DARKO

Dynamic Agile Production Robots That Learn and Optimise Knowledge and Operations

H2020-ICT-2020-2 Grant agreement no: 101017274

DELIVERABLE 4.2

Planning and control algorithms for efficient manipulation

Dissemination Level: PUBLIC

Due date: month 48 (May 2025)

Deliverable type: Report

Lead beneficiary: UNIPI

1 Introduction

The design of suitable planning and control algorithms for robotic manipulators is a key factor for enabling efficient manipulation in unstructured environments. In fact, these scenarios present several challenges like uncertainties on inertia parameters due to grasped objects, guaranteeing a safe interaction with the environment, and performing movements as predictable as possible by human workers who share the workspace with the robot. Furthermore, all these problems have to be dealt without losing the efficiency in task performances.

This deliverable reports the latest results we achieved in the DARKO project regarding motion planning and control of the manipulator. The document starts with the control part, where we start with the presentation of *Thunder Dynamics*, a software library for the implementation of adaptive control for robotic manipulators, and then we move to describe a novel formulation of the Adaptive Computed Torque Control which permits to avoid numerical problems given by the inversion of the estimated mass matrix. Then, we present an approach to deal with the presence of elastic joints in the manipulator, like elastic wrist available in the DARKO platform. After that, we move to the presentation of a novel motion planning algorithm for generating human-like movements in artificial systems. We start from the classical obstacle avoidance problem, and then we show the application of this approach for the grasping of moving objects. In the end, we report the framework developed for performing picking action with the pneumatic tools. For the sake of clarity, each section is related to a single work and contains an introduction which contextualize the presented results, providing to the reader all the necessary background information.

2 Thunder Dynamics: a C++ Library for Adaptive Control of Serial Manipulators

Nowadays the usage of robotic manipulators is exponentially increasing in a large number of fields such as manufacturing, logistics, and healthcare [31, 44, 64]. This trend is driven by the exceptional precision and ability of this type of system to handle heavy loads.

Behind the level of performance achieved, there is a strong theoretical background given by the study of the dynamics of rigid bodies [33]. Over time, researchers have developed a large set of algorithms to compute different entities related to rigid body systems, such as forward and backward dynamic, in an efficient way [32]. From these results, in the last decade, different software packages implementing in an easy way all these algorithms have been released [20, 34]. Several control techniques rely on a good knowledge of the dynamic model of the manipulator requiring precise information of the mechanical properties of each link. Also in this case, if the robot picks an object to perform the desired task, the inertia of the grasped object will change the dynamic behavior of the kinematic chain.

Different strategies have been proposed in the literature to deal with the effect of uncertainties in the model. A possible way is to design controllers which provide a desired robustness to model parameters uncertainties, like for example, Robust Model Predictive Controls [14]. However they usually come with drawbacks in terms of level of performance and computational load. Another possible approach is to estimate the discrepancy between the model and the real and compensate for it. This class of approaches can be separated into two types of solutions: the model identification and the adaptive control. The first type tackles the problem *offline* by setting up an estimation procedure to identify the unknown parameters [5, 76]. The latter targets the problem *online* from a control perspective by

adapting the control action to compensate for parameter uncertainties. These types of solutions are usually called *adaptive controllers* [57, 139]. The most famous algorithm of this family is the one proposed in [120], where the control law is composed of feedback and feedforward parts, while parameters are modified online to ensure stability.

Even if this approach is effective in compensating for model errors, it requires factorization of the robot dynamic equation to express it explicitly in a linear form with respect to the inertial parameters [62]. Usually, this factorization uses the classical inertial representation of rigid bodies (10 parameters for each link of the kinematic chain). However, this set of parameters is not completely observable through the joint-space dynamic of the manipulator itself, bringing to the construction of a larger regressor. In [41], the authors addressed this problem by developing an algorithm that allows finding the minimal dynamic parametrization starting from the classical one. However, most of the computational cost is related to the construction of the regressor matrix, which is addressed in this work.

The computation of the regressor matrix, which can be easily done by inspection for manipulators with a couple of Degrees of Freedom (DoFs), becomes a daunting task for many-DoF systems. Works that address this problem are [40] and [38], where the authors have proposed algorithms to build the regressor and dynamical matrices related to the inertial parameters. In [38], authors have also proposed a library, developed in Wolfram Mathematica©, that integrates the algorithm for generating this regressor, through the Denavit-Hartenberg (D-H) representation [27], concerning serial kinematic structures. However, the existing solutions cannot be integrated in a straightforward manner into a classical software control infrastructure, usually developed in a ROS framework using languages like C++ or Python. Recently, authors in [92] have developed a high-level Python toolbox based on *Pinocchio* library. However, the proposed tool is designed for the identification and calibration of robot inertial and kinematic parameters and, to the best of my knowledge, it does not provide information usable for the online implementation of adaptive controllers. Moreover, it is not easy to add functions that are not already present in the framework.

In this work, starting from the results obtained in [38], we have formulated a procedure that does not require tensor algebra to compute the dynamic regressor and can be easily implemented with standard software languages with low computational time. Moreover, we have developed a software library, based on C++ and CasADI [4], able to provide the information needed by adaptive control or similar techniques. The proposed library has two possible use cases: a high-level and a low-level one. The high-level one allows a non-specialized user to build a C++ system model that can be directly incorporated into the control loop. This implementation is the easiest and the fastest to use, due to the efficiency of the compiled C++ code. The low-level use case allows an experienced user to modify the internal model, written in CasADi, to extend the functionalities of the library or to solve user-defined problems. Furthermore, the CasADi base allows future implementation of control optimization and planning in a straightforward manner. Finally, the entire framework is developed inside a Docker Image to improve usability [16]. To test the effectiveness of this method, we exploited a task-space adaptive backstepping controller and we used this tool to implement the control algorithm for the 7 DoFs manipulator *Franka Emika*. This controller is applied both in simulation and to the real system in different scenarios with the aim of testing the capability of the overall system to handle the inertial model, both in terms of trajectory tracking error, and inertial parameter estimation. The results presented in this section are part of the manuscript "*Thunder Dynamics: a C++ Library for Adaptive Control of Serial Manipulators*", currently under review at IEEE Open Journal of Industrial Electronics Society.

2.1 Theorethical framework

2.1.1 Direct Formulation of the Slotine-Li Regressor

The manipulator regressor $Y(q,\dot{q},\ddot{q})$ is a matrix function employed to write the manipulator dynamics linearly in the inertial parameters π, i.e.,

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Y(q,\dot{q},\ddot{q})\pi = \tau, \tag{1}$$

where $q,\dot{q},\ddot{q} \in \mathbb{R}^n$ are the joint position, velocity and acceleration; $B(q), C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ are the inertia and Coriolis matrices; $G(q) \in \mathbb{R}^n$ is the gravitational force vector, and τ are the joint torques.

Starting from the standard formulation, the one used by Slotine-Li in [120] can be reported as:

$$B(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q) = Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)\pi, \tag{2}$$

where $\dot{q}_r(t)$ is the reference velocity generated for the adaptive control law and $C(q,\dot{q})$ has to be obtained via the Christoffel symbols. The difference between these two formulations is that the last regressor matrix does not need acceleration signals which are difficult to obtain.

To provide background context to this work, we briefly recall some of the results reported in [38] necessary to establish the notation. For a complete explanation of how these results are obtained, we refer the interested reader to [38]. Starting from equation (2), the term $B(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r$ can be written in the following element-wise form:

$$[B(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r]_h = \sum_{j=1}^{n} b_{hj}\ddot{q}_{r_j} + \sum_{j=1}^{n}\sum_{k=1}^{n}\frac{1}{2}\left(\frac{\partial b_{hj}}{\partial q_k} + \frac{\partial b_{hk}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_h}\right)\dot{q}_k\dot{q}_{r_j}. \tag{3}$$

The terms in (3) can be cast in tensor notation as

$$B(q)\ddot{q}_r + \left(\frac{1}{2}\left(\frac{\partial B}{\partial q} + \left(\frac{\partial B}{\partial q}\right)^{T_{132}} - \left(\frac{\partial B}{\partial q}\right)^{T_{231}}\right)\dot{q}\right)\dot{q}_r. \tag{4}$$

Here, the generalized transpose operator was employed which is defined for tensors of arbitrary order. If we let $(D)_{i_1,\ldots,i_j,\ldots,i_k} = d_{i_1,\ldots,i_j,\ldots,i_k}$ be a tensor of order k, the operator $(\cdot)^{T_{n_1,\ldots,n_j,\ldots,n_k}}$ allows to specify an arbitrary reordering to apply t the indices of a tensor, such that if $\tilde{D} = D^{T_{n_1,\ldots,n_j,\ldots,n_k}}$ the j-th level in $\tilde{D}$ corresponds to the n_j-th level in D, as follows

$$(\tilde{D})_{i_1,\ldots,i_j,\ldots,i_k} = d_{i_{n_1},\ldots,i_{n_j},\ldots,i_{n_k}}. \tag{5}$$

Given that, from the Lagrangian formulation of standard dynamic for a serial robot, $B(q)$ and $C(q,\dot{q})$ are link-wise additives, i.e.,

$$B(q) = \sum_{i=1}^{n} B(q)^{(i)}, \qquad C(q,\dot{q}) = \sum_{i=1}^{n} C(q,\dot{q})^{(i)}, \tag{6}$$

the terms can be expanded in

$$\begin{aligned} B(q)^{(i)} &= (J_{v_i}^T J_{v_i})m_i \\ &+ \left(J_{\omega_i}^T\, {}^0R_i Q\, {}^0R_i^T J_{v_i} - J_{v_i}^T\, {}^0R_i Q\, {}^0R_i^T J_{\omega_i}\right) m_i l_{iG_i} \\ &+ \left(J_{\omega_i}^T\, {}^0R_i E\, {}^0R_i^T J_{\omega_i}\right)\bar{J}_i, \end{aligned} \tag{7}$$

$$C(q,\dot{q})^{(i)} = \frac{1}{2}\left(\frac{\partial B(q)^{(i)}}{\partial q} + \left(\frac{\partial B(q)^{(i)}}{\partial q}\right)^{T_{132}} - \left(\frac{\partial B(q)^{(i)}}{\partial q}\right)^{T_{231}}\right)\dot{q}, \tag{8}$$

where $\bar{J}_i = [\bar{J}_{ixx}, \bar{J}_{ixy}, \bar{J}_{ixz}, \bar{J}_{iyy}, \bar{J}_{iyz}, \bar{J}_{izz}]^T$ is the vector containing the elements of the inertial tensor, $Q \in \mathbb{R}^{3\times3\times3}$ and $E \in \mathbb{R}^{3\times3\times6}$ are defined as:

$$Q = [Q_1 \quad Q_2 \quad Q_3], \qquad E = \begin{bmatrix} E_1 & E_2 & E_3 & E_4 & E_5 & E_6 \end{bmatrix},$$

$$Q_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad Q_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \qquad Q_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

$$E_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad E_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad E_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

To better highlight the influence of the various inertia terms, we start rewriting (7) and (8) in a factorized form as:

$$\begin{aligned} B(q)^{(i)} &= B_0^{(i)}\pi_0^{(i)} + B_1^{(i)}\pi_1^{(i)} + B_2^{(i)}\pi_2^{(i)}, \\ C(q,\dot{q})^{(i)} &= C_0^{(i)}\pi_0^{(i)} + C_1^{(i)}\pi_1^{(i)} + C_2^{(i)}\pi_2^{(i)}, \end{aligned}$$

where

$$\begin{aligned} B_0^{(i)} &= (J_{v_i}^T J_{v_i}), \\ B_1^{(i)} &= \left(J_{\omega_i}^T \, {}^0R_i Q \, {}^0R_i^T J_{v_i} - J_{v_i}^T \, {}^0R_i Q \, {}^0R_i^T J_{\omega_i}\right), \\ B_2^{(i)} &= \left(J_{\omega_i}^T \, {}^0R_i E \, {}^0R_i^T J_{\omega_i}\right), \\ C_j^{(i)} &= \frac{1}{2}\left(\frac{\partial B_j^{(i)}}{\partial q} + \left(\frac{\partial B_j^{(i)}}{\partial q}\right)^{T_{132}} - \left(\frac{\partial B_j^{(i)}}{\partial q}\right)^{T_{231}}\right)\dot{q}, \quad j = 0,1,2. \end{aligned} \tag{9}$$

It is worth noting that $B_0^{(i)}, C_0^{(i)} \in \mathbb{R}^{n\times n}$, $B_1^{(i)}, C_1^{(i)} \in \mathbb{R}^{n\times3\times n}$, $B_2^{(i)}, C_2^{(i)} \in \mathbb{R}^{n\times6\times n}$. Highlighting the linear dependencies of the inertia parameters we obtain

$$\begin{aligned} &\left(B_0^{(i)}\ddot{q}_r + C_0^{(i)}\dot{q}_r + Z_0^{(i)}\right)\pi_0^{(i)} \\ &+ \left(B_1^{(i)}\ddot{q}_r + C_1^{(i)}\dot{q}_r + Z_1^{(i)}\right)\pi_1^{(i)} + \left(B_2^{(i)}\ddot{q}_r + C_2^{(i)}\dot{q}_r\right)\pi_2^{(i)} \\ &= Y_0^{(i)}\pi_0^{(i)} + Y_1^{(i)}\pi_1^{(i)} + Y_2^{(i)}\pi_2^{(i)}. \end{aligned} \tag{10}$$

where $Z_0^{(i)}, Z_1^{(i)}$ are derived from the potential energy term, i.e.,

$$\left[\frac{\partial U^{(i)}}{\partial q}\right]^T = Z_0^{(i)} m_i + Z_1^{(i)} m_i l_{iG_i}. \tag{11}$$

where m_i are link masses and l_{iG_i} are the distance from the links to the centers of mass. The equation (10) can be expanded by splitting Q and E into every part of the regressor.

$$Y_r^{(i)} = \left[\, Y_0{}^{(i)} \middle| {}^1Y_1{}^{(i)} \middle| {}^2Y_1{}^{(i)} \middle| {}^3Y_1{}^{(i)} \middle| {}^1Y_2{}^{(i)} \middle| \ldots \middle| {}^6Y_2{}^{(i)} \,\right] \tag{12}$$

$$
\begin{aligned}
Y_0^{(i)} &= B_0^{(i)}\ddot{q}_r + C_0^{(i)}\dot{q}_r + Z_0^{(i)} \\
{}^{h}Y_1^{(i)} &= B(Q_h)_1^{(i)}\ddot{q}_r + C(Q_h)_1^{(i)}\dot{q}_r + {}^{h}Z_1^{(i)} && h = 1,2,3 \\
{}^{k}Y_2^{(i)} &= B(E_k)_2^{(i)}\ddot{q}_r + C(E_k)_2^{(i)}\dot{q}_r && k = 1,2,...,6
\end{aligned}
$$

With this notation, we move from third-order tensor to matrix algebra: indeed $B_0^{(i)}$, $C_0^{(i)}$, $B(Q_h)_1^{(i)}$, $C(Q_h)_1^{(i)}$, $B(E_k)_2^{(i)}$, $C(E_k)_2^{(i)} \in \mathbb{R}^{n\times n}$.

Equation (12) shows the columns of the regressor matrix of each link. The complete regressor can be written juxtaposing regressor blocks $Y_r^{(i)}$ as

$$
Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r) = \left[Y_r^{(1)}\middle|Y_r^{(2)}\middle|\ldots\middle|Y_r^{(n)}\right] \tag{13}
$$

The following step is the computation of the Coriolis matrix. One of the main difficulties encountered in computing this matrix is the derivative of the inertia matrix and the rearrangements of its elements. The first problem is easily overcome using the symbolic framework provided by the CasADi library [4]. The second is widely analyzed in literature and many techniques are proposed as in [38]. However, these cannot be easily integrated into standard control software for real hardware. The method proposed in this section is based on creating a specific matrix to select the necessary terms of mass matrix derivative to construct the Coriolis matrix. This avoids the use of tensor algebra which is not supported in many computer algebra systems.

To explain the new algorithm it is necessary to define some notations. Considering a matrix $A \in \mathbb{R}^{n\times m}$ and a vector $v \in \mathbb{R}^p$, Table 1 shows the utilized notation.

$$
A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{bmatrix}
$$

$a_{ij} = A(i,j)$	element in i-th row and j-th column of matrix A
$v_i = v(i)$	element in i-th position of vector v
A^{r_i}	i-th row of matrix A
A^{c_j}	j-th column of matrix A
$A_v \triangleq \dfrac{\partial A}{\partial v}$	derivative of matrix A with respect to v
${}^{n,m}[A]$	or *reshape*(A,p,q), command to reorganize the elements of a matrix, where $p\cdot q = n\cdot m$

Table 1: Summary of the mathematical notation used in this section.

In low-level programming languages, such as C++, matrices are typically represented as arrays of arrays. Alternatively, they can be seen as a sequence of pointers, each pointing to the memory addresses where the matrix elements are stored. Certain matrix operations can be more straightforward when the matrix is treated as a one-dimensional vector, which is achieved by reshaping the original matrix structure:

$$
{}^{1,n\cdot m}[A] = \left[\alpha_1, \alpha_2, \ldots, \alpha_k, \ldots \alpha_{n\cdot m}\right]
$$

The reshaping process involves converting between matrix and vector forms, where the conversion mapping follows this pattern:

$$
\begin{aligned}
A(i,j) &= \alpha_{i+(j-1)n}, \\
\alpha_k &= A(k-(\lceil k/n\rceil - 1)n, \lceil k/n\rceil),
\end{aligned} \tag{14}
$$

where $\lceil \cdot \rceil$ is the ceiling function. The derivative of matrix A with respect to v in a low-level programming language is usually defined as a matrix $A_v \in \mathbb{R}^{(n\cdot m)\times p}$:

$$\frac{\partial A}{\partial v} = A_v \triangleq \begin{bmatrix} \frac{\partial a_{11}}{\partial v_1} & \cdots & \frac{\partial a_{11}}{\partial v_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{ij}}{\partial v_1} & \cdots & \frac{\partial a_{ij}}{\partial v_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{nm}}{\partial v_1} & \cdots & \frac{\partial a_{nm}}{\partial v_p} \end{bmatrix} = \begin{bmatrix} \frac{\partial \alpha_1}{\partial v_1} & \cdots & \frac{\partial \alpha_1}{\partial v_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_k}{\partial v_1} & \cdots & \frac{\partial \alpha_k}{\partial v_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_{n\cdot m}}{\partial v_1} & \cdots & \frac{\partial \alpha_{n\cdot m}}{\partial v_p} \end{bmatrix}$$

The Coriolis matrix, written using Christoffel symbols, can be re-arranged as (15):

$$\begin{aligned} C_{ij} &= \sum_{k=1}^{n} \Gamma_{ijk}\dot{q}_k = \sum_{k=1}^{n} \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k \\ &= \sum_{k=1}^{n} \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k \\ &= \frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{\partial b_{ij}}{\partial q_k}\right)\dot{q}_k + \sum_{k=1}^{n}\left(\frac{\partial b_{ik}}{\partial q_j}\right)\dot{q}_k - \sum_{k=1}^{n}\left(\frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k\right) \\ &= \frac{1}{2}\left[{}^{I}\mathfrak{C}_{ij} + {}^{II}\mathfrak{C}_{ij} - {}^{III}\mathfrak{C}_{ij}\right] \end{aligned} \tag{15}$$

Bearing in mind the notation of this work:

$$\frac{\partial B}{\partial q} = B_q = \begin{bmatrix} \frac{\partial \beta_1}{\partial q_1} & \cdots & \frac{\partial \beta_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \beta_{n\cdot n}}{\partial q_1} & \cdots & \frac{\partial \beta_{n\cdot n}}{\partial q_n} \end{bmatrix} \in \mathbb{R}^{(n\cdot n)\times n}$$

The elements of (15) become:

$${}^{I}\mathfrak{C}_{ij} = \sum_{k=1}^{n}\left(\frac{\partial \beta_{h_1}}{\partial q_k}\right)\dot{q}_k = B_q^{r_{h_1}}\dot{q} \tag{16}$$

where $h_1 = i + (j-1)n$;

$$\begin{aligned} {}^{II}\mathfrak{C}_{ij} &= \sum_{k=1}^{n}\left(\frac{\partial \beta_{h_2}}{\partial q_j}\right)\dot{q}_k = \left({}^{n,n}\left[\beta_{q_j}\right]\right)^{r_i}\dot{q} \\ &= \sum_{k=1}^{n} B_q^{c_j}(h_2)\cdot\dot{q}_k = \sum_{k=1}^{n}\mathfrak{Q}^{r_i}(h_2)B_q^{c_j}(h_2) \end{aligned} \tag{17}$$

where $h_2 = i + (k-1)n$ and $\mathfrak{Q}^{r_i}$ is the r_i-th row of the matrix $\mathfrak{Q} \in \mathbb{R}^{n\times(n\cdot n)}$ defined as

$$\mathfrak{Q} = [I_{n\times n}\dot{q}_1 | \ldots | I_{n\times n}\dot{q}_n] \in \mathbb{R}^{n\times(n\cdot n)} \tag{18}$$

so that

$${}^{II}\mathfrak{C}_{ij} = \mathfrak{Q}B_q. \tag{19}$$

Noting that

$$^{III}\mathfrak{C}_{ij} = {}^{II}\mathfrak{C}_{ji}, \tag{20}$$

the Coriolis matrix becomes

$$C(q,\dot{q}) = \frac{1}{2}\left({}^{n,n}\left[B_q\dot{q}\right] + \mathfrak{Q}B_q - B_q^T\mathfrak{Q}^T\right) \tag{21}$$

2.1.2 Task-space backstepping control

In this section, we present a complete control law, in position and orientation, for the end-effector tracking which will be used to test the library. The presented approach is based on the more general adaptive backstepping control reported in [70]. The obtained closed-loop system is asymptotically stable also in the presence of uncertainties in dynamical parameters. In the literature, various works exploit backstepping theory to develop adaptive task-space control for robotic manipulators. For example, in [96], the authors use the theory of backstepping to develop an adaptive control law to follow a position path in the Cartesian space with the end-effector of a mobile manipulator. A similar problem is taken in [142] for a parallel manipulator with a fixed base. [90] uses an adaptive backstepping control for the position tracking of a manipulator taking into consideration also the motor currents. However, to the best of our knowledge, no work explicitly deals with Cartesian orientation in presenting the proof of this approach. For this reason, for the sake of completeness, in this section we report the proof taking into account explicitly the orientation error in terms of axis-angle representation.

It is instrumental for the understanding of the following parts to introduce the orientation error by the axis-angle representation, as exploited in [115]. We define the frames

- $\{S\}$: fixed frame
- $\{B\}$: actual end-effector frame
- $\{B_d\}$: desired end-effector frame

and consequently the following rotation matrices ($\in SO(3)$)

$R = [n\,s\,a]$: rotation of $\{B\}$ with respect to $\{S\}$
$R_d = [n_d\,s_d\,a_d]$: rotation of $\{B_d\}$ with respect to $\{S\}$
$\tilde{R} = R_d R^T$: rotation of $\{B_d\}$ with respect to $\{B\}$ in coordinates $\{S\}$

where $n, s, a, n_d, s_d, a_d \in \mathbb{R}^3$. Moreover, it has to be defined the *skew* $\hat{v}$ and the *vect* $V^{\vee}$ operators. If we apply the *skew* operator to the vector $v = [v_x, v_y, v_z]^T \in \mathbb{R}^3$, we obtain the following matrix

$$\hat{v} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \tag{22}$$

that verify $(uw^T)^{\vee} = \frac{1}{2}\hat{w}u$. On the other hand, if we apply the *vect* operator to a matrix V that is antisymmetric, we obtain the vector $V^{\vee} = [v_x, v_y, v_z]^T \in \mathbb{R}^3$.

We am looking for an error such that is zero when $\tilde{R} = I$ and $R = R_d$. Bearing this in mind, we define orientation error as:

$$e_o \triangleq r\sin(\theta), \tag{23}$$

where r and θ are angle-axis parametrization of $\tilde{R}$. As mentioned in [132], it is known that r and θ can be extracted from the matrix $\tilde{R}$, i.e.,

$$r = \frac{1}{2\sin(\theta)}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}, \quad \cos(\theta) = \frac{r_{11}+r_{22}+r_{33}-1}{2}. \tag{24}$$

Noting that e_o is the antisymmetric part of the matrix $\tilde{R}$, it is possible to write

$$\begin{aligned} e_o &= (n_d n^T + s_d s^T + a_d a^T)^{\mathrm{v}} \\ &= \frac{1}{2}(\hat{n} n_d + \hat{s} s_d + \hat{a} a_d). \end{aligned} \tag{25}$$

For the next steps, it is useful to introduce the Poisson Theorem [131]

$$\dot{R} = \hat{\omega} R \tag{26}$$

that permits to write $[\dot{n}\,\dot{s}\,\dot{a}] = \hat{\omega}[n\,s\,a]$, where ω is the angular velocity expressed in $\{S\}$ frame components. Moreover, the same relation also applies to the desired quantities R_d, ω_d. Then, the orientation error derivative $\dot{e}_o$ from (25) becomes

$$\begin{aligned} \dot{e}_o =& \frac{1}{2}(\hat{\dot{n}} n_d + \hat{\dot{s}} s_d + \hat{\dot{a}} a_d + \hat{n}\dot{n}_d + \hat{s}\dot{s}_d + \hat{a}\dot{a}_d) \\ =& \frac{1}{2}((\hat{\omega} n) \times n_d + (\hat{\omega} s) \times s_d + (\hat{\omega} a) \times a_d \\ &+ n \times (\hat{\omega}_d n_d) + s \times (\hat{\omega}_d s_d) + a \times (\hat{\omega}_d a_d)) \end{aligned}$$

and using the antisymmetry property of the cross-product, we obtain

$$\begin{aligned} \dot{e}_o =& \frac{1}{2}(\hat{n}_d \hat{n} + \hat{s}_d \hat{s} + \hat{a}_d \hat{a})\omega - \frac{1}{2}(\hat{n}\hat{n}_d + \hat{s}\hat{s}_d + \hat{a}\hat{a}_d)\omega_d \\ =& L_o^T \omega_d - L_o \omega \end{aligned} \tag{27}$$

where

$$L_o \triangleq -\frac{1}{2}(\hat{n}_d \hat{n} + \hat{s}_d \hat{s} + \hat{a}_d \hat{a}). \tag{28}$$

In a serial manipulator $\omega = J_o(q)\dot{q}$, so we can write:

$$\dot{e}_o = L_o^T \omega_d - L_o J_o(q)\dot{q} \tag{29}$$

where $J_o(q)$ is the orientation part of geometric Jacobian. As regards position error, it can be defined as

$$e_p = p_d - p. \tag{30}$$

Consider a task where the goal is to follow a desired trajectory in Cartesian space defined by position and orientation. Defining the position error e_p as (30) and the orientation error e_o as (23), the complete system (1) can be rewritten in

$$\dot{e} = \begin{bmatrix} \dot{e}_p \\ \dot{e}_o \end{bmatrix} = L^T \xi_d - LJ\dot{q} \tag{31}$$

$$\ddot{q} = B^{-1}(q)\big(\tau - C(q,\dot{q})\dot{q} - G(q)\big), \tag{32}$$

where, for compactness, the Jacobian conversion matrix $L \in \mathbb{R}^{6\times 6}$, and the desired end-effector twist $\xi_d \in \mathbb{R}^6$ are defined as follows

$$L \triangleq \begin{bmatrix} I & 0 \\ 0 & L_o \end{bmatrix}, \qquad \xi_d \triangleq \begin{bmatrix} \dot{p}_d(t) \\ \omega_d(t) \end{bmatrix}. \tag{33}$$

The overall control problem can be divided into three steps: 1) kinematic control, 2) dynamic control, and 3) adaptive control. First, in Proposition 1, the design of a kinematic control law generates joint velocity references $\dot{q}_r \in \mathbb{R}^n$ that ensures the end-effector position and orientation tracking. Then, in Proposition 2, the dynamic control law generates the necessary joint torque command to follow the desired joint velocity computed by the kinematic part. Finally, the entire loop is closed with inertial uncertainties in Proposition 3.

