the description of which is given in Section 4, can be found at the following link:

https://tinyurl.com/canopies-d5-1-videos

Table of Content

1 Dual-arm farming robot 6
1.1 Objectives 6
1.2 Farming Robot Kinematics 6
1.3 Tasks description 8
2 Farming Robot Kinematic Control 11
2.1 Intermediate-Level Control 11
2.2 High-Level Control 12
2.2.1 Set-Based Closed-Loop-Inverse Kinematics 12
2.2.2 Hierarchical Quadratic Programming 14
2.3 Implemented tasks 15
2.3.1 Kinematic limits and collision tasks 15
2.3.1.1 Joint position limits 15
2.3.1.2 Joint velocity limits 16
2.3.1.3 Virtual walls 16
2.3.1.4 Self-collisions 17
2.3.2 Operational tasks 18
2.3.2.1 Uncoordinated motion 18
2.3.2.2 Coordinated motion 18
2.3.3 Optimization task 19
2.3.3.1 Swivel angle 19
2.4 Kinematic singularity handling 20
3 Validation 22
3.1 Simulations 22
3.1.1 Set-based CLIK validation on the Movo robot 22
3.1.2 HQP validation on the Farming Robot 24

3.1.2.1 Safety tasks validation 24
3.1.2.2 Operational tasks validation 29
3.1.2.3 Optimization task validation 31
3.2 Experiments 32
3.2.1 Set-Based CLIK algorithm on the Movo robot 32
3.2.2 HQP control framework on the Farming Robot 33
4 Reference to Web Content 37
5 References 38
6 Bibliography 38

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

Abbreviations and Acronyms

FR	Farming Robot
RGB-D camera	Red, green, blue and depth camera
ROS	Robot Operating System
HQP	Hierarchical-Quadratic-Programming
CLIK	Closed-Loop-Inverse-Kinematics
DoFs	Degrees of Freedom
DH	Denavit-Hartenberg

1 Dual-arm farming robot

1.1 Objectives

One of the objectives of the CANOPIES project (*O2.1*) is to devise a prototype of farming robot (FR) able to perform agronomic operations, like autonomous or collaborative grape harvesting and pruning. To achieve this objective, the modeling and planning of manipulator arms is required, while considering all constraints the robot is subject to. In this deliverable, we provide a comprehensive description of the manipulation algorithms we developed, together with their experimental validations both in simulation and in real field.

More in detail, constrained optimal inverse kinematic functionalities were implemented for the design of kinematic control strategies for uncoordinated and coordinated arm functionalities starting from representative task variables; constraints consider, for example, self-collision, environment obstacles, dynamic and kinematic limits; to the scope, Set-Based Closed-Loop-Inverse-Kinematics (Di Lillo, Arrichiello, Antonelli, & Chiaverini, 2018) and Hierarchical Quadratic Programming (Escande , Mansard, & Wieber, 2014) approaches were investigated to handle inequalities constraints for the mobile robot at hand. These kinds of constraints appear, for example, in the case of joint configuration and velocity limits, and the proposed strategies handle any number of tasks arranged in priorities. Furthermore, the overall framework can handle the online activation and deactivation of extra degrees of freedom, like the lift and yaw joints of the torso depending on the context, the case of both arms moving independently or cooperatively and the online switching from one modality to the other according to the robot task.

1.2 Farming Robot Kinematics

The farming robot is composed of a non-holonomic mobile base and an upper body with a torso with 2 DoFs (rotational yaw and prismatic lift joints) and two manipulators made each one of 7 rotational joints.

Figure 1 shows the farming robot structures and frames of interest for the overall robotic platform, namely the world frame $\mathcal{W}$, the base frame $\mathcal{B}$, the right end-effector frame $\mathcal{E}_2$ and left end-effector frame $\mathcal{E}_1$, for which it holds:

$$\begin{cases} \boldsymbol{p}_{\mathcal{E}_i}^{\mathcal{W}} = \boldsymbol{p}_{\mathcal{B}}^{\mathcal{W}} + \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}} \boldsymbol{p}_{\mathcal{E}_i}^{\mathcal{B}} \\ \boldsymbol{R}_{\mathcal{E}_i}^{\mathcal{W}} = \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}} \boldsymbol{R}_{\mathcal{E}_i}^{\mathcal{B}} \end{cases} \quad i = 1, 2 \tag{1}$$

where $\boldsymbol{p}_{\mathcal{B}}^{\mathcal{W}}$ is the vector that describes the position of the mobile platform with respect to the world frame $\mathcal{W}$, and $\boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}$ is the rotation matrix that describes the orientation of the mobile platform with respect to the same frame.

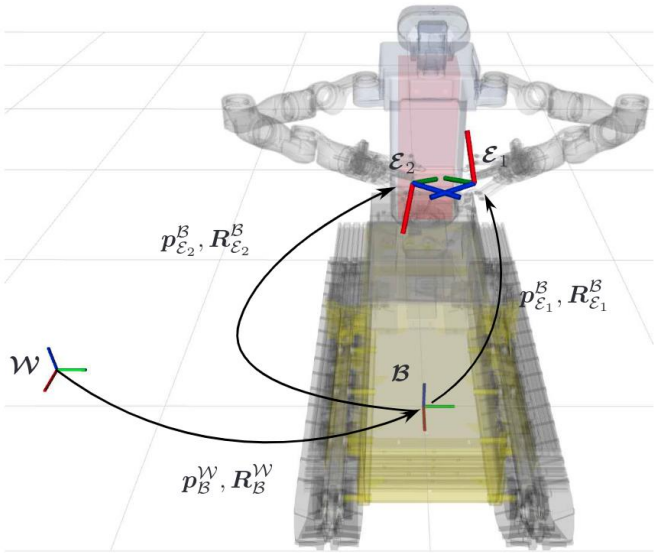
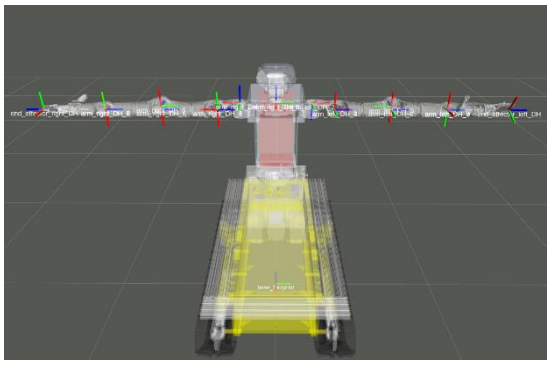


Figure 1: Frames of interest for the robotic platform

Let us define the system state $\boldsymbol{q} = [\boldsymbol{p}_{\mathcal{B}}^T \quad \phi_{\mathcal{B}} \quad \boldsymbol{q}_T^T \quad \boldsymbol{q}_L^T \quad \boldsymbol{q}_R^T]^T$ as the vector containing the position and the orientation of the base ($\boldsymbol{p}_{\mathcal{B}}$ and $\phi_{\mathcal{B}}$, respectively), torso, left arm and right arm joints ($\boldsymbol{q}_T$, $\boldsymbol{q}_L$ and $\boldsymbol{q}_R$, respectively), and the system velocity vector as $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{p}}_{\mathcal{B}}^T \quad \dot{\phi}_{\mathcal{B}}^T \quad \dot{\boldsymbol{q}}_T^T \quad \dot{\boldsymbol{q}}_L^T \quad \dot{\boldsymbol{q}}_R^T]^T$, where the first two components are the linear and angular velocities of the mobile base. Furthermore, we introduce the vector $\boldsymbol{q}_{UB} = [\boldsymbol{q}_T^T \quad \boldsymbol{q}_L^T \quad \boldsymbol{q}_R^T]^T$, which gathers the joints of the upper body of the robot. In order to derive the forward kinematics of the robot, which relates system state $\boldsymbol{q}$ to the position and the orientation of the two end-effectors, the standard Denavit-Hartenberg procedure was applied. Figure 2 shows the DH frames on the left, and the corresponding DH parameters on the



Joint	a	alpha	d	theta
1	0.181	3.14	0.0	0.0
2	0.203	3.14	0.0	1.57
3	0.13	-1.57	0.04	0.0
4	0.0	1.57	0.0	0.0
5	0.02	-1.57	0.32	0.0
6	0.02	-1.57	0.0	0.0
7	0.0	1.57	0.32	0.0
8	0.0	1.57	0.0	0.0
9	0.0	0.0	0.0	0.0

Joint	a	alpha	d	theta
1	0.181	3.14	0.0	0.0
2	0.203	0.0	0.0	-1.57
3	0.13	1.57	-0.04	0.0
4	0.0	1.57	0.0	0.0
5	0.02	1.57	-0.32	0.0
6	0.02	1.57	0.0	0.0
7	0.0	1.57	-0.32	0.0
8	0.0	1.57	0.0	0.0
9	0.0	0.0	0.0	0.0

Figure 2: Left: The Farming Robot in the home configuration. Center: DH table of the left arm. Right: DH table of the right arm. Angles are in radians and length in meters.

right of the farming robot prototype.

Concerning the arms differential kinematics, the velocity of the left end-effector

$$\boldsymbol{v}_{\mathcal{E}_1}^{\mathcal{W}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{\mathcal{E}_1}^{\mathcal{W}^T} & \boldsymbol{\omega}_{\mathcal{E}_1}^{\mathcal{W}^T} \end{bmatrix}^T$$

with $\dot{\boldsymbol{p}}_{\mathcal{E}_1}^{\mathcal{W}}$ and $\boldsymbol{\omega}_{\mathcal{E}_1}^{\mathcal{W}}$ the linear and angular velocity, respectively, can be expressed in virtue of Eq. (1) as:

$$\boldsymbol{v}_{\mathcal{E}_1}^{\mathcal{W}} = \boldsymbol{J}_{\gamma_1}\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & -\boldsymbol{S}(\boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{p}_{\mathcal{E}_1}^{\mathcal{B}}) & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_1,p_T}^{\mathcal{B}}(\boldsymbol{q}_T) & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_1,p_L}^{\mathcal{B}}(\boldsymbol{q}_L) & \boldsymbol{0}_{3\times7} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_1,o_T}^{\mathcal{B}}(\boldsymbol{q}_T) & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_1,o_L}^{\mathcal{B}}(\boldsymbol{q}_L) & \boldsymbol{0}_{3\times7} \end{bmatrix}\dot{\boldsymbol{q}}$$

where $\boldsymbol{S}(\cdot)$ is the skew symmetric matrix operator and:

- $\boldsymbol{J}_{\mathcal{E}_1,p_T}^{\mathcal{B}}(\boldsymbol{q}_T)$, $\boldsymbol{J}_{\mathcal{E}_1,p_L}^{\mathcal{B}}(\boldsymbol{q}_L)$ are the Jacobians relating torso and left arm velocities, respectively, to the linear velocity of left end-effector;
- $\boldsymbol{J}_{\mathcal{E}_1,o_T}^{\mathcal{B}}(\boldsymbol{q}_T)$, $\boldsymbol{J}_{\mathcal{E}_1,o_L}^{\mathcal{B}}(\boldsymbol{q}_L)$ are the Jacobians relating torso and left arm velocities, respectively, to the angular velocity of left end-effector.

In the same way, the velocity of the right end-effector

$$\boldsymbol{v}_{\mathcal{E}_2}^{\mathcal{W}} = \begin{bmatrix} \dot{\boldsymbol{p}}_{\mathcal{E}_2}^{\mathcal{W}^T} & \boldsymbol{\omega}_{\mathcal{E}_2}^{\mathcal{W}^T} \end{bmatrix}^T$$

can be expressed as:

$$\boldsymbol{v}_{\mathcal{E}_2}^{\mathcal{W}} = \boldsymbol{J}_{\gamma_2}\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & -\boldsymbol{S}(\boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{p}_{\mathcal{E}_2}^{\mathcal{B}}) & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_2,p_T}^{\mathcal{B}}(\boldsymbol{q}_T) & \boldsymbol{0}_{3\times7} & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_2,p_R}^{\mathcal{B}}(\boldsymbol{q}_R) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_2,o_T}^{\mathcal{B}}(\boldsymbol{q}_T) & \boldsymbol{0}_{3\times7} & \boldsymbol{R}_{\mathcal{B}}^{\mathcal{W}}\boldsymbol{J}_{\mathcal{E}_2,o_R}^{\mathcal{B}}(\boldsymbol{q}_R) \end{bmatrix}\dot{\boldsymbol{q}}$$

By stacking together the vectors $\boldsymbol{v}_{\mathcal{E}_1}^{\mathcal{W}}$ and $\boldsymbol{v}_{\mathcal{E}_2}^{\mathcal{W}}$, it holds

$$\boldsymbol{v}^{\mathcal{W}} = \begin{bmatrix} \boldsymbol{v}_{\mathcal{E}_1}^{\mathcal{W}} \\ \boldsymbol{v}_{\mathcal{E}_2}^{\mathcal{W}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{\gamma_1} \\ \boldsymbol{J}_{\gamma_2} \end{bmatrix}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\gamma}\dot{\boldsymbol{q}} \tag{2}$$

that represents the differential kinematic relationships that relates both end-effector linear/angular velocities to the system velocity vector.

1.3 Tasks description

A generic task $\boldsymbol{\sigma}_x$ for the farming robot can be expressed as a function of the system state vector:

$$\boldsymbol{\sigma}_x = \boldsymbol{f}_x(\boldsymbol{p}_{\mathcal{B}}, \boldsymbol{\phi}_{\mathcal{B}}, \boldsymbol{q}_T, \boldsymbol{q}_L, \boldsymbol{q}_R)$$

The relationship between the system velocity and the task velocity can be expressed in terms of a Jacobian matrix:

$$\dot{\boldsymbol{\sigma}}_x = \boldsymbol{J}_x(\boldsymbol{q})\dot{\boldsymbol{q}}, \qquad \boldsymbol{J}_x = \begin{bmatrix} \overbrace{\frac{\partial \boldsymbol{f}_x(\boldsymbol{q})}{\partial \boldsymbol{p}_{\mathcal{B}}} \quad \frac{\partial \boldsymbol{f}_x(\boldsymbol{q})}{\partial \boldsymbol{\phi}_{\mathcal{B}}}}^{\text{base}} & \overbrace{\frac{\partial \boldsymbol{f}_x(\boldsymbol{q})}{\partial \boldsymbol{q}_{\mathrm{T}}}}^{\text{torso}} & \overbrace{\frac{\partial \boldsymbol{f}_x(\boldsymbol{q})}{\partial \boldsymbol{q}_{\mathrm{L}}}}^{\text{left arm}} & \overbrace{\frac{\partial \boldsymbol{f}_x(\boldsymbol{q})}{\partial \boldsymbol{q}_{\mathrm{R}}}}^{\text{right arm}} \end{bmatrix}$$

The columns of this matrix can be divided in blocks. Each block is related to a DoF of the farming robot (e.g., the first six are related to the mobile platform) and describes the existing relationship between the task velocity and the DoFs corresponding to that block. Such a decomposition is exploited in the following agronomic tasks in order to activate or deactivate some degrees of freedom according to the context. For instance, we might want to keep the base and/or the torso still in case there is a human operator close to the robot.

