Dual-Arm control strategy in industrial environments

Sonny Tarbouriech

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Systèmes Automatiques et Microélectroniques (SYAM)

École doctorale : Information, Structures, Systèmes

Unité de recherche LIRMM

Stratégie de contrôle de robot bi-bras dans un environnement industriel

Présentée par Sonny Tarbouriech

Le 5 Décembre 2019

Sous la direction de Philippe FRAISSE

Devant le jury présidé par

Bruno SICILIANO, Professeur, University of Naples	Examinateur

Et composé de

Vincent PADOIS, Maître de Conférences HDR, Inria Bordeaux	Rapporteur
Michael MISTRY, Maître de Conférences, University of Edinburgh	Rapporteur
Philippe FRAISSE, Professeur, Université de Montpellier	Directeur de thèse
Andrea CHERUBINI, Maître de Conférences HDR, Université de Montpellier	Co-encadrant de thèse
André CROSNIER, Professeur, Université de Montpellier	Co-encadrant de thèse

UNIVERSITÉ DE MONTPELLIER

Titre : Contribution à la stratégie de commande d'un robot bi-bras en milieu industriel

Résumé :

Le besoin grandissant de flexibilité en milieu industriel conduit à reconsidérer la manière dont les robots sont utilisés dans de tels environnements. Il s'ensuit que la relation entre l'homme et les machines doit évoluer au profit d'une plus grande proximité, en leur permettant de partager un espace de travail commun et d'interagir physiquement. Dans cette optique, cette thèse a pour objectif de contribuer au contrôle de robots bi-bras à des fins collaboratives dans un contexte industriel. Pour ce faire, nous proposons une approche de contrôle cinématique réactif basée sur une loi de contrôle en admittance. Celle-ci permet une manipulation d'objets sécuritaire en collaboration physique avec des opérateurs humains. Le contrôleur résout un problème d'optimisation quadratique (QP) afin de trouver le déplacement articulaire permettant de satisfaire la commande spécifiée dans l'espace de la tâche, ceci tout en respectant un ensemble de contraintes (e.g. limites articulaires, évitement de collision). La résolution cinématique peut être adaptée afin de générer des solutions parcimonieuses au niveau des vitesses articulaires, ce qui signifie qu'un nombre minimal d'actionneurs est activé pour assurer la réalisation de la tâche. Cela induit un comportement potentiellement plus sûr dans un environnement évolutif partagé avec des individus. Les plateformes bi-bras comprennent parfois des extensions (par exemple, une base mobile, un torse articulé, etc.). Dans cette thèse, nous présentons une méthode hiérarchique originale pour le contrôle de systèmes multi-robots. Une implémentation open source du travail, Robot Kinematics Control Library (RKCL), a été développée. Cette librairie rassemble tous les composants décrits dans cette thèse et peut être facilement configurée pour inclure de nouveaux robots. Tout au long des développements, des validations expérimentales ont été effectuées sur le cobot mobile à deux bras BAZAR.

Mots-clefs : Manipulation Bi-Bras, Interaction Humain-Robot, Contrôle cinématique

Title: Dual-Arm control strategy in industrial environments

Abstract:

The growing need for flexibility in industrial settings leads to reconsidering the way robotic systems are exploited in such environments. It follows that the relationship between humans and machines has to evolve in favor of more proximity, by letting them share the same workspace and physically interact together. With this in mind, this thesis aims at contributing beyond the state of art in the control of dual-arm robots for collaborative purposes in an industrial context. We propose a generic online kinematic control approach based on an admittance control law which enables safe manipulation of objects in physical collaboration with humans. The controller solves a Quadratic Programming (QP) optimization problem to find the joint space motion that satisfies the task space command while respecting a set of constraints (e.g. joint

limits, collision avoidance). The kinematic solver can be tuned to generate parsimonious solutions at the joint velocity level, meaning that as few actuators as possible are activated to achieve the tasks. This induces potentially safer behavior in an unstructured environment shared with humans. Dual-arm platforms are sometimes extended to include additional robots (e.g., mobile base, articulated torso, ...). In this thesis, we also present an original hierarchical method for the control of multi-robot systems. An open-source implementation of the work, Robot Kinematics Control Library (RKCL), is available. It implements all the components described in this thesis and can be easily configured to work with new robots. Throughout the developments, experimental validations have been performed on the dual-arm mobile cobot BAZAR.

Keywords: Dual-Arm Manipulation, Human-Robot Interaction, Kinematic Control.

Discipline : Systèmes Avancés et Microélectronique

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
UMR 5506 CNRS/Université de Montpellier
Batiment 5 - 860 rue de St Priest

Contents

List of Figures

Nomenclature

Bold symbols are for vectors.

Acronyms

CoM	Center of Mass
CPU	Central Processing Unit
DoF	Degrees of Freedom
HMD	Head-Mounted Display
HQP	Hierarchical Quadratic Programming
pHRI	Physical Human-Robot Interaction
PID	Packages Integration and Deployment
QP	Quadratic Programming
RKCL	Robot Kinematics Control Library
V-REP	Virtual Robot Experimentation Platform

List of symbols

N_{dof}	Number of robot's DoF
N_{eq}	Number of equality constraints
N_{group}	Number of joint groups composing the robot
N_{ineq}	Number of inequality constraints
N_{task}	Number of tasks
T	Sampling time
$\boldsymbol{S}(\boldsymbol{\omega})$	Skew-symmetric matrix associated with $\boldsymbol{\omega}$
$\boldsymbol{\epsilon}$	Residual error on the kinematic task
$\mathbf{A}$	Linear coefficient matrix of the inequality constraint
$\mathbf{B}$	Diagonal damping matrix
$\mathbf{F}$	Task space force vector
$\mathbf{J}$	Jacobian matrix

$\mathbf{K}$	Diagonal stiffness matrix
$\mathbf{M}$	Task space moment vector
$\mathbf{R}_B^A$	Rotation matrix expressing the orientation of Frame B with respect to Frame A
$\mathbf{T}_B^A$	Homogeneous transformation matrix expressing the pose of Frame B with respect to Frame A
$\mathbf{W}$	Task space wrench vector
$\ddot{\mathbf{q}}$	Joint acceleration vector
$\ddot{\mathbf{x}}$	Task space acceleration vector
$\dot{\mathbf{q}}$	Joint velocity vector
$\dot{\mathbf{x}}$	Task space velocity vector
$\mathbf{b}$	Constant vector of the inequality constraint
$\mathbf{c}$	Center of mass vector (in the object's frame)
$\mathbf{g}$	Gravity vector (in the world frame)
$\mathbf{p}_B^A$	Translation vector expressing the position of Frame B with respect to Frame A
$\mathbf{q}$	Joint position vector
$\mathbf{x}$	Task space pose vector
$\mathcal{F}_{ctrl}$	Control frame
$\mathcal{F}_{ref}$	Reference frame
$\mathcal{F}_w$	World frame

Introduction

The vision of the industry in all the sectors of European manufacturing has been progressively changing over the past few years. The mass production approach that was widely deployed since the early 20th century has recently shifted to a new trend that leads to the reduction of production lot sizes.

This recent trend results from the combination of multiple factors. The two principal ones are the deployment of Lean methodology (Nic18) that tends to reduce stocks as much as possible and the mass customization of products that tends to multiply the number of product variants to be produced. Unlike the mass-production paradigm where the equipment is either based on fixed automation or reconfigured over an extensive-time period, mass-customization has proven more challenging in terms of production rates and reconfiguration efficiency. One main issue is that the advantages of industrial robots are not exploited in their full potential within the production plants due to poor acceptance.

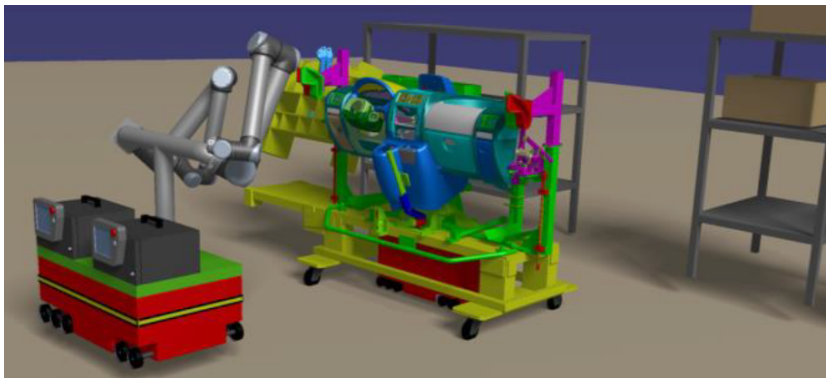


Figure 1 – Vision of the automotive case treated in the VERSATILE project. The dual-arm mobile platform operates in a changing environment.