Proposition 1 (Kinematic control asymptotic stability)**.** *Consider the kinematic error model* (31)*. Let* $e \triangleq [e_p^T, e_o^T]^T$ *be the Cartesian pose error as defined in* (30)*,* (23)*. If joint velocities are as in*

$$\dot{q} = \dot{q}_r \triangleq J^{\dagger} L^{-1}\big(L^T \xi_d + \Lambda e\big), \tag{34}$$

where $\Lambda \in \mathbb{R}^{6\times 6}$ *is a symmetric positive definite matrix and* $J^{\dagger}$ *is the Moore-Penrose inverse of the Jacobian matrix, the origin is an asymptotically stable (A.S.) equilibrium for the kinematic error dynamics* (31)*.*

Proof. Taking the candidate Lyapunov function

$$V(e) = \frac{1}{2} e^T K_p e, \tag{35}$$

with K_p positive definite and time-invariant matrix. The time derivative of is

$$\begin{aligned} \dot{V}(e) =& e^T K_p \dot{e} = e^T K_p \big(L^T \xi_d - LJ\dot{q}\big) \\ =& e^T K_p \Big(L^T \xi_d - \cancel{LJJ^{\dagger}L^{-1}}\big(L^T \xi_d + \Lambda e\big)\Big) \\ =& - e^T K_p \Lambda e. \end{aligned} \tag{36}$$

Being $V(e)$ positive definite time-invariant and being $\dot{V}$ negative definite, we can infer asymptotic stability from the Lyapunov theorem. □

For the dynamic part, we rewrite the dynamics (31), (32) in the new state coordinates (e, s), where $s \triangleq \dot{q}_r - \dot{q}$, with $\dot{q}_r$ defined as in (34).

Proposition 2 (Dynamic control asymptotic stability)**.** *Considering the state* (e, s)*, the torque control law*

$$\tau = \tau^* \triangleq B\ddot{q}_r + C\dot{q}_r + G + K_d s + J^T K_p L^T e, \tag{37}$$

where $K_d \in \mathbb{R}^{n\times n}, K_p \in \mathbb{R}^{6\times 6}$ *are the controller gains, makes system A.S. in the origin.*

Proof. Taking the time-variant Lyapunov candidate

$$W(e, s, t) = V(e) + \frac{1}{2} s^T B(q) s, \tag{38}$$

where $V(e)$ is the Lyapunov function used in the kinematic part (35). Computing the derivative, we have

$$\begin{aligned} \dot{W} &= e^T K p \dot{e} + s^T B \dot{s} + \frac{1}{2} s^T \dot{B} s \\ &= e^T K p \dot{e} + s^T (B\ddot{q}_r - B\ddot{q}) + \frac{1}{2} s^T \dot{B} s \\ &= e^T K p \dot{e} + s^T (B\ddot{q}_r + C\dot{q} + G - \tau^*) + \frac{1}{2} s^T \dot{B} s \\ &= e^T K p \dot{e} + s^T (B\ddot{q}_r + C\dot{q}_r + G - \tau^*) + \frac{1}{2} s^T (\dot{B} - 2C) s. \end{aligned} \tag{39}$$

If $C(q,\dot{q})$ is written in Christoffel form we have that $(\dot{B}-2C)$ is skew-symmetric and so $\frac{1}{2}s^T(\dot{B}-2C)s$ is zero. At this point, we have

$$\begin{aligned}\dot{W} &= -s^T K_d s + e^T K_p \dot{e} - \left(\dot{q}_r - \dot{q}\right)^T J^T L^T K_p^T e \\ &= -s^T K_d s + \cancel{e^T K_p \dot{e}} - \left(\cancel{L^T \xi_d} + \Lambda e\right) K_p^T e + \cancel{\dot{q}^T J^T L^T K_p^T e} \\ &= -s^T K_d s - e^T K_p \Lambda e.\end{aligned} \tag{40}$$

Applying Lyapunov's theorem for a time-variant system, the system in variables (e,s) has a time-variant positive-definite candidate, $W(e,s,t)$ that is limited by K_∞ function as shown in [60] and its derivative $\dot{W}(e,s)$ is time-invariant and negative-definite. Therefore, we can deduce the condition of asymptotic stability. □

Proposition 3 (Adaptive control asymptotic stability)**.** *Let $\hat{\pi}$ be an estimate of the real inertial parameters π and define $\tilde{\pi}(t) = \pi - \hat{\pi}(t)$ as the parameters's estimation error. Moreover, let the estimated dynamical matrices be $(\hat{B},\hat{C},\hat{G})$, obtained by computing the Euler-Lagrange equation using $\hat{\pi}$ instead of the real parameters π. The control action (37)*

$$\tau = \hat{B}\ddot{q}_r + \hat{C}\dot{q}_r + \hat{G} + K_d s + J^T K_p L^T e. \tag{41}$$

and the parameter update law

$$\dot{\hat{\pi}} = u_\pi \triangleq \Gamma^{-1} Y_r^T(q,\dot{q},\dot{q}_r,\ddot{q}_r)s. \tag{42}$$

where Γ is a positive-definite gain matrix and $Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ is defined as (2)*, , lead to asymptotic stability of the closed-loop system.*

Proof. Choosing $(e,s,\tilde{\pi})$ as state-space coordinates, a new candidate Lyapunov function can be defined as

$$W_\pi(e,s,\tilde{\pi},t) = W(e,s,t) + \frac{1}{2}\tilde{\pi}^T\Gamma\tilde{\pi}. \tag{43}$$

Computing the derivative, as in (40), we have

$$\dot{W}_\pi = e^T K_p \dot{e} + s^T(B\ddot{q}_r + C\dot{q}_r + G - \tau) + \tilde{\pi}^T\Gamma\dot{\tilde{\pi}}. \tag{44}$$

where, since real parameters are fixed in time, $\dot{\tilde{\pi}} = -u_\pi$. From (2), the system dynamics is linear in the dynamic parameters, and the control (41) can be rewritten in

$$\tau = \tau^* - Y_r\tilde{\pi}. \tag{45}$$

where τ^* is the control action (37). Substituting (45) into (44), it becomes

$$\dot{W}_\pi = \dot{W} = -s^T K_d s - e^T K_p \Lambda e. \tag{46}$$

Since exists U_1, U_2, U_3 that holds

$$\begin{aligned} U_1(e,s,\tilde{\pi}) \leq & W_\pi(e,s,\tilde{\pi},t) \leq U_2(e,s,\tilde{\pi}), \\ \dot{W}_\pi(e,s) \leq & U_3(e,s), \end{aligned} \tag{47}$$

Using Theorem 8.4 from [61], it holds $\dot{W}_\pi \to 0$. From this and (46), it is

$$\lim_{t\to\infty} e(t) = 0, \qquad \lim_{t\to\infty} \dot{e}(t) = 0.$$

□

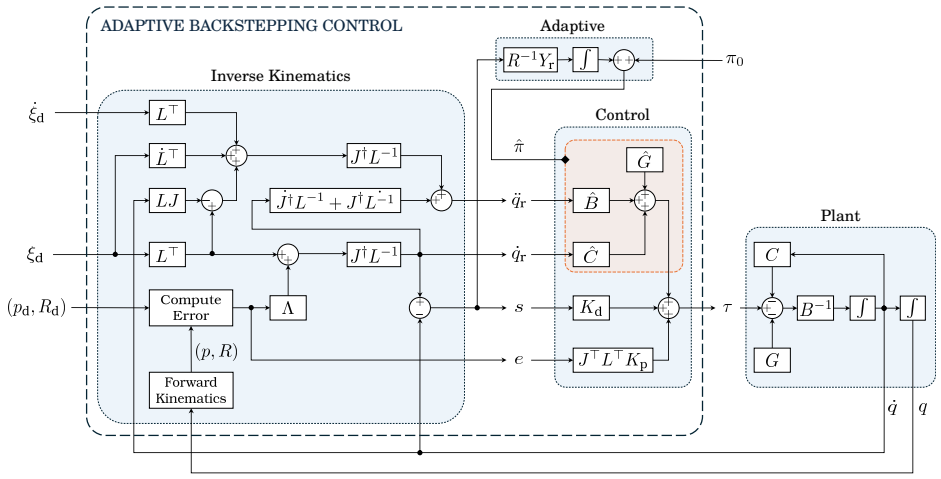


Figure 1: Overall scheme of the Adaptive Backstepping Control. The desired Cartesian trajectory serves as the input to the Inverse Kinematics block, which generates new joint references and new coordinates. These outputs are utilized by the Adaptive block for updating inertial parameters and by the Control block to ensure asymptotic stability. The Plant block completes the loop. All matrices are obtainable using the library proposed in this section.

Variable	Value	Ref.
τ	$\hat{B}\ddot{q}_r + \hat{C}\dot{q}_r + \hat{G} + K_d s + J^T K_p L^T e$	(41)
u_π	$\Gamma^{-1} Y_r^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)s$	(42)
$\dot{q}_r$	$J^\dagger L^{-1}\left(L^T \xi_d + \Lambda e\right)$	(34)
s	$\dot{q}_r - \dot{q}$	
e	$\begin{bmatrix} e_p \\ e_o \end{bmatrix} = \begin{bmatrix} p_d - p \\ r \sin\theta \end{bmatrix}$	(30),(23)

Table 2: Overall summary of the control law terms.

For the sake of clarity, we briefly report in Table 2 all the key components of the proposed control law while in Figure 1 is depicted an overall block diagram representing the control law.

Remark 1. *The control law* (41) *is similar to the one from [120], as it only differs for the term* $J^T L^T K_p^T e$*, which introduces a term proportional to the tracking error. We also explicitly include the inverse kinematic solution in the overall closed loop. The advantage of this inclusion is that it ensures the asymptotic stability of end-effector tracking, as opposed to the case when the adaptive controller tracks the solution in the joint space provided by a separate external inverse kinematic solver. In this sense, it is possible to view adaptive backstepping as a slight generalization of classical adaptive control techniques mainly used in literature.*

Remark 2. *In the case that not all parameters have uncertainties, it is possible to split the equation* (78) *into known* ($Y_k \pi_k$) *and unknown* ($Y_u \pi_u$) *parts, and reduce the number of symbolic parameters. However, the resulting equation is no longer linear in the unknown parameters* π_u*. A possible solution is adding* 1 *to the new parameters representing the known dynamics, i.e.,*

$$\tau = Y_k \pi_{dk} + Y_u \pi_u = \begin{bmatrix} Y_k \pi_{dk} & Y_u \end{bmatrix} \begin{bmatrix} 1 \\ \pi_{du} \end{bmatrix} \tag{48}$$

This method is used in the library to have symbolic selectivity and gain performances when a

parameter is perfectly known.

2.2 Software package description

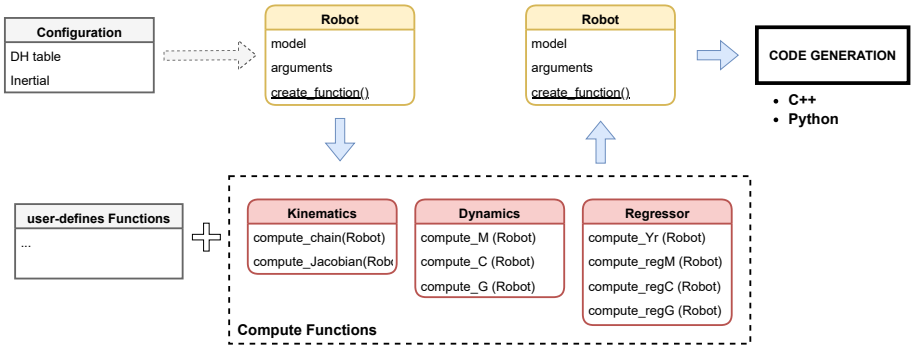


Figure 2: Simplified UML diagram of the framework. The main program takes a configuration file as input, creates the robot, and fills it with the computing functions. Then the code is generated on a C++ library.

In this section, we briefly describe the library. At the actual state, the library includes, among all the classical terms for a serial manipulator: forward kinematics, Jacobians, dynamical matrices, the dynamic regressor, and the regressor of each part of the dynamic equation $(M(q)\ddot{q}_r, C(q,\dot{q})\dot{q}_r, G(q))$. The framework exhibits a continuous commitment to evolution, with ongoing updates that introduce additional functionalities. It is important to say that adding features that have to be in the generated C++ library is very easy. The user has to create a function that adds the desired features to the robot, and then the code generation and the Python bindings are automatic. The software package is open-source and freely available[1]. For a detailed explanation regarding the usage of this library, we refer the interested reader to the documentation inside the repository.

The framework offers a command-line utility designed for the creation of a customized C++ library tailored to a specific robotic platform. To generate the library files for the custom robot, the package requires a configuration file (*YAML*) housing the D-H table and an initial estimate of inertia parameters. The only action required by the user is to execute the command:

```
./thunder gen <path>/<robot>.yaml
```

where *<path>* is the relative path from the thunder binary to the folder containing the .yaml configuration file, and *<robot>* is the robot name. The name will be used to create library files. Upon execution of this command, the framework generates the specified library within the folder *generatedFiles*. These files can be included in the control system of the specific robot and can be used to obtain the desired quantities. Moreover, it created automatically the Python bindings of the generated C++ library, allowing versatility and efficiency at the same time.

The proposed library has two possible use cases: a high-level and a low-level one. The high-level one allows a non-specialized user to build a C++ system model that can be directly incorporated into the control loop. This implementation is the easiest and the fastest to use, due to the efficiency of the compiled C++ code. The low-level use case allows an experienced user to modify the internal model, written in CasADi, to extend

[1] https://github.com/CentroEPiaggio/thunder_dynamics.git

the functionalities of the library or to solve user-defined problems. All the parameters that represent the robot can be expressed in a symbolic or numerical way. This permits building the robot structure and choosing which parameters have to be inputs or not at the high-level interface. Furthermore, the CasADi base allows future implementation of control optimization and planning in a straightforward manner. Finally, the entire framework is developed inside a Docker Image to improve usability [16].

2.3 Experimental Validation

In this section, we provide both the theoretical and the technical validation of this framework. We start evaluating the computational time required by our software package to handle kinematic structures with increasing complexities. Then, we apply this library to design the task-space adaptive backstepping controller described in the previous section for a Franka Emika Panda, a 7-DoF manipulator. Furthermore, we test it both in simulation and in real-world scenarios.

2.3.1 Execution time comparison

To evaluate the advantages of the new algorithm we focus on the execution time as a comparison parameter. Figure 3 shows the comparisons of the time needed to compute the Coriolis matrix and the regressor for different DoF serial manipulators. To perform this test we used a PC with an 11th Gen Intel®Core™ i7-1165G7. The main differences between algorithms are how to compute the Coriolis terms: the "C_{std}" uses the classic approach with Christoffel symbols, "C" uses the algorithm proposed in this section, the last three (C_{built}, C_{std_built}, Y_{built}) are obtained from C-code generation function, made available by CasADi, of the first two algorithms and of the complete regressor matrix Y_r.

As can be seen in Figure 3, the proposed approach to compute the Coriolis matrix shows computational times up to one order of magnitude below the classical method. This result leads to obtain a computational time for the whole regressor matrix Y_r lower than the one needed for C_{std} with the increasing of the number of DoF involved. In this test, we show also that the built code is faster by more than two orders of magnitude with respect to the direct use of CasADi symbols. Another important result is that the required time to evaluate these quantities meets the usual computational constraints in the implementation of torque control in robotic manipulators. In fact, in Figure 3, it can be seen that the evaluation time of the regressor always remains below $1\,ms$, which is coherent with the control loop frequency required in standard manipulators (generally around 1 kHz).

Regarding the time needed to create the symbolic robot and compute the standard quantities, and the time used to create the C++ library, they are resumed in Table 3. It is important to note that, despite it is easy to create the C++ library for a high DoF robot, problems can arise at compilation time because files are very large. In this case, for 15 and 30 DoF the generated .cpp files have sizes of 200 MB and 1.7 GB respectively.

DoF	3	5	7	9	15	30
Robot creation [ms]	23	81	326	954	5582	18454
Library creation [s]	0.6	0.8	1.1	2.3	12.1	155.1

Table 3: Time comparison for the robot and library creation

2.3.2 Adaptive control implementation

To test the proposed approach, we exploit the developed library to design a ROS package for adaptive control. This software package is developed for *Franka Emika Panda* manipulator

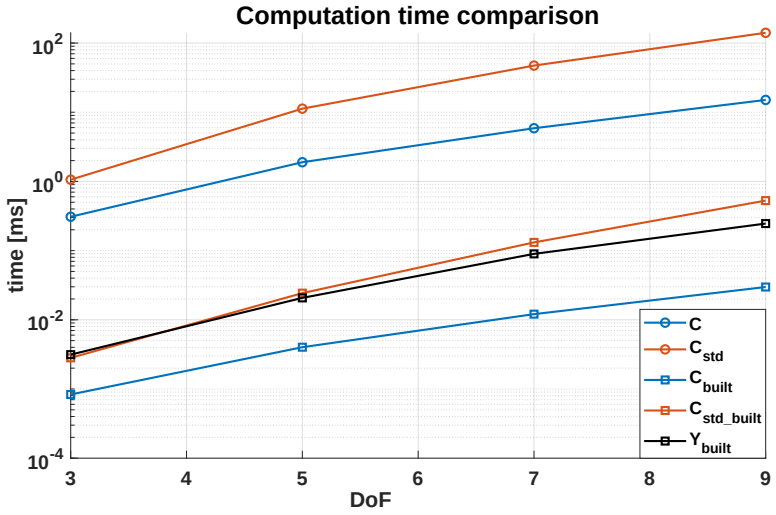


Figure 3: Execution time comparison of different algorithms. The proposed algorithm is generally faster and permits the real-time adaptive control of a 7 DoF manipulator

link_i	a_i	α_i	d_i	θ_i
1	0	0	0.3330	0
2	0	-$\pi/2$	0	0
3	0	$\pi/2$	0.3160	0
4	0.0825	$\pi/2$	0	0
5	-0.0825	-$\pi/2$	0.384	0
6	0	$\pi/2$	0	0
7	0.088	$\pi/2$	0.107	0

Table 4: Denavit-Hartenberg Table of Franka Emika Panda.

and it is available online in the adaptive branch of the repository *panda_controllers*[2]. The generated code of the new algorithm proposed in this work satisfies the strict constraints of real-time scheduling that require the control loop to be $1Khz$.

To use the direct formulation of the Slotine-Li regressor, we have to describe the robot in the Denavit-Hartenberg (D-H) parametrization. Figure 4 shows how frames of each link are chosen, in agreement with D-H rules, and the related D-H table is provided in Table 4. Dimensions of the robot and parameters are provided by *Franka Control Interface*[3].

The validation procedure can be divided into three phases. The first step is the implementation and testing of the control algorithm in simulation. This part is fundamental not only to perform a preliminary evaluation of the performance of the controller but also to select the best set of gains to guarantee the right motion behavior of the robotic arm. This phase is carried out by exploiting the robot model provided by Franka in the Gazebo simulator [68]. After that, we moved to perform tests on the real robot. In the second phase, we add errors in the dynamic model of the real arm and we evaluate the capacity of the adaptive control law to manage this uncertainty with respect to the non-adaptive version of the same control law. In the third and last phase, we mount an external end-

[2]https://github.com/CentroEPiaggio/panda_controllers.git
[3]https://frankaemika.github.io/docs/control_parameters.html

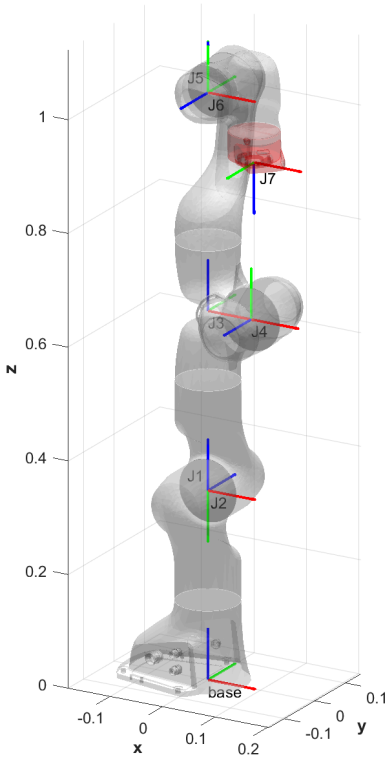


Figure 4: Frames obtained with Denavit-Hartenberg parametrization.

traj	A [m]	B [m]	C [m]	a [Hz]	b [Hz]	c [Hz]
3D	0.10	0.15	0.10	0.20	0.10	0.40

Table 5: Lissajous trajectories parameter

effector to perform similar trials also in case there is a real error in the inertial model of the manipulator to test both the capability to reject the disturbance and to estimate the new inertial parameters.

In all the steps mentioned above, to get meaningful results in terms of parameter estimation, the reference trajectories were chosen "rich enough" to make errors observable in the dynamic model of the manipulator. In the literature, different works try to address the problem of generating optimal movement to maximize the information gathered [85, 105]. However, given that the focus of the work is the adaptive controller, we exploited Lissajous, a family of parametric curves defined as:

$$p_d(t) = \begin{pmatrix} A\sin(2\pi a(t-t_0)) \\ B\cos(2\pi b(t-t_0)) \\ C\sin(2\pi c(t-t_0)) \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

For these tests, we used the curve depicted in Figure 5, which can be obtained with the coefficients reported in Table 5.

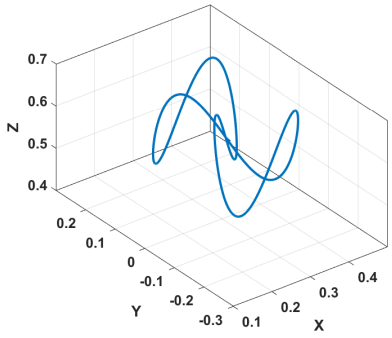


Figure 5: Lissajous Trajectory used for experimental validation with the manipulator ($f_x = 0.2$ Hz, $f_y = 0.1$ Hz, $f_z = 0.4$ Hz).

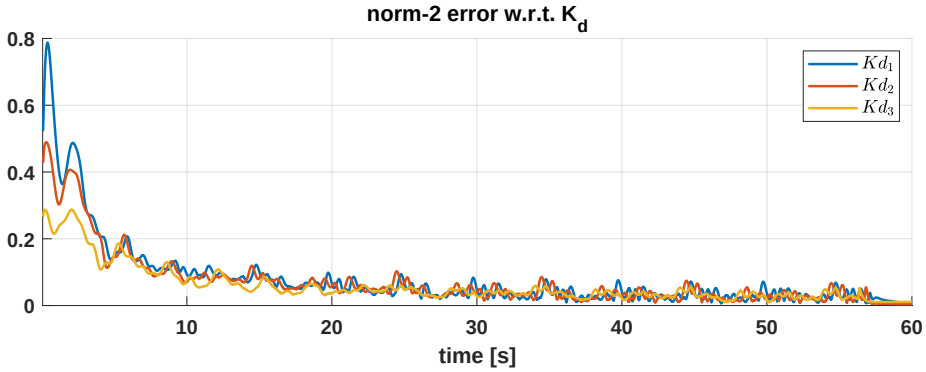


Figure 6: Error norm with respect to K_d (simulation test) (Real values: $Mass = 0.736$[kg], $CoM_x = 0.0105$[m], $CoM_y = -0.0043$[m], $CoM_z = -0.0454$[m]): $Kd_1 = [5,5,5,5,3,3,1]$ (Blue), $Kd_2 = [10,10,10,10,5,5,2]$ (Red), $Kd_3 = [30,30,30,30,15,15,5]$ (Yellow).

For the controller setup, we use matrix gains as follows:

$$\Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_6 \end{bmatrix} \in \mathbb{R}^{6\times 6}, \quad K_d = \begin{bmatrix} k_{d1} & & & \\ & k_{d2} & & \\ & & \ddots & \\ & & & k_{d7} \end{bmatrix} \in \mathbb{R}^{7\times 7} \tag{49}$$

$$\Gamma = \begin{bmatrix} \Gamma_{l_1} & & \\ & \ddots & \\ & & \Gamma_{l_7} \end{bmatrix} \in \mathbb{R}^{70\times 70}, \quad \Gamma_{l_i} = r_i \begin{bmatrix} r_m & & & \\ & r_{mCoM_x} & & \\ & & \ddots & \\ & & & r_{I_6} \end{bmatrix} \in \mathbb{R}^{10\times 10} \tag{50}$$

Simulation tests: The first step of validation is performed in a simulated environment. During this phase, we tested how the performances of the controller change with respect to the values assigned to K_d and Γ. To reproduce the case where there are uncertainties in the inertial parameters, we introduced an error in the internal dynamic model of the controller. The real parameters and the internal ones are (m=0.73kg, CoM=(1.05, −0.43, −4.52)cm) and (m=2.5kg, CoM=(20, 20, 20)cm) respectively for mass and center of mass, while the inertial tensor terms are set to zero.

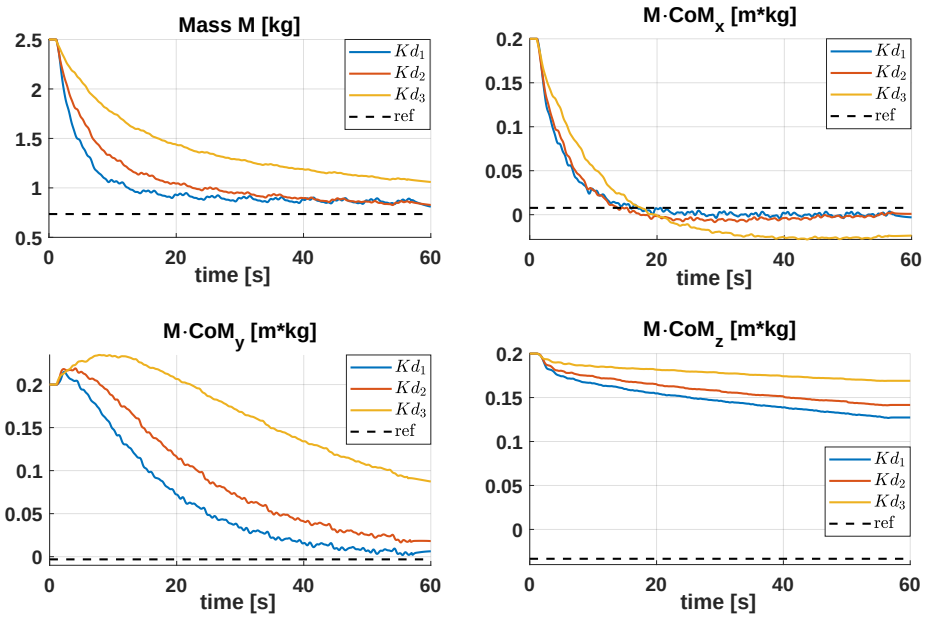


Figure 7: Parameters updating with respect to K_d (simulation test) (Real values: $Mass = 0.736[\text{kg}]$, $CoM_x = 0.0105[\text{m}]$, $CoM_y = -0.0043[\text{m}]$, $CoM_z = -0.0454[\text{m}]$): $Kd_1 = [5,5,5,5,3,3,1]$ (Blue), $Kd_2 = [10,10,10,10,5,5,2]$ (Red), $Kd_3 = [30,30,30,30,15,15,5]$ (Yellow).