In the following, the considered tasks are described in terms of Jacobian matrices and control objectives. Aiming at giving more flexibility to the system in terms of performing the needed operations in different manners depending on the specific operational conditions, we have foreseen two kinds of task to control.

The first one is the "uncoordinated motion", in which the objective is to control the position and the orientation of the two end-effectors separately. The Jacobian of the task is the $\boldsymbol{J}_{\sigma}$ matrix that relates the linear/angular velocities of the two end-effectors with the system velocity, and it has been already defined in Eq. (2). The task value is the vector stacking the position and the orientation of the two end-effectors:

$$\boldsymbol{\sigma}_{\gamma} = \begin{bmatrix} \boldsymbol{\sigma}_{\gamma,\mathcal{E}_1} \\ \boldsymbol{\sigma}_{\gamma,\mathcal{E}_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{\mathcal{E}_1}^{\mathcal{W}} \\ \boldsymbol{\mathcal{Q}}_{\mathcal{E}_1}^{\mathcal{W}} \\ \boldsymbol{p}_{\mathcal{E}_2}^{\mathcal{W}} \\ \boldsymbol{\mathcal{Q}}_{\mathcal{E}_2}^{\mathcal{W}} \end{bmatrix} \tag{3}$$

where $\mathcal{Q}_{\mathcal{E}_1}^{\mathcal{W}}, \mathcal{Q}_{\mathcal{E}_2}^{\mathcal{W}}$ are the unit quaternions expressing the orientation of the left and right end-effectors, respectively.

The second task is the "coordinated motion", in which the objective is to cooperatively move both the end-effectors in order to control the pose of the "absolute frame" and the pose of the "relative frame" (Chiacchio, Chiaverini, & Siciliano, 1996) (Caccavale & Uchiyama, 2016). More in detail, let us define the absolute position and the absolute orientation as:

$$\boldsymbol{p}_a^{\mathcal{W}} = \alpha \boldsymbol{p}_{\mathcal{E}_1}^{\mathcal{W}} + (1-\alpha)\boldsymbol{p}_{\mathcal{E}_2}^{\mathcal{W}}$$

$$\mathbf{R}_a^{\mathcal{W}} = \mathbf{R}_{\mathcal{E}_1}^{\mathcal{W}} \mathbf{R}_{\mathcal{E}_1,\mathcal{E}_2}^{\mathcal{E}_1}((1-\alpha)\vartheta_{\mathcal{E}_1,\mathcal{E}_2});$$

$\boldsymbol{p}_a^{\mathcal{W}}$ can be seen as the position of a point between the two end-effectors (with $\alpha = 0$ it corresponds to the left end-effector position, while with $\alpha = 1$ it corresponds to the right end-effector position and $\alpha = 1/2$ represents the case the control point is exactly in between the two end-effectors). Similar reasonings apply to the orientation, considering that $\mathbf{R}_{\mathcal{E}_1,\mathcal{E}_2}^{\mathcal{E}_1}((1-\alpha)\vartheta_{\mathcal{E}_1,\mathcal{E}_2})$ is the relative orientation matrix expressing the orientation of the right end-effector with respect to the left one, parametrized with respect to the axis-angle representation (only the dependency from the angle $\vartheta_{\mathcal{E}_1,\mathcal{E}_2}$ is highlighted).

The relative position and the relative orientation are defined as:

$$\mathbf{p}_r^{\mathcal{W}} = \mathbf{p}_{\mathcal{E}_2}^{\mathcal{W}} - \mathbf{p}_{\mathcal{E}_1}^{\mathcal{W}}$$

$$\mathbf{R}_r^{\mathcal{E}_1} = \mathbf{R}_{\mathcal{E}_2}^{\mathcal{E}_1}$$

and they can be seen as the position/orientation of the right end-effector with respect to the left one. The corresponding task value can be expressed as:

$$\boldsymbol{\sigma}_\sigma = \begin{bmatrix} \boldsymbol{\sigma}_a \\ \boldsymbol{\sigma}_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_a^{\mathcal{W}} \\ \boldsymbol{\mathcal{Q}}_a^{\mathcal{W}} \\ \boldsymbol{p}_r^{\mathcal{W}} \\ \boldsymbol{\mathcal{Q}}_r^{\mathcal{E}_1} \end{bmatrix} \tag{4}$$

where $\boldsymbol{\mathcal{Q}}_a^{\mathcal{W}}$ and $\boldsymbol{\mathcal{Q}}_r^{\mathcal{E}_1}$ are the quaternions extracted by the rotation matrices $\boldsymbol{R}_a^{\mathcal{W}}$ and $\boldsymbol{R}_r^{\mathcal{E}_1}$, respectively. Figure 3 shows a graphical representation of the absolute frame with $\alpha = 0.5$ and of the relative frame.

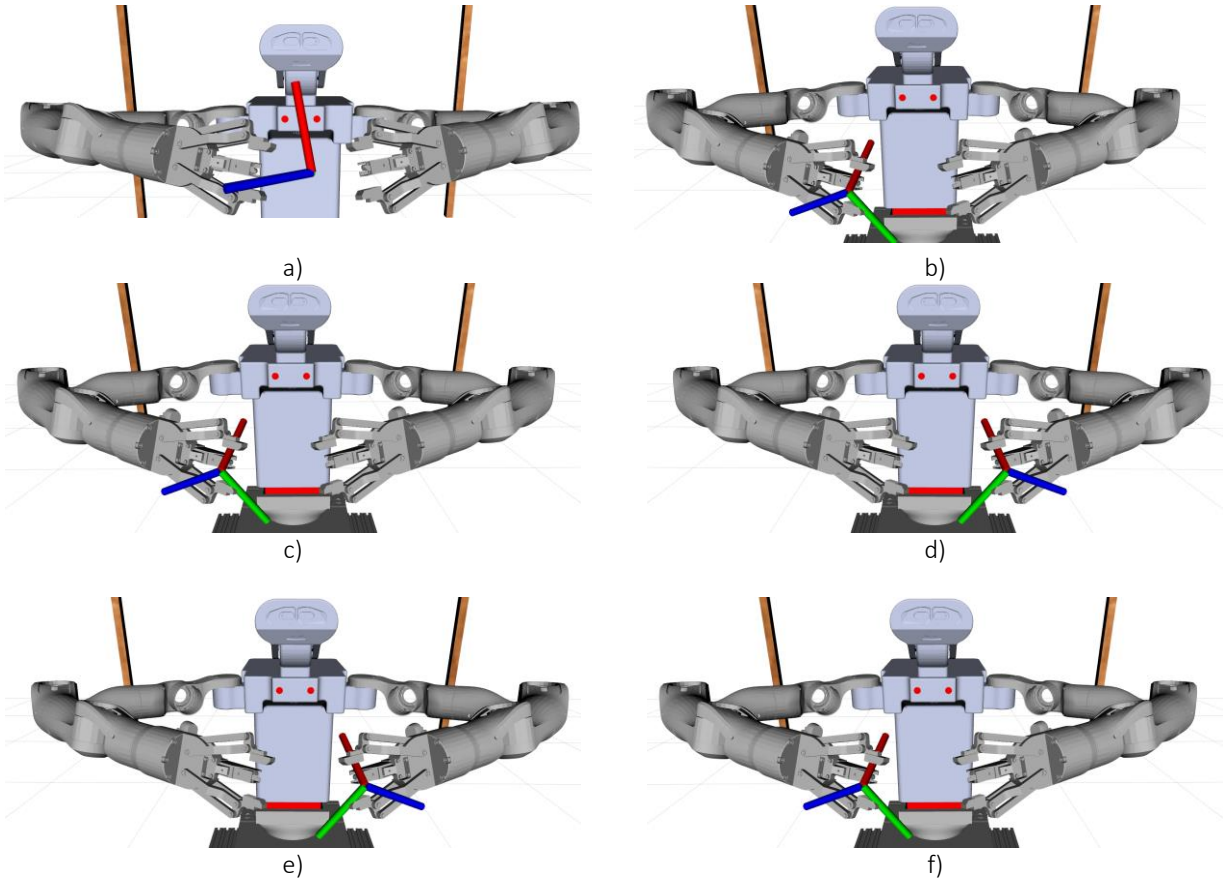


Figure 3: a) Graphical representation of the absolute frame ($\alpha = 0.5$); b) Graphical representation of the relative frame($\alpha = 0.5$); c) Graphical representation of the absolute frame ($\alpha = 0$); d) Graphical representation of the relative frame ($\alpha = 0$); e) Graphical representation of the absolute frame ($\alpha = 1$); f) Graphical representation of the relative frame ($\alpha = 1$).

Finally, the task Jacobian can be computed as (Almeida & Karayiannidis, 2019):

$$\boldsymbol{J}_\sigma = \begin{bmatrix} \alpha \boldsymbol{I}_3 & \boldsymbol{0_3} & (1-\alpha)\boldsymbol{I}_3 & \boldsymbol{0_3} \\ \boldsymbol{0_3} & \alpha \boldsymbol{I}_3 & \boldsymbol{0_3} & (1-\alpha)\boldsymbol{I}_3 \\ -\boldsymbol{I}_3 & \boldsymbol{0_3} & \boldsymbol{I}_3 & \boldsymbol{0_3} \\ \boldsymbol{0_3} & -\boldsymbol{I}_3 & \boldsymbol{0_3} & \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{J}_\gamma \tag{5}$$

The parameter α can be used for modulating motion between the two end-effectors, spanning from fully symmetric movements ($\alpha = 0.5$) to fully asymmetric ones ($\alpha = 0$ or $\alpha = 1$).

2 Farming Robot Kinematic Control

The upper body of the Farming Robot, made of the actuated torso, two arms and a mobile head, is provided with a built-in velocity interface which allows to send velocity reference commands at high frequency (100*Hz*) to the robot. Starting from the provided interface, a control scheme has been designed to implement the low-level control functionalities mentioned above as depicted in Figure 4. In detail,

- the *High Level Control* block is in charge of implementing the robot manipulation functionalities as described in Section 2.2. Based on the information coming from external sensors (e.g., cameras) and tasks to achieve, it generates desired joint velocities $\dot{\boldsymbol{q}}_d$ and configurations $\boldsymbol{q}_d$;
- these desired joint velocities and configurations are sent to *the Intermediate Control,* made of a controller *C* and a pre-filter *F* whose target is to achieve tracking of the reference coming from the *High Level Control*. The output of the *Intermediate-Control* is fed into the velocity controller provided by the robot.

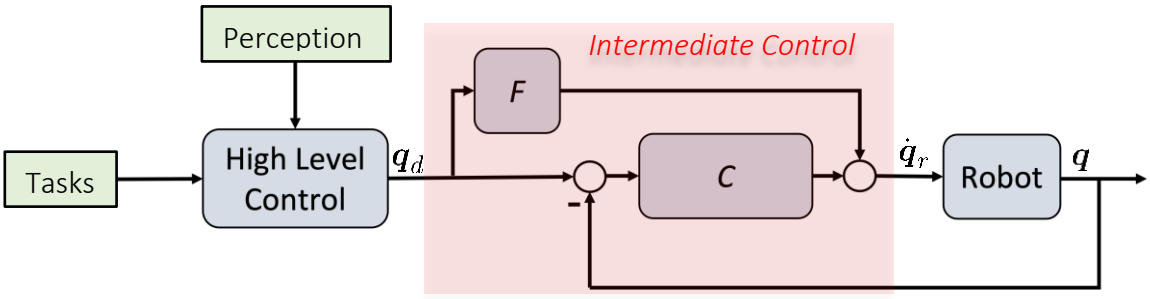


Figure 4: Block diagram showing the schematic of the robot control.

The idea underlying the proposed scheme is that the real joint vector $\boldsymbol{q}$ is not fed into the High Level Control in order to decouple the control and robot dynamics.

2.1 Intermediate-Level Control

With reference to the scheme in Figure 4, the Intermediate Control simply adopts a proportional action (*C*) plus a derivative one (*F*) leading to the following velocity reference for the low-level built-in control:

$$\dot{\boldsymbol{q}}_r = \dot{\boldsymbol{q}}_d + \boldsymbol{K}_q (\boldsymbol{q}_d - \boldsymbol{q}) \tag{6}$$

where $\boldsymbol{K}_q$ is a diagonal matrix gain. The simple structure of the adopted controller has showed good performance in terms of joint configuration tracking as highlighted by results reported in Section 3.2.2. Therefore, for the sake of notation simplicity, we assume $\dot{\boldsymbol{q}}_d \approx \dot{\boldsymbol{q}}$ in the following mathematical discussion.

2.2 High-Level Control

In the following, the control schemes investigated for providing the farming robot with all the manipulation functionalities are described. They are the Set-Based Closed-Loop-Inverse Kinematics and the Hierarchical Quadratic Programming detailed in Sections 2.2.1 and 2.2.2, respectively.

2.2.1 Set-Based Closed-Loop-Inverse Kinematics

Given a certain desired task value $\boldsymbol{\sigma}_{x,d}$, the system velocity vector that make the robot fulfil the task can be computed by resorting to the Closed-Loop Inverse Kinematics (CLIK) algorithm (Chiaverini, 1997):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_x^{\dagger}(\dot{\boldsymbol{\sigma}}_{x,d} + \boldsymbol{K}_x\tilde{\boldsymbol{\sigma}}_x) \tag{7}$$

where $\dot{\boldsymbol{\sigma}}_{x,d}$ is a desired task velocity to be given as a feed-forward term, $\tilde{\boldsymbol{\sigma}}_x = \boldsymbol{\sigma}_{x,d} - \boldsymbol{\sigma}_x$ is the task error, $\boldsymbol{K}_x$ is a positive-definite matrix of gains and $\boldsymbol{J}_x^{\dagger}$ is the Moore-Penrose pseudoinverse of the task Jacobian matrix.

When the number of DoFs is greater than ones strictly needed for the accomplishment of a certain task, the robot is said to be redundant with respect to that task, and the additional DoFs can be exploited to perform other control objectives simultaneously. To resolve the possible conflicts that might arise between the two tasks, a usual approach is to define a hierarchy, i.e., giving more importance to one task (which is called primary) over the other one (called secondary). The objective is to filter out the velocity contribution of the secondary task that would affect the execution of the primary one; in this way, the primary task is executed perfectly as long as it remains feasible, while the secondary one is executed at best. This is implemented by projecting the velocity contribution of the secondary task in the null space of the Jacobian matrix of the primary one, achieving the following solution:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_d + \boldsymbol{K}\tilde{\boldsymbol{\sigma}}) + \boldsymbol{N}\dot{\boldsymbol{q}}_0$$

where $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ is the null space projector and $\dot{\boldsymbol{q}}_0$ is an arbitrary vector that can be used to minimize or maximize a scalar value by setting:

$$\dot{\boldsymbol{q}}_0 = k_0 \left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^{\mathrm{T}}$$

where k_0 is a scalar gain and $w(\boldsymbol{q})$ is a secondary objective function.