One of the main reasons why companies, and particularly SMEs, do not adopt robots in their production lines involves the lack of flexible equipment. Additionally, complex manual tasks cannot be fully automated with a good ratio of cost vs robustness using the traditional manipulators due to three major reasons: 1) traditional manufacturing processes are designed for large scale serial production of the same product

and thus the robots are programmed for repetitive tasks (high-cost engineering and offline programming). This paradigm, however, does not apply to small scale production that may exhibit "one of a kind" product variability. 2) precise and costly jigs and tooling are required by robots while humans only require a table and their hands that are dexterous and sensitive enough to perform constrained assembly. 3) the high cost of cell maintenance and auxiliary systems for a variety of parts makes automation non-affordable/sustainable under the market uncertainty.

In terms of manufacturing processes, the transition to a small scale production implies major technological changes, such as the need for flexible equipment that can be easily adapted to perform new operations.

Motivations

The VERSATILE European project[1] in which this thesis lies has been launched to contribute to the development of the "factory of the future". Among different field of action, VERSATILE intends to increase flexibility and versatility throughout the set up of dual-arm robotic platforms in production lines, as shown in Fig. 1.

Indeed, dual-arm setups are attractive solutions to meet the growing need for versatility of production lines. Letting several manipulator arms share the same workspace and operate cooperatively brings the potential of industrial robotics at a higher level. Similarly to humans, the combination of two arms makes the realization of complex tasks possible and allows the manipulation of bigger and heavier objects than with a unique arm only.

Moreover, these platforms can be extended by connecting them to a torso with articulated joints or/and a mobile base. When the resulting system is controlled as a whole, this provides even more flexibility by increasing the dimensions of the workspace.

Finally, letting robots physically interact with humans for industrial purposes is a significant asset. Combining the strength of robots (accuracy, repeatability, efficiency) with human intelligence allows to rapidly adapt to changes in production lines.

From a control point of view, however, more versatility often means more complexity. Indeed, the resulting structure has a large number of DoF that should be properly managed to perform a desired operation in the task space. Also, dual-arm cooperation leads to some specific considerations:

1. Coordination of the arms has to be precisely handled to provide collaborative motions.

2. Closed kinematic chains are formed when manipulating a common object with the two arms. In that case, internal wrenches appear in the system and may create damages if not supervised.

[1] https://versatile-project.eu/

This thesis tackles the aforementioned challenges by proposing a complete framework for dual-arm collaborative manipulators. It relies on the online closed-loop control scheme depicted in Fig. 2.

Organization of the thesis

The document is organized as follows:

- Chapter 1 gives some background on dual-arm manipulation and task-solving methods for redundant robots. This chapter also covers the works done on human-robot physical collaboration in industry with a special focus on force control strategies and their applications to collaborative object transportation.
- Chapter 2 focuses on task space aspects of the control process. We treat representation concerns for dual-arm collaborative manipulators and show how to express both spatial and interaction data in a relevant and homogeneous manner. In particular, we include new considerations at the task level to make Physical Human-Robot Interaction (pHRI) more natural. We then detail our task space control law which uses this task formalism to provide a tunable compliant behavior combined with an obstacle avoidance strategy.
- Chapter 3 is dedicated to the joint motion control generation based on Quadratic Programming (QP) optimization of the inverse kinematics problem under joint velocity constraints. We introduce a Hierarchical Quadratic Programming (HQP) approach adapted to the cooperative dual-arm case. In case of high redundancy with respect to the tasks, we propose to exploit remaining DoF to generate parsimonious solutions, i.e. ones that reduce the number of active joints. We extend the HQP structure to integrate a second level of hierarchy, at the joint level. This allows to activate some groups of joints only when necessary, which is particularly suited for controlling a mobile base. A careful study of constraints is also given in that chapter.
- Chapter 4 exposes the complete framework organization and its software implementation. We present Robot Kinematics Control Library (RKCL), the library specially developed for the kinematic control of dual-arm robots.

Contributions

The main contributions of this work are (cf. Fig. 2):

① A robust and straightforward representation of tasks for bimanual cooperative operations, presented in Section 2.1.

② A Wrench interpretation at the cooperative task space level with extended considerations for pHRI improvement, as detailed in Section 2.2.

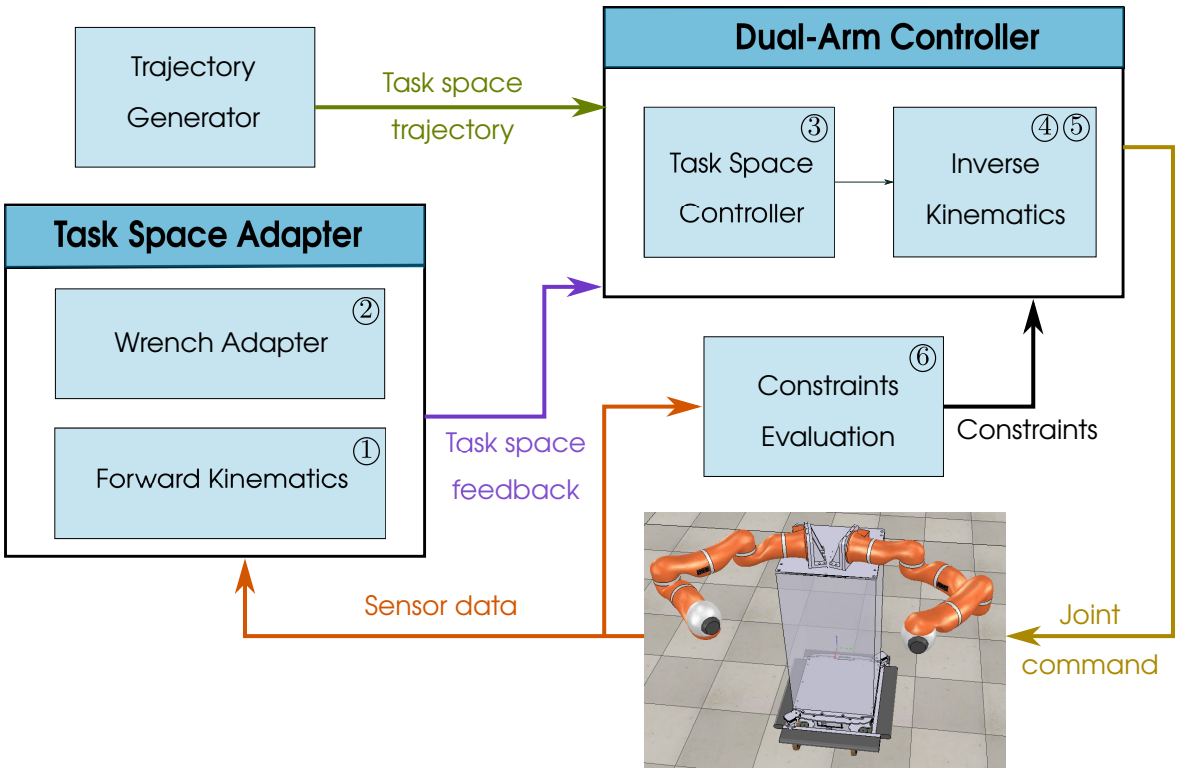


Figure 2 – High level overview of the dual-arm kinematic control scheme.

③ The development of an admittance based task space control law allowing safe dual-arm manipulation and interaction with the environment, as shown in Section 2.3.

④ An original inverse kinematics strategy based on a HQP architecture delivering a tunable parsimonious use of robot joints, as presented in Section 3.1.3.

⑤ A new method for controlling robotic extensions of the dual-arm platform (e.g mobile base) with prioritization at the joint group level to better match the component specifications. The approach is explained in Section 3.1.5.

⑥ An in-depth study of constraints to ensure adequate robot behavior. Special attention has been paid to prevent any kind of collision to occur with the generalization of the Velocity Damper method. Details are given in Section 3.2.

⑦ The open-source software implementation of the kinematic controller with all the above-mentioned features. The RKCL framework provides a generic and easy-to-use solution to control any type of dual-arm robots. We discuss the structure and give examples in Section 4.1.