First, we start with the discussion on the choice of the gain K_d. As it can be observed in Figure 6, while the steady-state error and the settling time are similar with different gains, the main effect is related to its initial amplitude. This happens because higher gain leads to a controlled system which is less sensible to inertial parameter errors. Moving instead to inertial parameter convergence (Figure 7), it can be observed that lower values of K_d lead to faster convergence of them. The reason behind this behavior is that the parameter's update rate is strictly connected with the tracking error and, if the error decreases too quickly, the update law slows down sooner and requires more time to converge. In those cases, the selection of a more exciting trajectory could help in achieving the convergence in a lower time.

After that, we move to the discussion on the effect of Γ on the system's behavior. In Figure 8 the tracking error convergence achieved with different values is depicted. As it can be observed, higher values of Γ lead to slower tracking and parameter convergence. This is because the update law is proportional to Γ^{-1} and the controlled system needs more time to adapt the inertial model to the real system. This slower convergence is also shown in Figure 9. However, it is evident from the plot that if we decrease Γ too much, undesired oscillatory behavior in the parameter estimation appears, leading to the conclusion that it is not convenient to decrease Γ arbitrarily.

Real manipulator tests: After the simulation campaign, we move on to the implementation of the controller on the real manipulator. The first step is to test the improvements provided by an adaptive controller. In fact, as said previously, torque-based control could be very sensitive in terms of tracking errors in the presence of inertial parameter uncertainties. For this reason, we implemented on the manipulator two versions of the Backstepping control, the classical version and the adaptive one, maintaining the same set of gains to check the improvements provided by the adaptive law. Figure 10 represents the tracking

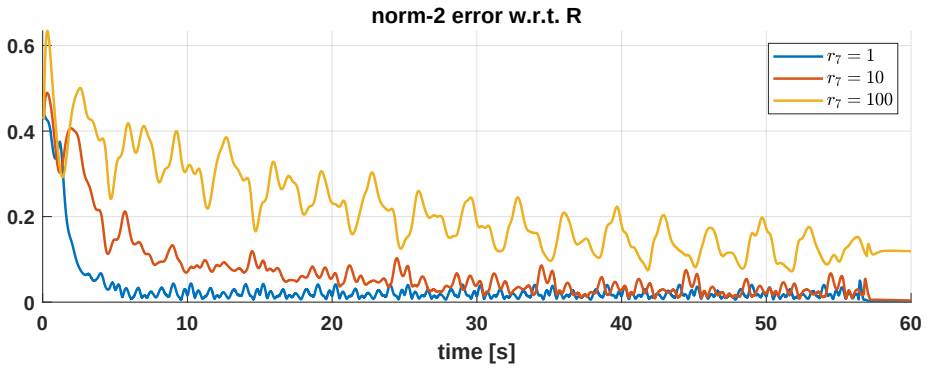


Figure 8: Norm-2 of error with respect to Γ (laboratory test): $r_7 = 100$ (Yellow), $r_7 = 10$ (Red), $r_7 = 1$ (Blue).

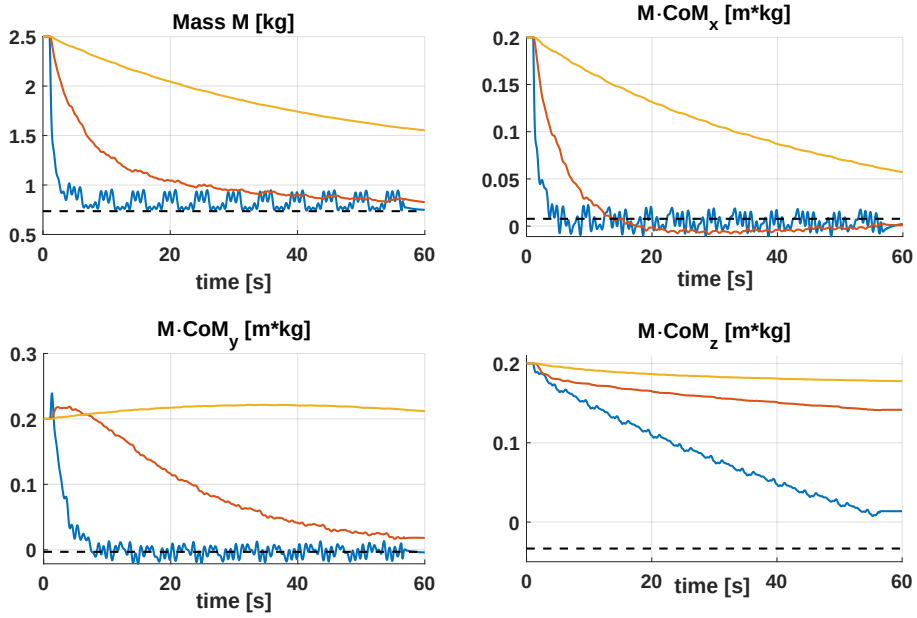


Figure 9: Parameters updating with respect to R (laboratory test): $r_7 = 100$ (Yellow), $r_7 = 10$ (Red), $r_7 = 1$ (Blue).

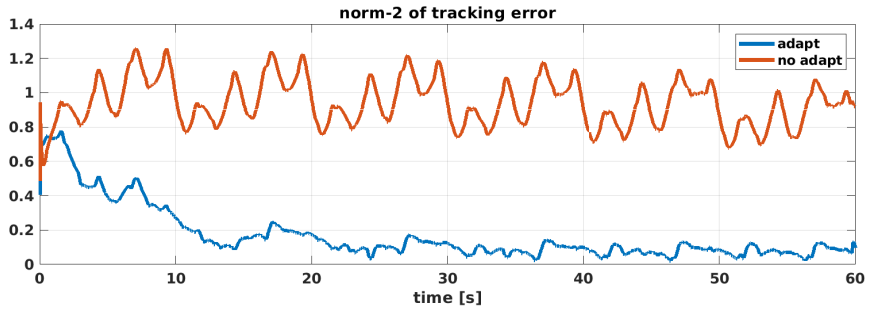


Figure 10: Norm-2 of error (laboratory test): no-adaptive controller (Red), adaptive controller (Blue).

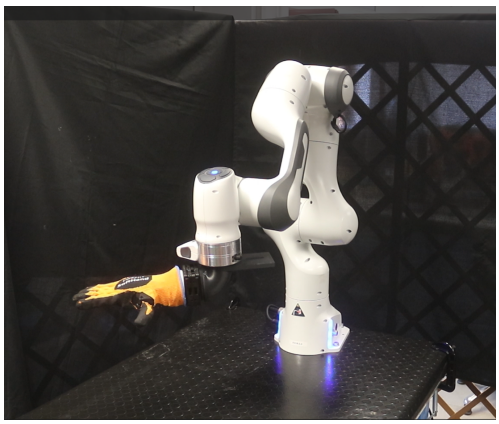


Figure 11: Setup used for test with real manipulator composed by a Franka Emika Panda with Softhand mounted as end-effector.

error of the two controllers under the same conditions, showing how the adaptive law permits handling the dynamic uncertainties in the model of the manipulator.

As the last experiments, a Softhand [21] was fixed to the robot's flange, and adaptive control law was used only on link 7. In this case, the inertial parameters of the Softhand were not considered in the model. This test aims to verify the correct tracking with the adaptive controller if an unidentified object is fixed on the end-effector. Figures 12 and 13 show the tracking error and the inertial parameter estimation respectively, proving that the adaptive controller is effective in compensating for the disturbance related to the unknown inertial parameters of the end-effector. The comparison between the true inertial parameters of link 7 with SoftHand and the estimated ones are reported in Table 6.

	Mass [kg]	CoM_x [m]	CoM_y [m]	CoM_z [m]
True	1.5655	0.06170	-0.0004	0.0009
Estimate	1.7542	0.0414	-0.0039	-0.0191
Initial	0.736	0.014	-0.004	-0.045

Table 6: Results of the estimation of Softhand + Link 7 parameters in $\{S_7\}$ through adaptive control. "True" indicates contains the real values, "Estimate" indicates the final values of the parameters, while "Initial" is the set of initial values of the parameters. The temporal evolution of the estimates can be seen in Figure 13.

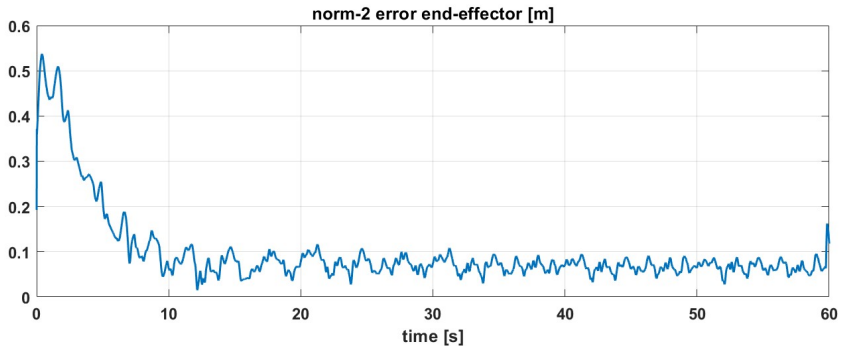


Figure 12: Norm-2 of error with Softhand on a 3D Lissajous trajectory.

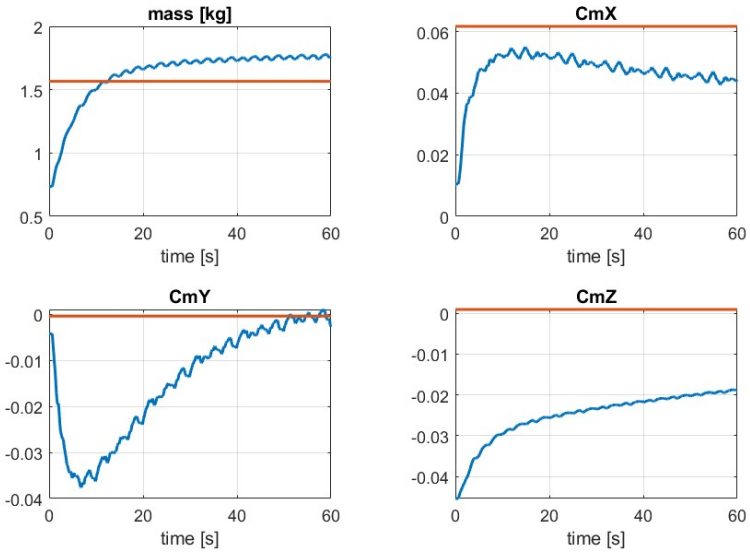


Figure 13: Temporal behaviour of the parameters updating with Softhand on a 3D Lissajous trajectory.

3 A Novel Formulation for Adaptive Computed Torque Control Enabling Low Feedback Gains in Highly Dynamical Tasks

These days, robotic manipulators are becoming more and more common in industries and everyday life, including manufacturing, logistics, and healthcare [45, 65, 141]. The ability of this kind of system to manage large loads and execute highly dynamic movements with a very high degree of precision is what drives this trend.

To move robotics manipulators into daily living situations, one has to guarantee a safe physical interaction with the environment and humans. For this purpose, most of the control algorithms used in these situations are based on active compliant control [111,121]. This technique facilitates the assignment of a desired compliant behavior to the manipulator through the design of an appropriate control law. This enables the manipulator to interact with the environment without the generation of excessive contact forces. Several works employ this strategy [1, 121]. However, in these controllers, errors in the dynamic model of the manipulator can result in tracking errors on the desired motion.

To overcome this limitation, adaptive control laws or model reference adaptive control (MRAC) laws are developed, see e.g.: [140]. One of the traditional control algorithms in the literature that reaches asymptotic stability under parameter uncertainties is the Adaptive Computed Torque control [24, 25, 28, 113], which consists in a feedback linearization with the joint torque as the input of the system. However, for parameter updates, the law incorporates the estimated mass matrix inverse and the joint accelerations, which may lead to numerical issues such as the ill-conditioning of the estimated mass matrix or the requirement for double derivation of the joint position measurements to obtain acceleration. Furthermore, the solution of the Lyapunov equation for the linearized model of the closed-loop system is also necessary. One possibility is to limit the variation of the parameters in a meaningful physical region [79]. An alternative approach, largely used in the literature, is the one proposed in [114, 118, 119], wherein authors introduce a new reference combining velocity and position errors. This leads to a distinct law for parameter updates that does not necessitate the aforementioned quantities. However, the control structure does not permit a straightforward manner of setting the desired stiffness of the manipulator. Indeed, imposing fixed stiffness gains in this type of control results in using time-variant gains and the inverse of the estimated mass matrix (leading to the same numerical problems of Adaptive Computed Torque control).

To address this challenge, we present a novel adaptive law for the Computed Torque algorithm that eliminates the need for the aforementioned information. We formulate the control law both in the joint and in the Cartesian domains, to encompass all the possible reference trajectory types. Subsequently, we validate the framework through simulations and experiments conducted on a seven-degree-of-freedom robot, executing various tests. Finally, we evaluate the framework's performance in a throwing task. We selected this task for two reasons: 1) the highly dynamic nature of this action amplifies the trajectory tracking error associated with uncertainties in inertial parameters, and 2) the short duration of the movement necessitates the controller to avoid relying solely on feedback action and instead possess a highly precise feedforward component. A crucial aspect of these types of tasks is the friction. This impedes precise tracking and generates errors in speed. Therefore, we have included this matter in the discussion. The results presented in this section are reported in the manuscript "*A Novel Formulation for Adaptive Computed Torque Control Enabling Low Feedback Gains in Highly Dynamical Tasks*" published in IEEE Access [116].

3.1 Adaptive control: background

In this section, we provide a comprehensive analysis of adaptive control for robotic manipulators, specifically addressing the primary limitations of conventional *Adaptive Computed Torque* (ACT) control algorithms and the existing solutions within the robotics domain.

Let us consider the dynamic model of a rigid robot manipulator described by Lagrange equation (see [54]):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau\,. \tag{51}$$

As pointed out in [55], (51) is linear in the inertia parameters $\pi_d \in \mathbb{R}^{p_d}$ (link masses, first and second moments of inertia), and can be hence rewritten as

$$\tau = Y_d(q,\dot{q},\ddot{q})\pi_d, \tag{52}$$

where $Y_d \in \mathbb{R}^{n\times p_d}$ is the regressor matrix and p_d is the number of parameters.

Let us define $\hat{\pi}_d \in \mathbb{R}^{p_d}$ as an estimate of π_d, and let $\big(\hat{M}(\cdot),\hat{C}(\cdot),\hat{G}(\cdot)\big)$ be the dynamic matrices evaluated in $\hat{\pi}_d$. Let us define $q_d(t) \in \mathbb{R}^n \times \mathbb{R}$ as the desired trajectory. Finally, let then $\tilde{\pi}_d \triangleq \pi_d - \hat{\pi}_d$ be the parameters' error and $e(t) = q_d(t) - q(t)$ be the tracking error.

Classical Computed Torque control [55] offers a linearization law for nonlinear systems. However, it necessitates the knowledge of the dynamic model to generate both feed-forward and feedback torque actions.

$$\tau = \hat{M}(q)\ddot{q}_d + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + k_v\dot{e} + k_p e, \tag{53}$$

where k_v, k_p are positive definite gain matrices. Substituting (**??**) in (51), the error dynamics becomes

$$\hat{M}(q)\ddot{e} + k_v\dot{e} + k_p e = \tilde{M}(q)\ddot{q} + \tilde{C}(q,\dot{q})\dot{q} + \tilde{G}(q). \tag{54}$$

Assuming perfect knowledge of the dynamic model, i.e., $\hat{\pi}_d \equiv \pi_d$, the error follows a mass-spring-damper dynamics with linear stiffness and damping, i.e.,

$$M(q)\ddot{e} + k_v\dot{e} + k_p e = 0, \tag{55}$$

with guarantees of asymptotic stability due to positive definiteness of $M(q), k_v, k_p$. In case of parameters uncertainties, i.e., if $\hat{\pi}_d \neq \pi_d$, (54) can be written as

$$\ddot{e} + \hat{M}^{-1}(q)\big(k_v\dot{e} + k_p e\big) = \hat{M}^{-1}(q)Y_d(q,\dot{q},\ddot{q})\tilde{\pi}_d.$$

A possible solution, exploited in [24, 81, 86, 113], is to consider a new state $x = (e^\top, \dot{e}^\top)^\top$, which represents the tracking error and its derivative, and rewrite (51) in state-space form as follows

$$\begin{cases} \dot{x} = Ax + B\hat{M}^{-1}(q)Y_d(q,\dot{q},\ddot{q})\tilde{\pi}_d \\ \dot{\tilde{\pi}}_d = -\dot{\hat{\pi}}_d \end{cases}$$

where $A \in \mathbb{R}^{2n\times 2n}, B \in \mathbb{R}^{2n\times 2n}$ are the dynamic and input matrices of a minimal state space realization of the error system. By choosing the parameters' update law as

$$\dot{\hat{\pi}}_d = R^{-1}Y_d^\top(q,\dot{q},\ddot{q})\hat{M}^{-\top}(q)B^\top Px \tag{56}$$

where $R \in \mathbb{R}^{p_d\times p_d}$ is a gain matrix and $P \in \mathbb{R}^{n\times n}$ is the solution of the Lyapunov equation, it is possible to show the asymptotic stability of the tracking error (we refer the interested reader to [24] for the theoretical proof). With this law, it is possible to assign the desired

stiffness k_p and damping k_v to the manipulator (see (55)). Although the theoretical foundation is robust, the practical implementation may encounter distinct challenges. Notably, the parameter update law necessitates the inversion of the estimated mass matrix $\hat{M}$, which may be ill-conditioned. Additionally, the actual joint acceleration $\ddot{q}$ cannot be directly measured and must be derived through differentiation. Furthermore, it requires also the solution P of the Lyapunov equation, which is related to the controller gains and must be computed each time they change.

In [119], Slotine and Li provide a control strategy that does not require all the above quantities. This approach is also known as *Adaptive Inertia-Related Control* (AIRC) [55]. Let us indeed define $\dot{q}_r \triangleq \dot{q}_d + \Lambda e$ as a new velocity reference, and $s \triangleq \dot{q}_r - \dot{q}$ as a joint velocity error, the control law proposed in [119] is

$$\tau = \hat{M}(q)\ddot{q}_r + \hat{C}(q,\dot{q})\dot{q}_r + \hat{G}(q) + k_d s. \tag{57}$$

with update law of parameters

$$\dot{\hat{\pi}}_d = R^{-1} Y_{r,d}^{\top}(q,\dot{q},\dot{q}_r,\ddot{q}_r)s, \tag{58}$$

where $Y_{r,d}(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ is known as the *Slotine-Li regressor matrix*, such that

$$Y_{r,d}(q,\dot{q},\dot{q}_r,\ddot{q}_r)\pi_d = M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + G(q)\,. \tag{59}$$

It is important to note that the Slotine-Li regressor $Y_{r,d}(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ is a generalization of the classical one $Y_d(q,\dot{q},\ddot{q})$. Indeed, substituting $(\dot{q}_r,\ddot{q}_r)$ with $(\dot{q},\ddot{q})$ the two matrices are identical.

Substituting (57) into (51), the error dynamics with the perfect model assumption is

$$M\ddot{e} + \big(M\Lambda + C + k_d\big)\dot{e} + \big(M\dot{\Lambda} + C\Lambda + k_d\Lambda\big)e = 0,$$

Notice that, by choosing the gains in (53) as follows

$$\begin{aligned} k_v &= \hat{M}\Lambda + \hat{C} + k_d, \\ k_p &= \hat{M}\dot{\Lambda} + \big(\hat{C} + k_d\big)\Lambda, \end{aligned} \tag{60}$$

the Computed Torque control law (53) acts exactly as (57). However, computing the necessary gains Λ and k_d to obtain the desired impedance k_p and k_v is not straightforward. It requires the inertia matrix inverse and time-varying gains.

3.2 Novel adaptive Computed Torque control

We propose a novel formulation of the Adaptive Computed Torque control that does not require the inverse of the estimated mass matrix, avoiding all the problems related to the ill-conditioning of the matrix. We start with the definition of the dynamic system representing the manipulator. Then we report the formulation and the proof of the novel control law for both the joint and the Cartesian spaces. In the end, we discuss the most critical parts from an implementation point of view and how to deal with them.

3.2.1 Model Definition

The primary objective of this section is to elucidate the system model employed throughout this work, along with the underlying assumptions. Starting from the classical Lagrangian model (51) [54], to address highly dynamic tasks (e.g., the throwing of objects), a friction term is incorporated, resulting in the following equation:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D(\dot{q}) + G(q) = \tau\,, \tag{61}$$

where $q(t), \dot{q}(t), \ddot{q}(t) \in \mathbb{R}^n$ are the joint positions, velocities, and accelerations, respectively, and n is the number of joints. $M \in \mathbb{R}^{n\times n}$ is the inertia matrix; $C \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, while $D(\dot{q}) \in \mathbb{R}^{n\times n}$ represents the damping terms. Finally, $\tau \in \mathbb{R}^n$ is the torque applied to the motors.

Let the system (61) meets the classical bounding properties [55]:

1. $0 < ||M(q)|| < \infty$, and $M(q)$ positive definite.
2. $||C(q,\dot{q})|| < l_1||\dot{q}||$, $l_1 < \infty$.
3. $||G(q)|| < \infty$.
4. $||D(\dot{q})|| < l_2||\dot{q}||$, $l_2 < \infty$.

For joint friction, we assume that it can be modeled as one described in [18], i.e.,

$$D(\dot{q})_i = d_{i,1}\dot{q}_i + d_{i,2}\text{sgn}(\dot{q}_i), \tag{62}$$

where i is the joint index, and $d_{i,1}, d_{i,2}$ are the viscous and Coulomb coefficients respectively. We disregard stiction because the considered task exhibits dynamic behavior. Moreover, for continuity reasons, the sign function is implemented as

$$\text{sgn}(\dot{q}_i) = \frac{\dot{q}_i}{|\dot{q}_i| + \delta},$$

with $\delta > 0$.

It is worth noting that the friction term is linear in a parameter vector $\pi_f \in \mathbb{R}^{2n}$, i.e., $D(\dot{q}) = Y_f\, \pi_f$ with $Y_f \in \mathbb{R}^{n\times 2n}$ the friction regressor matrix. As a consequence, the system dynamics can be easily rewritten with a general regressor matrix Y by writing

$$\tau = [Y_d, Y_f]\begin{bmatrix}\pi_d \\ \pi_f\end{bmatrix} \triangleq Y\pi,$$

The same is valid for the Slotine-Li regressor matrix where $Y_r \triangleq [Y_{r,d}, Y_{r,f}]$.

Finally, let us define the desired trajectory $q_d(t)$ as in Sec. 3.1, and let us assume the following assumptions:

1. q_d is of class $\mathscr{C}^2$.
2. $||q_d^{(3)}|| < \infty$.
3. Perfect knowledge of the kinematic model.

3.2.2 Control law definition

In this section, we present two control laws, one formulated in joint space and the other in Cartesian space, that address the limitations of Adaptive Computed Torque as analyzed in Section 3.1. Both control laws attain asymptotic stability, including a straightforward compensation of the friction terms.

Joint-space control law: The initial control law introduced is the one defined in the joint domain. In this case, the error is defined as $e(t) = q_d(t) - q(t)$. As previously mentioned, the proposed control law does not necessitate the computation of the mass matrix inverse or the solution of the Lyapunov equation.

Theorem 1. *Consider the system* (61)*, under properties and assumptions reported above. The control law*

$$\tau = \hat{M}(q)\ddot{q}_d + \hat{C}(q,\dot{q})\dot{q}_d + \hat{D}(\dot{q}) + \hat{G}(q) + k_v\dot{e} + k_p e, \tag{63}$$

with $k_p \in \mathbb{R}^{n\times n}, k_v \in \mathbb{R}^{n\times n}$ *positive gains, and the parameters update law*

$$u_\pi = R_p\Big(Y_r^\top \dot{e} + \gamma Y^\top R_t(\tau - Y\hat{\pi})\Big), \tag{64}$$

with $R_p \in \mathbb{R}^{p\times p}, R_t \in \mathbb{R}^{n\times n}$ *positive definite matrices and* $\gamma > 0$*, leads to asymptotic convergence of the tracking error.*

Proof. To improve the readability of the proof, we have eliminated the matrix dependencies. By applying (63) into (61), the error model is

$$M\ddot{e} + \big(k_v + C\big)\dot{e} + k_p e = \tilde{M}\ddot{q}_d + \tilde{C}\dot{q}_d + \tilde{D} + \tilde{G} = Y_r\tilde{\pi}. \tag{65}$$

Let us now consider the Lyapunov candidate:

$$V = \frac{1}{2}e^\top k_p e + \frac{1}{2}\dot{e}^\top M\dot{e} + \frac{1}{2}\tilde{\pi}^\top R_p^{-1}\tilde{\pi}. \tag{66}$$

Since k_p, R_t and M are positive definite matrices, the derivative of the Lyapunov candidate is

$$\dot{V} = -\dot{e}^\top k_v\dot{e} + \frac{1}{2}\dot{e}^\top(\dot{M} - 2C)\dot{e} + \dot{e}^\top Y_r\tilde{\pi} - \tilde{\pi}^\top R_p^{-1}u_\pi.$$

Applying the adaptive law (64) and considering the Coriolis matrix written with the Christoffel symbols, the matrix $(\dot{M} - 2C)$ is skew-symmetric and it results in

$$\dot{V} = -\dot{e}^\top k_v\dot{e} - \gamma\tilde{\pi}^\top Y^\top R_t Y\tilde{\pi}. \tag{67}$$

The Lyapunov derivative is negative semi-definite, and hence V is bounded. At this juncture, we are unable to definitively ascertain the asymptotic stability of the system. However, we require Barbalat's Lemma, which requires that $\ddot{V}$ must be bounded. Deriving (67), we obtain

$$\ddot{V} = -2\dot{e}^\top k_v\ddot{e} - \gamma\tilde{\pi}^\top Y^\top R_t\Big(\dot{Y}\tilde{\pi} - Yu_\pi\Big),$$

where the term $\dot{Y}\tilde{\pi}$ is

$$\dot{Y}\tilde{\pi} = \dot{\tau} - \Big(\hat{M}q^{(3)} + \big(\dot{\hat{M}} + \hat{C}\big)\ddot{q} + \dot{\hat{C}}\dot{q} + \dot{\hat{D}} + \dot{\hat{G}}\Big),$$

while

$$q^{(3)} = M^{-1}\Big(\dot{\tau} - \big(\dot{M} - C\big)\ddot{q} - \dot{C}\dot{q} - \dot{D} - \dot{G}\Big),$$

which is the third derivative of q. Moreover,

$$\ddot{q} = M^{-1}\Big(\tau - C\dot{q} - D - G\Big) < \infty.$$

All the quantities in $\ddot{V}$ are bounded because τ and $\dot{\tau}$ are also bounded. By Barbalat's Lemma, it is possible to conclude that $\dot{V} \to 0$, implying that also $\dot{e} = 0, \ddot{e} = 0$ and $Y\tilde{\pi} = 0$. As a consequence, the error model (65) becomes

$$k_p e = \tilde{M}\ddot{q}_d + \tilde{C}\dot{q}_d + \tilde{D} + \tilde{G} = Y_r\tilde{\pi}.$$

Taking in consideration that $\ddot{q}_d = \ddot{e} + \ddot{q}$ and $\dot{q}_d = \dot{e} + \dot{q}$, the error model reduces to

$$k_p e = Y\tilde{\pi} = 0\,,$$

which concludes the demonstration, showing the asymptotic stability of the closed-loop system. □

Remark 3. *The adaptive law* (64) *and* (56) *require the joint acceleration values. Despite it is possible to numerically filter the signals, Section 3.2.3 shows a method to obtain the regressor with the only use of velocities.*

Remark 4. *The proposed MACT architecture (*(63),(64)*) differs from the ACT laws (*(53),(56)*) in the parameter adaptation. Unlike the Adaptive Computed Torque laws, the proposed control architecture does not invert the inertia matrix $\hat{M}^{-1}$ and does not need the solution of the Lyapunov equation P. Furthermore, it differs from the Adaptive Inertia-Related control laws (*(57),(58)*) since, in the proposed laws, there is no influence of the position error in the velocity reference.*

Task-space control: Despite the simplest way to control a manipulator is to generate motion in the joint space, in terms of task planning, the most straightforward approach is to compute the reference trajectories in the Cartesian domain. Nowadays, it is common to have manipulators with more than six degrees of freedom. In such cases, the controller must possess the capability to manage the redundancy inherent in the system. Considering the system (61), defining $(\cdot)^+$ as the right pseudo-inverse operator, i.e., $A^+ \triangleq A^\top (AA^\top)^{-1}$, the system (61) can be written in the task space by substituting $\dot{q} = J^+\dot{\xi} + N\dot{a}$, and $\ddot{q} = J^+\big(\ddot{\xi} + \dot{J}\dot{q}\big) + N\ddot{a}$. The variables $\dot{a}, \ddot{a}$ are the joint velocity and acceleration that do not affect the motion of the end-effector, given that they are projected in the null of the Cartesian space by the null projector $N = I - J^+J$. Let us incorporate the following assumption

- $\exists\, J^+(q) \quad \forall\, q(t),\ t \in [t_0, t_f]$.