Another possible exploitation is to define a second task with a specific desired value and compute the solution that accomplishes both the tasks at the same time. Unfortunately, this solution may not exist due to the infeasibility of their simultaneous resolution. Also in this case, it is thus necessary to define a priority between the tasks and compute the solution that achieves the primary one while minimizes the error on the secondary one, as:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_1 + (\boldsymbol{J}_2\boldsymbol{N}_1)^{\dagger}\ (\dot{\boldsymbol{\sigma}}_{2,d} + \boldsymbol{K}_2\tilde{\boldsymbol{\sigma}}_2 - \boldsymbol{J}_2\dot{\boldsymbol{q}}_1)$$

To safely control complex robotic system, several tasks are required to be performed simultaneously. Usually, they can be grouped into three categories (Di Lillo, Arrichiello, Antonelli, & Chiaverini, 2018):

1. Safety tasks
2. Operational tasks
3. Optimization tasks

To solve possible conflicts that may arise among tasks, a common approach is to assign different priority levels among them, obtaining a hierarchy (as shown in Figure 5).

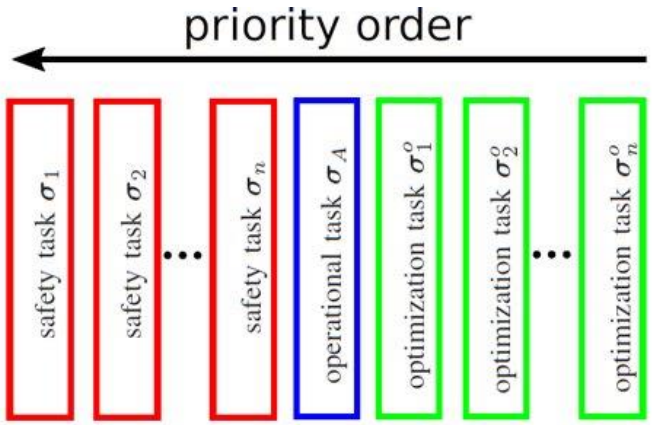


Figure 5: Example of hierarchy with tasks grouped into three categories. Priority is higher on the left and safety tasks have higher priority with respect to operational and optimization tasks.

For a hierarchy composed by h tasks, it is necessary to assign h levels of priority i, and sort them from the primary one ($i = 1$) to the lowest priority one ($i = h$). Given this assumption, the reference system velocity can be computed recursively as (Siciliano & Slotine, 1991):

$$\dot{\boldsymbol{q}}_h = \sum_{i=1}^{h} (\boldsymbol{J}_i \boldsymbol{N}_{i-1}^A)^\dagger (\dot{\boldsymbol{\sigma}}_{i,d} + \boldsymbol{K}_i \tilde{\boldsymbol{\sigma}}_i - \boldsymbol{J}_i \dot{\boldsymbol{q}}_{i-1}) \tag{8}$$

where $\boldsymbol{N}_0^A = \boldsymbol{I}, \dot{\boldsymbol{q}}_0 = \boldsymbol{0}$, $\dot{\boldsymbol{q}}_i$ is the reference system velocity that fulfils all the tasks from the first one to the i-th, and $\boldsymbol{N}_i^A$ is the null space projector of the augmented Jacobian matrix, obtained by stacking all the task Jacobian matrices from task 1 to i:

$$\boldsymbol{J}_i^A = \begin{bmatrix} \boldsymbol{J}_1^{\mathrm{T}} & \boldsymbol{J}_2^{\mathrm{T}} & \dots & \boldsymbol{J}_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

and it is defined as:

$$\boldsymbol{N}_i^A = \boldsymbol{I} - (\boldsymbol{J}_i^A)^\dagger \boldsymbol{J}_i^A$$

The control framework represented by Eq. (8), like all the main redundancy resolution methods, has been developed to handle equality-based tasks, i.e., control objectives in which the goal is to bring the task value to a specific one (e.g., moving the arm end-effector to a target position). However, several control objectives may require their value to lie in an interval, i.e., above a lower threshold and below an upper threshold. These are usually called set-based tasks or inequality constraints in literature. Classical task-priority frameworks have been extended to handle set-based tasks (Simetti & Casalino, 2016) (Moe, Antonelli, Teel, Pettersen, & Schrimpf, 2016). The key idea is that a set-based task can be seen as a mono-dimensional equality-based one which gets active or inactive depending on its current value. It is necessary to set different reference values for each set-based task: physical

thresholds σ_M (σ_m), safety thresholds $\sigma_{s,u} < \sigma_M$ $(\sigma_{s,l} > \sigma_m)$, and activation thresholds $\sigma_{a,u} = \sigma_{s,u} - \varepsilon$ $(\sigma_{a,l} = \sigma_{s,l} + \varepsilon)$. Figure 6 shows all the mentioned thresholds.

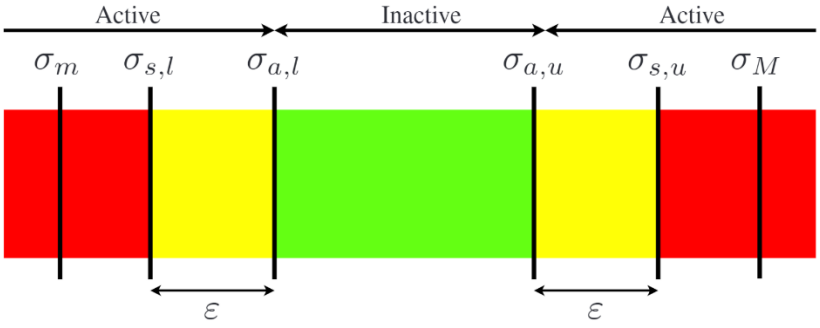


Figure 6: Activation and physical thresholds of a set-based task

When the task value exceeds an activation threshold, it must be added to the task hierarchy as a new equality-based task with desired value:

$$\sigma_d = \begin{cases} \sigma_{s,u} & \text{if } \sigma \geq \sigma_{a,u} \\ \sigma_{s,l} & \text{if } \sigma \leq \sigma_{a,l} \end{cases}$$

The task can be deactivated when the solution of the hierarchy that contains only the other tasks would push its value toward the valid set. More specifically, it is possible to check whether a generic solution $\dot{\boldsymbol{q}}$ makes a set-based task σ_A go beyond the desired limit or not by evaluating its projection in the task space. Defining $\boldsymbol{J}_A$ as the Jacobian matrix of σ_A, if $\boldsymbol{J}_A\dot{\boldsymbol{q}} > 0$ the solution would increase the set-based task value, otherwise if $\boldsymbol{J}_A\dot{\boldsymbol{q}} < 0$ the solution would decrease it. In this way, σ_A can be deactivated if:

$$\sigma_A \geq \sigma_{a,u} \ \wedge \ \boldsymbol{J}_A\dot{\boldsymbol{q}} < 0$$

or

$$\sigma_A \leq \sigma_{a,l} \ \wedge \ \boldsymbol{J}_A\dot{\boldsymbol{q}} > 0$$

2.2.2 Hierarchical Quadratic Programming

The other scheme investigated for the kinematic control of the farming robot leverage that the computation of the system velocity vector can be seen as a Quadratic Programming (QP) problem.

At the end of the implementation, this scheme was preferred with respect to the previous one since, from an implementation standpoint, it allows to more easily handle an arbitrary number of equality and inequality constraints, and obtained joint velocities are continuous, since constraints are not inserted and removed as in the previous scheme. Given a task $\boldsymbol{\sigma}_1$ with associated Jacobian $\boldsymbol{J}_1$ and desired task velocity $\dot{\boldsymbol{\sigma}}_{1,d}$, the problem to solve can be represented as:

$$\begin{aligned} \min_{\boldsymbol{w}_1,\dot{\boldsymbol{q}}} \quad & \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{Q}\dot{\boldsymbol{q}} + \boldsymbol{w}_1^T\boldsymbol{Q}_w\boldsymbol{w}_1 \\ \text{s.t.} \quad & \dot{\boldsymbol{\sigma}}_{1,d} = \boldsymbol{J}_1\dot{\boldsymbol{q}} + \boldsymbol{w}_1 \end{aligned} \tag{9}$$

If there are two tasks to be performed in a hierarchical manner, the solution can be computed by solving two separate QP problems, in which the second one has an additional constraint that takes into account the solution obtained by the first one:

$$\begin{aligned} \min_{\boldsymbol{w}_2, \dot{\boldsymbol{q}}} \quad & \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{Q}\dot{\boldsymbol{q}} + \boldsymbol{w}_2^T \boldsymbol{Q}_w \boldsymbol{w}_2 \\ \text{s.t.} \quad & \dot{\boldsymbol{\sigma}}_{2,d} = \boldsymbol{J}_2 \dot{\boldsymbol{q}} + \boldsymbol{w}_2 \\ & \dot{\boldsymbol{\sigma}}_{1,d} = \boldsymbol{J}_1 \dot{\boldsymbol{q}} + \boldsymbol{w}_1^\star \end{aligned}$$

where $\boldsymbol{w}_1^\star$ is the solution of the QP problem reported in Eq. (9).

The approach can be generalized for a hierarchy composed by k arbitrary tasks by solving the cascade of k QP problems, in which the level i includes the solutions of the previous QP problems from 1 to $i-1$. The last QP has thus the following structure (Escande, Mandard, & Wieber, 2014):

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

2.3 Implemented tasks

In this Section, the details about the tasks implemented for the CANOPIES robot are given. Each task is expressed in terms of Jacobian matrix $\boldsymbol{J}_x$ and task value $\boldsymbol{\sigma}_x$ to be used together with a certain desired task value $\boldsymbol{\sigma}_{x,d}$ for the Set-Based CLIK algorithm in Eq. (7) and in terms of equality or inequality constraints to be introduced in the i-th QP problem for the HQP algorithm Eq. (9). Each constraint is expressed in the general form $\boldsymbol{J}_x\dot{\boldsymbol{q}} = \boldsymbol{b}_x$ for equality ones and $\boldsymbol{J}_x\dot{\boldsymbol{q}} \geq \boldsymbol{b}_x$ or $\boldsymbol{J}_x\dot{\boldsymbol{q}} \leq \boldsymbol{b}_x$ for inequality ones.

2.3.1 Kinematic limits and collision tasks

2.3.1.1 Joint position limits

This task allows to consider the physical joint limits in terms of positions when computing the control signal. The Jacobian matrix is the identity matrix:

$$\boldsymbol{J}_{\text{jpl}} = \boldsymbol{I}$$

Since the mobile base has virtually no limits and is handled by the navigation stack, we only consider the joint position vector of the upper body $\boldsymbol{q}_{\text{UB}} = \begin{bmatrix} \boldsymbol{q}_T^T & \boldsymbol{q}_L^T & \boldsymbol{q}_R^T \end{bmatrix}^T$ and the task value is:

$$\boldsymbol{\sigma}_{\text{jpl}} = \boldsymbol{q}_{\text{UB}}$$

Regarding the HQP formulation, the vector $\boldsymbol{b}_{\text{jpl}}$ is obtained by $\boldsymbol{b}_{\text{jpl}} = \boldsymbol{K}_{\text{jpl}}(\boldsymbol{q}_{\max} - \boldsymbol{\sigma}_{\text{jpl}})$ or $\boldsymbol{b}_{\text{jpl}} = \boldsymbol{K}_{\text{jpl}}(\boldsymbol{q}_{\min} - \boldsymbol{\sigma}_{\text{jpl}})$, where $\boldsymbol{q}_{\min}$ and $\boldsymbol{q}_{\max}$ are vectors containing the minimum and maximum joint values respectively, and $\boldsymbol{K}_{\text{jpl}}$ is a matrix of gains. Overall, this task introduces two constraints in the QP problem:

$$\boldsymbol{J}_{\text{jpl}}\, \dot{\boldsymbol{q}}_{\text{UB}} \leq \boldsymbol{b}_{\text{jpl}}$$

$$\boldsymbol{J}_{\text{jpl}}\ \dot{\boldsymbol{q}}_{\text{UB}} \geq \boldsymbol{b}_{\text{jpl}}$$

2.3.1.2 *Joint velocity limits*

This task allows to consider the physical joint limits in terms of velocities when computing the control signal. The Jacobian matrix is again the identity matrix:

$$\boldsymbol{J}_{\text{jvl}} = \boldsymbol{I}$$

and the task value is the vector stacking the joint velocities of the upper body:

$$\boldsymbol{\sigma}_{\text{jvl}} = \dot{\boldsymbol{q}}_{\text{UB}}$$

For the HQP formulation, vector $\boldsymbol{b}_{\text{jvl}}$ is obtained as $\boldsymbol{b}_{\text{jvl}} = \dot{\boldsymbol{q}}_{\min}$ or $\boldsymbol{b}_{\text{jvl}} = \dot{\boldsymbol{q}}_{\max}$, which are vectors containing the minimum and maximum joint velocities. This task introduces two constraints in the QP problem:

$$\boldsymbol{J}_{\text{jvl}}\ \ \dot{\boldsymbol{q}}_{\text{UB}} \leq \dot{\boldsymbol{q}}_{\max}$$

$$\boldsymbol{J}_{\text{jvl}}\ \ \dot{\boldsymbol{q}}_{\text{UB}} \geq \dot{\boldsymbol{q}}_{\min}$$

2.3.1.3 *Virtual walls*

This task allows to define virtual walls that a point belonging to the serial chain of the manipulators cannot exceed. Given a certain point $\boldsymbol{p}_i$ belonging to the manipulator, the task objective is to keep the distance between the point and a plane above a certain minimum threshold. Defining as $\boldsymbol{p}^1$, $\boldsymbol{p}^2$ and $\boldsymbol{p}^3$ as three points belonging to the plane, the task value is distance between the point and the plane:

$$\sigma_{\text{vw},i} = \hat{\boldsymbol{n}}^T(\boldsymbol{p}_i - \boldsymbol{p}^1)$$

Where $\hat{\boldsymbol{n}}$ is the normal unit vector of the plane, computed as:

$$\hat{\boldsymbol{n}} = \frac{(\boldsymbol{p}^2 - \boldsymbol{p}^1) \times (\boldsymbol{p}^3 - \boldsymbol{p}^1)}{||(\boldsymbol{p}^2 - \boldsymbol{p}^1) \times (\boldsymbol{p}^3 - \boldsymbol{p}^1)||}$$

The task Jacobian matrix is defined as:

$$\boldsymbol{J}_{\text{vw},i} = -\hat{\boldsymbol{n}}^T \boldsymbol{J}_{p,i}$$

Where $\boldsymbol{J}_{p,i}$ is the position Jacobian related to the point $\boldsymbol{p}_i$. Regarding the HQP formulation, the $\boldsymbol{b}_{\text{vw},i}$ vector is defined as:

$$\boldsymbol{b}_{\text{vw},i} = k_{\text{vw},i}(d_{\min,i} - \sigma_{\text{vw},i})$$

where $d_{\min,i}$ is the minimum distance that the point $\boldsymbol{p}_i$ should keep from the plane and k_{vw} is a scalar gain.