Personal papers

International Conference Papers

S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Dual-arm relative tasks performance using sparse kinematic control," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2018

S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Admittance control for collaborative dual-arm manipulation," in *Int. Conf. on Advanced Robotics, ICAR*, 2019

International Journal Papers

A. Cherubini, R. Passama, B. Navarro, M. Sorour, A. Khelloufi, O. Mazhar, S. Tarbouriech, J. Zhu, O. Tempier, A. Crosnier *et al.*, "A collaborative robot for the factory of the future: Bazar," *The Int. Journal of Advanced Manufacturing Technology*, pp. 1–17, 2019

CHAPTER 1

State of the art

This thesis aims to bring new contributions to the control of dual-arm robots in an industrial context. The way we have conducted our project brings us to establish the baseline on the following areas of research: first, Section 1.1 gives some background on multi-arm manipulation with a particular focus on the dual-arm particular case. This section mainly deals with task representation and specific techniques to perform multi-arm operations. Then, in Section 1.2, we present the state of the art in task-solving strategies for redundant robots. We go from general paradigms concerning inverse kinematics and optimization processes to their application on dual-arm systems. Finally, Section 1.3 reviews the most relevant works on physical human-robot collaboration.

1.1 Multi-arm manipulation

Over the last thirty years, important research has been conducted on multi-manipulator systems (SKN+12). This type of cooperative systems, of which dual-arm is a special case, can be used to carry large or heavy loads as well as to handle objects firmly with both hands, providing more stability and accuracy during execution of the task. However, despite their undeniable usefulness, such systems are complex and require an advanced control architecture to deal with cooperative issues and internal constraint management to make adequate use of their potential.

Several approaches have been proposed to address the problem of multi-arm manipulation.

1.1.1 Augmented object and virtual-linkage

Khatib (Kha88) presented a control scheme using N robots with the same number of DoF and rigidly connected to a common manipulated object. He called the resulting

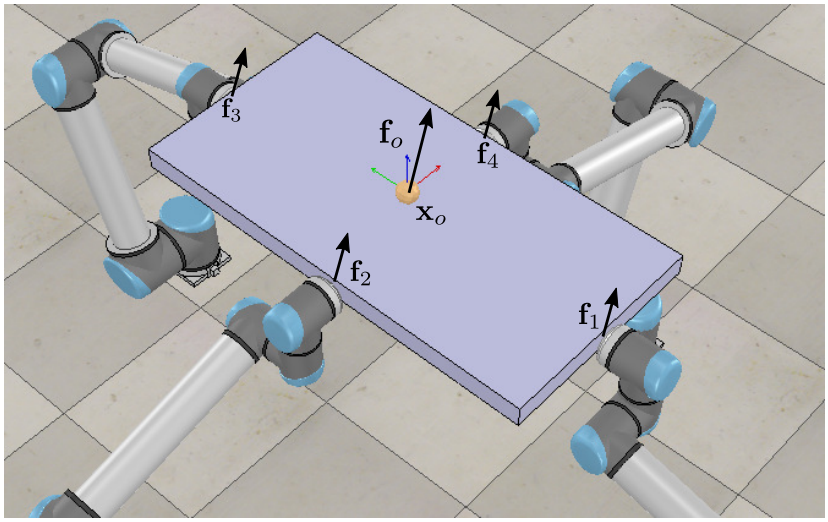


Figure 1.1 – Augmented object model. The operational force f_o at the operating point x_o is a function of the individual end-effectors' wrenches.

system (the end-effector plus the object) the augmented object, since the description of the overall system takes into account the inertial characteristics of all the effectors and the object. This augmented object is subjected to an operational force f_o at the operating point x_o, as shown in Fig. 1.1. The operational force f_o results from the contribution of all the end-effectors' wrenches.

By generalizing the operational space formulation from (Kha87) with the augmented object model, Khatib proposed a torque control approach to regulate the operational force f_o. To do so, he imposed the condition that the desired wrench f_i at the end-effector of arm i should be aligned with f_o, as depicted in Fig. 1.1. Given the targeted wrench f_i for each manipulator, the torque command can be directly deduced through the Jacobian matrix at the several end-effectors.

The main inherent weakness of this method is that internal forces (the ones that produce stress in the manipulated object) are not taken into account.

To address that problem, the principle of virtual-linkage between the grasping points was added to the previous model (WK93). Because stress throughout the object is caused by internal forces while moments imply local stress at the interaction points, virtual-linkage can be in the form of two different elements: prismatic joints to represent internal forces due to the interaction between two arms and spherical joints connecting the prismatic joints to represent the internal moments caused by each actuator (see Fig. 1.2).

However, internal forces are computed from a quasi-static analysis, meaning that object velocities and accelerations are not taken into account in this formalism. Moreover, internal moments are not used to resolve internal forces and so the approximation is only valid when internal moments are very small.

More recently, Sentis et al. (SPK10) applied the virtual-linkage model in the context of whole-body control of humanoid robots. The model provides a representation of the

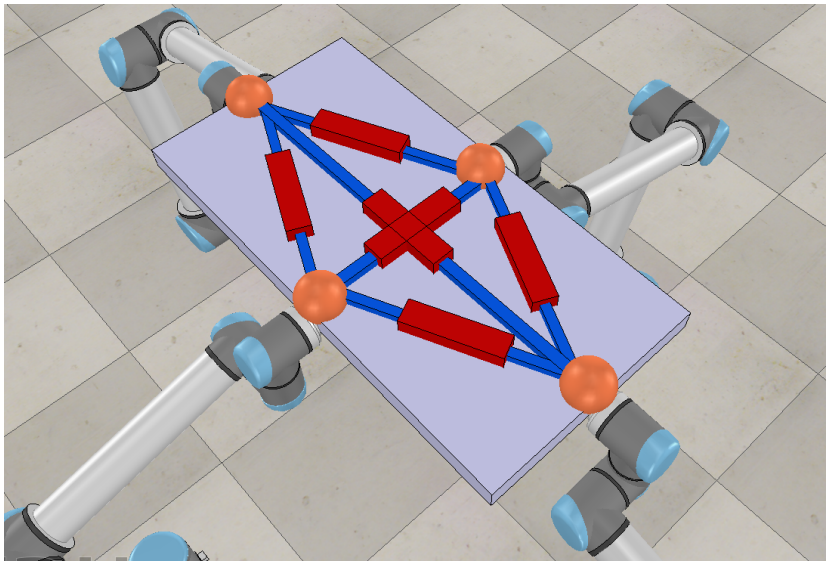


Figure 1.2 – Virtual-linkage representation: each arm is connected to the others through prismatic joints to represent internal forces and spherical joints to represent the internal moments.

internal and CoM resultant forces in multi-contact interaction tasks, which allows to explicitly control the desired contact forces through inverse dynamics using the torques acting in the nullspace of the motion.

1.1.2 Symmetric control scheme

In (UD88), Uchiyama and Dauchez introduced the concept of symmetric control scheme for dual-arm robots. To be usable, the method assumes that two arms manipulate a common tightly grasped object with a very small deformation of it. The principle is to perform a hybrid control of the robotic system using the relationship between forces and velocities applied to the object, which involves the following steps:

- Determination of internal/external forces/moments using the principle of "virtual sticks"; that is, vectors originating from the end-effector frame and ending at the origin of the reference frame $\mathcal{F}_a$ attached to the object, as shown in Fig. 1.3.
- Computation of absolute and relative velocities based on the principle of virtual work.
- Deduction of absolute and relative poses after integration.

This process allows the two robots to carry an object safely, either by grasping, pulling, or pushing it, using an absolute and relative description of the task.

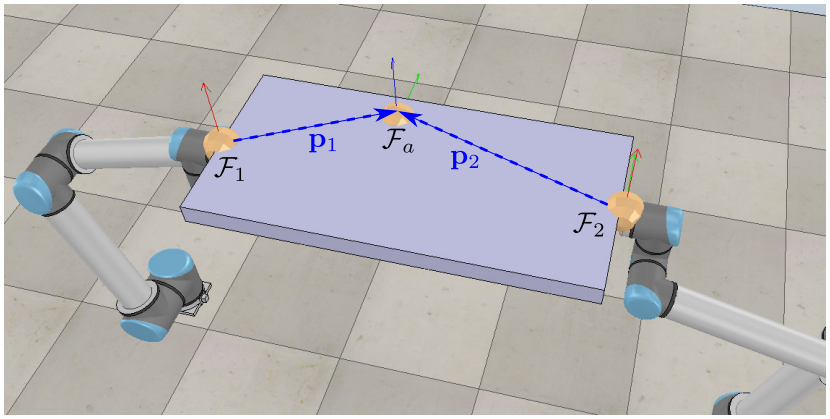


Figure 1.3 – Dual-arm manipulation using the symmetric control representation. Virtual sticks $\mathbf{p}_1$ and $\mathbf{p}_2$ join at the absolute frame $\mathcal{F}_a$, which is attached to any arbitrary location on the object.