This assumption implies that the Jacobian is non-singular along the trajectory.

Theorem 2. *Given the system* (61)*, under the properties and the assumptions reported above, define a desired trajectory in Cartesian space, denoted as $\xi_d(t) \in SE(3) \times \mathbb{R}$. Subsequently, define the end-effector error as $e_\xi = \xi_d - \xi$. By using the control law*

$$\tau = \hat{M}\alpha + \hat{C}\beta + \hat{D} + \hat{G} + k_v(\beta - \dot{q}) + J^\top k_{\xi_p} e_\xi, \tag{68}$$

where $\alpha \triangleq J^+\big(\ddot{\xi}_d - \dot{J}\dot{q}\big) + N\ddot{a}_d$, $\beta \triangleq J^+\dot{\xi}_d + N\dot{a}_d$ are desired velocity and acceleration respectively, and $k_{\xi_p} \in \mathbb{R}^{6\times 6}$ is the gain matrix for the error in the Cartesian space, in conjunction with the parameters update law

$$u_\pi = R_p\Big(Y_r^\top(q, \dot{q}, \beta, \alpha)\big(\beta - \dot{q}\big) + \gamma Y^\top(\cdot) R_t\big(\tau - Y(\cdot)\hat{\pi}\big)\Big), \tag{69}$$

with $R_p \in \mathbb{R}^{p\times p}, R_t \in \mathbb{R}^{n\times n}$ positive definite matrices and $\gamma > 0$, the closed loop system is stable and the end-effector follows the desired trajectory, i.e.,

$$\lim_{t\to\infty} ||e_\xi|| = 0.$$

Proof. In the demonstration, we eliminate the matrix dependencies that are superfluous. Defining $\dot{e}_q, \ddot{e}_q$ the joint velocity and acceleration errors, i.e.,

$$\begin{aligned} \ddot{e}_q &= J^+\big(\ddot{e}_\xi - \dot{J}\dot{e}_q\big) + N\ddot{e}_N, \\ \dot{e}_q &= J^+\dot{e}_\xi + N\dot{e}_N, \end{aligned}$$

and applying (68) in (61), the error dynamics becomes

$$\ddot{e}_q = M^{-1}\Big(Y_r(q, \dot{q}, \beta, \alpha)\tilde{\pi} - \big(C + k_v\big)\dot{e}_q - J^\top k_{\xi_p} e_\xi\Big). \tag{70}$$

Taking the Lyapunov candidate

$$V_\xi = \frac{1}{2} e_\xi^\top k_{\xi_p} e_\xi + \frac{1}{2} \dot{e}_q^\top M \dot{e}_q + \frac{1}{2} \tilde{\pi}^\top R_p^{-1} \tilde{\pi}, \tag{71}$$

its derivative is

$$\dot{V}_\xi = -\dot{e}_q^\top k_v^\top \dot{e}_q - \gamma \tilde{\pi}^\top Y^\top(\cdot) R_t Y(\cdot) \tilde{\pi}.$$

that is negative semi-definite. By applying Barbalat's lemma we can show, similarly to Theorem 1, that $\dot{e}_\xi = 0, \ddot{e}_\xi = 0, Y(\cdot)\tilde{\pi} = 0$. By substituting in (70), it results in

$$J^\top k_{\xi_p} e_\xi = 0. \tag{72}$$

Since the kernel of $J^\top$ is composed of structural wrenches, and the robot is not in singularity by assumptions, the equation (72) implies $k_{\xi_p} e_\xi = 0$, concluding the demonstration. □

3.2.3 Implementation consideration

In addition to the theoretical considerations presented thus far, several implementation-related aspects merit further discussion. The following sections provide a brief overview of these elements.

Choosing control gain matrices: With regard to control action, the gains k_p, k_v, k_{ξ_p} can be selected to achieve the desired behavior of the closed-loop system. High-gain matrices result in a stiff behavior and robustness. Conversely, if interaction with the environment or cooperation is required, low stiffness facilitates a safe attitude. On the other hand, the choice of the adaptive gain matrices depends on the choice of control parameters. There are three quantities on the control law that determine the adaptive behavior: γ, R_p, R_t. Intuitively, the parameters update law is divided into two components. The first one can be viewed as the classical direct law, which guarantees tracking convergence and stability. The second one is more general and permits parameter convergence regardless of error convergence. However, the latter suffers from delays or noise due to the acceleration estimate. The union of the two instead ensures the asymptotic stability of the overall system. Finally, the gain γ balances the two parts. We propose to appropriately select the adaptive gains in proportion to the magnitudes of the model parameters, so that

$$R_p^{-1} * \hat{\pi}(t_0) \approx \eta * I$$

with η being a scalar gain. In this manner, the values are updated with quantities that are proportional to their magnitude. For the purpose of R_t, it regulates the update by the torques applied to the joints. A suitable choice is to set this gain to the anticipated torque on the actuators.

Parameters observability: In the context of adaptive control, the observability of dynamic parameters presents a fundamental challenge. Starting from the classical representation of the dynamic of a rigid body (mass m, center of mass (c_x, c_y, c_z), and moments of inertia $(I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz})$, the most common choice to represent the dynamic of a serial manipulator is the union of the dynamic parameters of each link of the kinematic chain. However, it is well known that this set of dynamic parameters is not fully observable. Several works have been conducted to identify minimum sets of parameters [43] and define estimation procedures for complex kinematic chains [19].

The other main aspect related to the observability of the inertial parameter regards the motion performed during the estimation procedure. The performed movement should

be rich enough to permit each parameter to have an effect on the manipulator dynamics. This condition is called *persistent excitation* [118, 119], and it is verified if exists l_0, l_1, l_2 such that

$$\forall\, t < 0, \quad l_1 I \leq \int_t^{t+l_0} Y^\top(r) Y(r) dr \leq l_2 I.$$

In the case of cyclic trajectories, it permits the controller to learn a set of parameters that ensure tracking of that movement. The problem can also be addressed within the active sensing control framework, as outlined in [84, 105]. By optimally designing the reference trajectory, a substantial amount of information about the unknown parameters can be collected. This approach assumes the perfect knowledge of the kinematic information. In reality, the regressor Y is nonlinearly dependent on these quantities, making it challenging to quantify the impact of errors or uncertainties on performance. The literature contains some works proposing solutions for kinematic uncertainties [82, 133], but this falls outside the scope of this manuscript.

Eliminating the Requirement for Accelerations in the Regressor Computation: The regressor $Y(q, \dot{q}, \ddot{q})$ used in (56), (64) needs precise kinematic information. While commercial manipulators generally provide precise information about joint angle and velocity, joint acceleration is not provided. The classical solution is to derive it numerically from velocity measures, filtering them to avoid distortion due to measurement noise. However, in this section, we also propose a possible solution to address this issue.

As noted in [118, 123], it is possible to analytically obtain a filtered regressor that does not depend on accelerations. Let us define the filtered torque $y \in \mathbb{R}^n$ as

$$y \triangleq f\big(\tau(t)\big) = \int_0^t \phi(t-r)\big(M\ddot{q} + C\dot{q} + D + G\big) dr,$$

where ϕ is the impulsive response of a low-pass filter f. Then, it is possible to eliminate $\ddot{q}$ integrating by parts the first term

$$y = \big[\phi(t-r) M \dot{q}\big]_0^t - \int_0^t \Big(\frac{d\big(\phi(t-r)M\big)}{dt} \dot{q} \quad + \phi(t-r)\big(C\dot{q} + D + G\big)\Big)\, dr$$

Since real parameters are not affected by the filter, a regressor matrix $W(q, \dot{q}) \in \mathbb{R}^{n \times n}$ exists for the filtered dynamics that do not depend on accelerations, i.e.,

$$y = W(q, \dot{q})\pi = \left(W_a(q, \dot{q}) - \int_0^t W_b(q, \dot{q})\, dr \right) \pi,$$

where W_a, W_b are the regressors of the two parts inside and outside the integral. This filtered regressor $W(q, \dot{q})$ can be used in the adaptive law (64), (56) substituting terms $(\tau, Y(q, \dot{q}, \ddot{q}))$ with $(y, W(q, \dot{q}))$.

3.3 Validation

The entire validation covers both aspects of control in joint and Cartesian spaces, both in simulations and experiments. The adaptive framework is developed on ROS Noetic and is based on the library *Thunder_dynamics*[1]. The code for both simulations and experiments is available at the repository *panda_mact_controller*[2]. All the tests are performed using a *Franka Emika Panda*, a 7 degrees of freedom collaborative manipulator [51]. The validation is composed of three parts:

[1] https://github.com/CentroEPiaggio/thunder_dynamics
[2] https://github.com/CentroEPiaggio/panda_mact_controller.git

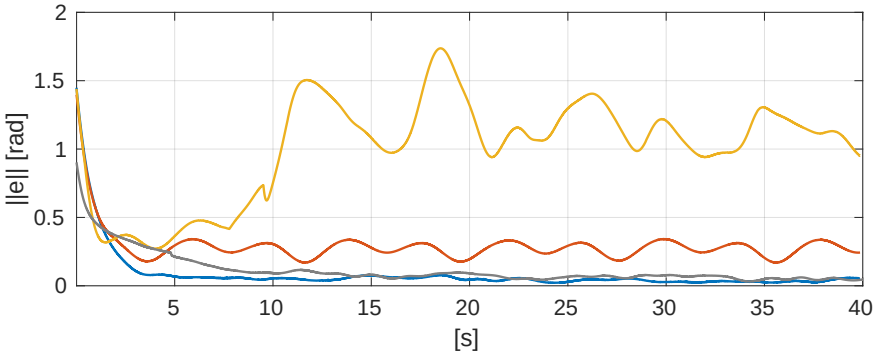


Figure 14: Simulations in joint-space. The MACT (**blue**) achieves a better error correction than the non-adaptive CT (**red**), and a similar convergence of the AIRC (gray). In (**yellow**) the classical ACT suffers from the inertia conditioning problem (Figure 15b).

1. simulation tests, where both controllers (joint-space and Cartesian space) are tested introducing uncertainties across the entire kinematic chain;
2. real robot test, where both controllers are tested on the real hardware with an unknown payload attached at the end-effector;
3. real dynamic task test, where we choose the throwing as example task for evaluating the improvement that our control law could bring in the execution of these tasks (Figure 25).

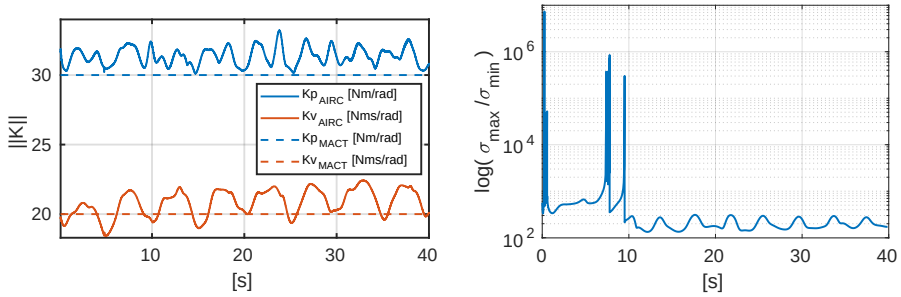
In all the tracking tests we compare our MACT control with the non-adaptive version (CT) and with AIRC.

3.3.1 Simulations

Simulations are performed using the Franka manipulator model provided by the manufacturer within the Gazebo environment [69]. For the controllers, the initial parameter estimation is set so that the error is approximately 15% relative to the actual robot parameters. Notice that parameter error is challenging to quantify, as each parameter exerts a distinct impact on the error and should be analyzed independently. Gains are chosen low ($k_p \approx 30N/rad$ and $k_{\xi_p} \approx 30N/m$) to have compliancy. In the AIRC the gains are chosen empirically, following equations (60), to have similar equivalent stiffness and damping w.r.t. the *Computed Torque*. It is noteworthy that gains not only encompass the potential interaction with the environment but also are instrumental in mitigating uncertainties. High gains facilitate rapid error correction, but they also result in substantial interaction forces in the event of collisions. For this reason, we use low gains during the tests.

Joint-space simulations: The simulation entails executing a trajectory-tracking task on a joint-space Lissajous curve while simultaneously monitoring joint errors. These trajectories are cyclic and well-suited for dynamic pick-and-place tasks. The gains chosen for CT, ACT, MACT controllers are

$$\begin{aligned} k_p &= \text{diag}(30, 30, 30, 30, 15, 15, 3), \\ k_v &= \text{diag}(15, 15, 15, 15, 7.5, 7.5, 1.5), \end{aligned} \tag{73}$$



(a) Equivalent gains of IRC. **(b)** Mass matrix condition number.

Figure 15: Problems of the existing controllers: AIRC has time-varying stiffness and damping gains. ACT presents ill-conditioning of the Inertia matrix.

	CT	ACT	AIRC	MACT
RMSE [m]	0.3112	1.0727	0.1729	0.1371

Table 7: RMSE of the joint trajectory tracking obtained in simulation with four different control laws: classical Computed Torque (CT), classical Adaptive Computed Torque (ACT), Adaptive Inertia Related Control (AIRC) and our Modified Adaptive Computed Torque (MACT).

while the AIRC controller has

$$\begin{aligned} \Lambda &= \text{diag}(2,2,2,2,1,1,0.2), \\ k_d &= k_v. \end{aligned} \tag{74}$$

Figure 14 shows the tracking error, represented as the norm of the joint trajectory error. From the plot, we can notice that our approach outperforms both the classical adaptive and non-adaptive computed torque controllers and achieves similar results of the AIRC. The Root Mean Square Error (RMSE) for each control law is reported in Table 7. While for the non-adaptive controller (shown in red) the error is due to the not perfect knowledge of the inertial parameters, for the classical adaptive computed torque control (shown in yellow) the main problem is related to the ill-conditioned mass matrix. In fact, we can observe that when the condition number grows up (Figure 15b) also the tracking error diverges. Despite the AIRC control obtains very good tracking, the equivalent gains (60) are time-variant and depend on the manipulator movement. Figure 15a plots these equivalent gains and the fixed ones.

Cartesian-space simulations: In a similar vein to the joint-space scenario, the simulation involves the tracking of a Lissajous curve in Cartesian space. We compare our Cartesian controller with the adaptive inertia-related control. Since AIRC works in joint space, the Cartesian trajectory is obtained by a second-order CLIK algorithm [115]. The controller gains are

$$\begin{aligned} k_{\xi_p} &= \text{diag}(30,30,30,3,3,3), \\ k_v &= \text{diag}(15,15,15,15,7.5,7.5,1.5), \end{aligned} \tag{75}$$

for CT and MACT controllers, while the AIRC has the same gains of the joint controller. We show two different trajectories: a slow tridimensional movement defined by a Lissajous, and a fast linear movement on the YZ plane. Figure 17 shows the error in position (Figure 17a) and orientation (Figure 17b). As observed, there is no significant difference in the

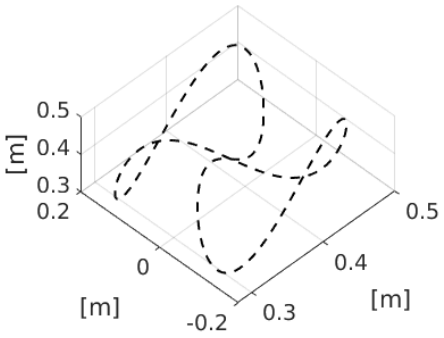
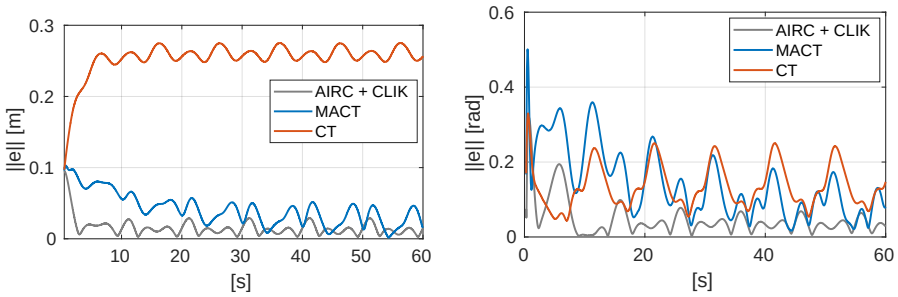
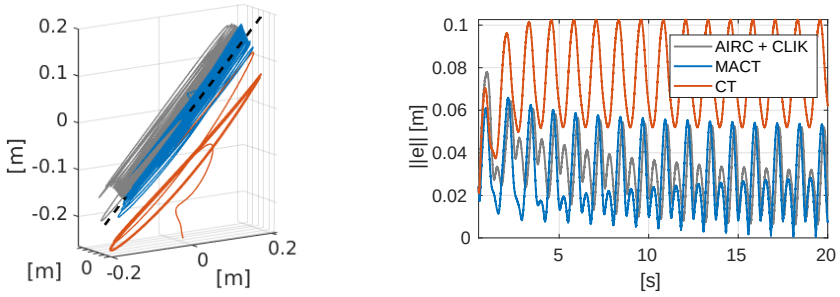


Figure 16: Slow Cartesian trajectory.



(a) Position error. **(b)** Orientation error.

Figure 17: Simulation error on the slow 3D Lissajous Cartesian trajectory. While the difference is clear on the position plot, the orientation is less affected by uncertainties.

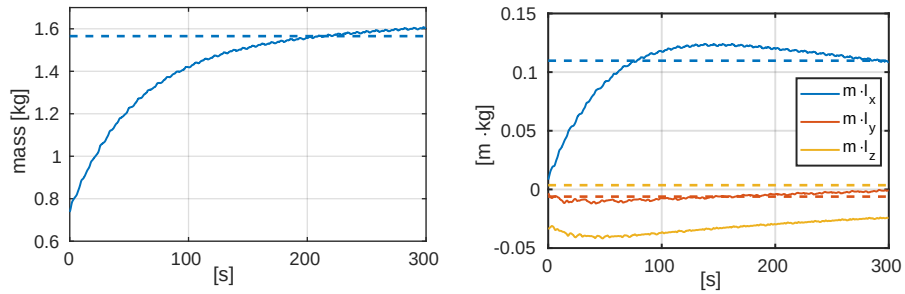


(a) Cartesian trajectory. **(b)** Cartesian error.

Figure 18: Simulation of fast Cartesian linear trajectory. The trajectories of MACT and AIRC are similar, while CT has an increased error. The dashed line on Figure 18a is the desired path.

	CT	AIRC	MACT
RMSE liss3D [m]	0.2545	0.0266	0.0505
RMSE linear [m]	0.0768	0.0313	0.0315

Table 8: RMSE of the Cartesian trajectory tracking obtained in simulation with three different control laws: classical Computed Torque (CT), Adaptive Inertia Related Control (AIRC) and our Modified Adaptive Computed Torque (MACT).



(a) Mass parameter. **(b)** First moment parameters.

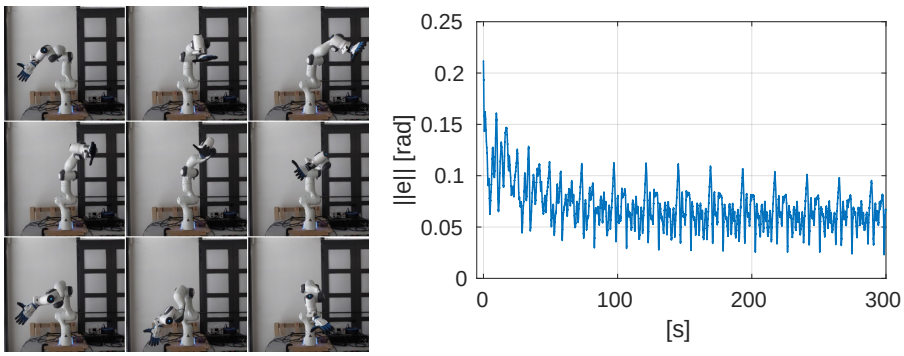
Figure 19: Principal parameters estimate behavior on the real robot. The manipulator follows a joint-space reference while updating the internal model of the end-effector.

orientation plot, indicating that, with our chosen gains, the controller corrects orientation by feedback. Therefore, we have chosen to display only the position error in the following plots. In this slow trajectory, most of the error is due to incorrect gravity compensation. For what concerns the fast linear trajectory, in Figure 18b the cartesian view and the errors are plotted. Table 8 resumes the RMSE for the simulations. In this case, both gravity compensation and dynamical quantities like inertia and friction were sources of errors.

3.3.2 Experiments

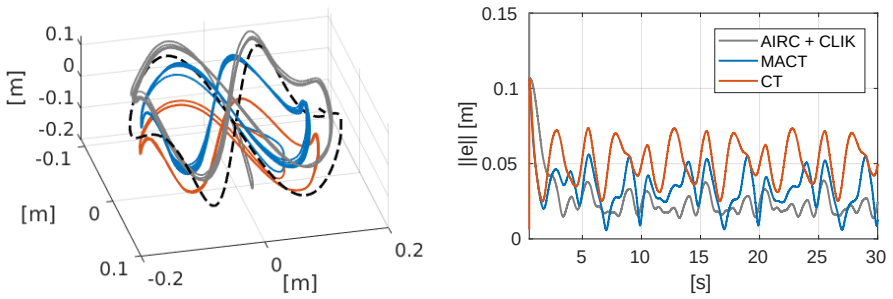
Moving beyond simulation, experimental validation is conducted using the physical *Franka Emika Panda* robot. To introduce an error in the manipulator's inertial model, we mounted an additional gripper (a Pisa/IIT SoftHand [21]) at the end-effector. This decision was made for two primary reasons: 1) the constructor incorporates the inertial parameters of the gripper, thereby facilitating the evaluation of the framework's capability for inertial parameter estimation, and 2) the SoftHand's center of mass is positioned farther from the attachment flange of the robot compared to the conventional two-finger gripper provided by Franka, resulting in a more pronounced impact of the model error on the system's performance. The SoftHand have a mass of $0.83\,kg$ and the center-of-mass in $[0.102, -0.003, 0.037]^T\ m$ w.r.t. its base. Also in this case the experiments are divided into joint and Cartesian space. For what concerns acceleration measurements, it is a traditional problem. Nowadays, with very high sampling rates and the high resolution of sensors, the measurements obtained from numerical filters are more than acceptable. For this reason, we used a numerical filter to take the derivative of the velocity and smooth the signal.

Joint-space experiments: The first experiment involves executing a joint Lissajous trajectory while monitoring actual states. In this phase, we evaluate both the trajectory tracking error and the estimation of the unknown end-effector's inertial parameters of the MACT controller. In this test, the gains are set to the same values as those used in the simulation, as defined in (73).



(a) Robot movements. **(b)** Joint errors on estimating trajectory.

Figure 20: Robot movements and trajectory error on the estimated trajectory. During this phase, our algorithm attempts to estimate the dynamic parameters of the unknown hand.

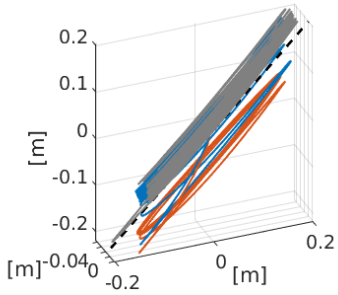


(a) Cartesian trajectory. **(b)** Position error.

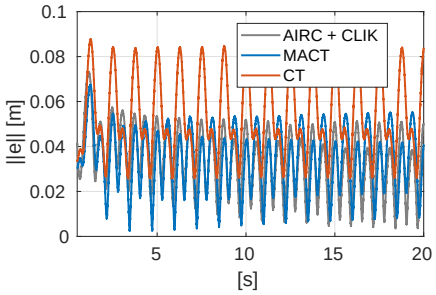
Figure 21: Experimental error on the slow 3D Lissajous Cartesian trajectory. The tracking error is comparable among the three controllers, primarily due to the high gains employed. The dashed line represents the desired trajectory.

As shown in [42] and Section 3.2.3, the classical representation of the rigid body inertia (the mass, the three coordinates of the center of mass, and the six components of the central inertia tensor) for each link of the kinematic chain is not completely observable from the measurements of the robot's torque. To achieve convergence to the last link's inertial parameters, the adaptive controller's update law was restricted to the last link of the kinematic chain (where the unknown end-effector is attached). To obtain the reference for the last link's inertial parameters, we merge the Franka link 7 inertia with the one of the SoftHand using the Huygens-Steiner theorem.

Figure 19 represents the estimation of the mass (19a) and the 3 first inertia moment (19b) parameters (those are the ones that have a major impact on the dynamics) and related real values showing the capability of our adaptive controller to correct the known dynamic model to go toward the real parameters of the system. The error norms of the final parameters are $e_m = 0.0340\,kg$, $e_{COM} = 0.0284\,m{\cdot}kg$, $e_I = 0.0282\,m^2{\cdot}kg$ for the mass, first moment, and second moment of Inertia. In Figure 20a snapshots of the trajectory executed by the Franka robot are merged to give an overall idea of the robot movements. Figure 20b shows the joint errors on the same trajectory. The overall RMSE value of the entire experiment is $||e||_{rms} = 0.741$ rad. Since from time $t = 100s$ the error is quite repetitive, the RMSE value from this time is 0.655 rad.

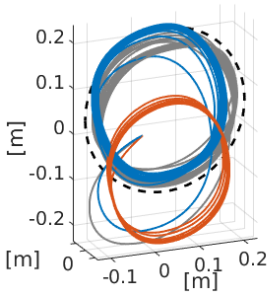


(a) Cartesian trajectory.

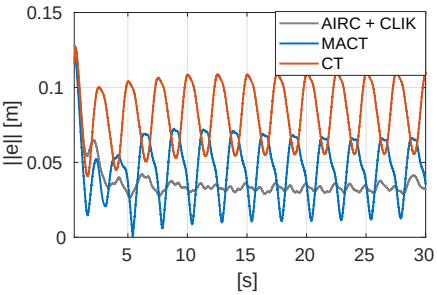


(b) Position error.

Figure 22: Fast Cartesian linear trajectory Experiment. The high dynamic movement generates more error in the non-adaptive control. The dashed line is the desired path.



(a) Cartesian trajectory.



(b) Position error.