A group of L virtual wall task, associated with different points $\boldsymbol{p}_i$ that have to be kept above a certain distance from the same wall can be grouped in a single task by stacking the corresponding Jacobian matrices $\boldsymbol{J}_{\text{vw},i}$ and vectors $\boldsymbol{b}_{\text{vw},i}$, obtaining:

$$\boldsymbol{J}_{\mathrm{vw}} = \begin{bmatrix} \boldsymbol{J}_{\mathrm{vw},1} \\ \boldsymbol{J}_{\mathrm{vw},2} \\ \vdots \\ \boldsymbol{J}_{\mathrm{vw},L} \end{bmatrix}$$

$$\boldsymbol{b}_{\mathrm{vw}} = \begin{bmatrix} \boldsymbol{b}_{\mathrm{vw},1} \\ \boldsymbol{b}_{\mathrm{vw},2} \\ \vdots \\ \boldsymbol{b}_{\mathrm{vw},L} \end{bmatrix}$$

Overall, all these tasks introduce L inequality constraints in the QP formulation:

$$\boldsymbol{J}_{\mathrm{vw}}\ \dot{\boldsymbol{q}}_{\mathrm{UB}} \geq \boldsymbol{b}_{\mathrm{vw}}$$

2.3.1.4 Self-collisions

This task is aimed at avoiding collisions between the different parts of the robot, i.e., the manipulators and the torso. The idea is to define M points $\boldsymbol{p}_{m,i}$ belonging to the two manipulators and O points $p_{o,i}$ belonging to the robot structure and define a constraint that keeps the distance between the two points above a minimum threshold $d_{\min}$. Let us define the task value as distance between two generic points as:

$$\sigma_{\mathrm{sc},i,l} = ||\boldsymbol{p}_{m,i} - \boldsymbol{p}_{o,l}||$$

The Jacobian matrix can be expressed as:

$$\boldsymbol{J}_{\mathrm{sc},i,l} = -\hat{\boldsymbol{n}}_{i,l}^{T} \boldsymbol{J}_{p,i}$$

Where:

$$\hat{\boldsymbol{n}}_{i,l} = \frac{\boldsymbol{p}_{m,i} - \boldsymbol{p}_{o,l}}{||\boldsymbol{p}_{m,i} - \boldsymbol{p}_{o,l}||}$$

is the unit vector connecting the two points and $\boldsymbol{J}_{p,i}$ is the position Jacobian matrix of the point $\boldsymbol{p}_i$. The $b_{\mathrm{sc},i,l}$ element of the constraint is expressed as:

$$b_{\mathrm{sc},i,l} = k_{i,l}(d_{\min,i} - \sigma_{\mathrm{sc},i,l})$$

and each one of the tasks impose a constraint in the QP problem such as:

$$\boldsymbol{J}_{\mathrm{sc},i,l}\ \dot{\boldsymbol{q}}_{\mathrm{UB}} \geq b_{\mathrm{sc},i,l}$$

where only torso joints are considered since self-collision are not affected by the mobile base degrees of mobility. The constraints referring to the same point of the manipulator can be gathered in the same Jacobian matrix:

$$\boldsymbol{J}_{\mathrm{sc},i} = \begin{bmatrix} \boldsymbol{J}_{\mathrm{sc},i,1} \\ \boldsymbol{J}_{\mathrm{sc},i,2} \\ \vdots \\ \boldsymbol{J}_{\mathrm{sc},i,O} \end{bmatrix}$$

and the $b_{\mathrm{sc},i,l}$ elements can be gathered in the vector:

$$\boldsymbol{b}_{\mathrm{sc},i} = \begin{bmatrix} b_{\mathrm{sc},i,1} \\ b_{\mathrm{sc},i,2} \\ \vdots \\ b_{\mathrm{sc},i,O} \end{bmatrix}$$

In the same way, all the constraints referring to the M points of the manipulator can be expressed via the Jacobian matrix:

$$\boldsymbol{J}_{\mathrm{sc}} = \begin{bmatrix} \boldsymbol{J}_{\mathrm{sc},1} \\ \boldsymbol{J}_{\mathrm{sc},2} \\ \vdots \\ \boldsymbol{J}_{\mathrm{sc},M} \end{bmatrix}$$

and the vector

$$\boldsymbol{b}_{\mathrm{sc}} = \begin{bmatrix} \boldsymbol{b}_{\mathrm{sc},1} \\ \boldsymbol{b}_{\mathrm{sc},2} \\ \vdots \\ \boldsymbol{b}_{\mathrm{sc},M} \end{bmatrix}$$

Overall, the tasks introduce an inequality constraint that can be expressed as:

$$\boldsymbol{J}_{\mathrm{sc}}\, \dot{\boldsymbol{q}}_{\mathrm{UB}} \geq \boldsymbol{b}_{\mathrm{sc}}$$

2.3.2 Operational tasks

2.3.2.1 Uncoordinated motion

This task allows to set two separate desired poses to the two end-effectors. The constraint matrix is the Jacobian matrix $\boldsymbol{J}_{\gamma}$ reported in Eq. (2) and also the task value has been already defined in Eq. (3). Regarding the HQP formulation, the $\boldsymbol{b}_{\gamma}$ vector is expressed as:

$$\boldsymbol{b}_{\gamma} = \begin{bmatrix} \dot{\boldsymbol{p}}^{\mathcal{W}}_{\mathcal{E}_1,d} \\ \boldsymbol{\omega}^{\mathcal{W}}_{\mathcal{E}_1,d} \\ \dot{\boldsymbol{p}}^{\mathcal{W}}_{\mathcal{E}_2,d} \\ \boldsymbol{\omega}^{\mathcal{W}}_{\mathcal{E}_2,d} \end{bmatrix} + \boldsymbol{K}_{\gamma} \left(\begin{bmatrix} \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1,d} - \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1} \\ \tilde{\boldsymbol{\mathcal{Q}}}^{\mathcal{W}}_{\mathcal{E}_1} \\ \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2,d} - \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2} \\ \tilde{\boldsymbol{\mathcal{Q}}}^{\mathcal{W}}_{\mathcal{E}_2} \end{bmatrix} \right)$$

where the first term is the vector stacking the desired linear/angular velocities for the two end-effectors, $\boldsymbol{K}_{\gamma}$ is a positive-definite matrix of gains, $\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1,d} - \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1}$ and $\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2,d} - \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2}$ are the position errors for the left and right end-effectors, $\tilde{\boldsymbol{\mathcal{Q}}}^{\mathcal{W}}_{\mathcal{E}_1}$ and $\tilde{\boldsymbol{\mathcal{Q}}}^{\mathcal{W}}_{\mathcal{E}_2}$ are the quaternion errors representing the orientation errors of the two end-effectors. This task introduces in the QP problem the following equality constraint:

$$\boldsymbol{J}_{\gamma}\, \dot{\boldsymbol{q}} = \boldsymbol{b}_{\gamma}$$

2.3.2.2 Coordinated motion

This task allows to assign a desired position and orientation trajectories for the absolute and relative frames described in Section 1.3. The Jacobian matrix is reported in Eq. (5), and the task value is expressed as in Eq. (4). Regarding the HQP formulation, the $\boldsymbol{b}_{\sigma}$ of the constraint can be expressed as:

$$\boldsymbol{b}_\sigma = \begin{bmatrix} \dot{\boldsymbol{p}}^{\mathcal{W}}_{a,d} \\ \boldsymbol{\omega}^{\mathcal{W}}_{a,d} \\ \dot{\boldsymbol{p}}^{\mathcal{W}}_{r,d} \\ \boldsymbol{\omega}^{\mathcal{W}}_{r,d} \end{bmatrix} + \boldsymbol{K}_\sigma \left(\begin{bmatrix} \boldsymbol{p}^{\mathcal{W}}_{a,d} - \boldsymbol{p}^{\mathcal{W}}_{a} \\ \tilde{\boldsymbol{Q}}^{\mathcal{W}}_{a} \\ \boldsymbol{p}^{\mathcal{W}}_{r,d} - \boldsymbol{p}^{\mathcal{W}}_{r} \\ \tilde{\boldsymbol{Q}}^{\mathcal{E}_1}_{r} \end{bmatrix} \right)$$

where the first term is the vector stacking the desired linear/angular for the absolute and relative frames, $\boldsymbol{K}_\sigma$ is a positive-definite matrix of gains, $\boldsymbol{p}^{\mathcal{W}}_{a,d} - \boldsymbol{p}^{\mathcal{W}}_{a}$ and $\boldsymbol{p}^{\mathcal{W}}_{r,d} - \boldsymbol{p}^{\mathcal{W}}_{r}$ are the position errors for the two frames, $\tilde{\boldsymbol{Q}}^{\mathcal{W}}_{a}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{E}_1}_{r}$ are the quaternion errors representing the orientation errors of the absolute and relative frames. This task introduces in the QP problem the following equality constraint:

$$\boldsymbol{J}_\sigma \, \dot{\boldsymbol{q}} = \boldsymbol{b}_\sigma$$

2.3.3 Optimization task

2.3.3.1 Swivel angle

This task aims at controlling the swivel angle of the manipulators, which is defined as the angle between the plane identified by the base, the shoulder and the wrist of a manipulator and the plane identified by the shoulder, the elbow, and the wrist (Jiang, 2022). The reason behind this additional task is that, since the robot is meant to work in cooperation with human farmers, controlling this angle in order to let the robot arms assume a configuration similar to human ones would increase the acceptability of the robot and the predictability of its motion. Figure 7 shows a graphical representation of the swivel angle for the right manipulator. Defining the position of the elbow as $\boldsymbol{p}_e$, the position of the wrist as $\boldsymbol{p}_w$, the position of the shoulder as $\boldsymbol{p}_s$, and the following vectors:

$$\boldsymbol{a} = [0,\, 0,\, 1]^T \quad \boldsymbol{b} = \boldsymbol{p}_s - \boldsymbol{p}_e \quad \boldsymbol{c} = \boldsymbol{p}_s - \boldsymbol{p}_w \quad \boldsymbol{d} = \boldsymbol{p}_e - \boldsymbol{p}_w$$

the swivel angle can be computed employing the following relationship:

$$\varphi = sgn(\boldsymbol{a} \times \boldsymbol{b} \cdot \boldsymbol{c}) \ \arccos\left(\frac{(\boldsymbol{a}\times\boldsymbol{c})(\boldsymbol{b}\times\boldsymbol{d})}{||\boldsymbol{a}\times\boldsymbol{c}||\ ||\boldsymbol{b}\times\boldsymbol{d}||} \right)$$

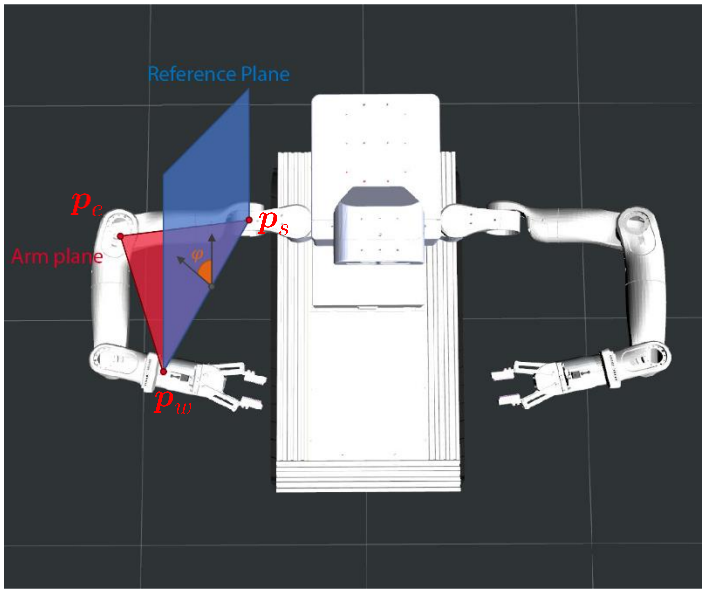


Figure 7: Graphical representation of the swivel angle φ for the right arm

The Jacobian matrix is defined as the gradient of φ with respect to the joint variable $\boldsymbol{q}$:

$$\boldsymbol{J}_{\mathrm{sw}} = \frac{\partial \varphi}{\partial \boldsymbol{q}_{\mathrm{UB}}}$$

whose expression as function of the joint vector has been computed symbolically in MATLAB and it is not reported here for the sake of brevity.

This task introduces an equality constraint in the QP problem as:

$$\boldsymbol{J}_{\mathrm{sw}}\, \dot{\boldsymbol{q}}_{\mathrm{UB}} = k_{\mathrm{sw}} \left(\varphi_d - \varphi\right)$$

where φ_d is a desired swivel angle value and k_{sw} is a scalar gain.

2.4 Kinematic singularity handling

If the task Jacobian matrix $\boldsymbol{J}$ is singular, the reference velocity in output of Eq. (7) for the CLIK algorithm and of Eq. (9) for the HQP algorithm diverges, potentially leading the system to instability (Di Vito, Natale, & Antonelli, 2017). This kind of situations are undesirable and need to be properly handled or avoided to guarantee the safety of the operations. Regarding the CLIK algorithm, a possible solution is to employ Damped Least-Square (DLS) pseudoinverse of the Jacobian matrix that applies a damping factor only on the directions associated to the loss of mobility. Given the Singular-Value Decomposition (SVD) of the Jacobian:

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \sum_{i=1}^{m} s_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$$

The DLS pseudoinverse $\boldsymbol{J}^{\#}$ is computed as:

$$\boldsymbol{J}^{\#} = \boldsymbol{V}\widetilde{\boldsymbol{\Sigma}}^{\dagger}\boldsymbol{U}^{\mathrm{T}} = \boldsymbol{V}\begin{bmatrix}\widetilde{\boldsymbol{S}}^{-1} \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{U}^{\mathrm{T}}$$

The diagonal elements of the matrix $\widetilde{\boldsymbol{S}}^{-1}$ are:

$$\widetilde{\boldsymbol{S}}^{-1}_{i,i} = \frac{s_i}{s_i^2+\lambda_i^2}$$

with:

$$\lambda_i = \begin{cases} 0 & \text{if } s_i \geq s^\star \\ \Lambda\left(\frac{1}{2}\cos(\frac{s_i}{s^\star}\pi) + \frac{1}{2}\right) & \text{if } s_i < s^\star \end{cases}$$

where Λ and $s^\star$ express the maximum damping factor and the lower threshold on the singular values to be tuned according to the specific system used. Figure 8 shows the evolution of the damping factor λ_i in function of the i-th singular value s_i of the Jacobian matrix.

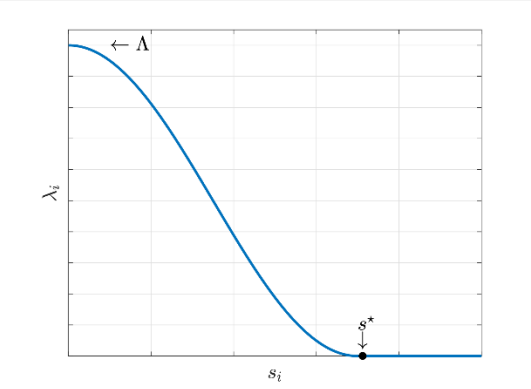


Figure 8: Evolution of the damping factor over the singular value

The DLS $\boldsymbol{J}^{\#}$ can be then employed in Eq. (8) in place of the standard pseudoinverse $\boldsymbol{J}^{\dagger}$ to mitigate the effect on the joint velocities in proximity of a singular configuration.