1.1.3 The cooperative task space

Following the same perspective, Chiacchio et al. (CCS96) introduced a kinematic representation for dual-arm systems performing cooperative operations. In this approach, the two arms (referred to with subscripts 1 and 2) are seen as a unique entity and the collaborative aspect of the process is made possible through the definition of two complementary tasks, which define the cooperative task space, as depicted in Fig. 1.4:

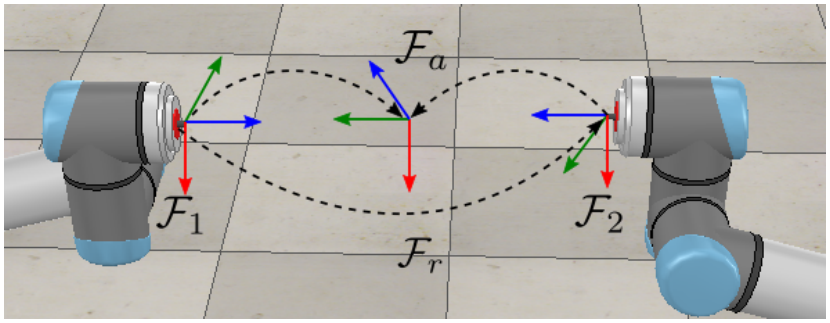


Figure 1.4 – Representation of the cooperative tasks.

- The absolute task which controls the motion of the robot in the workspace. The control frame associated with this task is defined in function of the two end-effector poses, such as:

$$\begin{aligned} \mathbf{p}_a &= \frac{\mathbf{p}_2 + \mathbf{p}_1}{2} \\ \mathbf{R}_a &= \mathbf{R}_1 \mathbf{R} \left\{ n_{12}, \frac{\phi_{12}}{2} \right\}, \end{aligned} \tag{1.1}$$

  with $\mathbf{p}$ and $\mathbf{R}$ denoting the position vector and rotation matrix, respectively. The operator $\mathbf{R}\{n, \phi\}$ generates the orientation matrix corresponding to a rotation of

an angle ϕ around the unit vector n. In Eq. (1.1), $\mathbf{R}\left\{n_{12}, \frac{\phi_{12}}{2}\right\}$ makes a rotation about the axis n_{12} (vector originating at $\mathbf{p}_1$ and directed towards $\mathbf{p}_2$) by half the angle ϕ_{12} necessary to align $\mathbf{R}_2$ with $\mathbf{R}_1$.

- The relative task which regulates the relative motion between the two end-effectors. The control frame associated with this task is attached to the end-effector of *Arm 2* and the reference frame is attached to the one of *Arm 1*. The relative task position $\mathbf{p}_r$ and orientation $\mathbf{R}_r$ are obtained through:

$$\begin{aligned} \mathbf{p}_r &= \mathbf{p}_2 - \mathbf{p}_1 \\ \mathbf{R}_r &= \mathbf{R}_2^1, \end{aligned} \tag{1.2}$$

  where $\mathbf{R}_2^1$ is the rotation matrix expressing the orientation of *Arm 2* with respect to the frame attached to *Arm 1*. Note that the choice of the reference arm is arbitrary, here *Arm 1* has been selected.

Contrary to the Symmetric control scheme, this approach only uses spatial information to characterize the task. The advantage is that we can relax the assumption of manipulating a firmly grasped object and apply the concept to any kind of dual-arm collaborative operations. Nevertheless, operational forces are not taken into account in this case.

To avoid representation singularities, the unit quaternion has been used in (CCC00) to describe orientations of the frames. In that paper, they also proposed an independent control of the manipulators to manage the cooperative task regulation. They implemented a torque control law based on kinetostatic filtering of the control action and internal force feedback. Applied to the manipulation of a firmly grasped object, the method allows to minimize the internal forces.

Intending to provide a more compact and uniform representation, Adorno et al. (AFD10) introduced the cooperative dual task-space. They use dual quaternions to simultaneously describe the position and the orientation of rigid bodies. This simplifies the task description and provide a singularity free representation for bimanual systems.

Solving the inverse kinematic problem using the cooperative task space formalism induces the definition of an absolute $\mathbf{J}_a$ and relative $\mathbf{J}_r$ tasks Jacobian that maps the joint velocities of the two robots to their motion in task-space. In the original approach (CCS96), they suggested straightforward definitions of the cooperative task Jacobian matrices given the Jacobian matrices of each manipulator ($\mathbf{J}_1$ and $\mathbf{J}_2$), leading to simple and efficient implementation:

$$\begin{aligned} \mathbf{J}_a &= \begin{bmatrix} \frac{1}{2}\mathbf{J}_1 & \frac{1}{2}\mathbf{J}_2 \end{bmatrix} \\ \mathbf{J}_r &= \begin{bmatrix} -\mathbf{J}_1 & \mathbf{J}_2 \end{bmatrix} \end{aligned} \tag{1.3}$$

Note that these simple definitions are applicable only when the cooperative tasks are expressed with respect to a fixed frame. However, to fully benefit from the relative representation, the relative task should be entirely disconnected from the environment.

This is made possible by expressing the relative task in the frame attached to the end-effector of the reference arm. In that case, the rotational velocity of the reference frame can be non-null (SS12) and the initial formulation is no longer valid. Recent work (JR15) addressed this issue. Considering a fixed world frame referred to as w, they proposed a new definition of the relative Jacobian:

$$\mathbf{J} = \begin{bmatrix} -\mathbf{\Psi}_r^1 \mathbf{\Omega}_w^1 \mathbf{J}_1^w & \mathbf{\Omega}_w^1 \mathbf{J}_2^w \end{bmatrix}, \tag{1.4}$$

with

$$\mathbf{\Psi}_r^1 = \begin{bmatrix} \mathbf{I} & -\mathbf{S}(\mathbf{p}_r^1) \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \mathbf{\Omega}_w^1 = \begin{bmatrix} \mathbf{R}_w^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_w^1 \end{bmatrix}. \tag{1.5}$$

When comparing to the original definition, we remark that only the skew-symmetric matrix $\mathbf{S}(\mathbf{p}_r^1)$ with input vector $\mathbf{p}_r^1$ has been added to the equation through the term $\mathbf{\Psi}_r^1$. Normally present in parallel mechanisms (Tsa99), the contribution of this term is negligible as long as the rotational velocity of the reference frame is close to zero, which is not necessarily the case during dual-arm operations.

1.2 Task-solving approaches for redundant robots

For any robotic system, a kinematic task consists in describing the motion of a frame of interest (usually the end-effector) attached to the robot. To realize a kinematic control, the specification of the task in the operational space should be converted into the joint space, where actuation takes place.

Controlling directly the location of the task frame requires to solve the inverse geometric model of the robot (KD04). However, this method involves complex geometric computations for high DoF robots and is not suited for multi-task considerations.

Instead, the inversion of the differential kinematics easily allows to execute a desired Cartesian motion of redundant robots (COW08).

Indeed, there exists a simple linear relation between joint velocities $\dot{\mathbf{q}}$ and task space velocities $\dot{\mathbf{x}}$ through the Jacobian matrix $\mathbf{J}$:

$$\dot{\mathbf{x}} = \mathbf{J}(q)\dot{\mathbf{q}}. \tag{1.6}$$

In the context of motion control, a robot is qualified as redundant if it has more DoF than what the task requires. In other words, redundancy arises when there exists more than one valid solution $\dot{\mathbf{q}}$, with respect to the differential kinematic equation Eq. (1.6), to fulfill a given task $\dot{\mathbf{x}}$.

1.2.1 Explicit solutions to the inverse kinematics problem

An explicit resolution of the differential kinematic equation consists in using a generalized inverse of the Jacobian matrix $\mathbf{J}$. Most often, the Moore-Penrose pseudoinverse $\mathbf{J}^+$ is utilized to solve the problem:

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{x}}. \tag{1.7}$$

The pseudoinverse provides the least square solution to Eq. (1.6). However, one may want to exploit redundancy for other purposes. In particular, redundant robots are particularly attractive for multitasking. In fact, extra actuation capabilities can be used to simultaneously complete several objectives.