Figure 23: Cartesian Circle Trajectory Experiment. This experiment demonstrates the effect of decreased gains on the performance of a non-adaptive controller. As gains are reduced, the error in the controller's trajectory increases. The dashed line represents the desired path.

	CT	AIRC	MACT
RMSE liss3D [m]	0.0537	0.0267	0.0350
RMSE linear [m]	0.0571	0.0347	0.0387
RMSE circle [m]	0.0862	0.0357	0.0491

Table 9: RMSE of the Cartesian trajectory tracking obtained with the actual hardware, employing three distinct control algorithms: classical Computed Torque (CT), Adaptive Inertia Related Control (AIRC) and our Modified Adaptive Computed Torque (MACT).

Cartesian-space experiments: We tested the MACT controller on three different trajectories performed by the *Franka* arm against CT and AIRC + CLIK controllers. In the tests, the end-effector had unknown dynamic parameters, including its mass, center of mass, and moments of inertia. We used two different sets of gains for k_{ξ_p} for the modified adaptive computed torque, i.e.,

$$\begin{aligned} k_{\xi_p}{}^a &= \text{diag}(100, 100, 100, 10, 10, 10), \\ k_{\xi_p}{}^b &= \text{diag}(50, 50, 50, 5, 5, 5), \end{aligned} \tag{76}$$

while k_v is the same of the simulations. The AIRC has the same gains as the joint controller. For each trajectory, Figure (a) shows the desired and actual Cartesian trajectory, while Figure (b) shows the tracking error over time. In the first two trajectories (Lissajous 3D and linear) our cartesian controller uses $k_{\xi_p} = k_{\xi_p}{}^a$. In the third trajectory (a circle) the gains are switched to $k_{\xi_p} = k_{\xi_p}{}^b$ to decrease the stiffness. The first test is shown in Figure 21, showing a slow 3D Cartesian trajectory (Figure 21a) and the associated tracking error (Figure 21b). We did the same test on the fast Cartesian linear trajectory in Figure 22 where the dynamic uncertainties should have a major impact on the error. In both trajectories, there is not so much difference in the tracking error, so we chose to decrease the stiffness on the circle trajectory. Figure 23 shows the trajectory (Figure 23a) and the error (Figure 23b) using these new gains. As a result, decreasing the gains leads to increasing the error on the non-adaptive computed torque, while the adaptive controllers maintain similar errors reached on the other trajectories. The root-mean-square of the experiments conducted above is presented in Table 9.

3.3.3 The throwing problem

This section presents the throwing task, as an example of a dynamic task where an accurate knowledge of the dynamic model has an important effect in terms of its success rate. In our scenario, the controller lacks prior knowledge regarding the inertial characteristics of the object and must acquire this information during the task it-self. In this type of task, the object has to reach a desired landing position $p_d = (x_d, y_d, z_d)$. For this purpose, the gripper must release the object in a precise Cartesian pose and with a precise velocity. Considering the projectile motion, the object movement is planar on the plane defined by the end-effector velocity at the release time and the gravity acceleration vector $\vec{g}$. In our reference system is aligned with the z axes. The final end-effector position and velocity influence directly the landing distance from the releasing pose through the range equation,

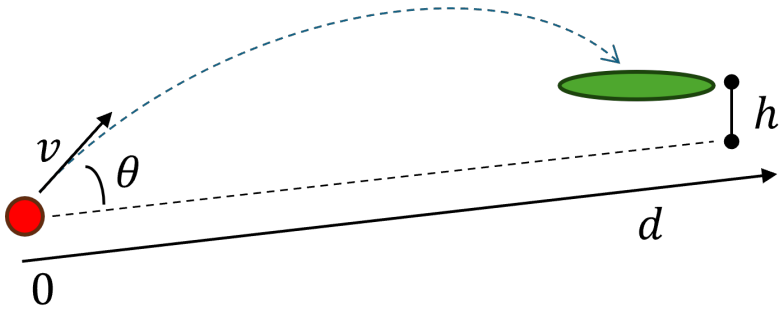


Figure 24: Projectile Motion Scheme. This figure illustrates the factors that influence the landing points of a thrown object. The primary factors are the initial velocity (v) and the angle (θ).

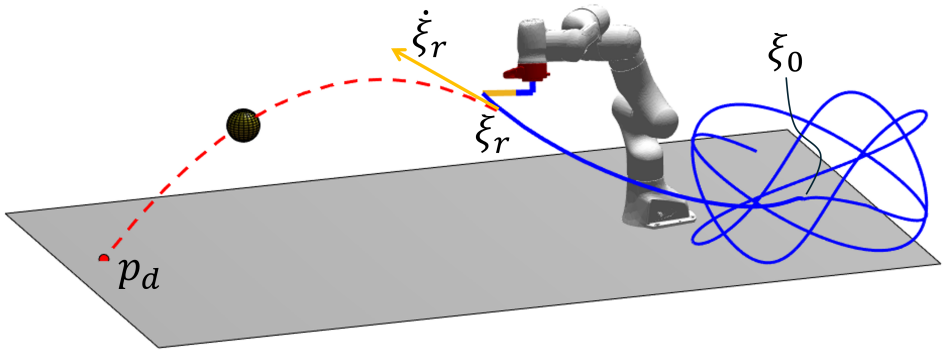


Figure 25: Example of a dynamic task. The robot refines dynamic parameters to improve the throwing performance.

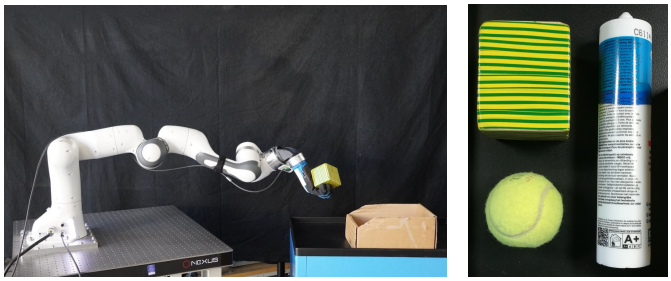
i.e.,

$$d = \frac{v\cos(\theta)}{g}\Big(v\sin(\theta) + \sqrt{v^2\sin^2(\theta) + 2gh}\Big), \tag{77}$$

where $d \in \mathbb{R}$ is the landing distance, $v \in \mathbb{R}$ is the initial speed, $\theta \in \mathbb{R}$ is the throwing angle, $g \in \mathbb{R}$ is the gravity acceleration and $h \in \mathbb{R}$ is the height difference between the releasing and the landing points. Figure 24 provides a visual representation of the formula (77).

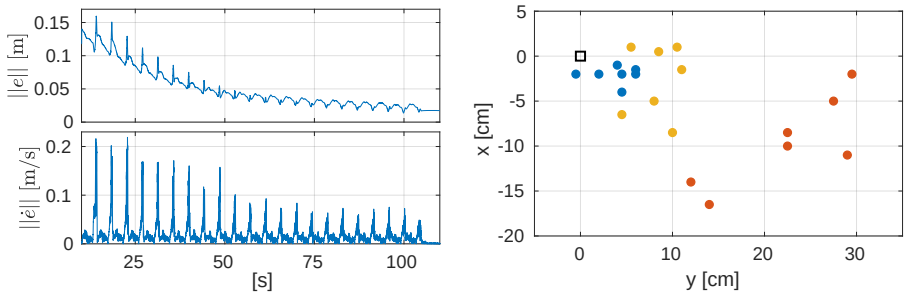
In this analysis, we presume that the object's release time is instantaneous. Consequently, the trajectory followed by the robot prior to the release does not influence the trajectory of the object after release. We select a minimum-jerk trajectory to traverse from the initial end-effector position $\xi_0 \in \mathbb{R}^3$ to the release point $\xi_r \in \mathbb{R}^3$ with velocity $\dot{\xi}_r \in \mathbb{R}^3$. The selection of this trajectory for task execution was made to ensure smooth movements, characterized by continuous acceleration and minimal jerk. The orientation is considered fixed to maintain no angular velocity and simplify the hand opening motion. It is worth noting that the solution proposed in this work to perform the throwing is not necessarily the optimal one. There are several works in the literature that address the problem of generating optimal and robust throwing motions [80, 138]. However, the primary focus of this section is on evaluating the enhancement introduced by our controller in dynamic tasks. The integration of this approach with an optimal throwing motion will be a subject of future research endeavors.

The experimental setup comprises a Franka Emika Panda equipped with a SoftHand



(a) Setup. **(b)** Objects.

Figure 26: Experimental setup and objects used in the throws. The objects are a wooden box (221 g) in the top right, a heavy tennis ball (427 g), and a silicon tube (445 g).



(a) Tracking error. **(b)** Landing point distribution.

Figure 27: Throwing experiments. (a) position and velocity tracking errors: to reduce the errors in the inertia parameter, the manipulator executes a series of repetitions of the throwing trajectory without releasing the object at the end of the movement. Subsequently, the launch is performed. (b) landing point distribution for **adaptive low gains**, **non-adaptive low gains**, **non-adaptive high gains** (the black rectangle is the desired landing point).

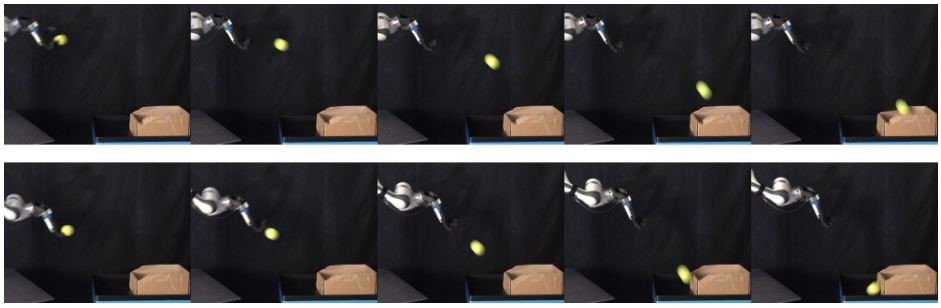


Figure 28: The throwing sequence of the robot. The heavy tennis ball is only able to reach the box in the scenario where adaptive control is employed.

as the gripper. The manipulator, commencing from a predetermined initial position, is tasked with propelling the grasped object to the designated target. The dynamic model of the manipulator incorporates the inertia of the SoftHand, but not that of the grasped object. This decision was made to simulate the scenario where the manipulator is required to launch an unfamiliar object acquired using a soft, under-actuated gripper. In such a situation, even if the system possesses complete knowledge of the object's inertia, the structure of the gripper precludes the ability to ascertain the precise manner in which the object is grasped, thereby rendering it infeasible to modify the dynamic model of the robot with the necessary accuracy.

Precise throwing: In this study, we conducted tests with three distinct controllers: our adaptive computed torque with low gains ($k_p \approx 30N/m$), the conventional non-adaptive computed torque with low gains ($k_p \approx 30N/m$), and the conventional non-adaptive computed torque with high gains ($k_p \approx 300N/m$). The decision to evaluate the non-adaptive controller with two gain levels was motivated by the common practice of increasing manipulator stiffness to mitigate inertial model errors. However, this approach entails elevated interaction forces with the environment, compromising system safety. The primary objective of this test is to demonstrate that our methodology can achieve superior performance while maintaining a low stiffness profile. The gain k_v was selected in each case to achieve critical or overdamped damping behavior, thereby preventing oscillations.

As previously introduced, the motion performed in this series of tests comprises the

minimum jerk trajectory that connects the designated starting point to the desired release point, thereby propelling the object to the intended target. For the purpose of throwing, we utilize a silicon tube (Figure 26b), which weighs 445 g. Prior to executing the throwing motion, the adaptive controller executes a four-repeat sequence of the desired trajectory. This allows the update law to correct the dynamic model. It is noteworthy that this strategy is not the optimal approach for estimating inertial parameters. Various works in the literature address this issue, including [19, 105]. However, the optimal trajectory generation for inertial parameter estimation is not the primary focus of this work and warrants a separate discussion.

To assess the performance of the controllers, we evaluated the dispersion of the object's position after being thrown relative to the desired target. As depicted in Figure 27b, the results obtained by the three tested controllers demonstrate the superiority of our approach over the others. In Figure 27a, the convergence of the position and velocity errors during the "learning phase" before the throw is shown. The RMSE for the position and velocity of the throwing movement are $6.03 \cdot 10^{-4}\, m$, $0.0174\, m/s$, respectively.

Throwing an object into a box: To further validate the performance of our adaptive control system, we conducted a second phase of throws. The goal of this phase was to successfully land the object within a box positioned over the robot's natural workspace. The experimental setup for these tests is depicted in Figure 26a. In this comparative analysis, we evaluate the adaptive and non-adaptive controllers against two novel objects (see Figure 26b): 1) a wooden box and 2) a heavy tennis ball. In these tests, we observe that the experiments involving non-adaptive controllers fail to reach the box due to their inability to compensate for the errors caused by the weight of the grasped object. Figure 28 presents a photographic sequence illustrating this phenomenon with the heavy tennis ball.

4 Throwing with an Elastic Wrist

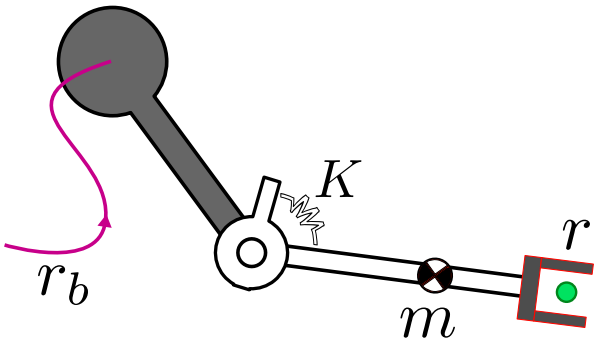


Figure 29: Floating Elastic Wrist

In applications that need precise manipulation and contact with the environment, robotic wrists are essential for improving the dexterity and flexibility of robotic systems. Numerous methods, concentrating on both rigid and compliant structures, have been developed throughout time to model and control robotic wrists [10]. Although they provide strong control and stability, rigid wrist designs frequently falter in jobs that call for compliance and flexibility in response to outside pressures. On the other hand, as shown in [77], compliant wrists which are made of elastic or flexible components, are interesting because of their capacity to absorb shocks and adjust to environmental uncertainty.

Within the realm of compliant wrists, elastic joints, and stiffness-based control schemes are receiving growing attention. Work on variable stiffness actuators (VSAs) [134] and other compliance mechanisms highlights how adding elasticity to joints can produce smoother robot–environment interactions and enhance safety in human–robot collaboration. Elastic elements also offer an avenue for energy storage and release, as demonstrated in [39]. The European DARKO project further underscores these advantages: by leveraging an elastic wrist, the robot can perform pick-and-place operations, and particularly high-speed throwing more efficiently, achieving the end-effector velocities required for such dynamic tasks.

The floating nature of the wrist permits the decoupling of the wrist system from the connected manipulator. However, in the final part, we integrate the wrist with the Franka manipulator, considering the constraints that it requires. This section describes in depth the theoretical and control aspects of elastic throwing using a Franka arm combined with an elastic wrist system. It systematically moves from the comprehensive nonlinear model to a simplified linearized representation, followed by separate solutions for optimal wrist and base motions.

4.1 Complete Model of the Floating Wrist

We consider a robotic system comprising two rigid links: a floating base and an end-effector (EE). These two bodies are connected by a Series Elastic Actuator (SEA). The floating base has full six degrees of freedom, three translational and three rotational, and can move freely in three-dimensional space. The EE is coupled to the base through an internal joint, driven by a motor described by coordinate θ, and elastically connected to the EE's coordinate q. By so, the difference $\theta - q$ is the spring deflection in the series elastic actuator.

The model is obtained from the Euler-Lagrange equation of motion (see [54]) using the base frame trajectory and the wrist angle as coordinates. The base frame is free to move and follow a trajectory $r_b \in SE(3)$ in position and orientation, meanwhile the wrist angle $\phi \in \mathbb{R}$ is the rotation of the wrist joint. Defining $q = (r_b, \phi)$ as the configuration, and $\nu_b \in \mathbb{R}^6$ as the base twist, the system dynamics is

$$M(q)\dot{\nu} + C(q, \dot{\nu})\dot{\nu} + G(q) + \begin{bmatrix} 0 \\ K(\phi - \theta) \end{bmatrix} = \begin{bmatrix} \tau_b \\ 0 \end{bmatrix}, \tag{78}$$

where $\nu = [\, \nu_b^T, \dot{\phi}\,]^T \in \mathbb{R}^7$ are the system velocities, $M \in \mathbb{R}^{7\times7}$ is the positive definite mass matrix, $C \in \mathbb{R}^{7\times7}$ is the Coriolis matrix, $G \in \mathbb{R}^7$ is the gravity vector, $K \in \mathbb{R}$ is the stiffness of the wrist joint, $\theta \in \mathbb{R}$ is the wrist motor angle, and $\tau_b \in \mathbb{R}^6$ is the wrench at the base. Finally, define $r \in SE(3)$ as the end-effector pose, $\xi \in \mathbb{R}^6$ as the end-effector twist, and let $J \in \mathbb{R}^{6\times7}$ be the Jacobian matrix that maps the system velocities to the end-effector twist with the relation $\xi = J\nu$. Here, we assume that the inputs of the system are the wrist motor angle θ and the base trajectory r_b. Moreover, external forces are considered zero in order to simplify the discussion.

4.2 Simplified Model of the Floating Wrist

For analytic traceability, we focus on the wrist dynamics only, assuming zero inertia and damping of the base. This simplification results in a linear second-order differential equation:

$$I\ddot{q} = -k\big(q - \theta\big) - c\dot{q} + mr_c\Big(\big(\ddot{x}_b - g\sin(\alpha)\big)\sin(q) - \big(\ddot{z}_b + g\cos(\alpha)\big)\cos(q)\Big) \tag{79}$$

where $I \in \mathbb{R}$ the wrist inertia, $k \in \mathbb{R}$ is the spring stiffness, $c \in \mathbb{R}$ is the damping coefficient, $m \in \mathbb{R}$ is the end-effector(EE) mass, $r_c \in \mathbb{R}$ the distance from the EE's center of mass, $g \in \mathbb{R}$ gravitational acceleration, $\alpha \in \mathbb{R}$ is the angle of the base relative to gravity, and $\ddot{x}_b, \ddot{z}_b \in \mathbb{R}$ are the base accelerations in x and z direction respectively. Then, we linearize the nonlinear dynamics around a nominal operating point ($q \approx 0$), leading to

$$I\ddot{q} = \Big(r_c m\big(\ddot{x}_b - g\sin(\alpha)\big) - k\Big)q - c\dot{q} + k\theta - r_c m\big(\ddot{z}_b + g\cos(\alpha)\big). \tag{80}$$

This simplified model is used in the optimal problem definitions to find the optimal wrist motor movement and base accelerations. This part is divided into two sections: the first related only to the wrist motor movement 4.3, and the second related to only the base accelerations 4.4.

4.3 Optimization of the Wrist Movement

In this section, the system (80) is considered without base accelerations ($\ddot{x}_b = \ddot{z}_b = 0$). Considering an input change for static gravity compensation as $\theta = u_b + k^{-1} r_c mg\cos(\alpha)$, and an effective stiffness $k_{eff} = k + r_c mg\sin(\alpha)$, it is possible to write the dynamic equation (80) as a secondary order ODE:

$$\ddot{q} + 2\zeta\omega_0\dot{q} + \omega_0^2 q = \frac{k}{I}u_b \tag{81}$$

where $\omega_0^2 = \frac{k_{eff}}{I}$ is the natural frequency and $2\zeta\omega_0 = \frac{c}{I}$ is the damping factor. In this tractation, we consider the underdamped case ($0 \le \zeta < 1$) because the overdamped case eliminates the benefits of having an elastic system. Having $x = [q, \dot{q}]^T$, and rearranging the terms in matrix form we have

$$\dot{x} = A_b x + B_b u_b = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix} x + \begin{bmatrix} 0 \\ k/I \end{bmatrix} u_b \tag{82}$$

In the general case, the solution of the differential equation (81) is

$$q(t) = e^{-\zeta\omega_0 t}\left(q_0\cos(w_d t) + \frac{\dot{q}_0 + \zeta\omega_0 q_0}{\omega_d}\sin(\omega_d t)\right) + \int_0^t g(t-h)u(h)dh, \tag{83}$$

with $g(t)$ as the impulse response of the system, i.e.

$$g(t) = \frac{k}{I\omega_d} e^{-\zeta\omega_0 t}\sin(\omega_d t), \tag{84}$$

where $\omega_d \triangleq \omega_0\sqrt{1-\zeta^2}$.

Our objective is to maximise the end-effector angular velocity $\dot{q}(T)$ at a prescribed time T while ensuring the tip is at its neutral position and considering the wrist motor angle constraint $\theta_{\max}$. The optimal problem is formulated as

$$\begin{aligned} \max_{u_b} \quad & \frac{1}{2}\dot{q}(T) \\ \text{s.t.} \quad & q(T) = 0 \\ & \dot{x}(t) = A_b x(t) + B_b u_b(t) \\ & |\theta(t)| \le \theta_{\max} \end{aligned} \tag{85}$$

A single switch of θ is allowed at the time $t_1 \in [0, T]$, reflecting the one-stroke nature of a throwing motion. A similar approach is used in [39], where the purpose is to use elastic elements to have a kicking motion.

Introducing the co-states $\lambda = [\lambda_1, \lambda_2]^T$, the Hamiltonian is

$$H = \lambda_1 x_2 + \lambda_2\big(\frac{k}{I}u - \omega_0^2 x_1 - 2\zeta\omega_0 x_2\big). \tag{86}$$

Because u enters (86) linearly and the admissible set is compact, the optimal control is *bang-bang*:

$$u^\star(t) = \theta_{\max}\,\mathrm{sign}\big(\lambda_2(t)\big). \tag{87}$$

or,

$$u^\star(t) = \begin{cases} -\theta_{\max}, & 0 \le t < t_1, \\ \theta_{\max}, & t_1 \le t \le T. \end{cases} \tag{88}$$

The purpose of the following sections is to find the switching function $\lambda_2(t)$ in order to obtain the control switching time t_1. Different from [39], the gravity and the damping are taken into account in the optimal problem. The slope of the wrist in space α acts as a stiffness change, and so it modifies the system oscillation frequency ω_0. In order to simplify the discussion, we first introduce the simplest case. Then, we introduce the viscous damping.

4.3.1 Optimal control for the lossless level case ($\zeta = 0, \alpha = 0$)

With $\zeta = 0$ and $\alpha = 0$, equation (82) reduces to the well-known undamped oscillator. The Pontryagin Hamiltonian

$$H = \lambda_1 x_2 + \lambda_2 \omega_0^2 (u - x_1) \tag{89}$$

leads to the co-state ODE

$$\dot{\lambda} = \begin{bmatrix} 0 & -1 \\ \omega_0^2 & 0 \end{bmatrix}\lambda$$

with terminal condition $\lambda(T) = [\nu; 1]^\mathsf{T}$, and $\nu \in \mathbb{R}$ a constant value. The unique solution is

$$\lambda_2(t) = \cos\big(\omega_0(T-t)\big) + \frac{\nu}{\omega_0}\sin\big(\omega_0(T-t)\big). \tag{90}$$

Enforcing a single zero of (90) together with $q(T) = 0$, exploiting equation (83), yields closed-form expressions for the Lagrange multiplier and the switching instant:

$$\nu = -\omega_0 \cot\big(\omega_0(T-t_1)\big), \qquad t_1 = \frac{T\omega_0 - \arccos\big(\frac{1}{2}\big(1+\cos\omega_0 T\big)\big)}{\omega_0}.$$

The attainable tip speed is

$$\dot{q}(T) = \theta_{\max}\omega_0\Big[2\sqrt{1-\cos^4(\tfrac{\omega_0 T}{2})} - \sin(\omega_0 T)\Big]. \tag{91}$$

4.3.2 Viscously damped case ($\zeta > 0$)

Considering the Hamiltonian (86), the co-state now satisfies

$$\dot{\lambda} = \begin{bmatrix} 0 & \omega_0^2 \\ -1 & 2\zeta\omega_0 \end{bmatrix}\lambda, \qquad \lambda(T) = \begin{bmatrix} \nu \\ 1 \end{bmatrix}. \tag{92}$$

Solving gives the damped sinusoid

$$\lambda_2(t) = e^{\zeta\omega_0(T-t)}\Big(\cos(\omega_d(T-t)) + \tfrac{\nu+\zeta}{\sqrt{1-\zeta^2}}\sin(\omega_d(T-t))\Big). \tag{93}$$

At this point, t_1 can be obtained numerically by integrating equation (83) on the entire horizon T, keeping in mind that $x(T) = 0$. Once t_1 is known, ν follows from the boundary condition $\lambda_2(T) = 0$, i.e.

$$\nu = -\Big(\zeta + \sqrt{1-\zeta^2}\cot\big(\omega_d(T-t_1)\big)\Big) \tag{94}$$

4.4 Optimization of the Base Movement

Next, we isolate the optimization problem for base movements, assuming constant wrist position q. In this way, we can directly use the rigid optimization already developed in [117] to obtain the base movement. This strategy combines the simplicity of the base optimization with the effectiveness of the elasticity in the wrist, leading to a general improvement of the performance.

Initially, a throwing point p_t and velocity v_t are selected inside the manipulator workspace, solving an optimization problem. Then, the method computes an intermediate robot configuration q_i that, using a constant acceleration of joints, reaches the throwing point p_t and the velocity v_t in a time T. Then, the robot computes the minimum-jerk trajectory to go from the initial configuration q_0 to the intermediate one in the predefined time t_i. Finally, the complete trajectory is sent to the robot that performs the throwing motion. In this framework, some aspects require a bit of attention, and they are:

- After reaching the throwing point, the robot has to stop. This is achieved by creating a breaking trajectory that decrease all the joint velocities in a small amount of time.
- Since the hand has an opening delay, the opening time is anticipated with respect to the reaching of the throwing point.
- The hand orientation is fixed with the palm oriented towards the throwing velocity to permit the object to leave the hand freely.

5 A general approach for generating artificial human-like motions from functional components of human upper limb movements

Human-likeness (HL) is a key characteristic for artificial systems designed for a safe, effective and trustworthy human-machine interaction, e.g. with humanoid robots or virtual avatars [13, 102]. HL is a broad term which encompasses several characteristics of the system, such as its design, appearance and motion [104]. While much has been done to increase the realism of body design, with a large spread of hyper-realistic humanoids [22, 48, 52], the motion of these systems is far from the naturalness of human movements. The problem of generating Human-Like movements is still under-explored, albeit representing a key aspect for the realism, acceptability and predictability of human-machine interaction [2, 29, 37, 137], even in non-anthropomorphic systems (such as industrial manipulators) where ensuring these features also only at the end-effector could improve the interaction between human and robot [56, 75].

Human motion has different key features which make it peculiar with respect to movements generated by classical planning algorithms. For example, in [49] the authors analyzed human hand motion in 3D space during reaching tasks, with and without obstacles, finding that the movement paths are largely planar. In [74], the authors found that there is a relation between the curvature of a path and the velocity at which humans have to follow it. In [36] the authors observed that humans tend to minimize jerk during movement execution.

Many researchers proposed strategies for generating Human-Like movements in different applications [50, 91]. One of the most popular solutions to achieve HL is to formalize an optimization problem whose functional cost is devised from neuroscientific observations. For example, in [97] the authors developed an optimization-based framework to generate minimum-jerk trajectories building on [36], while [67] exploited the minimization of the torque-change following the model proposed in [126]. However, such optimization approaches usually build upon hypotheses on motion generation, which can reduce the

variability of the planned movement (and, sometimes also lack experimental support [87]). To cope with this problem, an interesting approach was the one proposed in [103] where the authors proposed a model of motion planning based on constraint hierarchy instead of optimizing some cost functions. In this way, the model focuses more on satisfying the constraints related to the desired task (e.g. final target arm posture, obstacle avoidance, etc.) instead of minimizing a predefined cost index.