In order to have the same behaviour while employing the HQP algorithm, it is necessary to manipulate the $\boldsymbol{Q}$ matrix in the objective function to minimize in Eq. (10). Given the Singular Value Decomposition of the task Jacobian matrix:

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$$

The weighting matrix $\boldsymbol{Q}$ can computed as:

$$\boldsymbol{Q} = \boldsymbol{V}\ \boldsymbol{\Delta S}\ \boldsymbol{V}^{T}$$

where $\boldsymbol{\Delta S}$ is a diagonal matrix, whose diagonal elements are computed as:

$$\boldsymbol{\Delta S}_{i,i} = \begin{cases} 0 & \text{if } s_i \geq s^\star \\ \Lambda\left(\frac{1}{2}\cos(\frac{s_i}{s^\star}\pi) + \frac{1}{2}\right) & \text{if } s_i < s^\star \end{cases}$$

3 Validation

The control frameworks described in the previous sections has been extensively validated through simulations and experiments on several platforms. In detail, the control strategies were tested on three different platforms shown in Figure 9:

- Movo robot: this is a dual arm robot, already available at the LAI-Robotics laboratory at UNICLAM, with variable-height torso, two 7DoFs arms and omnidirectional mobile base. This robot was selected for preliminary tests in the first part of the project since, apart the rotational joint on the torso, it has a similar kinematics to the Farming Robot;
- Tiago++-based Farming Robot: the Tiago++ was adapted to be mounted on the mobile platform by partner PAL, as part of the mitigation plan put in place by the Consortium because of the delay in the delivery of the Farming Robot prototype. The platform was delivered to UNICLAM facilities at the end of June 2022;
- Farming Robot protype: this is the Farming Robot prototype equipped with scissors and two finger grippers suitable to work on the experimental field. In terms of kinematics, the upper body has, differently from the previous platforms, an addition revolute joint on the torso to increase the workspace the robot. The platform was delivered at UNICLAM in the middle of January 2023.

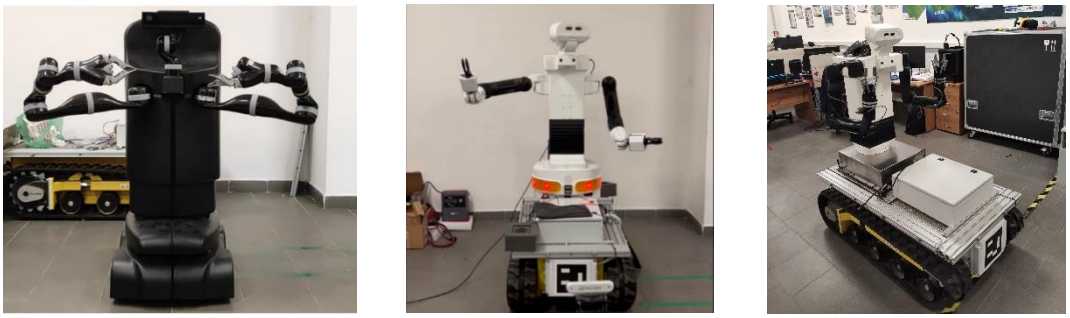


Figure 9: The three platforms adopted to test the devised methodologies. Left: Movo robot, already available at the LAI-Robotics laboratory. Center: Tiago++ mounted on the mobile base. Right: First prototype of the CANOPIES Farming Robot.

3.1 Simulations

This section reports the obtained simulative results. In Section 3.1.1, the Set-Based CLIK algorithm is validated on the Movo robot, while in Sections 3.1.2-3.1.2.3, the HQP approach is validated on the Farming Robot model.

3.1.1 Set-based CLIK validation on the Movo robot

In the following, we report the results obtained in the case of the Movo robot and CLIK algorithm described in Section 2.2.1. In order to validate the methodology in a complex case study, a typical domestic use case scenario has been considered, in which the robot has to bring a tray to a desired location holding it with two hands (see, Figure 10).

The mobile base frame $\mathcal{B}$ of the robot is initially placed in:

$$\boldsymbol{p}_{\mathcal{B},0}^{\mathcal{W}} = [0 \quad 0 \quad 0.2]^{\mathrm{T}} \ \mathrm{m}$$

with respect to the world frame $\mathcal{W}$ while holding a 40 cm-long tray with both hands. The task is designed in terms of a coordinated motion with $\alpha = 0$. The initial absolute position is:

$$\boldsymbol{p}_{a,0}^{\mathcal{W}} = [0.66 \quad 0.15 \quad 0.85]^{\mathrm{T}} \ \mathrm{m}$$

with an orientation expressed by the following quaternion:

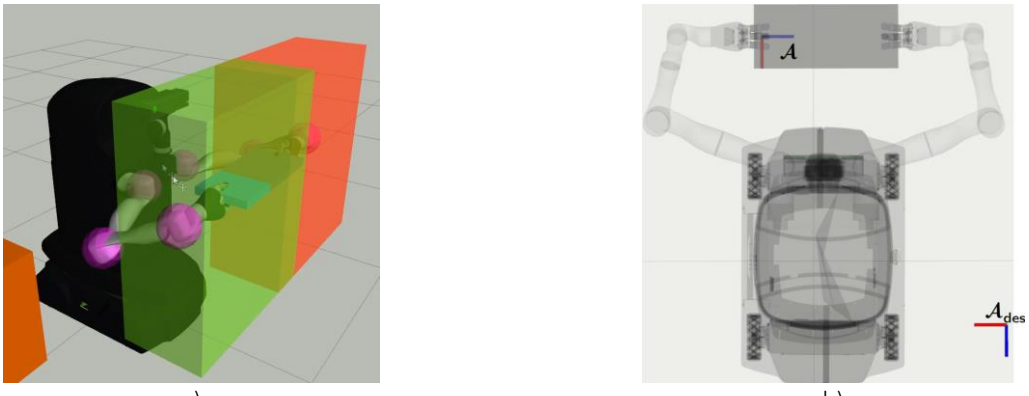
$$\mathcal{Q}_{a,0}^{\mathcal{W}} = [0 \quad -0.7071 \quad 0.7071 \quad 0]^{\mathrm{T}}$$

The desired absolute position and orientation are given as a constant they are chosen to be behind the robot, with an orientation of -90 degrees around the y-axis with respect to the initial orientation. The reference for the relative pose task is set equal to the initial one, allowing it to hold horizontally the tray.

In the following, the results obtained by applying the following task hierarchy are discussed:

1. **Joint limits:** upper/lower limits on the 2nd, 4th and 6th joints of both arms are set, being the ones that have actual mechanical limits on the Movo robot.
2. **Virtual walls**: six virtual walls, forming a virtual box, are placed around the base frames of both arms.
3. **Coordinated motion**: a coordinated motion in terms of absolute/relative pose ($\alpha = 0$) has been designed to make the robot move the tray to the desired location.

Figure 10 shows graphical representations of the initial and desired absolute frame poses, the virtual boxes around the arm base frames and the joint taken into consideration for the joint limits task.



a) b)

Figure 10: a) Graphical representation of the joint limits and virtual wall tasks; b) Graphical representation of the initial and desired absolute frame configuration

The obtained results in Figure 11 show that both the absolute and relative position and quaternion errors reach a null steady-state value, meaning that the tray eventually reaches the desired location, while keeping the relative pose between the two end-effectors at a constant value. The limits imposed by the safety tasks are not violated during the entire experiment. Additionally, it can be noticed that several safety tasks become active during the motion and that they get stuck at the

desired safety thresholds (red-dashed horizontal lines in the plots): the second and fourth joints of the right arm, the wall on the z direction of the right arm base frame and the wall on the x direction of the left arm base frame.

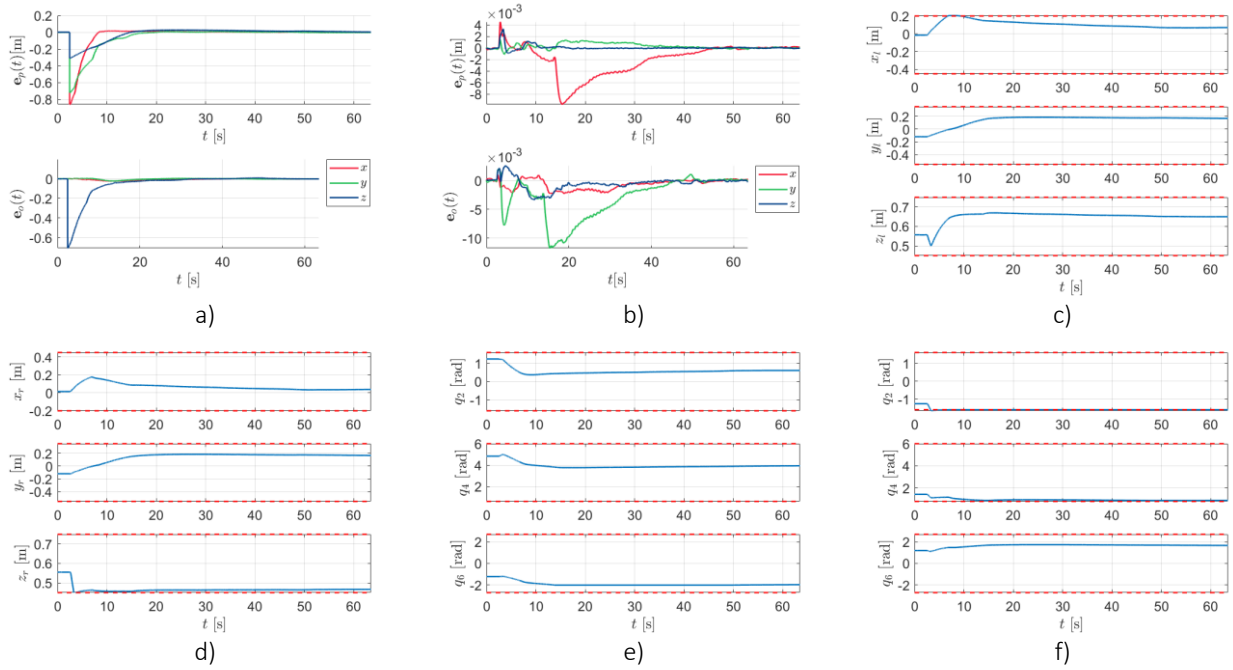


Figure 11: a) absolute pose error; b) relative pose error; c) position of the left end-effector expressed in the arm base frame and the limits imposed by the virtual wall tasks; d) position of the right end-effector expressed in the arm base frame and the limits imposed by the virtual wall tasks; e) left arm joint positions and imposed limits; f) e) right arm joint positions and imposed limits.

3.1.2 HQP validation on the Farming Robot

In the following, we report the results obtained in the case of the Farming Robot and the HQP algorithm described in Section 2.2.2. In order to validate the methodology, we report simulation of the three task categories separately, i.e. the safety, operational and optimization tasks described in Section 2.3.

3.1.2.1 Safety tasks validation

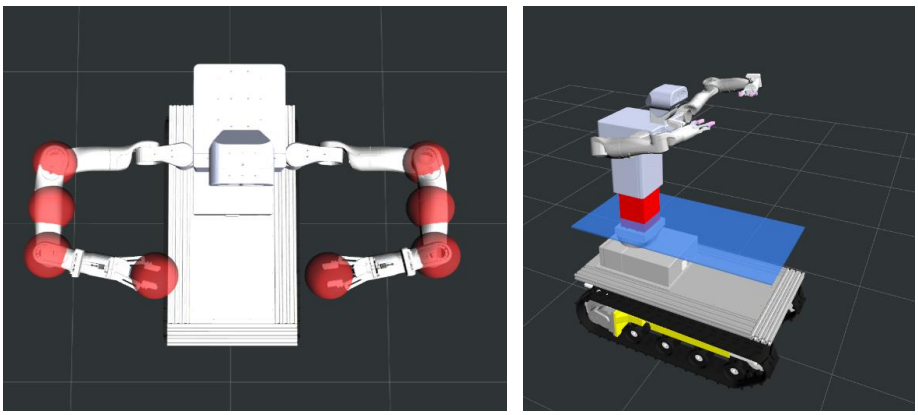
This simulation is aimed at validating all the implemented safety tasks by exploiting the HQP approach. A sinusoidal velocity is imposed to all the joints, and it is implemented as an equality constraint in the HQP framework with $\boldsymbol{J} = \boldsymbol{I}$ and each element of the $\boldsymbol{b}$ vector is composed as predefined sinusoidal functions. The overall task hierarchy taken into consideration for this simulation is:

1. **Joint position/velocity limits**: upper/lower thresholds are imposed for all the joint position and velocity. They are set in order to match the actual limits that the robot exhibits.
2. **Virtual wall**: it is a horizontal plane placed at the mobile base height in order to avoid collisions between three points belonging to the manipulators and the mobile base. In particular, the points are chosen as the elbows, the wrists and the end-effectors and the minimum distance that must be kept is set to 0.5m
3. **Self-collisions avoidance:** these tasks are inserted to avoid collisions between four points of the manipulators (joint 6, mid-point between joint 6 and joint 8, joint 8 and end-effector) and

the closest point of the head, the closest point of the torso, the mid-point between joint 3 and 4, between joint 4 and 6 and between joint 6 and 8, and the same four points (joint 6, mid-point between joint 6 and joint 8, joint 8 and end-effector) of the other arm.

4. Sinusoidal joint velocity: a sinusoidal velocity of $\dot{q}_i = \frac{3}{2}\pi \sin(\frac{\pi}{4}t)$ is imposed to all the joints of the two manipulators.

It is worth noticing that the desired joint trajectories of the lowest-priority task have been intentionally chosen to exceed the imposed minimum/maximum thresholds of the higher-priority tasks, triggering the activation of the associated constraints. Figure 12 shows graphical representations of the points taken into consideration for the self-collision tasks and of the virtual wall placed to avoid collisions with the mobile base.



a) b)

Figure 12: a) Graphical representation of the points taken into consideration for the self-collision tasks; b) Graphical representation of virtual wall for the corresponding task

Figure 13 and Figure 14 show the evolution of the joint positions and velocities of the two manipulators over time during the simulation. The red-dashed horizontal lines represent the minimum/maximum thresholds, and it is worth noticing that the control algorithm successfully manages to make both the position and the velocity respect the constraints. Regarding the virtual wall task, Figure 15 shows the evolution of the three distances for the two arms, and it is clear that they never exceed the imposed constraint. Figure 16 shows the results of the self-collisions tasks. Figure 17 reports some snapshots of the first simulation in Rviz environment.