There exists an analytical formulation leading to a strict hierarchy of tasks based on the projection on the null space of the Jacobian (HYN81; SS91). To do so, let us consider a set of N tasks with decreasing level of priority. The corresponding task velocity vectors to track are $\dot{\mathbf{x}}_1, ..., \dot{\mathbf{x}}_N$ and the associated Jacobian matrices are $\mathbf{J}_1, ..., \mathbf{J}_N$. The pseudoinverse resolution Eq. (1.7) can be extended to consider the ith ($i \in [2; N]$) first successive tasks in a prioritized manner, leading to the joint velocity solution $\dot{\mathbf{q}}_i$ defined by:

$$\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{i-1} + (\mathbf{J}_i\mathbf{P}_{i-1})^{+}(\dot{\mathbf{x}}_i - \mathbf{J}_i\dot{\mathbf{q}}_{i-1}), \quad \dot{\mathbf{q}}_1 = \mathbf{J}_1^{+}\dot{\mathbf{x}}_1 \tag{1.8}$$

where $\mathbf{P}_i$ is projector into the null space of the ith first tasks, such as:

$$\begin{aligned} \mathbf{J}_{1i} &= \begin{bmatrix} \mathbf{J}_1 & ... & \mathbf{J}_i \end{bmatrix}^T \\ \mathbf{P}_i &= (\mathbf{I} - \mathbf{J}_{1i}^{+}\mathbf{J}_{1i}), \end{aligned} \tag{1.9}$$

$\mathbf{I}$ being an identity matrix whose size corresponds to the number of DoF of the robot.

This method guarantees that the tasks of lower priority do not affect the performance of the highest priority task. However, there is no assurance that the robot provides sufficient redundancy to solve secondary tasks. Moreover, conflicts at the joint level are likely to occur as the task-solving approach does not take into account the position and velocity limits of the robot. In particular, the joint contributions intended for lower priority tasks may fail the execution of higher priority tasks because of exceeding the admissible solution range. Antonelli et al. (AIC09) proposed to address this problem by scaling down the tasks according to their priorities until the global solution obeys the joint constraints. In the same perspective, the principle of the Saturation in the Null Space algorithm (FDLK12) is to successively disable the use of actuators that would exceed their motion bounds and to redistribute the saturated contribution of these joints to ensure the satisfaction of all joint constraints.

In recent literature, numerous applications of null space projection for dual-arm redundancy resolution have been considered. Most of them consist in improving some arbitrary criteria while solving the relative task only, since relaxing the absolute part of the cooperative task representation leaves room for optimization. In (HHY15), the authors extended the Saturation in the Null Space algorithm to handle joint constraints in the case of dual-arm robots performing relative tasks. The framework is also able to deal with multi-tasking conflicts both in task and joint spaces by keeping the same strategy of null space distribution of exceeded joint velocity, which allows sacrificing

secondary tasks when necessary. Based on a similar hierarchical task-solving method, Faroni et al. (FBVT16) chose to exploit redundancy to increase manipulability while taking into account the mechanical boundaries of the joints. An interesting aspect of that work is that, unlike common approaches, they do not exclusively consider the local configurations of manipulators to evaluate their criteria. Instead, they apply an iterative optimization method to assess manipulability for remaining configurations as well and select the best solution from a global perspective.

Other research has addressed the problem of redundancy resolution for the full cooperative task. In (FLMO16), Freddi et al. studied redundancy of cooperative manipulators and also proposed the Jacobian null space method to solve several prioritized tasks. They provided a case study in which two planar manipulators are mounted on a mobile base. Using this technique, the platform can successfully move an object in the space while keeping the relative pose constant and avoiding obstacles. Ortenzi et al. (OMF+18) used the null space projection to establish a strict priority between three tasks: the relative part of the cooperative tasks is solved with maximum priority while the absolute part is projected in its null space. Finally, if the robot is sufficiently redundant, a lower priority task is processed to avoid joint limits. Experiments with varying number of DoF for the manipulators demonstrated the proper behavior of the hierarchical approach.

1.2.2 Numerical optimization approaches

The main drawback of the analytical solutions presented below is that hard constraints (e.g. physical limits of the robot) cannot be explicitly handled. Indeed, even if some cost functions aim at preventing from exiting the admissible space, there is no guarantee that the resulting solution will effectively satisfy the constraints.

To overcome this issue, numerical solvers have been widely adopted in the literature to solve the inverse kinematics problem. A common approach is to solve a QP with inequality constraints, as proposed in (DSBDS09). It allows to minimize the l_2-norm of a cost function with the possibility to define a set of constraints to be satisfied at any time. Solving the inverse kinematic problem consists in minimizing the least square error on the task tracking, which can be formulated formulated as follows:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \quad \mathbf{C}\dot{\mathbf{q}} = \mathbf{d} \end{aligned} \tag{1.10}$$

where $\mathbf{A}$, $\mathbf{C}$ are the linear coefficients matrices and $\mathbf{b}$, $\mathbf{d}$ the constant vectors in the equality and inequality constraints.

Based on QP solvers, it is possible to solve a sequence of prioritized tasks using a Quadratic Programming (QP) architecture (KLW+09; KLW11). As before, let us consider a set of N tasks sorted by decreasing priority level. The iterative process to obtain the final joint velocity vector solution of the HQP problem is given in Algorithm (1), where input data $\mathbf{J}$ and $\dot{\mathbf{x}}$ are containers vectors of Jacobian matrices and task space

command vectors, respectively, such as $\mathbf{J} = \begin{bmatrix}\mathbf{J}_1 & \dots & \mathbf{J}_N\end{bmatrix}^T$ and $\dot{\mathbf{x}} = \begin{bmatrix}\dot{\mathbf{x}}_1 & \dots & \dot{\mathbf{x}}_N\end{bmatrix}^T$. The basic idea of this method is to iteratively augment the equality constraint $\overline{\mathbf{C}}\dot{\mathbf{q}} = \overline{\mathbf{d}}$ with the latest optimal solution in the task space. Let us suppose that we just solved the ith task and $\dot{\mathbf{q}}_i$ is the solution vector, we augment the equality constraint with $\mathbf{J}_i\dot{\mathbf{q}} = \mathbf{J}_i\dot{\mathbf{q}}_i$ to enforce the solution of task $i+1$ to be in the null space of the ith first tasks. The approach is equivalent to the analytical formulation of strictly hierarchized tasks presented previously, but with hard constraints specification.

Algorithm 1 Hierarchical Inverse Kinematics

Input: $\mathbf{J}, \dot{\mathbf{x}}, \mathbf{A}, \mathbf{b}, \mathbf{C}, \mathbf{d}$
Output: $\dot{\mathbf{q}}$
  procedure HQP
    $\overline{\mathbf{C}} \leftarrow \mathbf{C}, \quad \overline{\mathbf{d}} \leftarrow \mathbf{d}$
    for $i \leftarrow 1 : N$ **do**
      $\dot{\mathbf{q}} \leftarrow \mathrm{QP}(\mathbf{J}_i, \dot{\mathbf{x}}_i, \mathbf{A}, \mathbf{b}, \overline{\mathbf{C}}, \overline{\mathbf{d}})$ ▷ Eq. (1.10)
      $\overline{\mathbf{C}} \leftarrow \begin{bmatrix}\overline{\mathbf{C}} & \mathbf{J}_i\end{bmatrix}^T$
      $\overline{\mathbf{d}} \leftarrow \begin{bmatrix}\overline{\mathbf{d}} & \mathbf{J}_i\dot{\mathbf{q}}\end{bmatrix}^T$
    end for
  end procedure

in (EMW10), Escande et al. improved the computation time of the solver using a complete orthogonal decomposition to obtain the null spaces of tasks with a prioritized hierarchy. This improvement allowed a satisfactory real-time implementation of the method on humanoid robots (EMW).