Another possible approach to generate Human-Like movements exploits learning and data-driven methods. This is a solution used very often in the field of animations and computer graphics where, after an extensive campaign of data recording via motion capture systems, the recorded datasets are used to train neural networks to animate the avatars [72, 89, 136]. For example, in [88] the authors used a recurrent neural network to act as a near-optimal feedback controller generating stable and realistic behaviour. Another example is [101], where the authors used Generative Adversarial Neural Networks for synthesizing gestures directly from speech. Some of these approaches are also applied in robotic applications for the generation of human-like movements with humanoid robots [110]. However, the common limitation of learning-based methods is related to the need for reliable datasets, whose dimensionality can be significant.

An interesting approach that lies in between model and learning-based solutions exploits Dynamic Movement Primitives (DMP) [109]. The idea is to use a dynamical system with convenient stability properties and modulate it with nonlinear terms such that it achieves a desired attractor behaviour. One of the strengths of this framework is the low number of demonstrations required to handle different situations [78]. This approach has been extensively studied and used in literature [108]. One problem that this method has to deal with is related to obstacle avoidance. The solution presented in several papers is to add a second nonlinear term to guide the evolution of the dynamical system around obstacles. There are two ways to compute this term: 1) through potential fields to repulse the system from the obstacles [46, 59]; 2) through neural networks learning the coupling term from a set of examples [99, 125]. These approaches permit to handle also time-varying environments given the negligible computational time required to compute this term. However, the selection of the specific potential field could influence the behaviour of the trajectory, which could lose its desired characteristics, while implementing learning-based techniques could require a large number of examples to generalize for a wide range of obstacle setups.

A possible solution to design an efficient Human-Like motion planning framework, overcoming the aforementioned issues, is to directly embed main human motion characteristics in the algorithmic structure. Many works in the literature addressed the analysis of human motion to extract movement patterns and obtain a reduced yet meaningful characterization of human kinematics [107]. Regarding the upper limb motion, in [8] the authors exploited functional Principal Component Analysis (fPCA) to identify a geometrical basis of mathematical functions whose elements can be combined to reconstruct the overall trajectory. These basis elements were also used to develop a planning algorithm in the joint space domain, which intrinsically embeds HL in the generated motion [9]. This planner, however, is strictly related to the kinematic description used to acquire human upper limb data, and a mapping strategy is needed to generalise the planning outcomes to manipulators with different kinematic structures. A solution to the latter problem was proposed in [7], where Cartesian impedance control was used to implement fPCA-based planning with manipulators with redundant anthropomorphic kinematic architectures - although dissimilar with respect to the human model used for functional mode extraction. However, these approaches in the joint space are associated with non-negligible computational time: for example, while obstacle-free planning can be solved in a closed form, devising a trajectory in presence of obstacles requires solving an optimization problem,

which can require up to several seconds.

To address both the problem of mapping and the reduction of the computational time, we propose a novel planning algorithm able to compute Human-Like trajectories of artificial upper limb/arm directly in the Cartesian domain. To this aim, we built upon the results we presented in [12], where we showed that a geometrical representation of the human end-effector trajectory in terms of functional elements still holds in the Cartesian space, confirming the outcomes reported in [8] at the joint level. This approach permits to obtain a reference trajectory with an intrinsic Human-Like behaviour which can be applied to any kinematic chain used for describing an artificial manipulator in a lower time than the previous approach developed in the joint domain (less than 7 ms on average). The results reported in this section are reported in the work "*A general approach for generating artificial human-like motions from functional components of human upper limb movements*" published in the journal Control Engineering Practice [11].

5.1 Method

5.1.1 Functional Principal Component Analysis (fPCA)

Functional Principal Component Analysis (fPCA) is a statistical method to identify a geometrical basis of functions whose elements can be combined to reconstruct time series. In this section, we will provide a brief introduction to the underpinning theory and its application - without loss of generality - to the description of hand trajectories (i.e. the trajectories of the end-effector of the upper limb kinematic chain), while referring the interested reader to [100] for more details. Given a dataset of hand motions, the generic motion $x(t)$ can be represented as a weighted sum of a set of basis functions $S_i(t)$, or functional Principal Components (fPCs) extracted from the dataset, that is:

$$x(t) \simeq \bar{x} + S_0(t) + \sum_{i=1}^{s_{max}} \alpha_i \circ S_i(t) \tag{95}$$

where $\bar{x}$ is the average pose of the hand, $S_0(t)$ is the average trajectory across all the trajectories in the dataset, α_i is a vector of weights, s_{max} is the number of basis elements, $S_i(t)$ is the i^{th} basis element, the symbol $\circ$ represents the Hadamard product (i.e. the element-wise product) and $t \in [0, 1]$ is the normalized time axis.

The first element of the functional basis or first fPC can be computed from the R motions of the dataset as:

$$\max_{S_1} \sum_{j=1}^{R} \left(\int S_1(t) x_j(t) dt \right)^2 \tag{96}$$

subject to

$$||S_1(t)||_2^2 = 1 \tag{97}$$

The other components $S_i(t)$ can be computed as:

$$\max_{S_i} \sum_{j=1}^{R} \left(\int S_i(t) x_j(t) dt \right)^2 \tag{98}$$

subject to

$$||S_i(t)||_2^2 = 1 \tag{99}$$

$$\int_0^{t_{end}} S_i(t) S_k(t) dt = 0, \forall k \in \{1, ..., i-1\} \tag{100}$$

In this manner, we can identify a basis of functional elements, ordered in terms of the explained variance that each element accounts for.

For our purpose, to obtain a general representation of the human hand motion, we used the dataset proposed in [6], containing the recording of 30 different activities of daily living performed by 30 different subjects belonging to three main classes of actions: Transitive, Intransitive and Tool-mediated, which were assumed to be representative of the human example [8]. However, in [12], we recomputed fPCA independently for each participant (90 trajectories for each participant) showing that, with the same set of tasks, a stable functional representation in terms of the shape of the fPCs can be extracted from a reduced dataset (90 movements) even if it contains heterogeneous tasks. This is possible thanks to the structure of fPCA, which allows us to handle datasets containing time series that are very different from each other without requiring a large number of examples. Instead learning-based methods, to generalize on more variable patterns, require higher dimensionality of the datasets. For example, in [53], the authors recorded, from 10 different subjects, a total of 400 trials taking into account only 8 different reaching movements (similar to the ones that, in our classification, belong to the Intransitive movements). Deep learning based solutions are notably even more data-hungry. To provide an example, recent generative based solutions require thousands of independent motion examples replicated on tenths of different characters [98].

5.1.2 Planning Algorithm

The fPCs extracted from a dataset that can be considered representative of the most common upper limb movements can be used to plan trajectories that intrinsically embed HL. In the following section, we provide a formalization of the planning problem starting with the no-obstacle case, and then we extend the approach to deal with the presence of an arbitrary number of fixed obstacles. Of note, fPCA is performed for each Degree of Freedom (DoF) separately that, in our case, are the Cartesian position and orientation of the hand with respect to the center of the chest. In the following, we report the equations for a single DoF of the end effector, while the extension to multiple DoFs (e.g. the six DoFs describing the pose of the end effector) is trivial.

The reconstruction of the single DoF trajectory can be attained as:

$$x(t) \simeq \bar{x} + S_0(t) + \sum_{i=1}^{s_{max}} \alpha_i S_i(t) \tag{101}$$

To find the coefficients $\bar{x}$ and α_i given a set of constraints to be satisfied we can define an equation system to obtain the desired trajectory to be planned. For example, setting the initial and final position, velocity and acceleration, the following equation system is defined:

$$\begin{bmatrix} 1 & S_1(t_0) & \dots & S_5(t_0) \\ 1 & S_1(t_f) & \dots & S_5(t_f) \\ 0 & \dot{S}_1(t_0) & \dots & \dot{S}_5(t_0) \\ 0 & \dot{S}_1(t_f) & \dots & \dot{S}_5(t_f) \\ 0 & \ddot{S}_1(t_0) & \dots & \ddot{S}_5(t_0) \\ 0 & \ddot{S}_1(t_f) & \dots & \ddot{S}_5(t_f) \end{bmatrix} \begin{bmatrix} \bar{x} \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} x(t_0) - S_0(t_0) \\ x(t_f) - S_0(t_f) \\ \dot{x}(t_0) - \dot{S}_0(t_0) \\ \dot{x}(t_f) - \dot{S}_0(t_f) \\ \ddot{x}(t_0) - \ddot{S}_0(t_0) \\ \ddot{x}(t_f) - \ddot{S}_0(t_f) \end{bmatrix} \tag{102}$$

by solving the system we can obtain the desired planned trajectory

$$x(t) = \bar{x} + S_0(t) + \sum_{i=1}^{5} \alpha_i S_i(t) \tag{103}$$

In the presence of obstacles, instead of numerical optimization, it is possible to define a set of via points (e.g. points defined in the trajectory domain through which the trajectory

itself has to go [130]) and build a similar extended system of equations to solve the problem in a closed form. In a nutshell, the idea is to plan the trajectory in pieces between two successive points, ensuring continuity at the junction point. This strategy moves the problem of collision avoidance to the selection of the best via points to guide the trajectory around the obstacles. The optimal via point guarantees collision avoidance with a minimum path length. To find it, we can use a sampling-based algorithm that selects a set of possible points (that ensure obstacle avoidance) inside the workspace and then identify the one with the minimum path length. This approach exploits the low computational load in generating a single trajectory to perform a random sampling in space to find the best solution in a short time. For our purpose, the velocity and acceleration values have to be set in the via point as constraints to solve the equation system. Without loss of generality, we set the velocity as the mean velocity to go from the initial to the final target point, while we decide to pass the via point with an acceleration equal to zero.

5.2 Simulation Framework

To evaluate the performance of our approach we tested it in simulation against a classical DMP approach. We have chosen this type of planner as a comparison for testing our method because it is placed in the middle between optimization-based techniques (usually computationally expensive) and learning-based techniques (that are faster to compute a solution but greedier in terms of data required for training and generalisation). Briefly, a DMP for a single DoF trajectory y of a discrete movement (point-to-point) is defined by the following set of nonlinear differential equation

$$\begin{aligned} \tau \dot{z} &= K(g-y) - Dz + (g-y_0)f(x) \\ \tau \dot{y} &= z \\ \tau \dot{x} &= -\alpha x \end{aligned} \tag{104}$$

where x is the phase variable and z is an auxiliary variable. Parameters K and D are respectively the spring and the damping terms which define the behaviour of the second-order system described by in (104). With the choice $\tau > 0$, $D = 2\sqrt{K}$ and $\alpha > 0$ the convergence of the underlying dynamic system to a unique attractor point at $y = g, z = 0$ is ensured. The forcing term $f(x)$ is defined as a linear combination of N nonlinear basis functions, which enables the robot to follow any smooth trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Phi_i(x)}{\sum_{i=1}^{N} \Phi_i(x)} x \tag{105}$$

Usually, the classical implementation of DMP uses for the forcing term a basis of Gaussian functions [59]. However, as discussed in [47], there are also other types of basis functions that could be used, such as the mollifiers-like basis functions given by:

$$\Phi_i(x) = \begin{cases} \exp\left(-\frac{1}{1-r(x)^2}\right), & \text{if } r < 1 \\ 0, & \text{otherwise} \end{cases} \tag{106}$$

with $r(x) = |a_i(x - c_i)|$, where c_i is the center and a_i is the width of the function. To select the suitable number of functions for implementing DMPs, we run a preliminary analysis with different cardinality of the basis of function elements. We started with 50 elements (which was stated as the usual maximum number of functions to be used in [47]) and we brought it down to 10 elements evaluating the jerk and the tracking performances. We found that the performance levels of all the tested setups were comparable. For

our purpose, we used 21 functions. This solution exhibited a slightly lower level of jerk compared to the other cardinalities we tested. To learn the weights for the forcing term, we considered human reaching motions extracted from the same dataset we used to perform fPCA [6] and we trained DMPs on one of these movements. For additional details on the building of this subset of motions, we refer the reader to Section 5.2.1.

To implement obstacle avoidance, we used the strategy presented in [95], where a dynamic potential field was used in combination with DMP to deviate the trajectory around the obstacles. In this work, the repulsive field is defined as:

$$U_{dyn}(y,z) = \begin{cases} \lambda(-\cos\theta)^{\beta}\frac{\|z\|}{p(y)}, & \frac{\pi}{2} < \theta \leq \pi \\ 0, & 0 \leq \theta \leq \frac{\pi}{2} \end{cases} \tag{107}$$

where θ is the angle taken between the current velocity z and the relative position of the trajectory with respect to the obstacle (with respect to the latter the distance is defined as $p(y)$, while β is a scalar coefficient). Mathematically, given the equation in (107), the potential field generates a repulsion force only when the trajectory moves towards the obstacle, while in all the other cases it does not influence the evolution of the dynamic system.

The validation of the planning algorithm is composed by different parts. First, given that our approach is based on sampling, we have studied how the output of the algorithm is influenced by the number of samples taken. After that, we move to the evaluation of the Human-Likeness of the generated trajectory which is divided into three parts: 1) comparison with a set of real reaching motions extracted by the original dataset; 2) large-scale simulation with different sets of obstacles; and 3) a reduced set of scenarios that reproduce some daily living activities, where the trajectories produced by the two planners served as references for the controllers of an anthropomorphic kinematic chain. More details regarding the simulation scenarios are provided in the following. All the validation is performed using MATLAB R2020b.

5.2.1 Comparison with Real Human Motion

To evaluate the Human-Likeness of the proposed approach and DMPs, we compared it with a set of recorded human motions to check the similarity of the output of the planning algorithm with them. Given that the algorithm is designed to compute a trajectory to reach a desired position from an initial point, we extracted from the dataset proposed in [6] 180 human upper limb reaching movements and compared the similarity of the generated path with respect to the human example. More specifically, given that the original dataset does not contain this type of movement, we extracted for 3 subjects the 30 intransitive motions performed (3 repetitions of 10 different tasks) and we manually segmented the forward and the backward motion of each trial. For each example movement contained in this subset, we extracted the initial and the final point, which were then used as start and goal positions for the planning algorithms.

5.2.2 Large Scale Obstacle Avoidance Simulation

Given that we do not have a dataset of real human motion performing reaching motion in the presence of obstacles, we performed this evaluation in simulation comparing the output of our planning algorithm with respect to DMP. To do this, we generated a large number of simulation settings, where each of them is defined in terms of a starting point, an ending point and a set of spherical obstacles. We have decided to use spherical obstacles due to the ease of verifying the collision on this shape. However, this choice does not limit the extension of this framework to any general obstacle shape [94]. We defined 4

different types of possible scenarios and, for each group, we generated 1000 different realizations of it. The 4 different classes are: 1) 5 spherical obstacles with $r = 0.1m$; 2) 1 single spherical obstacle with $r = 0.2m$; 3) 3 spherical obstacles with $r = 0.1m$ and 4) 3 spherical obstacles with $r = 0.2m$. All these scenarios were generated randomly extracting starting points, target points and the centres of the obstacles.

5.2.3 Dynamic Arm Simulation

The second part of the validation regards the application of the planner to daily living sample tasks and the usage of these trajectories as a reference to control an anthropomorphic kinematic chain. The dynamic of the arm was simulated using the "Robotics Toolbox for MATLAB" developed by Peter Corke [23].

To follow the Cartesian reference, we implemented a Cartesian computed torque controller defined as:

$$\tau = J^T M_X(\ddot{\xi}_d + K_V\dot{\tilde{\xi}} + K_P\tilde{\xi}) + h(q,\dot{q}) - J^T M_X \dot{J}\dot{q} \tag{108}$$

where ξ_d, $\dot{\xi_d}$ and $\ddot{\xi_d}$ are Cartesian desired position, velocity and acceleration, $\tilde{\xi} = \xi - \xi_d$, $\dot{\tilde{\xi}} = \dot{\xi} - \dot{\xi_d}$, J is the Jacobian, M_X is the Cartesian generalized mass and $h(q,\dot{q})$ is the term related to gravity and coriolis effects.

However, the most common kinematic representation of the human arm has 7 DoFs (3 for the shoulder, 2 for the elbow and 2 for the wrist), while a Cartesian controller has only 6 independent inputs. To solve the redundancy of the kinematic chain, a common solution is to design additional controllers projected in the null space of the primary controller. In our case, we exploited this additional DoF to avoid collision with the whole arm (while the planner guarantees collision avoidance only for the end-point trajectory) and to control the elevation of the elbow. For collision avoidance, we can generate a set of repulsive forces that push away the arm from the obstacles and use them to control torque as:

$$\tau_{obs} = J_{elb}^T F_{elb} + J_{arm}^T F_{arm} + J_{forearm}^T F_{forearm} \tag{109}$$

The single repulsive force applied on the link is generated as:

$$F_i = \frac{\xi_i - \xi_{obs}}{||\xi_i - \xi_{obs}||} \cdot \frac{1}{d_i} \tag{110}$$

where d_i is the distance between the i^{th} point in the kinematic chain and the centre of the obstacle. Regarding elbow posture, we can simply implement a joint impedance controller to keep a desired static posture. The control torque can be computed as:

$$\tau_{elb} = -K_v\dot{q} + K_p(\bar{q} - q) \tag{111}$$

where $\bar{q}$ is the desired arm posture. In our implementation, $\bar{q}$ is computed as mean posture when the participants were in the rest position during data recording in [6]. These two control actions can be added to the main controller (108) as τ_{null}:

$$\tau_{null} = P(\tau_{obs} + \tau_{elb}) \tag{112}$$

where $P = I - JJ^T$ is the Cartesian space null projector.

To perform these simulations, we selected 5 different tasks from the list of tasks contained in [6] (Hitchhiking, Block out sun from own face, Stop gesture, Exultation and Self-feeding). These 5 tasks span most of the possible movement directions of the human hand. For each task, four different scenarios are simulated: 1) without obstacles, 2) a single obstacle in the middle of the hand path, 3) a single obstacle in the elbow proximity and 4) the previous 2 obstacles at the same time. Furthermore, to avoid collision with any part of the simulated hand, we set a clearance zone of $0.2m$ from the surface of the obstacles to take into account the volumetric occupancy of the end-effector.

5.3 Results

5.3.1 Sampling Density Evaluation

Since our planning strategy is based on sampling, the first step is to evaluate how the number of samples used to find a possible trajectory affects the behavior of the algorithm both in terms of computational time and characteristics of the trajectory returned. For this reason, we considered the same scenario described in Section 5.2.2 and we tested our algorithm using different numbers of samples, ranging from 10 to 10^4.

In terms of failure rate, the proposed algorithm achieved an overall failure rate which ranges from 0.8% using 10 samples only, to 0.18% using 10^4 samples. Regarding the computational time, it ranges from 0.848*ms* using 10 samples to 46*ms* using 10^4 samples. Merging these two pieces of information we found that 10^3 represents the most effective trade-off to fulfill performance and computational requirements, with an average computational time of 6.88*ms* and a failure rate of 0.2%. In Figure 30 the average computational time and failure rate are computed for each scenario separately for different numbers of samples. Of note, even if the computational time shown is low and suitable for one-shot

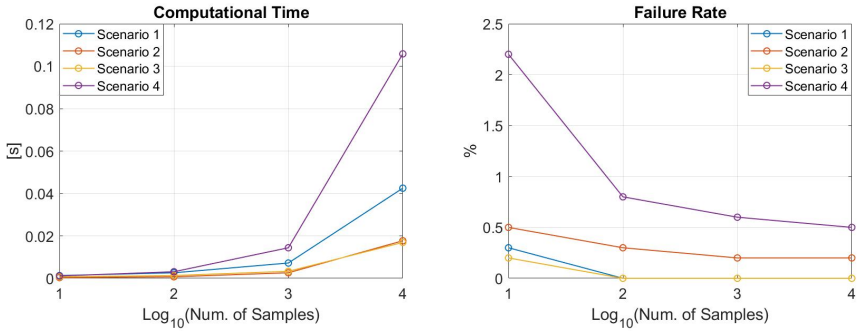


Figure 30: Average computational time and failure rate for each scenario varying the number of samples used by the planning algorithm to find a suitable trajectory.

planning of relatively long trajectories, it is generally larger compared to DMPs. In fact, while our approach has to compute the entire trajectory in advance, DMP allows computing it in real time, while the system is moving, making the latter more suitable for dealing with unstructured environments, e.g. where obstacles move and the trajectory must be recomputed at run time.

Then, we evaluated if the different sampling densities affect the average jerk of the computed trajectory. To do this, we applied the Kruskal-Wallis test [71] to check if the average jerk values obtained for each sampling density come from the same distribution. We applied this test for each scenario separately and we found that the number of samples used by the algorithm does not affect the jerk of the computed trajectory.

5.3.2 Comparison with Real Human Motion

The first index we considered is the similarity between the Cartesian path of the human example movements and the ones produced by our proposed approach and DMPs. To this aim, we computed the Root Mean Square (RMS) distance between the real path and the one produced by the two algorithms. In Table 10 the values of this metric are reported for both our algorithm and DMP-based one, showing that they have similar performances in generating a path similar to the ones done by humans.

Table 10: Average RMS Distance $[m]$ between real human movement and the ones generated by the proposed algorithm (fPC) and DMP

	fPC	**DMP**
RMS Distance	0.0306 ± 0.0194	0.0411 ± 0.0256

Another metric is related to the jerk of the computed trajectories. We computed the jerk for each trajectory contained in this subset and compared it with the ones obtained by the two planning algorithms. In Table 11 the median of the average jerk for each group is reported. We can see that DMP returns trajectory with a higher level of jerk with respect to our approach. We can also observe that real movements tend to have a jerk slightly higher with respect to the one returned by our approach. This can be explained by the fact that, given that the data of the real movements comes from measurements which, also after filtering, can contain residual noise that could amplify the level of computed jerk.

Table 11: Median of average jerk of the real movements, our approach (fPC) and DMP $[m/s^3]$

Real	**fPC**	**DMP**
0.1976	0.0884	1.1503

5.3.3 Large Scale Obstacle Avoidance Simulation

To compare the two planning algorithms we used different metrics to assess the effectiveness in reaching the target and the capability to maintain the human-likeness of the computed trajectory.

The first step is to check the capability of the proposed approaches to compute a trajectory which reaches the target in a given time. In our comparison, the two algorithms are based on completely different planning techniques that can have different problems in guaranteeing the reaching of the goal. For example, our approach is based on viapoint sampling and, as shown in Section 5.3.1, it could be possible to not find a suitable trajectory. On the other hand, DMP generates trajectories exploiting a dynamic system and the convergence behaviour to the target point could strongly depend on the obstacle configuration in the scenario. To evaluate this aspect we have computed the distance from the desired point at the final time. In Table 12 the mean values and their relative standard deviation computed with respect to the five scenarios are reported. We can observe from these results that the fPCA-based planner can reach the target exactly at the desired time while DMP has a variable error depending on the configuration of the obstacles.

Table 12: Final target error at desired final time $[m]$

	fPC	**DMP**
Scenario 1	$2.12 \pm 2.02 \cdot 10^{-14}$	$3.24 \pm 5.42 \cdot 10^{-2}$
Scenario 2	$1.87 \pm 1.75 \cdot 10^{-14}$	$0.80 \pm 1.25 \cdot 10^{-2}$
Scenario 3	$1.87 \pm 1.71 \cdot 10^{-14}$	$1.84 \pm 3.24 \cdot 10^{-2}$
Scenario 4	$2.06 \pm 1.98 \cdot 10^{-14}$	$2.30 \pm 4.56 \cdot 10^{-2}$

After this first assessment, we can move to evaluate the HL of the two algorithms. To do this we used the jerk of the trajectory. We can observe from Table 13 that the fPC-based solution outperforms consistently the DMP.

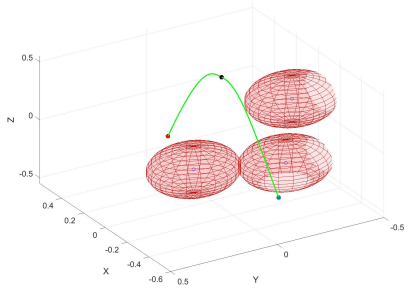


Figure 31: An example of trajectory computed by our algorithm in the presence of obstacles. The starting point, the goal point and the viapoint are respectively drawn in blue, red and black.

Table 13: Median of the mean jerk $[m/s^3]$

	fPC	**DMP**
Scenario 1	0.22	1.19
Scenario 2	0.21	1.59
Scenario 3	0.22	1.31
Scenario 4	0.26	1.28

Higher level of jerk ($p < 0.01$ Wilcoxon rank sum test) obtained by the DMP planner is related to the specific structure of the planner itself. In fact, the usage of a basis of function to learn the desired behaviour as done in DMP produces a ripple in the velocity profile, which brings to a higher level of jerk.

In Figure 31 we have reported an example of a Cartesian trajectory computed with our approach while in Figure 32 its position and velocity temporal evolutions are depicted. It is interesting to note that the behaviour shown by our planner, in terms of velocity profiles, is similar to the one reported in [36].

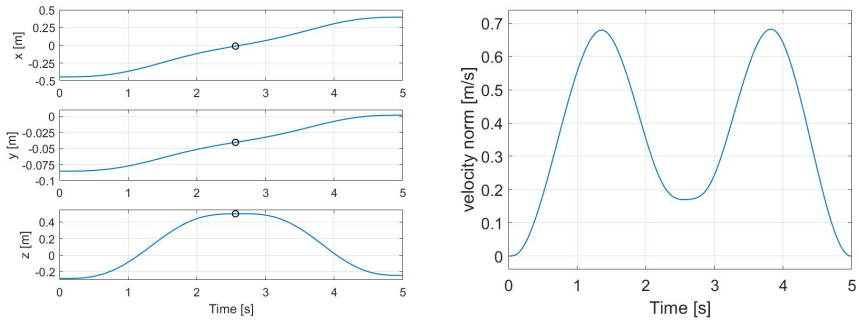


Figure 32: Temporal evolution of the trajectory depicted in Figure 31. In the first image, each Cartesian direction is shown separately (in black the viapoint used by the planning algorithm). In the second image, the plot of the velocity norm achieved by the computed motion is depicted.

5.3.4 Dynamics Arm Simulation

The second part of the validation involves also the dynamic of the kinematic chain. The goal is to check if the properties found in the previous analysis are also preserved by

a manipulator which uses a Cartesian controller to follow the computed trajectory as reference. A snapshot of these simulations is reported in Figures 33 and 34.

The first step, as done in the previous analysis, is to test the capability of the two planners to compute a trajectory to reach the target pose. Our planning algorithm is able to find a feasible trajectory for each of the 20 scenarios taken into account. Instead, the DMP has a final median error norm of $0.049m$. However, with this technique, there are two scenarios ("Hitchhiking" task with 2 obstacles and "Stop" task with 2 obstacles) which have noticeable errors in reaching the final pose (respectively $0.13m$ and $0.48m$).

After that, we can evaluate the HL of the movement generated by the planners themselves and the combination between the planning algorithm and the controller. To quantify this property the main index used in this work is the jerk of the motion produced. Given that the end-effector trajectory has 6 DoFs, to keep the consistency of the measurement units, we split the jerk analysis into two values, one for the end-effector translation and one for the rotation. The first step is to evaluate jerk for the reference trajectories computed by the two algorithms. For each trajectory, the mean of the jerk norm of the movement is computed. The results are reported in Table 14. We can observe a significant difference between the jerk values produced by the two methods. The statistical difference between the two planning algorithms was tested with the Wilcoxon rank sum test which fails to reject the null hypothesis with a $p < 0.01$.