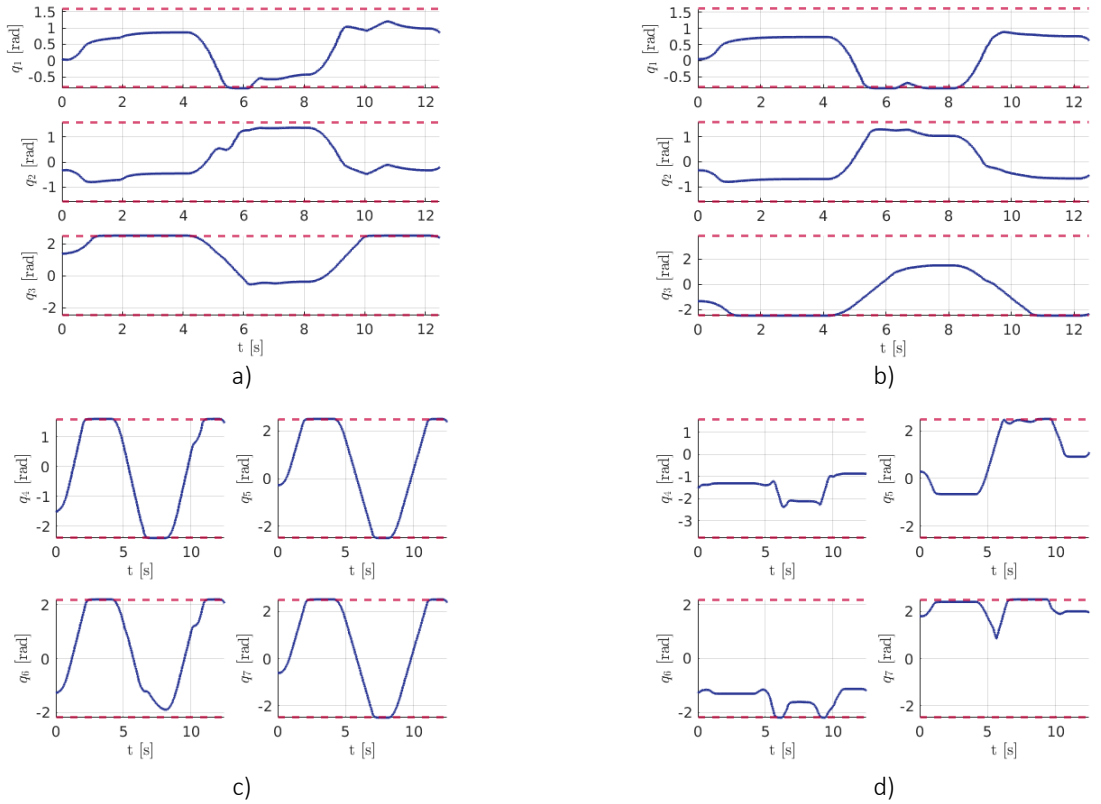


Figure 13. a) and c): Joint positions over time and imposed minimum/maximum values for the left arm; b) and d): Joint positions over time and imposed minimum/maximum values for the left arm

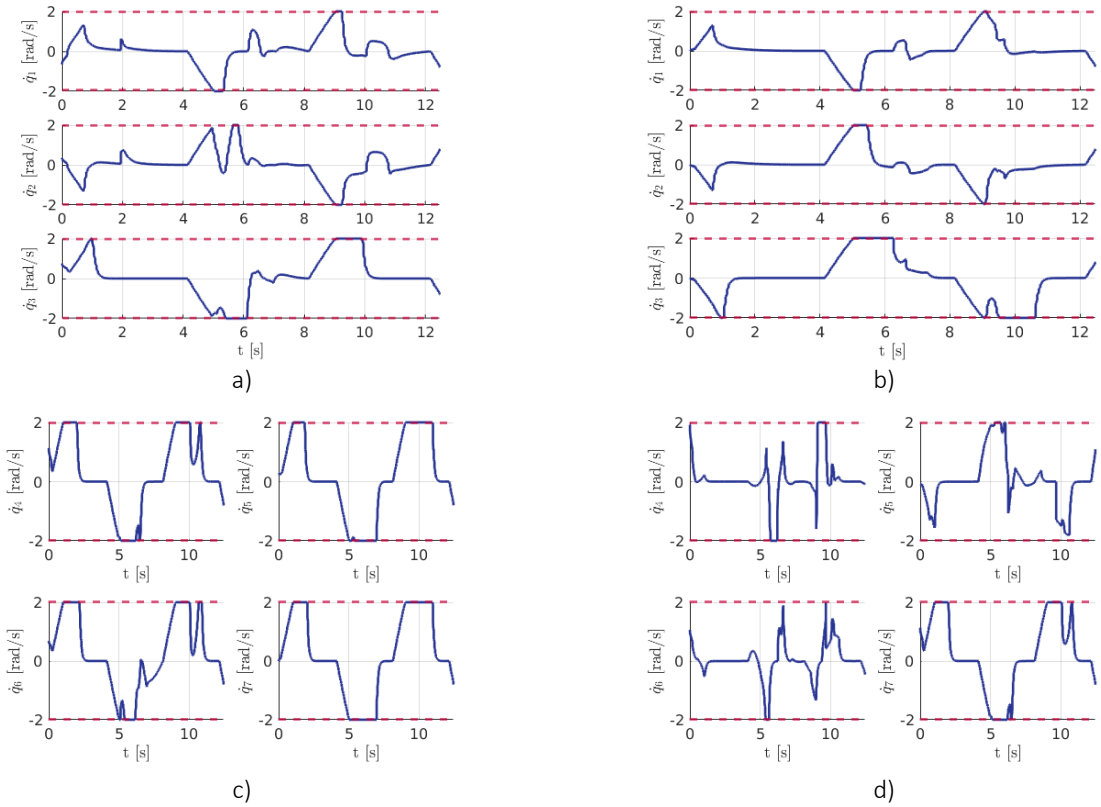


Figure 14. a) and c): joint velocities over time and imposed thresholds for the left arm; b) and d): joint velocities over time and imposed thresholds for the right arm

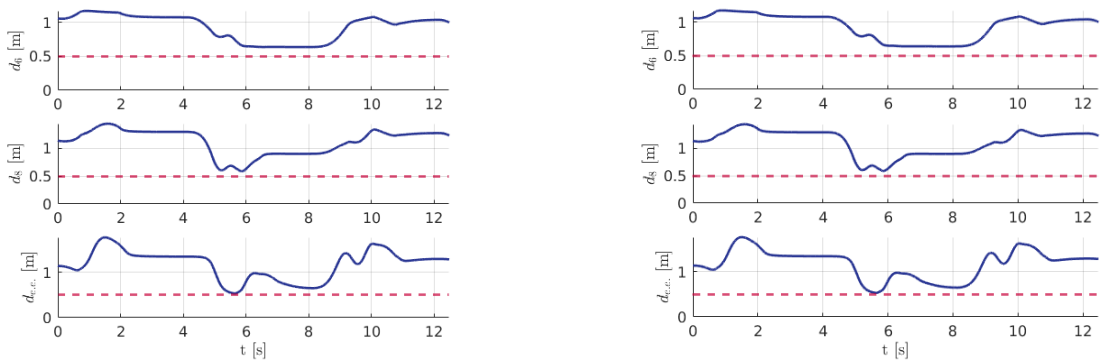


Figure 15: a) Distance between the three control points on the manipulators and the virtual wall placed on the mobile base for the left arm; b) Distance between the three control points on the manipulators and the virtual wall placed on the mobile base for the right arm

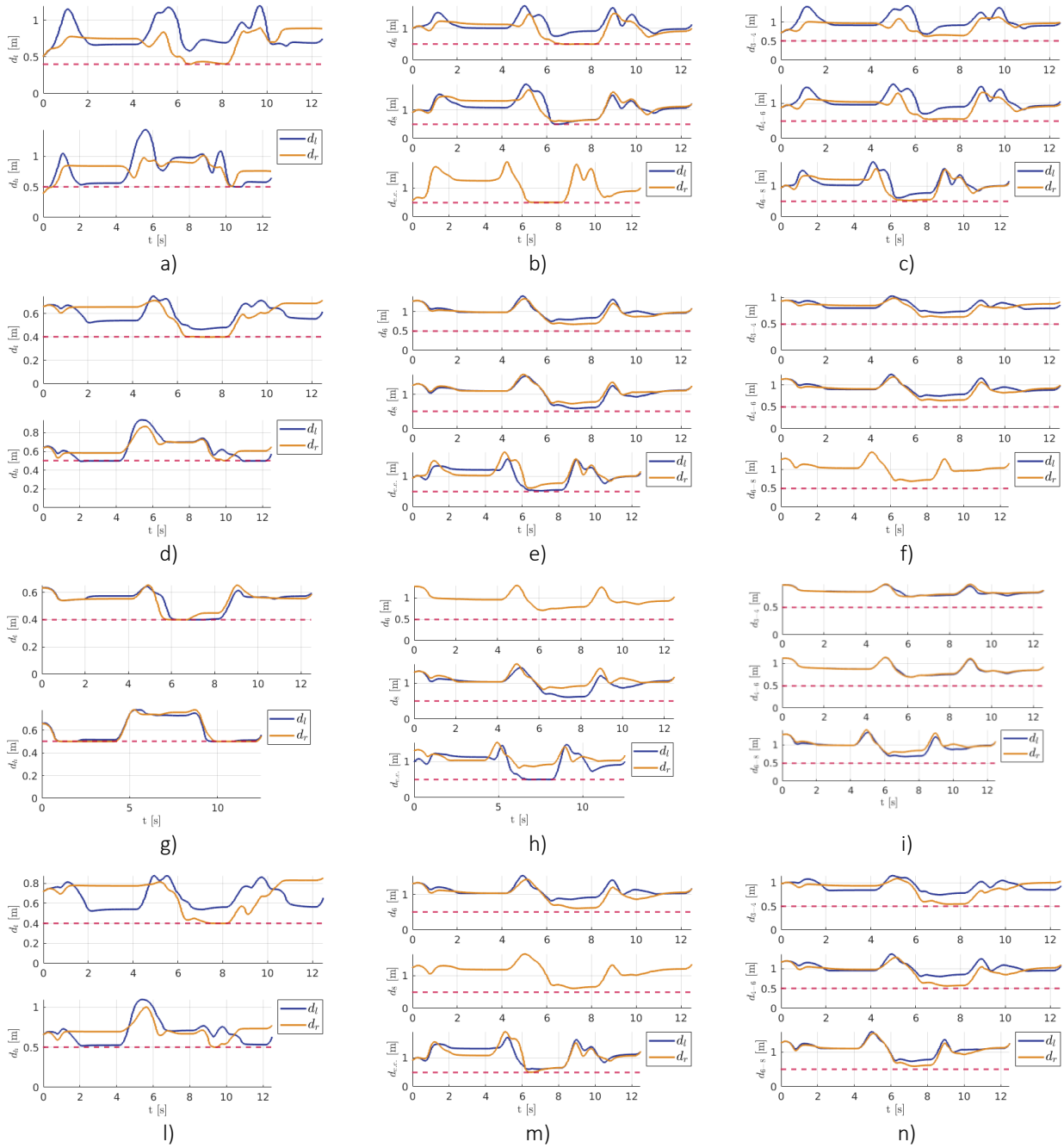


Figure 16: a) Distances between the two end-effectors and the closest point of the head and the torso; b) Distances between the end-effector of one arm and the joint 6, joint 8 and end-effector of the other arm; c) Distances between the end-effector of one arm and the mid-point between joint 3 and joint 4, between joint 4 and joint 6, and between the joint 6 and joint 8 of the other arm; d) Distances of the mid-point between joint 6 and joint 8 of both the arms and the closest point of the head and the torso; e) Distances of the mid-point between joint 6 and joint 8 of one arm and the joint 6, joint 8 and end-effector of the other arm; f) Distances of the mid-point between joint 6 and joint 8 of one arm and the mid-point between joint 3 and joint 4, between joint 4 and joint 6, and between the joint 6 and joint 8 of the other arm; g) Distances between the two joint 6 and the closest point of the head and the torso; h) Distances between the joint 6 of one arm and the joint 6, joint 8 and end-effector of the other arm; i) Distances between the joint 6 of one arm and the mid-point between joint 3 and joint 4, between joint 4 and joint 6, and between the joint 6 and joint 8 of the other arm; l) Distances between the two joint 8 and the closest point of the head and the torso; m) Distances between the joint 8 of one arm and the joint 6, joint 8 and end-effector of the other arm; n) Distances between the joint 8 of one arm and the mid-point between joint 3 and joint 4, between joint 4 and joint 6, and between the joint 6 and joint 8 of the other arm.

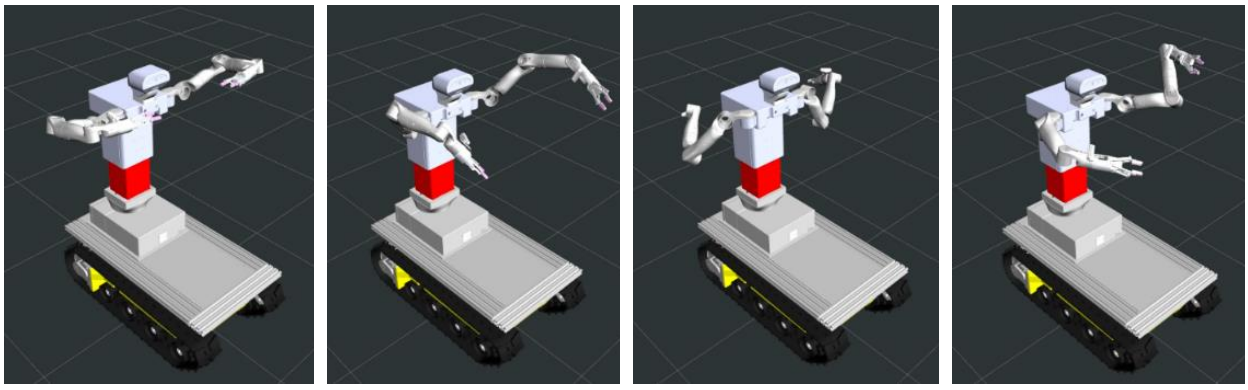


Figure 17: Snapshots of some robot configurations in the second simulation case study.

3.1.2.2 Operational tasks validation

This section shows two simulations aimed at validating the implementation of the coordinated and uncoordinated motion tasks.