Multi-tasking problems can also be tackled with weighting strategies. The idea here is to assign a weighting factor to each task and combine them to get a unique cost function. By solving the corresponding optimization problem, the process is able to find a solution which is a compromise between the different tasks. In this case, the tasks of lower priority does no longer belong to the null space of higher priority tasks, meaning that they affect their performance. In particular, weighted approaches based on QP have been proposed in the literature (SPB11; BK11). If we consider the same set of N tasks introduced for the HQP problem, the solution of the weighted QP is obtained by solving the following problem:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \sum_{i=1}^{N} w_i \left\|\mathbf{J}_i\dot{\mathbf{q}} - \dot{\mathbf{x}}_i\right\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \quad \mathbf{C}\dot{\mathbf{q}} = \mathbf{d} \end{aligned} \tag{1.11}$$

where w_i is the weighting factor associated to task i.

The advantages of such methods are that they are easy to implement and require to solve only one optimization problem which is more computationally efficient than the HQP approach. However, it is difficult to anticipate the behavior of a robot which

adopts this strategy, especially when there are more than two tasks to handle at the same time.

In recent years, several works have tackled the inverse kinematics problem of dual-arm robots using numerical solvers. A Constraint-based Programming method (ZM07; DSBDS09), combining the concepts of the additional tasks (Ser89), the user-defined objective functions (PA93), and the sub-tasks (TBBD09), has been used to control dual-arm motion during bi-manual cooperative operations (ÖSKK12). Rather than searching for the optimal solution, as done in previous works, the authors proposed to find a feasible *good enough* solution by reformulating the optimization problem. Instead of using equality constraints to project a secondary task in the null space of higher priority tasks, that reduce the dimension of the feasible subset, they expressed additional tasks in terms of inequalities. This allows to increases the number of tasks simultaneously manageable, since inequality constraints do not reduce the dimensionality of the feasible set in joint space. Wang et al. (WVK+14) extended that work by including time-dependent equality constraints in a compact and uniform way. In particular, this makes it possible to track timed trajectories or specify force feedback constraints.

A HQP framework for dual-arm robot performing relative tasks has been newly proposed (SD18). In the developed framework, several criteria are optimized in a prioritized order, while ensuring the satisfaction of hard constraints all the time. However, they do not explicitly explain how the HQP is set up in terms of cost functions and strict constraints, making it difficult to evaluate.

1.3 Human-robot physical collaboration in industry

Enabling pHRI has tremendous potential and is particularly attractive for industrial purposes (CPC+16). Combining the strength of robots (accuracy, repeatability, efficiency) with human intelligence allows to rapidly adapt to changes in production lines. However, requirements for interacting and collaborating with humans in industrial setups are not well formalized and are still a hot topic of research. In particular, safety concerns have to be clearly established to allow physical contacts between humans and robots. With a view towards standardization, the International Organization for Standardization published the ISO10218 standard (ISO11), which specifies velocity, power and force constraints for any manipulator arm working in presence of a human. Recently, the ISO15066 standard (15016) was defined to provide guidelines for the design and organization of a cooperative workspace with the aim of reducing the risks to which people may be exposed. Control of industrial robots for interacting with humans have to deal with special considerations, as we will see in what follows.

1.3.1 Robot force control for physical collaboration

During physical interaction, haptic data (force/torque) are valuable quantities to consider for the motion control of robots. Hence, a lot of concepts have derived from force control paradigms in robotics. In particular, impedance control (Hog84) has been largely adopted in the context of human-robot collaboration (II95; IMM02). Many works have included human parameters in the design of their controllers. In (DG07), the authors define a variable damping controller which uses the derivative of the interaction force as a natural sensor of human intention. An admittance control strategy with estimation of the intended human motion has been proposed in (CAB+07). Thanks to that, the level of assistance is adjusted to match the needs of the human collaborator. Some of the approaches are model-based. For instance, in (GKB11), Gribovskaya et al. combined machine learning techniques with an adaptive impedance control framework to model an interactive task.

Recently, the cooperative task space representation has been used to perform physical human-robot collaboration. In (BNU+17), Compliant Movement primitives (DGUP15) are extended to bimanual cooperative tasks. In this context, a torque controller adopts a stiff behavior on the relative task while more compliance is given to the absolute task. In (NLGU16; NLGU18), a cooperative control scheme based on an impedance law allows performing kinesthetic guiding operations. Adaptation of the stiffness along the trajectory provides more accuracy to the human co-worker during critical parts of the task.

1.3.2 Collaborative object transportation

Within the numerous applications engaging pHRI, collaborative carrying has raised great interest for its ability to facilitate the transportation of bulky objects. Many researchers have addressed the control problem of a robot system handling an object in cooperation with a human. Early work involving human-robot object comanipulation was done by Kosuge et al. (KYT+94). They studied the general case of several humans collaborating with a multi-arm robotic system. Assuming a rigid grasp and no relative motion between the arms, the mechanical impedance of the commonly held object is controlled and the necessary impedance for each robotic arm is deduced.

The need for robotic assistance is particularly desirable for moving cumbersome objects. In (TAHT02), the author proposes a method to facilitate the cooperative manipulation of long parts. By setting a virtual nonholonomic constraint at the tip of the robotic arm, the object's motion is restricted to a plane to prevent sideslip. Combining horizontal and vertical movements enables the 6 DoF manipulation of the object in 3D space. The load sharing during comanipulation is also an important aspect and effort sharing strategies have been proposed to improve the task performance (LMH10).

To fully benefit from such cooperative tasks, the mobility of robots is an essential asset. Initial work in this field involved wheeled mobile manipulators. In (KSK00), the

dual-arm mobile robot "MR helper" is able to collaborate with humans using wrench feedback from each arm's wrist. The apparent impedance of the object is estimated to generate the command in the task space. The mobile base and the manipulators are also connected through an impedance system, ensuring automatic motion coordination. More recent work on multi-robot cooperative transportation has been proposed in (ESH13). They presented an impedance-based control architecture that efficiently manages the coordination of the mobile manipulators by compensating kinematic errors. A decoupled control of the mobile base and manipulators aims at reducing the impact of disturbances generated by the mobile platforms on the end-effectors. In (WSKÖ15), a kinematic control strategy for serial-to-parallel structure is given. The concept of Virtual Kinematic Chains is used to specify the common motion of the parallel manipulators, instead of using the two manipulators kinematics. Other research activities on collaborative carrying have focused on human-humanoid interactions (BKCK12; ACB+14). In this case, Mobile manipulators use legs instead of wheels for locomotion, making the control more complex. Although the locomotion mode is different, the same issue arises in motion coordination between the arms and the mobile base.

1.4 Conclusion

This section gave an overview of the state of the art in three robotic areas which are at the core of this thesis: first, we presented the main works done on multi-arm manipulation and specifically on kinematic representation paradigms for bimanual coordinated operations. Although the notion of cooperative task space has been widely adopted in the context of dual-arm manipulation, we will see in the next chapter that our extension improves the original version in terms of robustness and convenience. Then, we introduced the existing task-solving approaches for redundant robots and their applications to the inverse kinematic control of dual-arm manipulators. We showed how QP optimization allows to solve kinematic tasks subject to constraints and how it can be extended to a stack of QP problems to solve several tasks simultaneously and in a prioritized manner. In this thesis, we base our task-solving process on HQP adapting it to the dual-arm case. Relying on this method, we propose several strategies to enhance the behavior of dual-arm platforms in industrial setups, including parsimonious joint control and management of mobile capabilities. Finally, we discussed previous works related to pHRI in industry. We mentioned various contributions in force control for pHRI and focused on collaborative object transportation. Our work goes beyond the state of the art by proposing an admittance control law for dual-arm cooperative operations with additional human considerations which improve the performances of collaborative object transportation tasks.

The next chapter is dedicated to dual-arm control in the task space. After introducing our strategy to represent geometric and force information in a unified cooperative space, we will propose some methods to better interpret force interactions arising during

pHRI. Then, we will detail the implementation of our closed-loop admittance controller intended for human-robot collaboration.