Table 14: Median of Reference Hand Trajectory Jerk

	Translation $[m/s^3]$	**Orientation** $[rad/s^3]$
fPC	0.28	0.31
DMP	1.65	3.19

The same analysis was performed also on the actual simulated arm motion to check if the control policy affects the smoothness of the movement. In this case, we have evaluated both the Cartesian performance (as done for the reference movements) and the joint behaviour. For the joint domain, we have computed for each movement the average jerk across all the seven joints of the kinematic chain. From the results reported in Table 15, we can see that the proposed approach performs better than the DMP. Also in this case the statistical difference was tested using the Wilcoxon rank sum test and it fails to reject the null hypothesis with a $p < 0.01$.

Table 15: Median of Real Arm Trajectory Jerk

	Translation $[m/s^3]$	**Orientation** $[rad/s^3]$	**Joint** $[rad/s^3]$
fPC	0.32	0.97	0.78
DMP	1.87	5.77	3.00

6 Human-like motion for grasping moving objects

Grasping is one of the basic actions which permit us to perform complex tasks and interact with the environment. For this reason, object manipulation with robotic systems generates huge interest in research groups worldwide. [15, 17, 135]. The problem has been studied from several perspectives, from the exploitation of environmental constraints [26, 30] to grasping in cluttered space [35, 124].

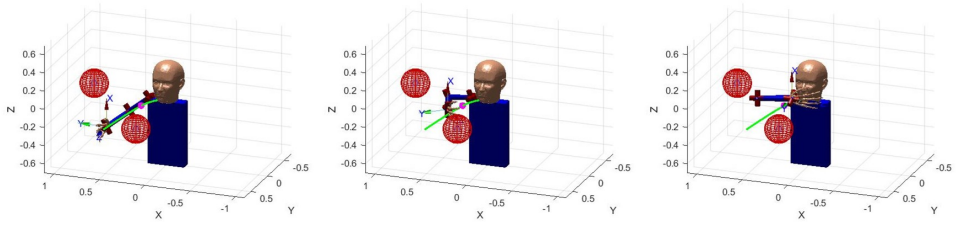


Figure 33: Simulation of self-feeding task with two obstacles. The desired trajectory is depicted in green and the viapoint in purple.

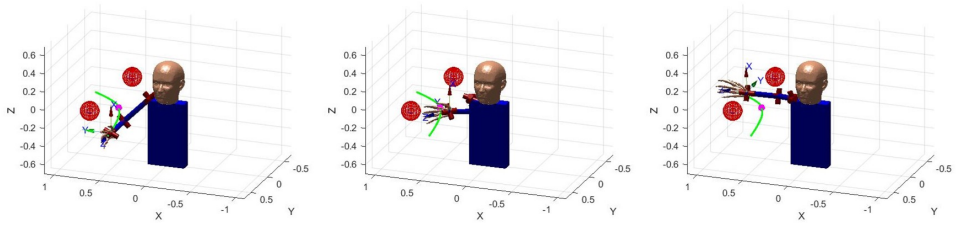


Figure 34: Simulation of hitchhiking task with two obstacles. The desired trajectory is depicted in green and the viapoint in purple.

Impact-aware manipulation is one of these hot topics in this research branch in the last period. In fact, in the real world, there are several situations where humans need to handle dynamic tasks, such as pushing or stopping objects with large inertia [63, 122] or performing dual-arm picking [127–129]. Among these, the dynamic grasping of moving objects has shown great interest in the robotic community for a large range of applications not only in real life but also in industrial setups where objects can be moved in different places of a warehouse using conveyor belts. Furthermore, handling relative motions between the robot and the target during grasping could also be beneficial for mobile robots, which could perform the task without stopping.

Historically, one of the most used techniques to solve this problem is visual servo control [58], where the information provided by a camera is used as feedback to control robot motion and reach the desired target. With this approach, the most used setup is the in-hand configuration, where the camera is integrated into the manipulator end-effector [3, 112]. Over the years, several other approaches were developed to handle more complex scenarios. In [83] the authors proposed a framework able to automatically switch between a local planner, which moves the robot to maintain a certain pre-grasp relative pose while the object is moving, and a global planner which moves the gripper in a different relative pose to change the grasping point on the target object. In [66, 106], the authors targeted the problem of catching flying objects with robotic manipulators using approaches based on dynamic systems.

These approaches target only the task of object grasping without giving any guarantees on the motion behavior of the robotic arm. However, to build a system capable of working beside humans it is necessary to guarantee a predictable and acceptable behavior. In this sense, generating human-like motion can sensibly improve the human-robot interaction [137]. However, dealing with the problem of ensuring human-likeness in combination with the time constraint given by the task makes the problem not trivial. Furthermore, as mentioned in the Introduction, we live in a world designed for humans: under this regard, the human inspiration, for what concerns both the end-effector design and the primitives used for grasping, should be also considered.

In this section, we focus on the exploitation of the human-like motion planning algorithm presented previously [11] as a base to build a framework capable of grasping moving objects. Leveraging on the low computational time, the proposed algorithm can recompute a new trajectory to correct the reaching motion with respect to the motion state estimation of the object. To validate this approach, we proposed as testing scenario the grasping of objects placed on a conveyor belt moving at constant unknown velocity.

6.1 Method

The problem of reaching a moving object to perform grasping can be seen as a rendezvous manoeuvre, where the end effector of the manipulator has to intercept the target object with a null relative velocity. This is necessary especially when the time required to close the fingers of the gripper is not negligible.

Under ideal assumption (perfect knowledge of the motion state of the target over the whole trajectory), one can decide the interception position or time, compute all the other kinematic quantities, and plan the desired motion for the grasping. Taking as example the use case proposed here, knowing exactly the initial position of the object and its velocity permits to perfectly predict any position in the future of the object itself. In this case, the problem can be simply solved by fixing the interception position, computing the time to reach the desired point, and performing a single-shot planning solving (102).

However, in the real world, the presence of noise in the sensing system could lead to not negligible errors in the state estimation (for example a drift in the position prediction due to a wrong velocity estimate). This problem makes single-shot planning unfeasible and it is necessary to correct the motion in real time requiring low computational load algorithms.

6.1.1 Missing condition

The first step is to define the missing condition which triggers the replanning routine. To do this, we decided to fix a reference position p_{ref}, based on the workspace of the manipulator, around which the grasp is performed. Based on the estimation of the initial position $\hat{p}_0$ and velocity $\hat{v}_0$, the time of arrival at the reference position can be computed as

$$\hat{t}_{\text{OA}} = \frac{(p_{\text{ref}} - \hat{p}_0)}{\hat{v}_0}. \tag{113}$$

At the generic instant k, the missing condition can be written as

$$|p_{\text{ref}} - (\hat{p}_{\text{k}} + \hat{v}_{\text{k}}\hat{t}_{\text{OA}})| > p_{\text{th}}, \tag{114}$$

which means that, using the actual estimation of position and velocity, at the desired time of arrival the distance between the reference position and the object position is higher than p_{th}. This condition triggers the replanning routine recomputing also the new time of arrival in function of the strategy used.

6.1.2 Adaptation of the planning algorithm for online replanning

Even if the planner presented as in Section 5 could be already feasible for the task, two main modifications were made to improve the output.

The first is to not use the information of the initial and final acceleration to compute the trajectory, exploiting a reduced version of the linear system (102). This choice was made to avoid the computation through numerical differentiation of the actual Cartesian acceleration of the manipulator which should be necessary as an initial condition for replanning the trajectory.

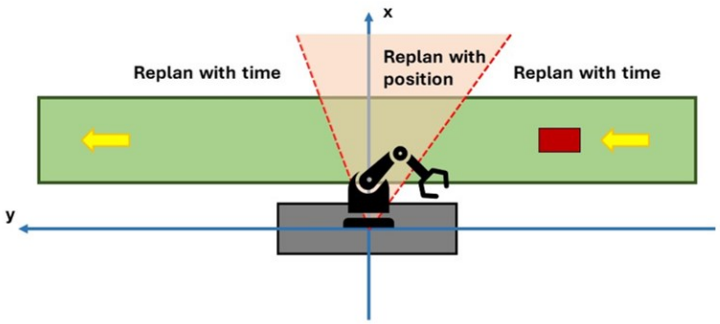


Figure 35: Representation of the Variable structure strategy. Inside the red area, the planning algorithm computes the trajectory toward the new final position keeping the same final time. Outside the red area, the algorithm recomputes the trajectory with a new final time toward a fixed desired position.

The other modification is related to the definition of the time axis of the fPCs. In fact, instead of using the entire fPC to replan the rest of the trajectory as if it were a new motion, we decided to use only the last part of the components to plan the missing part of the motion. This choice was made to avoid uncontrolled increasing in the sampling density in the trajectory.

Following the two modifications described, the new linear system to replan the rest of the motion at the time t_k can be defined as

$$\begin{bmatrix} 1 & S_1(t_k) & S_2(t_k) & S_3(t_k) \\ 1 & S_1(t_f) & S_3(t_f) & S_3(t_f) \\ 0 & \dot{S}_1(t_k) & \dot{S}_2(t_k) & \dot{S}_3(t_k) \\ 0 & \dot{S}_1(t_f) & \dot{S}_2(t_f) & \dot{S}_3(t_f) \end{bmatrix} \begin{bmatrix} \bar{x} \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} x(t_k) - S_0(t_k) \\ x(t_f) - S_0(t_f) \\ \dot{x}(t_k) - \dot{S}_0(t_k) \\ \dot{x}(t_f) - \dot{S}_0(t_f) \end{bmatrix}. \tag{115}$$

6.1.3 Replanning strategy

Once the basic structure of the planner is designed, it is necessary to design the replanning strategy. As said before, given the trajectory of the target object there are infinite possible interception positions that can be chosen. Here we present two possible strategies that can be used to solve this problem.

Variable structure strategy: This method is very simple and consists of keeping one between t_f or $x(t_f)$ constant during replanning and only recalculating the other variable based on the updated motion state estimation. In particular, the idea was to change t_f for larger scale corrections, while choosing different grasping positions $x(t_f)$ in the vicinity of p_{ref} was used for fine-tuning of the trajectory. A narrow area was defined as shown in Figure 35, and the replanning method was changed based on whether the estimated $\hat{p}_k(t_f)$ fell inside or outside of the depicted area. The region was chosen asymmetric with respect to p_{ref}, to favour grasp conditions with a lead on it rather than delayed.

Time augmentation strategy The main idea behind this method is to always replan towards p_{ref} as an initial guess, taking as t_f the $\hat{t}_{\text{OA}}$ calculated by the vision node. Then, for fine-tuning, the value of t_f was incremented or decremented based on a number of corrective terms, and the new $x(t_f)$ was derived from the target's trajectory. The new

final position and time are computed as:

$$t_f = t_{\text{OA}} + \alpha \frac{N-F}{N}(p_o + \lambda(f_{\text{new}} - f_{\text{old}}) + \phi(p_{\text{KF},y} - p_{\text{EE},y})) \tag{116}$$

$$p(t_f) = \hat{p}_{\text{k}} + \hat{v}_{\text{k}} \hat{t}_f \tag{117}$$

where $\hat{p}_{\text{k}}$ and $\hat{v}_{\text{k}}$ are the estimated object kinematics, p_{EE} is the current position of robot end effector, N is the total number of fPCA samples, F is number of already crossed samples, f_{old} and f_{new} are the previous and updated planning node frame rate, and $\alpha, p_o, \lambda, \phi$ are numeric coefficients that were determined by trial and error. A brief explanation for the various corrective terms:

- α and p_o are respectively a scale factor and a constant corrective term, which can be used to correct repeated biases of the system. p_o in particular skews correction towards lead or delay
- $\lambda(f_{\text{new}} - f_{\text{old}})$ was added to weight negatively sudden frequency changes in software operation, to smooth the variation of the velocity reference as much as possible
- $\phi(p_{\text{KF},y} - p_{\text{EE},y})$ was used to account for the asymmetry of the system along the direction of the conveyor belt, by increasing the lead in grasp position when the robot end effector was well behind the targeted object
- $\frac{N-F}{N}$ is a coefficient that becomes 0 when $F \to N$, reducing the dimension of the corrective terms when the robot end effector approached the targeted object, to avoid overshoots.

6.2 Experimental validation

6.2.1 Experimental setup

The experimental setup comprises a 7 DoF Franka Emika manipulator equipped with a SoftHand [21] as gripper, a conveyor belt and an RGB-D camera for the perception of the object (Figure 36). The relative pose between the camera and the robot was calibrated through a custom procedure exploring the AprilTag library [93].

The perception pipeline to get the relative pose from the point cloud recorded by the camera is composed of the following steps: 1) downsampling to decrease the weight of the data processed; 2) cluster extraction to isolate the target; 3) centroid computation to obtain the reference position of the target; 4) segmentation of the planar surface of the cluster; 5) Principal Component Analysis on the surface cluster to obtain the principal dimension of the target.

While the output of the PCA is used to set a reference frame on the object, the information about the position is used as input of a Kalman filter to estimate the velocity of the object. In our case, we chose to model the motion of the object as a constant velocity motion

$$p_{k+1} = p_k + v_k \cdot dt. \tag{118}$$

Starting from this model, defining the state as $x_k = [p_k^T, v_k^T]^T$, the prediction step of the Kalman filter can be defined as

$$\begin{aligned} x_{k|k-1} &= A_t x_{k-1|k-1} \\ P_{k|k-1} &= A_t P_{k-1|k-1} A_t^T + D_{k-1}\, \nu D_{k-1}^T \end{aligned} \tag{119}$$

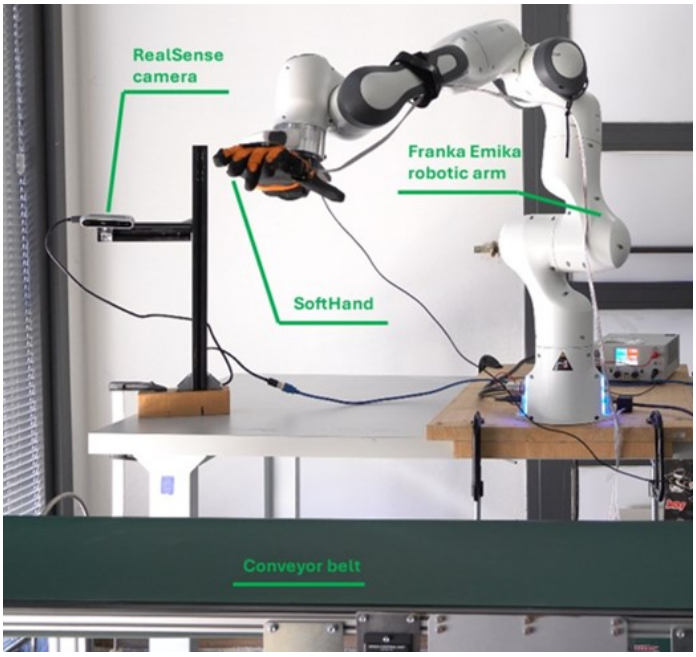


Figure 36: Picture representing the entire experimental scenario. The setup is composed of a Franka Emika Panda manipulator equipped with a SoftHand, a conveyor belt and a RealSense RGB-D camera mounted on a pole aside to the manipulator.

where

$$A_t = \begin{bmatrix} I_3 & dtI_3 \\ 0_3 & I_3 \end{bmatrix}$$
$$D_k = [v_{k-1}^T, v_{k-1}^T]^T.$$

The update step can be defined as

$$\begin{aligned} e_k &= y_k - Cx_{k|k-1} \\ S_k &= R + CP_{k|k-1}C^T \\ L_k &= P_{k+1|k}C^T S_k^{-1} \\ J_k &= I_6 - L_k C \\ x_{k|k} &= x_{k|k-1} + L_k e_k \\ P_{k|k} &= J_k P_{k|k-1} J_k^T + L_k R L_k^T. \end{aligned} \tag{120}$$

where $C = [I_3, 0_3]$ is the observation matrix and R is the noise covariance matrix. Given the assumption of constant unknown velocity of the desired target, the state of the Kalman filter is initialized with the first position registered by the camera and a null velocity.

6.2.2 Replanning strategy testing

The first step was to test the two replanning strategies proposed. In Figure 37 the different behaviour between the two strategies is represented. It can be observed that the Time Augmentation strategy is capable of replanning a lower number of times (shown by the lower number of peaks in the jerk plot) and consequently has a lower level of jerk. This is also confirmed by the mean jerk obtained by the two methods ($2.6 \cdot 10^{-3} m/s^3$ for Variable structure against $0.4 \cdot 10^{-3} m/s^3$ for Time Augmentation).

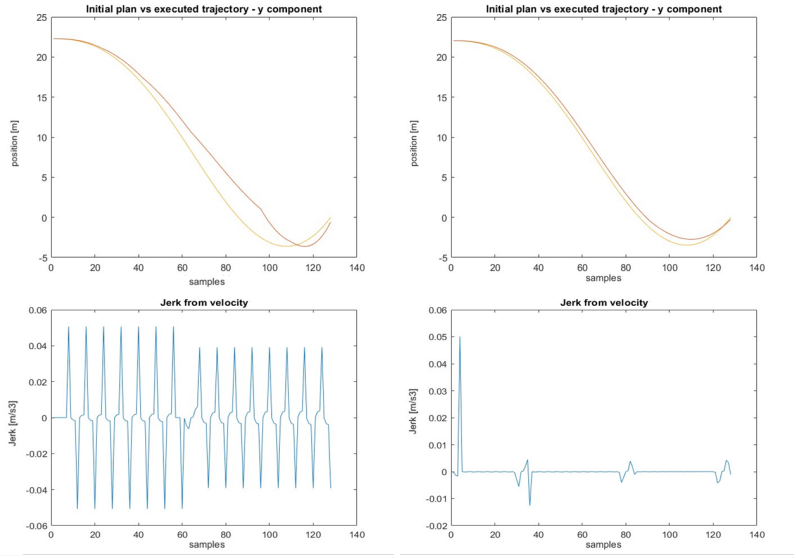


Figure 37: Comparisons between Variable structure replanning strategy (left) and Time Augmentation replanning strategy (right) in terms of planned motion along y direction (direction of motion of the conveyor belt) and of jerk.

6.2.3 Moving object grasping test

The last part of the experimental validation consists of evaluating the success rate in grasping moving objects. To do this, we tested different objects and different velocities of the conveyor belt. In Figure 40 we reported a set of snapshot of one of the trials performed. The grasp strategy implemented with the SoftHand is a top grasp choosing the orientation of the hand in a way that the principal dimension of the object identified through the vision pipeline lies between the thumb and the index finger.

We performed a grand total of 117 trials obtaining an overall success rate of 76%. In Figure 38 we reported the representation of the success rate pie chart of both the total set of trials and the results split between box-like objects and other objects. The results show that the success rate is stable also for heterogeneous objects.

We also evaluated the success rate obtained at different velocities of the conveyor belt. We used three different velocities ($0.16m/s$, $0.26m/s$ and $0.38m/s$). As expected, the success rate decreased with the increasing of the velocity obtaining respectively 95%, 91% and 52% for the tested velocities (Figure 39). This is due to the assumption of not knowing the velocity of the conveyor belt. In this scenario, the estimation of the Kalman filter has to converge to the right velocity estimation. Higher velocities give less time to the filter to converge and, consequently, the planner has less margin to adjust the trajectory and intercept the target. However, we tested as comparison the same framework without the replanning strategy obtaining a success rate of 81% with the low velocity and 31% with the medium velocity (the higher velocity was not tested extensively due to not succeeding in any grasp in the preliminary setup of the validation scenario). These results prove the effectiveness of the replanning strategy proposed.

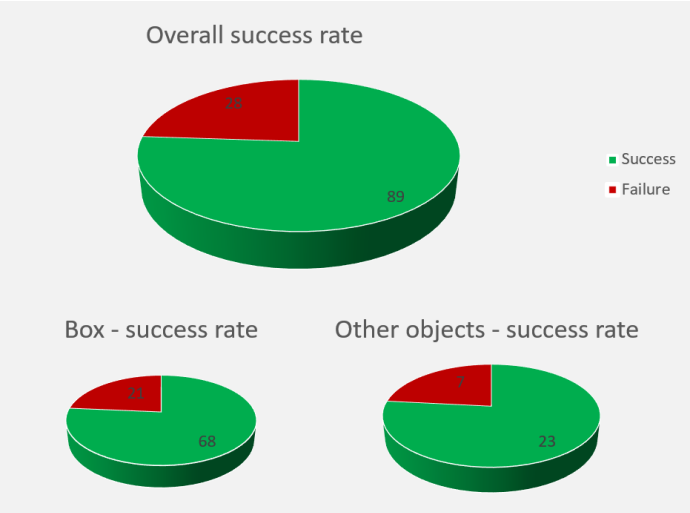


Figure 38: Representation of the success rate obtained during the validation test. The results are presented both as overall success rate and split with respect the type of object (box-type vs other).

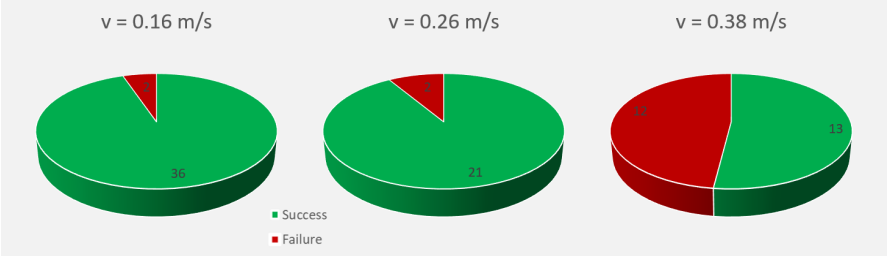


Figure 39: Representation of the success rate obtained at different velocities of the conveyor belt.

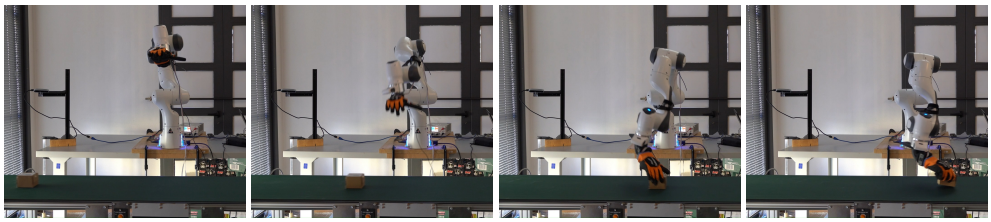


Figure 40: Snapshot of one of the trials performed during the validation tests.

Table 16: Success rate at different velocities

	$0.16\,m/s$	$0.26\,m/s$	$0.38\,m/s$
replan	95%	91%	52%
no replan	81%	31%	0%

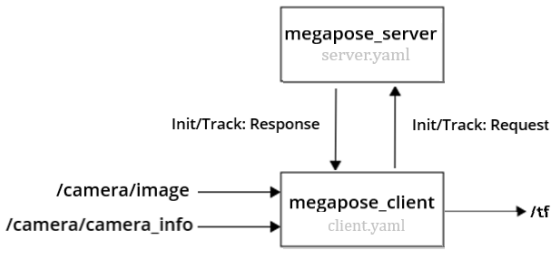


Figure 41: Structure of the ROS wrapper of MegaPose.

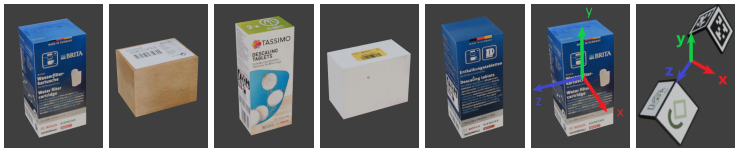


Figure 42: CAD models for MegaPose.

7 Object picking with pneumatic tool

In classical warehouse automation, one of the most used type of end-effector is the suction gripper. This choice was made because the simplicity and efficiency of this solution, allowing robots to pick up a wide range of objects, including fragile or irregularly shaped items, without causing damage. However, this solution directly integrated into the manipulator relies on the fact that robots are grounded in these scenarios, permitting them to have an air supply easily, and they are heavily specialised in picking specific objects.

In the DARKO project, these conditions are not satisfied, given that we deal with a mobile system which has to be capable of doing different types of task. For this reason, for the project a custom hand pneumatic tool has been designed, to permit the system to use it if needed. However, handling this tool with a soft anthropomorphic gripper could lead to additional uncertainties in performing picking with it, related to how the robot grasps the tool itself. In this section, we present a framework for handling this uncertainties and perform picking with pneumatic tool in an efficient way.

7.1 Vision layer

To identify the pose of the tool and the objects in the scenarios, we used MegaPose [73], a method to estimate the 6D pose of novel objects, that is, objects unseen during training, relying on the CAD model of the object itself. To use it in the DARKO framework, we build up a ROS1 wrapper for the MegaPose framework. The package consists of the nodes `megapose_client` and `megapose_server`. These two works in the classical client-server configuration (see Figure 41), where the client subscripts to the camera topics and publishes the pose of the desired object (Figure 42). To exploit this framework, we built the CAD model of the pneumatic tool and some objects in the DARKO scenarios using Blender.

We tested the precision of pose estimation obtained from MegaPose using as groundtruth a marker-based system [93]. For the pneumatic tool, the system is capable to obtain a pose with position error always less than $1\,cm$ even if the pneumatic tool is moving, which permits the overall system to operate in a robust way. For the objects, we tested the framework both with single object in motion, obtaining results similar to the ones obtained

with the tool, and in cluttered scenarios. In this case, the vision system is capable to work even in the case of partial occlusions (Figure 43).

(1) box1 **(2)** box4 **(3)** box2

Figure 43: Three cases of the cluttered scenario tests.

7.2 Visual servoing control

For solving the problem of bringing the end-effector in the desired position for picking, we decide to use visual serving. We can define the Cartesian error using the axis-angle representation as:

$$e = \begin{bmatrix} e_p \\ e_o \end{bmatrix} = \begin{bmatrix} x_d - x \\ rsin(\theta) \end{bmatrix} \tag{121}$$

From this, we can define the dynamic of the Cartesian error e as:

$$\dot{e} = \begin{bmatrix} e_p \\ e_o \end{bmatrix} = \begin{bmatrix} \dot{x}_d \\ L^T \omega_d \end{bmatrix} - \begin{bmatrix} I & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} J_p \\ J_o \end{bmatrix} \dot{q} \tag{122}$$

where $\dot{x}_d$ e ω_d are the Cartesian velocity references, while the matrix L can be obtained from the rotation matrices $R_d = (n_d, s_d, a_d)$ e $R = (n, s, a)$ as:

$$L = -\frac{1}{2}(\widehat{n_d}\widehat{n} + \widehat{s_d}\widehat{s} + \widehat{a_d}\widehat{a}) \tag{123}$$

Here we can define the control law as:

$$\dot{q} = J^+ \begin{bmatrix} I & 0 \\ 0 & L^{-1} \end{bmatrix} \left(\begin{bmatrix} \dot{x}d \\ L^T \omega_d \end{bmatrix} + \begin{bmatrix} K_p & 0 \\ 0 & K_o \end{bmatrix} \begin{bmatrix} e_p \\ e_o \end{bmatrix} \right) \tag{124}$$

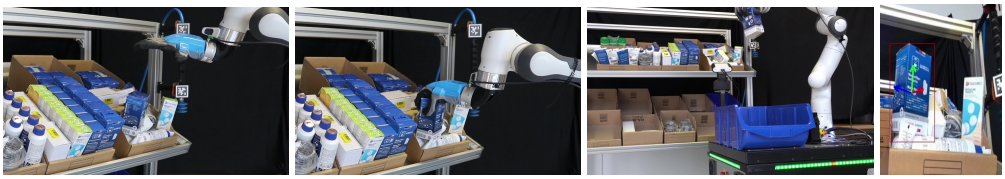
where K_p e K_o are the gains and J is the Jacobian matrix.