In the first one, an uncoordinated motion is designed for the two manipulators. The desired position for the left arm is set as:

$$\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1,d}(t) = \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1}(0) + \begin{bmatrix} 0.1\ \cos(1-\cos(\frac{\pi}{4}t)) \\ 0.2\ \cos(1-\cos(\frac{\pi}{4}t)) \\ 0.05\ \cos(1-\cos(\frac{\pi}{4}t)) \end{bmatrix}$$

where $\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_1}(0)$ is the initial position of the left end-effector. The desired orientation is expressed by the following rotation matrix:

$$\boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_1,d}(t) = \boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_1}(0)\boldsymbol{R}(\phi(t),\theta(t),\psi(t))$$

where $\boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_1}(0)$ is the rotation matrix expressing the initial orientation of the left end-effector and $\boldsymbol{R}(\phi(t),\theta(t),\psi(t))$ is the rotation matrix obtained from a roll-pitch-yaw rotation with:

$$\phi = 0 \quad \theta(t) = -\tfrac{\pi}{8}(1-cos(\tfrac{\pi}{4}t)) \quad \psi = 0$$

Regarding the right arm, the desired end-effector position is given as:

$$\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2,d}(t) = \boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2}(0) + \begin{bmatrix} 0.1\ \cos(1-\cos(\frac{\pi}{4}t)) \\ 0.4\ \cos(1-\cos(\frac{\pi}{4}t)) \\ -0.05\ \cos(1-\cos(\frac{\pi}{4}t)) \end{bmatrix}$$

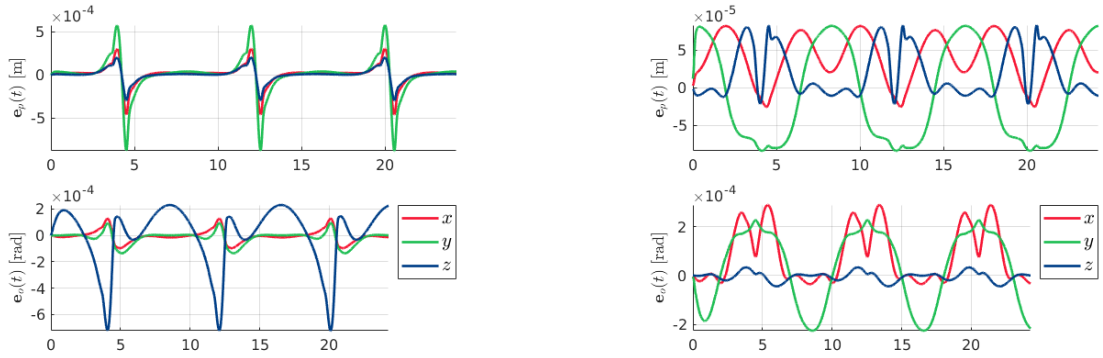
where $\boldsymbol{p}^{\mathcal{W}}_{\mathcal{E}_2,0}$ is the initial right end-effector position. The desired orientation is expressed by the following rotation matrix:

$$\boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_2,d}(t) = \boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_2}(0)\boldsymbol{R}(\phi(t),\theta(t),\psi(t))$$

Where, as before, $\boldsymbol{R}^{\mathcal{W}}_{\mathcal{E}_2}(0)$ is the rotation matrix expressing the initial orientation of the right end-effector and $\boldsymbol{R}(\phi(t),\theta(t),\psi(t))$ is the rotation matrix obtained from a roll-pitch-yaw rotation with

$$\phi = 0 \quad \theta = 0 \quad \psi(t) = -\tfrac{\pi}{8}(1-cos(\tfrac{\pi}{4}t))$$

Figure 18 shows the position and orientation errors for the two end-effectors, and they successfully track the desired trajectory both in position and orientation. Figure 19 shows some snapshots of the conducted simulation.



a) b)

Figure 18: a) Top: position error for the left arm; Bottom: orientation error for the left arm; b) Top: position error for the right arm; Bottom: orientation error for the right arm

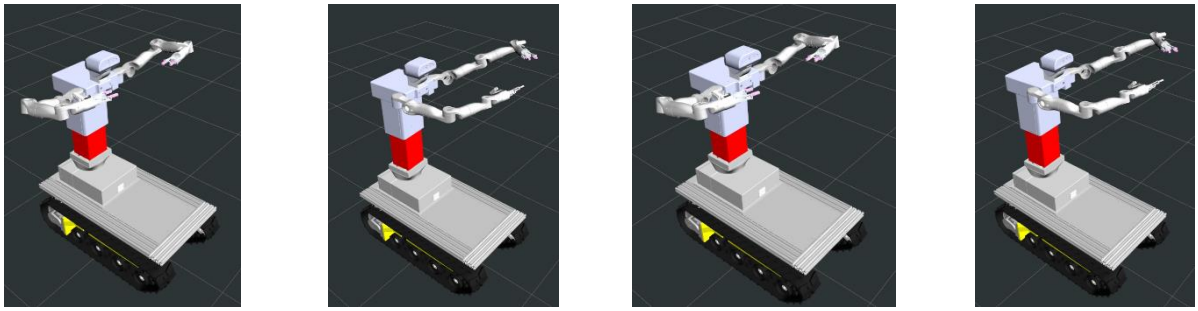


Figure 19: Snapshots of the simulation aimed at validating an uncoordinated motion

For the second simulation, a coordinated motion is designed for the system in terms of absolute and relative pose, with the parameter α set to 0.5. The desired absolute position is set as:

$$\boldsymbol{p}_{a,d}^{\mathcal{W}}(t) = \boldsymbol{p}_{a,0}^{\mathcal{W}} + \begin{bmatrix} 0.05(1-\cos(\frac{\pi}{2}t)) \\ 0.1(1-\cos(\frac{\pi}{2}t)) \\ 0.0 \end{bmatrix}$$

where $\boldsymbol{p}_{a,0}^{\mathcal{W}}$ is the initial position of the absolute frame, and the desired absolute orientation is kept constant at initial value. The desired relative position is set as:

$$\boldsymbol{p}_{r,d}^{\mathcal{W}}(t) = \boldsymbol{p}_{r,0}^{\mathcal{W}} + \begin{bmatrix} 0.1(1-\cos(\frac{\pi}{2}t)) \\ 0.0 \\ 0.1(1-\cos(\frac{\pi}{2}t)) \end{bmatrix}$$

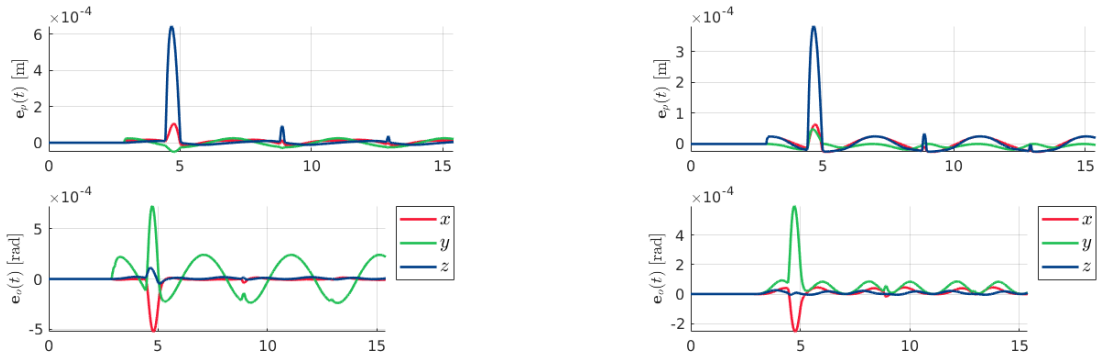
where $\boldsymbol{p}_{r,0}^{\mathcal{W}}$ is the initial position of the relative frame, and the desired relative orientation is expressed by the following rotation matrix:

$$\boldsymbol{R}_{r,d}^{\mathcal{W}} = \boldsymbol{R}_{r,0}^{\mathcal{W}} \boldsymbol{R}_{r,\psi(t)}^{\mathcal{W}}$$

where $\boldsymbol{R}_{r,0}^{\mathcal{W}}$ is the rotation matrix expressing the initial relative orientation, and $\boldsymbol{R}_{r,\psi(t)}^{\mathcal{W}}$ is the rotation matrix obtained from the following roll, pitch and yaw angles:

$$\phi = 0 \quad \theta = 0 \quad \psi = -\frac{\pi}{8}(1 - \cos(\frac{\pi}{2}(t)))$$

Figure 20 shows the evolution of the absolute and relative configuration errors, while Figure 21 shows some snapshots of the conducted simulation.



a) b)

Figure 20: a) Absolute position and orientation errors over time; b) Relative position and orientation errors over time

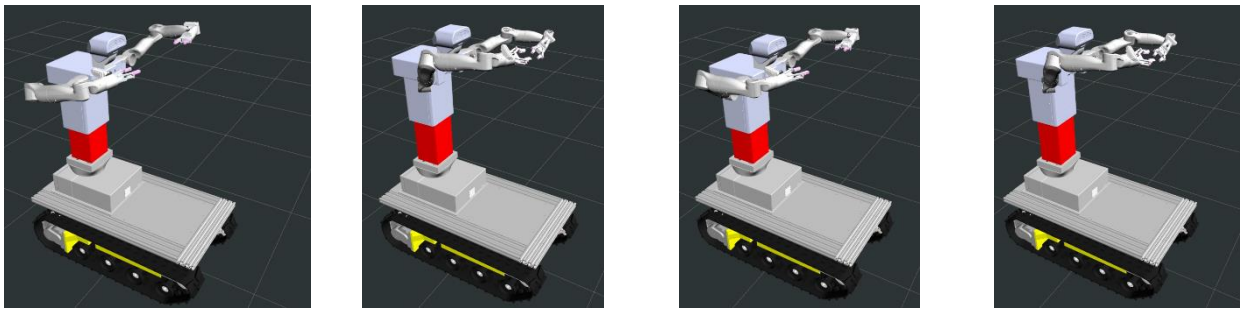
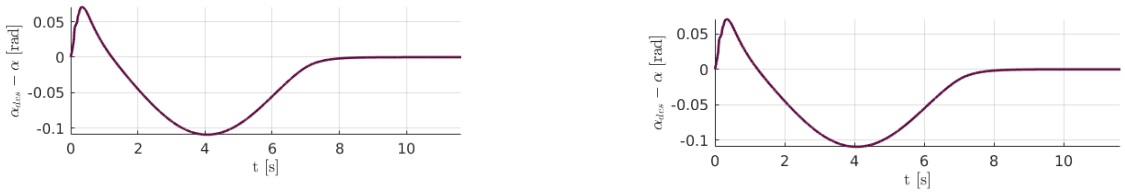


Figure 21: Snapshots of the coordinated task simulation

3.1.2.3 Optimization task validation

The following simulation has been carried out to show the effectiveness of the implemented swivel angle task. The manipulators are asked to keep the initial end-effector configurations in terms of position and orientation, while a smooth desired trajectory for the swivel angles of the two manipulators is given as reference for the swivel angle task, starting from the initial values and ending at a desired final value of $\frac{\pi}{4}$ rad. Figure 22 shows the swivel angle error and for the two manipulators. It is worth noticing that the error converges to zero, as the arms successfully manage to generate an internal motion that make their swivel angle follow the desired trajectory while keeping the end-effectors still. Figure 23 shows some snapshots of the simulation.



a) b)

Figure 22: a) Swivel angle error for the left arm. b) Swivel angle error for the right arm.

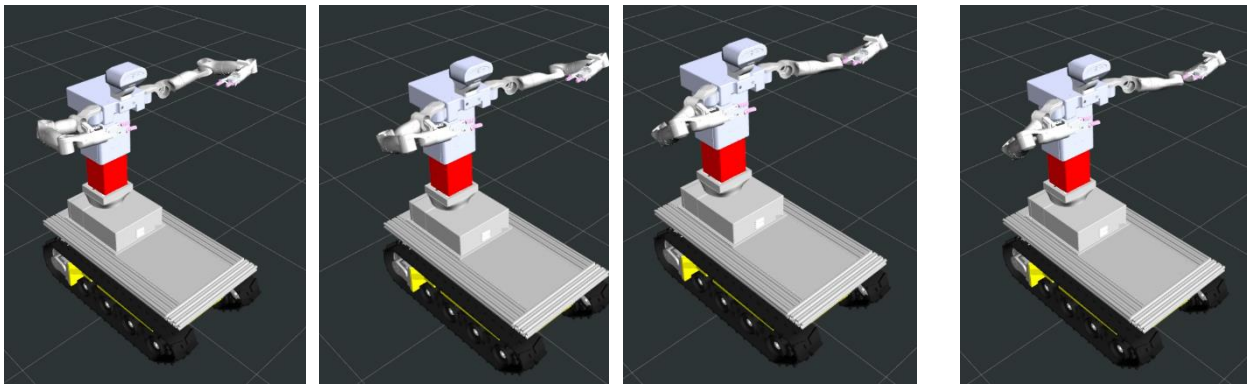


Figure 23: Snapshots of the simulation aimed at validating the swivel angle task

3.2 Experiments

In this Section, we show experimental results obtained employing the Set-Based CLIK algorithm on the Movo robot and employing the HQP algorithm on the Farming Robot. This choice is motivated by the fact that Set-Based CLIK algorithm has been first implemented and tested on the only robotic platform that was available in LAI-Robotics laboratory at UNICLAM (i.e., the Movo robot); only later, the HQP control framework has been implemented and tested on the Tiago++-based Farming robot and finally on the Farming Robot prototype.

For all experiments that we show in Section, the use case scenario is an example of harvesting motion, obtained with one or two arms.

3.2.1 Set-Based CLIK algorithm on the Movo robot

The first experiment is related to a coordinated motion executed with both arms with the Movo robot performed at UNICLAM. The references for the absolute and relative poses have been designed in order to execute a possible harvesting motion, dividing the overall task in a series of waypoints expressed in terms of position and orientation to be reached by employing a smooth trapezoidal velocity profile.

In particular, the first waypoint for the absolute pose is chosen in order to reach a pre-grasp configuration, while the desired relative pose is set to keep the left end-effector below the right one of a quantity that takes into account the length of the grape to harvest. The second waypoint allows the end-effectors to reach the peduncle while keeping the relative pose constant with respect to the first waypoint; then, both the end-effectors move to a pre-release configuration close to the mobile base, simulating the presence of a box in which the harvested grape has to be placed; finally, the left end-effector reaches a predefined configuration, while the right one releases the grape. Figure 24 shows some snapshots of the conducted experiment.

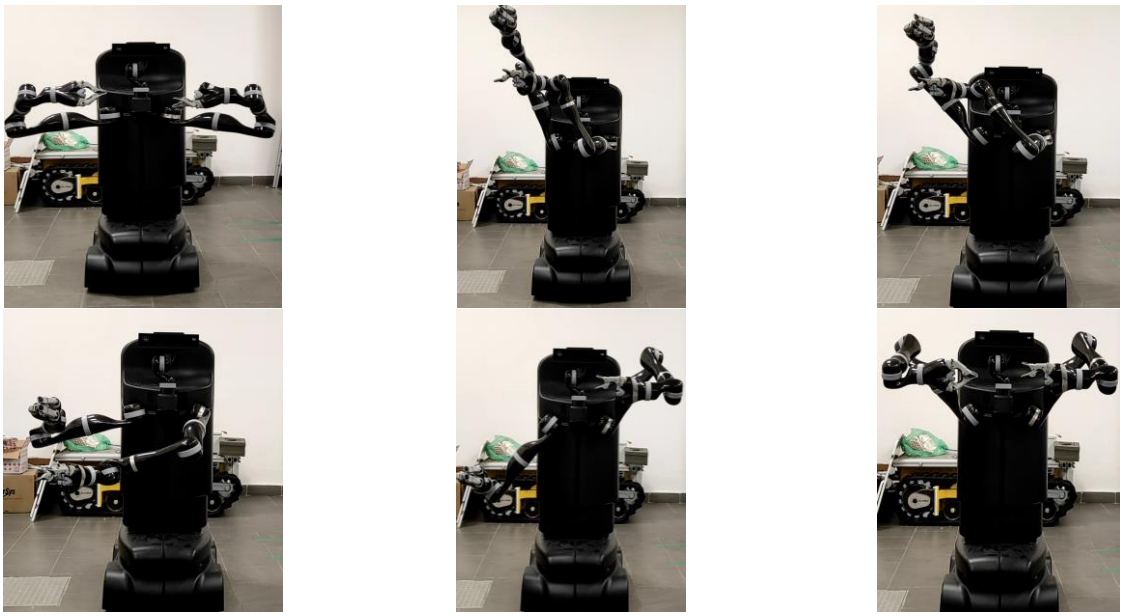


Figure 24: Snapshots of the coordinated harvesting motion with the Movo robot

In the second experiment, we show a possible harvesting motion obtained employing only one arm, i.e., in an uncoordinated manner. Also in this case, a proper sequence of waypoints in terms of position and orientation for the right end-effector has been chosen in order to reach a pre-grasp configuration, grasp the simulated peduncle, reach a pre-release pose over a simulated box and finally release the grape. Figure 25 shows some snapshots of the conducted experiment.