CHAPTER 2

Dual-arm task space control

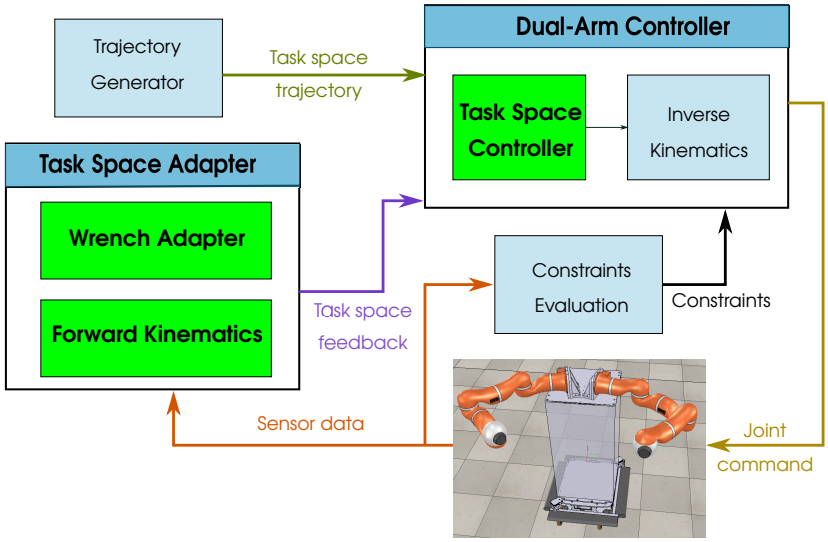


Figure 2.1 – Components of the dual-arm kinematic control scheme tackled in this chapter.

This chapter tackles the collaborative control of dual-arm robots at task level. To be usable in a control process, all the sensory feedback should be expressed in a common space. This is the role of the *Task Space Adapter* block from Fig. 2.1: spatial data are treated by the *Forward Kinematics* block while the *Wrench adapter* block is in charge of converting efforts measured at the tip of each arm.

Once processed, the task space feedback is transmitted to the *Task Space Controller* which combines this information with the task space trajectory to generate the task command. The aim of this block is to adequately manage the input to deliver a command which allows to safely manipulate objects and adequately react to interactions with the environment.

2.1 Task representation for dual-arm robots

The task description in a kinematic controller consists in using spatial information about a frame of interest to regulate its motion through the actuation of the robot joints. To characterize a task, the following two elements should be specified (see Fig. 2.3):

- A control frame ($\mathcal{F}_{ctrl}$): the frame in space whose motion has to be regulated.
- A reference frame ($\mathcal{F}_{ref}$): the frame with respect to which the control frame is moved.

A task can thus be represented by a homogeneous transformation matrix $\mathbf{T}_{ctrl}^{ref}$ expressing the pose of the control frame with respect to the reference frame.

As seen in the previous chapter, defining a task for dual-arm cooperative robots requires specific considerations. In the next section, we will present our extended version of the Cooperative task representation.

For a single manipulator arm, the control frame is generally attached to the end-effector. However, for dual-arm coordinated tasks, independent control of the arms is not an appropriate solution. In Chapter 1, we introduced the cooperative task-space representation (CCS96) which has been commonly adopted when dealing with dual-arm robots. Indeed, this paradigm allows defining bimanual operations by way of meaningful variables divided into absolute and relative components.

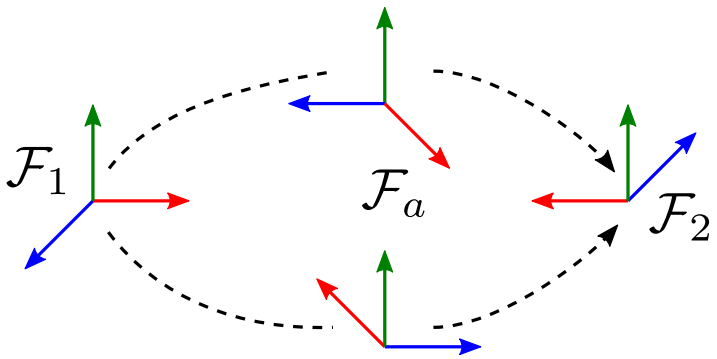
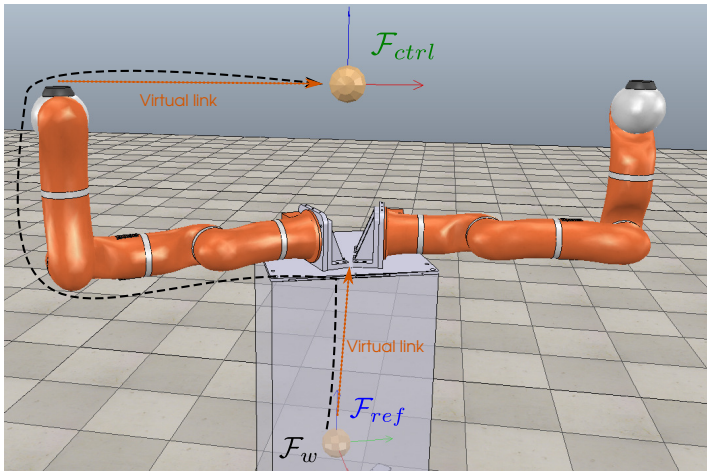
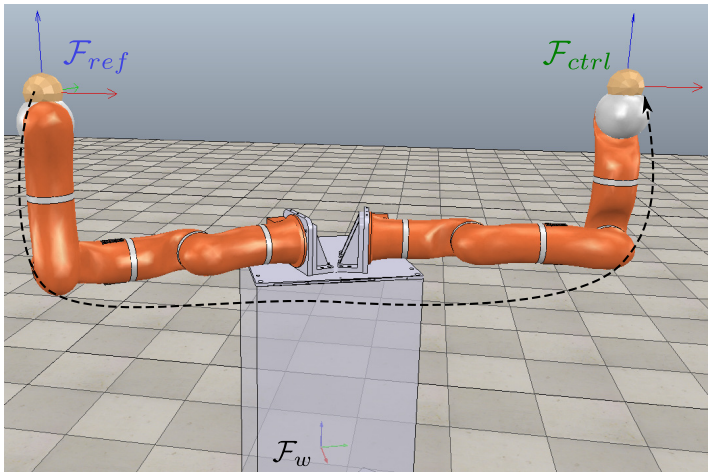


Figure 2.2 – Discontinuity of the absolute task orientation. In the original version, there exist two solutions for the orientation of $\mathcal{F}_a$ since it is obtained after applying half the rotation needed to align $\mathcal{F}_1$ with $\mathcal{F}_2$.

Keeping the same idea, we decided to modify the absolute task definition from (CCS96) given in Eq. (1.1) for two reasons. First, in the original version, the absolute frame has a discontinuous orientation since it is defined in function of the orientation of the two end-effectors. Indeed, as depicted in Fig. 2.2, there exist two distinct solutions for the orientation part $\mathbf{R}_a$ of the absolute task pose depending on the direction of rotation used to go from $\mathcal{F}_1$ to $\mathcal{F}_2$. No matter which convention we use, the computation of $\mathbf{R}_a$ may switch from one solution to the other leading to instabilities in the control



(a) The absolute task



(b) The relative task

Figure 2.3 – Cooperative task representation. (a) The absolute task expresses the pose of any arbitrary frame in the space ($\mathcal{F}_{ctrl}$) with respect to a fixed world frame ($\mathcal{F}_{ref}$). The kinematic chain associated with this task uses only one arm and virtual links for joining the frame of interest: one virtual link to attach the robot reference frame with the world frame and another one to consider the control frame as an extension of one arm. (b) The relative task expresses the pose of one end-effector ($\mathcal{F}_{ctrl}$) with respect to the other one ($\mathcal{F}_{ref}$).

process. Second, this approach is not straightforward when it comes to controlling any arbitrary frame of the workspace. For instance, if one wants to rotate a manipulated object around a specific frame, it is more convenient to attach the absolute frame to it.

Thus, we chose to define the absolute frame with respect to only one arm by creating a virtual link between its end-effector and a point of interest (see Fig. 2.3a). It is not necessary to explicitly consider the two arms in the absolute task definition because the relative task automatically handles the motion of the other manipulator. Assuming

that *Arm 1* is the reference arm, Eq. (1.1) becomes:

$$\begin{aligned} \mathbf{p}_a &= \mathbf{p}_1^w + \mathbf{R}_1^w \mathbf{p}_a^1 \\ \mathbf{R}_a &= \mathbf{R}_1^w \mathbf{R}_a^1, \end{aligned} \tag{2.1}$$

where $\mathbf{p}_a^1$, $\mathbf{R}_a^1$ express the pose of the absolute frame with respect to the end-effector of *Arm 1*, obtained through the virtual link. Since the orientation part is now expressed as a function of the orientation of one arm only, there is no more discontinuity problem.

The relative and absolute task are depicted in Fig. 2.3.

Throughout the rest of the document, we assume the following:

- The absolute task expresses the end-effector pose of *Arm 1* with respect to some fixed world frame $\mathcal{F}_w$ (superscript w). The corresponding homogeneous transformation matrix is $\mathbf{T}_a = \mathbf{T}_1^w$.
- *Arm 1* is taken as reference in the relative task definition, leading to $\mathbf{T}_r = \mathbf{T}_2^1$.