7.3 Experimental validation

We tested the framework composed by the vision layer and the visual servoing control both for picking directly with the SoftHand (Figure 44) and through the pneumatic tool handles by the SoftHand (Figure 45). The system shows a good versatility in being used both to grasp objects directly with the SoftHand and by using the pneumatic tool to perform the task. Several scenarios have been tested with different objects, and we reached an overall 95% of success rate.

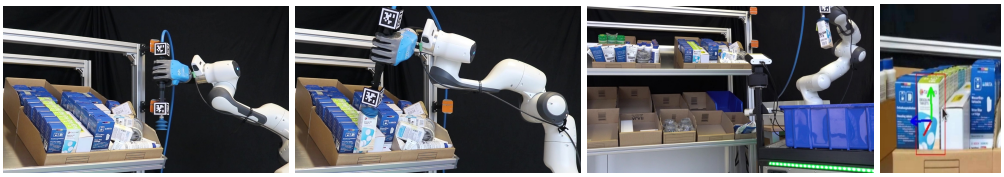
8 Conclusions

In this document, we reported an exhaustive description of the latest results achieved in the DARKO project about motion planning and control for efficient manipulation. All



(1) Reach **(2)** Grasp **(3)** Release **(4)** Pose

Figure 44: Picking with SoftHand: (1) The SoftHand reaches the desired object; (2) The desired object is grasped; (3) The object is released in the target ; (4) Estimated pose from the vision layer.



(1) Reach **(2)** Pick **(3)** Release **(4)** Posa

Figure 45: Picking with SoftHand-Tool: (1) In the first part, the SoftHand grasp the pneumatic tool and reach the target object; (2) The pneumatic tool is activated and the object extracted form the shelf; (3) The object is released in the target box; (4) Estimated pose from the vision layer.

the proposed works have been designed to address the specific needs of the use-case taken into account in this project, and they proved to be effective in fulfilling the desired requirements.

From a control point of view, we have presented a software library for implementing adaptive control for a robotic serial manipulator. To test the feasibility of the proposed library to be used for real-time control implementation, we evaluated the computation time needed to evaluate the required quantities, showing that the results obtained meet the computational constraints for real-time torque control (which generally stands around 1 kHz). Exploiting this library, we have then propose a novel adaptive Computed Torque strategy that is asymptotically stable. The new law does not require the inertia matrix inversion and the solution of the Lyapunov equation. The proposed controller, formulated both in the joint and the Cartesian domains, permits setting the desired stiffness of the manipulator and being robust to inertial parameters error at the same time. We tested them both in general trajectory tracking and in the accomplishment of dynamic tasks like throwing, showing how much this approach can be effective in increasing the efficiency and the precision in the tasks related to the DARKO use case.

From a planning point of view, in this deliverable, we proposed a framework capable to exploit elastic actuators to increase the performances of manipulators which has both rigid and elastic joints, as the one mounted on the DARKO platform. Another result showed here is the development of a new planning algorithm for the generation of human-like motion in artificial arm. The proposed algorithm can compute motions similar to the ones produced by humans, improving the predictability of the system during human-robot interactions, with reduced computational time, leading to a system capable to replan trajectory in real time. This particular feature has been exploited to extend the planning algorithm to the grasping of moving objects. In fact, the low computational load permits to adjust the computed trajectory to intercept the moving target. In the end, we also proposed a visual servo-based framework to perform picking with the pneumatic tool included in the DARKO system, enabling an efficient picking routine for cases where the

SoftHand could struggle (like a well-stacked box on the shelves).

Furthermore, the results reported here not only provide suitable tools to handle manipulation problems related to the DARKO use-case right now, but could also provide a strong base for future improvements of the state of the art regarding motion planning and control of manipulators with rigid and elastic joints, robotic manipulation, and human-robot interaction.

References

[1] Fares J Abu-Dakka and Matteo Saveriano. Variable impedance control and learning—a review. *Frontiers in Robotics and AI*, 7:590681, 2020.

[2] Abdulaziz Abubshait and Eva Wiese. You look human, but act like a machine: agent appearance and behavior modulate different aspects of human–robot interaction. *Frontiers in psychology*, 8:1393, 2017.

[3] Peter K Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, 1993.

[4] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[5] Christopher G. Atkeson, Chae H. An, and John M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. 5(3):101–119, 1986. Publisher: SAGE Publications Ltd STM.

[6] Giuseppe Averta, Federica Barontini, Vincenzo Catrambone, Sami Haddadin, Giacomo Handjaras, Jeremia PO Held, Tingli Hu, Eike Jakubowitz, Christoph M Kanzler, Johannes Kühn, et al. U-limb: A multi-modal, multi-center database on arm motion control in healthy and post-stroke conditions. *GigaScience*, 10(6):giab043, 2021.

[7] Giuseppe Averta, Danilo Caporale, Cosimo Della Santina, Antonio Bicchi, and Matteo Bianchi. A technical framework for human-like motion generation with autonomous anthropomorphic redundant manipulators. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3853–3859. IEEE, 2020.

[8] Giuseppe Averta, Cosimo Della Santina, Edoardo Battaglia, Federica Felici, Matteo Bianchi, and Antonio Bicchi. Unvealing the principal modes of human upper limb movements through functional analysis. *Frontiers in Robotics and AI*, 4:37, 2017.

[9] Giuseppe Averta, Cosimo Della Santina, Gaetano Valenza, Antonio Bicchi, and Matteo Bianchi. Exploiting upper-limb functional principal components for human-like motion generation of anthropomorphic robots. *Journal of NeuroEngineering and Rehabilitation*, 17(1):1–15, 2020.

[10] Neil M. Bajaj, Adam J. Spiers, and Aaron M. Dollar. State of the art in artificial wrists: A review of prosthetic and robotic wrist design. 35(1):261–277. Conference Name: IEEE Transactions on Robotics.

[11] Marco Baracca, Giuseppe Averta, and Matteo Bianchi. A general approach for generating artificial human-like motions from functional components of human upper limb movements. *Control Engineering Practice*, 148:105968, 2024.

[12] Marco Baracca, Paolo Bonifati, Ylenia Nisticò, Vincenzo Catrambone, Gaetano Valenza, A Bicchi, Giuseppe Averta, and Matteo Bianchi. Functional analysis of upper-limb movements in the cartesian domain. In *Converging Clinical and Engineering Research on Neurorehabilitation IV: Proceedings of the 5th International Conference on Neurorehabilitation (ICNR2020), October 13–16, 2020*, pages 339–343. Springer, 2022.

[13] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics*, 1:71–81, 2009.

[14] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 2007.

[15] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[16] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[17] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

[18] B. Bona and Marina Indri. Friction compensation in robotics: an overview. volume 2005, pages 4360–4367.

[19] Vincent Bonnet, Philippe Fraisse, André Crosnier, Maxime Gautier, Alejandro González, and Gentiane Venture. Optimal exciting dance for identifying inertial parameters of an anthropomorphic structure. *IEEE Transactions on Robotics*, 32(4):823–836, 2016.

[20] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[21] Manuel G Catalano, Giorgio Grioli, Edoardo Farnioli, Alessandro Serio, Cristina Piazza, and Antonio Bicchi. Adaptive synergies for the design and control of the pisa/iit softhand. *The International Journal of Robotics Research*, 33(5):768–782, 2014.

[22] Lorenzo Cominelli, Gustav Hoegen, and Danilo De Rossi. Abel: integrating humanoid body, emotions, and time perception to investigate social interaction and human cognition. *Applied Sciences*, 11(3):1070, 2021.

[23] Peter Corke. Matlab toolboxes: robotics and vision for students and teachers. *IEEE Robotics & automation magazine*, 14(4):16–17, 2007.

[24] J. Craig, Ping Hsu, and S. Sastry. Adaptive control of mechanical manipulators. In *1986 IEEE International Conference on Robotics and Automation Proceedings*, volume 3, pages 190–195.

[25] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005.

[26] Cosimo Della Santina, Matteo Bianchi, Giuseppe Averta, Simone Ciotti, Visar Arapi, Simone Fani, Edoardo Battaglia, Manuel Giuseppe Catalano, Marco Santello, and Antonio Bicchi. Postural hand synergies during environmental constraint exploitation. *Frontiers in neurorobotics*, 11:41, 2017.

[27] Jacques Denavit and Richard S Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 1955.

[28] Rajeshree Deotalu and Shital Chiddarwar. Trajectory tracking of the manipulator using adaptive computed torque control. In *2020 IEEE International Conference for Innovation in Technology (INOCON)*, pages 1–6, 2020.

[29] Brian R Duffy. Anthropomorphism and the social robot. *Robotics and autonomous systems*, 42(3-4):177–190, 2003.

[30] Clemens Eppner, Raphael Deimel, José Alvarez-Ruiz, Marianne Maertens, and Oliver Brock. Exploitation of environmental constraints in human and robotic grasping. *The International Journal of Robotics Research*, 34(7):1021–1038, 2015.

[31] Linn Danielsen Evjemo, Signe Moe, Jan Tommy Gravdahl, Olivier Roulet-Dubonnet, Lars Tore Gellein, Vegard Br, et al. Additive manufacturing by robot manipulator: An overview of the state-of-the-art and proof-of-concept results. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[32] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[33] Roy Featherstone and David E. Orin. *Springer Handbook of Robotics*. Springer Cham, 2016.

[34] Martin L Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511, 2017.

[35] David Fischinger and Markus Vincze. Empty the basket-a shape based learning approach for grasping piles of unknown objects. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2051–2057. IEEE, 2012.

[36] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[37] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003.

[38] Marco Gabiccini, Andrea Bracci, D De Carli, M Fredianelli, Antonio Bicchi, et al. Explicit lagrangian formulation of the dynamic regressors for serial manipulators. In *Proceedings of the XIX Aimeta Congress, Ancona, Italy*, pages 14–17, 2009.

[39] Manolo Garabini, Andrea Passaglia, Felipe Belo, Paolo Salaris, and Antonio Bicchi. Optimality principles in stiffness control: The VSA kick. In *2012 IEEE International Conference on Robotics and Automation*, pages 3341–3346. ISSN: 1050-4729.

[40] Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. On the closed form computation of the dynamic matrices and their differentiations. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2364–2359, 2013. ISSN: 2153-0866.

[41] M. Gautier and W. Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on Robotics and Automation*, 6(3):368–373, 1990. Conference Name: IEEE Transactions on Robotics and Automation.

[42] Maxime Gautier and Wisama Khalil. Identification of the minimum inertial parameters of robots. In *1989 IEEE International Conference on Robotics and Automation*, pages 1529–1530. IEEE Computer Society, 1989.

[43] Maxime Gautier and Wisama Khalil. Direct calculation of minimum set of inertial parameters of serial robots. *IEEE Transactions on robotics and Automation*, 6(3):368–373, 1990.

[44] Nooshin Ghodsian, Khaled Benfriha, Adel Olabi, Varun Gopinath, and Aurélien Arnou. Mobile manipulators in industry 4.0: A review of developments for industrial applications. *Sensors*, 23(19):8026, 2023.

[45] Nooshin Ghodsian, Khaled Benfriha, Adel Olabi, Varun Gopinath, and Aurélien Arnou. Mobile manipulators in industry 4.0: A review of developments for industrial applications. *Sensors*, 23(19):8026, 2023.

[46] Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto, and Paolo Fiorini. Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. *Journal of Intelligent & Robotic Systems*, 101:1–20, 2021.

[47] Michele Ginesi, Nicola Sansonetto, and Paolo Fiorini. Overcoming some drawbacks of dynamic movement primitives. *Robotics and Autonomous Systems*, 144:103844, 2021.

[48] Dylan F Glas, Takashi Minato, Carlos T Ishi, Tatsuya Kawahara, and Hiroshi Ishiguro. Erica: The erato intelligent conversational android. In *2016 25th IEEE International symposium on robot and human interactive communication (RO-MAN)*, pages 22–29. IEEE, 2016.

[49] Britta Grimme, John Lipinski, and Gregor Schöner. Naturalistic arm movements during obstacle avoidance in 3d and the identification of movement primitives. *Experimental brain research*, 222:185–200, 2012.

[50] Gianpaolo Gulletta, Wolfram Erlhagen, and Estela Bicho. Human-like arm motion generation: A review. *Robotics*, 9(4):102, 2020.

[51] Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022.

[52] David Hanson. Hanson robotics website, 2023.

[53] Chang He, Xiao-Wei Xu, Xiong-Fei Zheng, Cai-Hua Xiong, Quan-Lin Li, Wen-Bin Chen, and Bai-Yang Sun. Anthropomorphic reaching movement generating method for human-like upper limb robot. *IEEE Transactions on Cybernetics*, 52(12):13225–13236, 2021.

[54] John Hollerbach, Wisama Khalil, and Maxime Gautier. *Dynamics*, pages 37–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[55] John Hollerbach, Wisama Khalil, and Maxime Gautier. *Motion Control*, pages 163–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[56] Damian Hostettler, Simon Mayer, and Christian Hildebrand. Human-like movements of industrial robots positively impact observer perception. *International Journal of Social Robotics*, 15(8):1399–1417, 2023.

[57] T Hsia. Adaptive control of robot manipulators-a review. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 183–189. IEEE, 1986.

[58] Seth Hutchinson, Gregory D Hager, and Peter I Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[59] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[60] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[61] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[62] Pradeep K. Khosla and Takeo Kanade. Parameter identification of robot dynamics. In *1985 24th IEEE Conference on Decision and Control*, pages 1754–1760, 1985.

[63] Harshit Khurana and Aude Billard. Motion planning and inertia-based control for impact aware manipulation. *IEEE Transactions on Robotics*, 40:2201–2216, 2023.

[64] Jung Kim, Gwang Min Gu, and Pilwon Heo. Robotics for healthcare. *Biomedical Engineering: Frontier Research and Converging Technologies*, pages 489–509, 2016.

[65] Jung Kim, Gwang Min Gu, and Pilwon Heo. Robotics for healthcare. *Biomedical Engineering: Frontier Research and Converging Technologies*, pages 489–509, 2016.

[66] Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014.

[67] Mary D Klein Breteler, Stan CAM Gielen, and Ruud GJ Meulenbroek. End-point constraints in aiming movements: effects of approach angle and speed. *Biological Cybernetics*, 85:65–75, 2001.

[68] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[69] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[70] Miroslav Krstic, Ioannis Kanellakopoulos, and Petar V. Kokotovic. *Nonlinear and Adaptive Control Design*. Wiley, 1995.

[71] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[72] Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. A survey on reinforcement learning methods in character animation. In *Computer Graphics Forum*, volume 41, pages 613–639. Wiley Online Library, 2022.

[73] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. Megapose: 6d pose estimation of novel objects via render & compare. *arXiv preprint arXiv:2212.06870*, 2022.

[74] Francesco Lacquaniti, Carlo Terzuolo, and Paolo Viviani. The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica*, 54(1-3):115–130, 1983.

[75] Marta Lagomarsino, Marta Lorenzini, Elena De Momi, and Arash Ajoudani. Promind: Proximity and reactivity optimisation of robot motion to tune safety limits, human stress, and productivity in industrial settings. *arXiv preprint arXiv:2409.06864*, 2024.

[76] Quentin Leboutet, Julien Roux, Alexandre Janot, Julio Rogelio Guadarrama-Olvera, and Gordon Cheng. Inertial parameter identification in robotics: A survey. *Applied Sciences*, 11(9):4303, 2021.

[77] Yu-Feng Lee, Cheng-Yu Chu, Jia-You Xu, and Chao-Chieh Lan. A humanoid robotic wrist with two-dimensional series elastic actuation for accurate force/torque interaction. 21(3):1315–1325. Conference Name: IEEE/ASME Transactions on Mechatronics.

[78] Gianluca Lentini, Giorgio Grioli, Manuel G Catalano, and Antonio Bicchi. Robot programming without coding. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7576–7582. IEEE, 2020.

[79] W. Li and J.-J.E. Slotine. Indirect adaptive robot control. In *1988 IEEE International Conference on Robotics and Automation Proceedings*, pages 704–709 vol.2.

[80] Yang Liu and Aude Billard. Tube acceleration: robust dexterous throwing against release uncertainty. *IEEE Transactions on Robotics*, 2024.

[81] Jadesada Maneeratanaporn, Pakpoom Patompak, Siripong Varongkriengkrai, Itthisek Nilkhamhang, and Kanokvate Tungpimolrut. Adaptive inverse dynamics control for triple rotary joint mainipulator. In *ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 523–527, 2010.

[82] Murilo Marques Marinho and Bruno Vilhena Adorno. Adaptive constrained kinematic control using partial or complete task-space measurements. *IEEE Transactions on Robotics*, 38(6):3498–3513, 2022.

[83] Naresh Marturi, Marek Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Aleš Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Autonomous Robots*, 43:1241–1256, 2019.

[84] Nikos Mavrakis and Rustam Stolkin. Estimation and exploitation of objects ' inertial parameters in robotic grasping and manipulation: A survey. 124:103374.

[85] Nikos Mavrakis and Rustam Stolkin. Estimation and exploitation of objects' inertial parameters in robotic grasping and manipulation: A survey. *Robotics and Autonomous Systems*, 124:103374, 2020.

[86] R.H. Middleton and G.C. Goodwin. Adaptive computed torque control for rigid link manipulations. 10(1):9–16.

[87] Sylvain Miossec and Abderrahmane Kheddar. Human motion in cooperative tasks: Moving object case study. In *2008 IEEE International Conference on Robotics and Biomimetics*, pages 1509–1514, 2009.

[88] Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. *Advances in neural information processing systems*, 28, 2015.

[89] Lucas Mourot, Ludovic Hoyet, François Le Clerc, François Schnitzler, and Pierre Hellier. A survey on deep learning for skeleton-based human animation. In *Computer Graphics Forum*, volume 41, pages 122–157. Wiley Online Library, 2022.

[90] D. Nganga-Kouya, M. Saad, L. Lamarche, and C. Khairallah. Backstepping adaptive position control for robotic manipulators. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 1, pages 636–640 vol.1, 2001. ISSN: 0743-1619.

[91] Clautilde Nguiadem, Maxime Raison, and Sofiane Achiche. Motion planning of upper-limb exoskeleton robots: a review. *Applied Sciences*, 10(21):7626, 2020.

[92] Dinh Vinh Thanh Nguyen, Vincent Bonnet, Sabbah Maxime, Maxime Gautier, Pierre Fernbach, and Florent Lamiraux. FIGAROH: a python toolbox for dynamic identification and geometric calibration of robots and humans. 2023.

[93] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

[94] Joseph O'Rourke and Norman Badler. Decomposition of three-dimensional objects into spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3):295–305, 1979.

[95] Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 91–98. IEEE, 2008.

[96] Bhavik Patel, Ya-Jun Pan, and Usman Ahmad. Adaptive backstepping control approach for the trajectory tracking of mobile manipulators. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1769–1774, 2017.

[97] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE transactions on industrial electronics*, 47(1):140–149, 2000.

[98] Sigal Raab, Inbal Leibovitch, Peizhuo Li, Kfir Aberman, Olga Sorkine-Hornung, and Daniel Cohen-Or. Modi: Unconditional motion synthesis from diverse data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13873–13883, 2023.

[99] Akshara Rai, Giovanni Sutanto, Stefan Schaal, and Franziska Meier. Learning feedback terms for reactive planning and control. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2184–2191. IEEE, 2017.

[100] James O Ramsay, Giles Hooker, and Spencer Graves. *Functional data analysis with R and MATLAB*. Springer Science & Business Media, 2009.

[101] Manuel Rebol, Christian Gütl, and Krzysztof Pietroszek. Real-time gesture animation generation from speech for virtual human interaction. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–4, 2021.

[102] Laurel D Riek, Tal-Chen Rabinowitch, Bhismadev Chakrabarti, and Peter Robinson. How anthropomorphism affects empathy toward robots. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 245–246, 2009.

[103] David A Rosenbaum, Ruud GJ Meulenbroek, Jonathan Vaughan, and Chris Jansen. Posture-based motion planning: Applications to grasping. *Psychological Review*, 4(108):709–734, 2001.

[104] Nina Rothstein, John Kounios, Hasan Ayaz, and Ewart J de Visser. Assessment of human-likeness and anthropomorphism of robots: A literature review. In *Advances in Neuroergonomics and Cognitive Engineering: Proceedings of the AHFE 2020 Virtual Conferences on Neuroergonomics and Cognitive Engineering, and Industrial Cognitive Ergonomics and Engineering Psychology, July 16-20, 2020, USA*, pages 190–196. Springer, 2021.

[105] Paolo Salaris, Marco Cognetti, Riccardo Spica, and Paolo Robuffo Giordano. Online optimal perception-aware trajectory generation. *IEEE Transactions on Robotics*, 35(6):1307–1322, 2019.

[106] Seyed Sina Mirrazavi Salehian, Mahdi Khoramshahi, and Aude Billard. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics*, 32(2):462–471, 2016.

[107] Marco Santello, Matteo Bianchi, Marco Gabiccini, Emiliano Ricciardi, Gionata Salvietti, Domenico Prattichizzo, Marc Ernst, Alessandro Moscatelli, Henrik Jörntell, Astrid M.L. Kappers, Kostas Kyriakopoulos, Alin Albu-Schäffer, Claudio Castellini, and Antonio Bicchi. Hand synergies: Integration of robotics and neuroscience for understanding the control of biological and artificial hands. *Physics of Life Reviews*, 17:1–23, 2016.

[108] Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13):1133–1184, 2023.

[109] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. *Adaptive motion of animals and machines*, pages 261–280, 2006.

[110] Trenton Schulz, Jim Torresen, and Jo Herstad. Animation techniques in human-robot interaction user studies: A systematic literature review. *ACM Transactions on Human-Robot Interaction (THRI)*, 8(2):1–22, 2019.

[111] Marie Schumacher, Janis Wojtusch, Philipp Beckerle, and Oskar von Stryk. An introductory review of active compliant control. *Robotics and Autonomous Systems*, 119:185–200, 2019.

[112] Navid Shahriari, Silvia Fantasia, Fabrizio Flacco, and Giuseppe Oriolo. Robotic visual servoing of moving targets. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 77–82. IEEE, 2013.

[113] Wei-Wei Shang, Shuang Cong, and Yuan Ge. Adaptive computed torque control for a parallel manipulator with redundant actuation. *Robotica*, 30:457 – 466, 05 2012.

[114] Mojtaba Sharifi, Saeed Behzadipour, and Gholamreza Vossoughi. Nonlinear model reference adaptive impedance control for human–robot interactions. *Control Engineering Practice*, 32:9–27, 2014.

[115] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2010.

[116] Giorgio Simonini, Marco Baracca, Tommaso V Cavaliere, Antonio Bicchi, and Paolo Salaris. A novel formulation for adaptive computed torque control enabling low feedback gains in highly dynamical tasks. *IEEE Access*, 2025.

[117] Giorgio Simonini, Riccardo Di Majo, Lorenzo Boccalini, Matteo Guerci, Antonio Bicchi, and Paolo Salaris. Optimality principles in stiffness control: The vsa kick. In *I-RIM Conference, Rome, Italy*, 2023.

[118] Jean-Jacques E. Slotine and Weiping Li. Composite adaptive control of robot manipulators. 25(4):509–519.

[119] Jean-jacques E. Slotine and Weiping Li. Adaptive robot control: A new perspective. In *26th IEEE Conference on Decision and Control*, volume 26, pages 192–198, 1987.

[120] Jean-Jacques E Slotine and Weiping Li. On the adaptive control of robot manipulators. *The international journal of robotics research*, 6(3):49–59, 1987.

[121] Peng Song, Yueqing Yu, and Xuping Zhang. A tutorial survey and comparison of impedance control on robotic manipulation. *Robotica*, 37(5), 2019.

[122] Theodoros Stouraitis, Lei Yan, Joao Moura, Michael Gienger, and Sethu Vijayakumar. Multi-mode trajectory optimization for impact-aware manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9425–9432. IEEE, 2020.

[123] Tairen Sun, Liang Peng, Long Cheng, Zeng-Guang Hou, and Yongping Pan. Composite learning enhanced robot impedance control. 31(3):1052–1059. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[124] Ashok M Sundaram, Werner Friedl, and Máximo A Roa. Environment-aware grasp strategy planning in clutter for a variable stiffness hand. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9377–9384. IEEE, 2020.

[125] Giovanni Sutanto, Zhe Su, Stefan Schaal, and Franziska Meier. Learning sensor feedback models from demonstrations via phase-modulated neural networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1142–1149. IEEE, 2018.

[126] Yoji Uno, Mitsuo Kawato, and Rika Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological cybernetics*, 61(2):89–101, 1989.

[127] Jari van Steen, Daan Stokbroekx, Nathan van de Wouw, and Alessandro Saccon. Impact-aware robotic manipulation: Quantifying the sim-to-real gap for velocity jumps. *arXiv preprint arXiv:2411.06319*, 2024.

[128] Jari Van Steen, Gijs Van Den Brandt, Nathan van de Wouw, Jens Kober, and Alessandro Saccon. Quadratic programming-based reference spreading control for dual-arm robotic manipulation with planned simultaneous impacts. *IEEE Transactions on Robotics*, 2024.

[129] Jari J van Steen, Abdullah Coşgun, Nathan van de Wouw, and Alessandro Saccon. Dual arm impact-aware grasping through time-invariant reference spreading control. *IFAC-PapersOnLine*, 56(2):1009–1016, 2023.

[130] Jonathan Vaughan, David A Rosenbaum, and Ruud GJ Meulenbroek. Planning reaching and grasping movements: The problem of obstacle avoidance. *Motor control*, 5(2):116–135, 2001.

[131] Nguyen X. Vinh. *Flight Mechanics of High-Performance Aircraft*. Cambridge University Press, 1993. Google-Books-ID: ND9dDeOARkMC.

[132] Kenneth J. Waldron and James Schmiedeler. *Springer Handbook of Robotics*. Springer Cham, 2016.

[133] Hanlei Wang. Adaptive control of robot manipulators with uncertain kinematics and dynamics. *IEEE Transactions on Automatic Control*, 62(2):948–954, 2017.

[134] Sebastian Wolf, Giorgio Grioli, Oliver Eiberger, Werner Friedl, Markus Grebenstein, Hannes Höppner, Etienne Burdet, Darwin G. Caldwell, Raffaella Carloni, Manuel G. Catalano, Dirk Lefeber, Stefano Stramigioli, Nikos Tsagarakis, Michaël Van Damme, Ronald Van Ham, Bram Vanderborght, Ludo C. Visser, Antonio Bicchi, and Alin Albu-Schäffer. Variable stiffness actuators: Review on design and components. 21(5):2418–2430. Conference Name: IEEE/ASME Transactions on Mechatronics.

[135] Zhen Xie, Xinquan Liang, and Canale Roberto. Learning-based robotic grasping: A review. *Frontiers in Robotics and AI*, 10:1038658, 2023.

[136] Wenjie Yin, Hang Yin, Danica Kragic, and Mårten Björkman. Graph-based normalizing flow for human motion generation and reconstruction. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 641–648. IEEE, 2021.

[137] Andrea Maria Zanchettin, Luca Bascetta, and Paolo Rocco. Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution. *Applied ergonomics*, 44(6):982–989, 2013.

[138] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[139] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[140] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[141] Krishna S. Dhir Zoltan Dobra. Technology jump in the industry: human–robot cooperation in production. *Emerald Publishing Limited*, 2020.

[142] Jing Zou and John Schueller. Adaptive backstepping control for parallel robot with uncertainties in dynamics and kinematics. 32:1–24, 2014.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017274