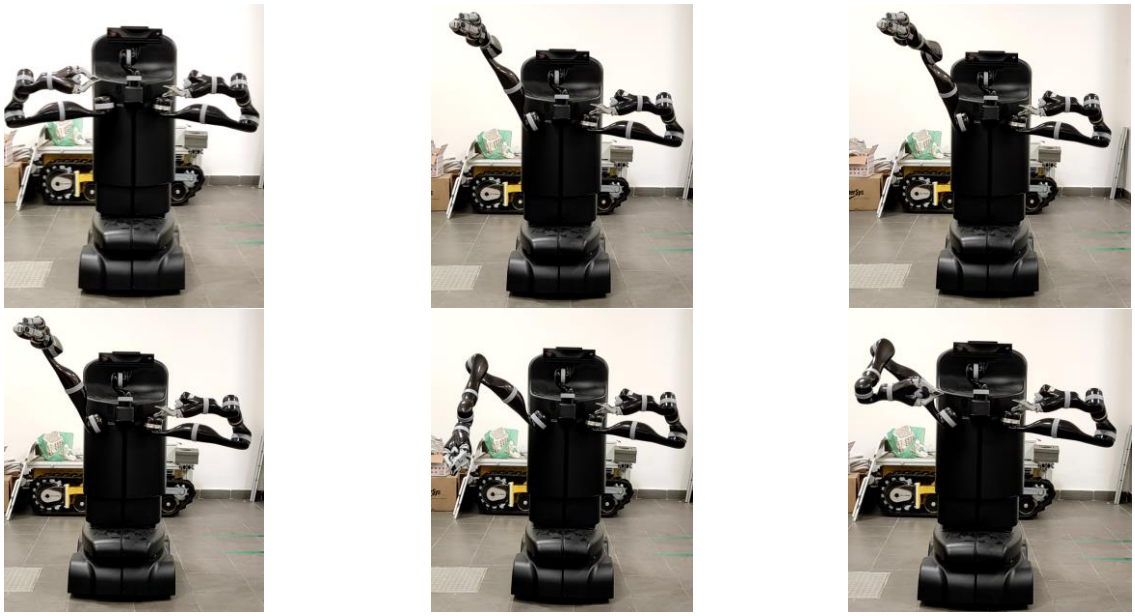


Figure 25: Snapshots of the single arm harvesting motion with the Movo robot. Right arm is the active arm, the left arm is kept still.

3.2.2 HQP control framework on the Farming Robot

Experiments were carried out both in the small-scale vineyard planted in 2021 by partner AGRIMESSINA in the nearby of UNICLAM facilities and in the CANOPIES experimental field in Aprilia, Italy. As an example, Figure 26 shows two pictures, one regarding the Tiago++-based farming robot

in the UNICLAM small-scale vineyard (September, 2022) and the Farming Robot prototype in the Aprilia experimental field (January, 2023).



Figure 26: Left: Tiago++-based farming robot in the UNICLAM small-scale vineyard. Right: the 1st farming robot prototype in the Aprilia experimental field (January, 2023).

In the following, we show the results obtained during one of the harvesting experiments conducted during the experimental campaign. In particular, we show here results relatively to harvesting operations performed both with a single arm and in a cooperative dual arm mode.

It is worth noticing that the position of the peduncle has been obtained in real-time by the actual perception software developed by UPC partner (Ciarfuglia, et al., 2023) which make use of the RGB-D camera mounted on the head of the robot (see Figure 27). The desired right end-effector position and orientation trajectories are obtained by connecting a series of proper waypoints through trajectories characterized by a trapezoidal velocity profile. The waypoints list is the following (see Figure 28):

- 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 grape;
- a pre-release configuration at a certain distance from the box placed on the mobile base;
- a release configuration over the box.

In the aforementioned experiment, we refer to the case of an end-effector equipped both with scissors to cut the peduncle and fingers to grasp the grape (to be delivered in spring 2023), while the version of the end-effector mounted on the right arm of the FR used for experiments is equipped by fingers only.

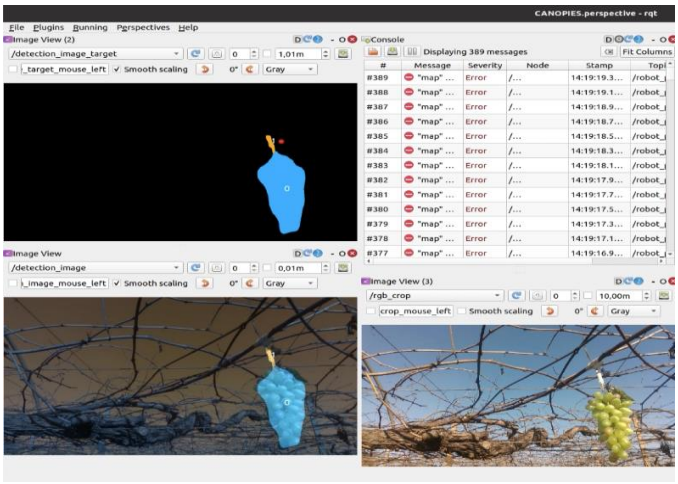


Figure 27: The output of detection and localization of both grape and peduncle (by UPC).



Figure 28: Snapshots of the single arm harvesting operation. From the left, robot in: (i) initial configuration; (ii) pre-grasping configuration; (iii) grasping configuration; (iv) release configuration.

Figure 29 shows the desired position/orientation and the position/orientation errors during the experiment. It is worth noticing that the peaks in both position and orientation errors occur when there is a change of waypoints, and they are due simply to the fact that we set a threshold for the errors below which the waypoint is considered reached. Figure 30 shows the evolution of the joint positions read by the actual encoders over time, together with the imposed limits.

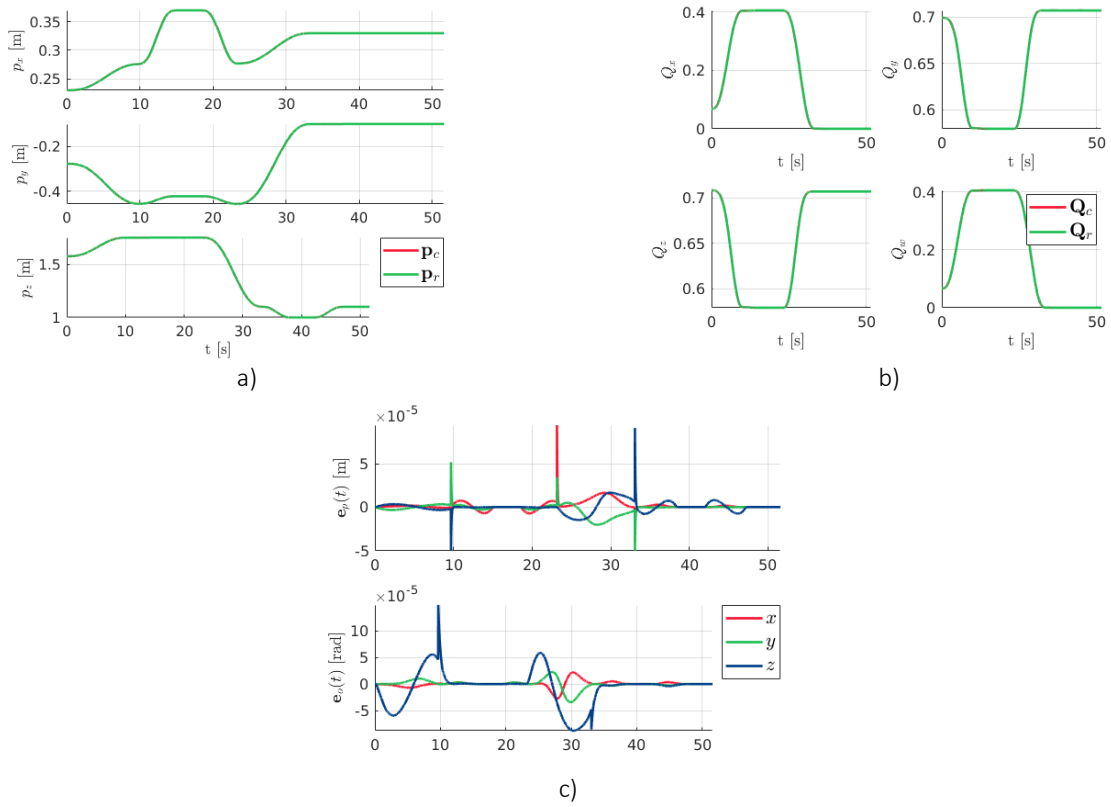


Figure 29: a) Desired position for the single arm harvesting experiment; b) Desired quaternion; c) Position and orientation errors

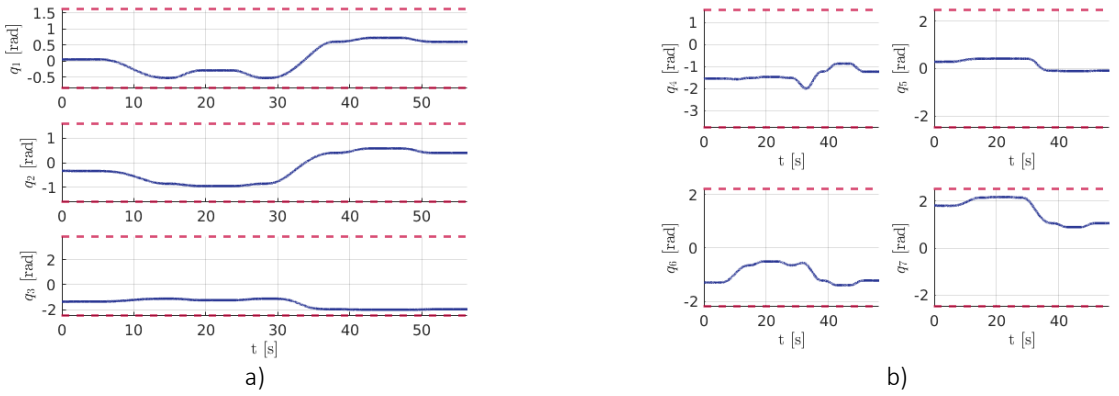


Figure 30: a) joints 1-3 positions over time and imposed minimum/maximum limits; b) joints 4-7 positions over time and imposed minimum/maximum limits.

Figure 31 shows the performance of the Intermediate-Level-Controller in Section 2.1 designed for the Farming Robot prototype. The error is relative to the right arm only (other joints are kept still in this experiment and shows similar results) and is obtained as $\boldsymbol{q}_d - \boldsymbol{q}$ (see Figure 4 and Eq. (6)). It is clear that the robot tracks the desired joint positions well-enough to successfully perform the task. Finally, Figure 32 shows some snapshots of the farming robot performing a harvesting operation.

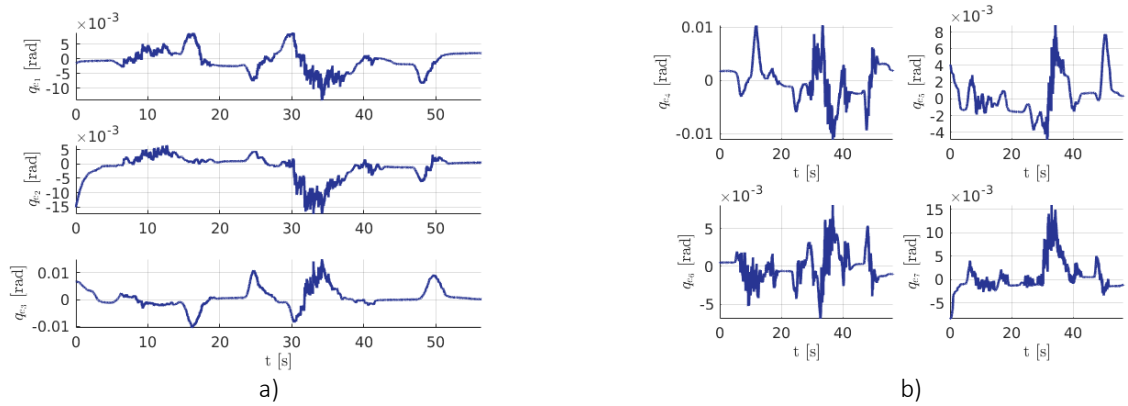


Figure 31: a) joints 1-3 position errors for the low-level controller during the experiment; a) joints 4-7 position errors for the low-level controller during the experiment.



Figure 32: Snapshots of the farming robot in a cooperative harvesting operation. Top Left: arms in the initial configuration. Top Right: arms in pre-grasping position. Bottom Left: arms in post-grasping position. Bottom Right: right arm in releasing grape configuration.

4 Reference to Web Content

Title	Description	URL
Simulations.mp4	Video of the simulations described in Section 3.1.2	https://tinyurl.com/canopies-d5-1-sim
Laboratoryexperiments.mp4	Video of the experiments described in Section 3.2.1 and Section 3.2.2	https://tinyurl.com/canopies-d5-1-lab
FieldExperiments.mp4	Video of the field experiments described in Section 3.2.2	https://tinyurl.com/canopies-d5-1-field

5 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

6 Bibliography

Almeida, D., & Karayiannidis, Y. (2019). Asymmetric dual-arm task execution using an extended relative jacobian. *arXiv preprint arXiv:1905.01248*.

Caccavale, F., & Uchiyama, M. (2016). Cooperative manipulation. *Springer Handbook of Robotics*.

Chiacchio, P., Chiaverini, S., & Siciliano, B. (1996). Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system. *Journal of Dynamic Systems Measurement and Control*.

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

Ciarfuglia, T., Motoi, I., Saraceni, L., Fawakherji, M., Sanfeliu, A., & Nardi, D. (2023). Weakly and semi-supervised detection, segmentation and tracking of table grapes with limited and noisy data. *Computers and Electronics in Agriculture*.

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).*

Di Vito, D., Natale, C., & Antonelli, G. (2017). A Comparison of Damped Least Squares Algorithms for Inverse Kinematics of Robot Manipulators. *IFAC-PapersOnLine.*

Escande , A., Mansard, N., & Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 1006-1028.

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

Jiang, J. (2022). A Robust Visual Servoing Controller for Anthropomorphic Manipulators With Field-of-View Constraints and Swivel-Angle Motion: Overcoming System Uncertainty and Improving Control Performance. *Robotics and Automation Magazine*.

Moe, S., Antonelli, G., Teel, R., Pettersen, K., & Schrimpf, J. (2016). Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results. *Frontiers in Robotics and AI*.

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

Simetti, E., & Casalino, P. (2016). A Novel Practical Technique to Integrate Inequality Control Objectives and Task Transitions in Priority Based Control. *Journal of Intelligent & Robotic Systems*.