2.2 Wrench feedback for safe and collaborative manipulation

Dual-arm mobile robots are particularly suited for transporting large and heavy objects. The versatility of such platforms can be better exploited by allowing human-robot cooperation. Notably, co-manipulation of a shared object is an interesting type of interaction in an industrial context. This can be used to facilitate the displacement of an object, by letting the human physically guide the robot towards the target location without lifting the weight of the object. Also, it can be applied to perform "teaching by demonstration" operations. This allows to quickly reconfigure the task of the robot without programming.

In an open-loop system, the task space command would be directly issued from the trajectory generator and converted into joint commands without having to worry about the evolution of the robot and the environment. However, in an unstructured workspace in which physical interaction with human operators may occur, it is necessary to close the feedback loop. In this regard, the proposed dual-arm strategy uses wrench feedback measurements at the wrist of each arm to manage internal constraints and perceive external forces.

To properly exploit the wrench information, it has to be meaningful in the task space. This conversion is made by the *Wrench Adapter* block depicted in Fig. 2. To improve the quality of the interaction during human-robot collaborative carrying, this block implements sequential processes to generate the cooperative task wrenches, as shown in Fig. 2.4.

First, wrench feedback coming from sensors is interpreted at the task frames. Then, the object weight is removed from the external wrench, as this should not be consid-

ered when interacting with the human. By knowing the interaction location, external moments are finally adapted to follow the direction of the wrench exerted by the human.

Further, by performing damping control on selected absolute task variables (see Section 2.3.1), the robot will follow the motion driven by the human operator interacting with it.

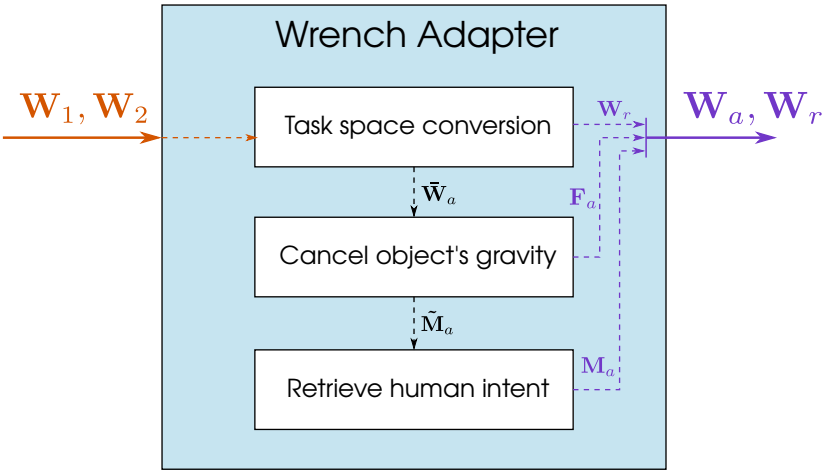


Figure 2.4 – The Wrench Adapter block is in charge of computing the cooperative task wrenches $\mathbf{W}_a$, $\mathbf{W}_r$ from wrench measurements at the end-effectors $\mathbf{W}_1$, $\mathbf{W}_2$. After conversion in the task space, sequential operations are performed to first remove object weight from the absolute task wrench $\bar{\mathbf{W}}_a$ and then retrieve human-robot interaction wrench from the absolute task moment $\widehat{\mathbf{M}}_a$.

2.2.1 Wrench in the task space

Let us assume that an object is grasped by the two end-effectors, as depicted in Fig. 2.5. Two kinds of efforts apply to the object: the closed kinematic chain causes internal stress $\mathbf{W}_{int}$ while external wrenches $\mathbf{W}_{ext}$ arise from environmental interactions (e.g. gravitational forces, interactions with humans, ...). The combination of all these actions is perceived at the arms' wrists through forces $\mathbf{F}_i^i$ and moments $\mathbf{M}_i^i$ $(i = 1, 2)$[1] and gathered in the wrench vectors $\mathbf{W}_1^1$ and $\mathbf{W}_2^2$ that we write for simplicity $\mathbf{W}_1 = \begin{bmatrix}\mathbf{F}_1 & \mathbf{M}_1\end{bmatrix}^T$ and $\mathbf{W}_2 = \begin{bmatrix}\mathbf{F}_2 & \mathbf{M}_2\end{bmatrix}^T$.

Let $\mathcal{F}_o$ denotes a frame attached to the object where $\mathbf{x_o}$ and $\boldsymbol{\upsilon_o}$ are respectively the pose and twist of $\mathcal{F}_o$ with respect to a fixed world frame $\mathcal{F}_w$. The dynamic equation of the object can be written as follows:

$$\mathbf{M}_o(\mathbf{x_o})\dot{\boldsymbol{\upsilon}}_\mathbf{o} + \mathbf{C}_o(\mathbf{x_o}, \boldsymbol{\upsilon_o}) + \mathbf{G}_o(\mathbf{x_o})\boldsymbol{\eta} = \mathbf{W}_{o,ext}, \tag{2.2}$$

where $\mathbf{M}_o(\mathbf{x_o})$ is the object inertial matrix, $\mathbf{C}_o(\mathbf{x_o}, \boldsymbol{\upsilon_o})$ is the vector of generalized centripetal, Coriolis, and gravity forces, $\mathbf{W}_{o,ext}$ is the resultant of wrenches exerted by external sources on the object and perceived at $\mathcal{F}_o$, $\mathbf{G}_o(\mathbf{x_o}) = \begin{bmatrix}\mathbf{G}_{o1}(\mathbf{x_o}) & \mathbf{G}_{o2}(\mathbf{x_o})\end{bmatrix}$ is

[1] Superscript i refers to the frame with respect to which forces and moments are expressed.

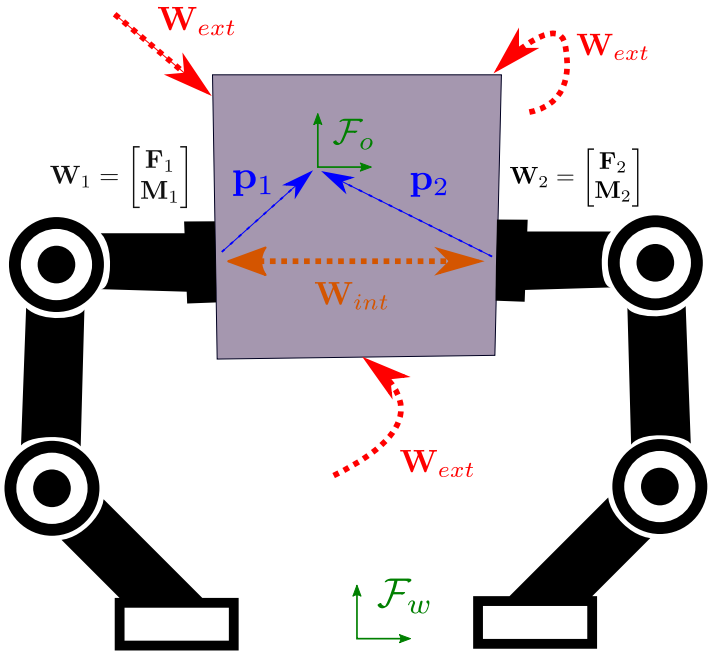


Figure 2.5 – Wrench representation during dual-arm manipulation of a rigid object. Wrenches $\mathbf{W}_1$ and $\mathbf{W}_2$ are measured at the wrist of each arm. They result from external wrenches $\mathbf{W}_{ext}$ exerted on the object and from the internal wrench $\mathbf{W}_{int}$ induced by the tight grasp.

the Grasp matrix (PT16) where $\mathbf{G}_{oi}(\mathbf{x_o})$ $(i \in \{1, 2\})$ is a transformation matrix which maps the velocities of the object onto the velocities of the corresponding end-effector given as

$$\mathbf{G}_{oi}(\mathbf{x_o}) = \mathbf{H}_{oi}\tilde{\mathbf{G}}_{oi}(\mathbf{x_o}). \tag{2.3}$$

The matrix $\mathbf{H}_{oi}$ is used to provide a contact model between the object and the end-effector i. In our study, we consider a rigid grasp of the object, meaning that all the translational and rotational velocity components of the contact point are transmitted through the contact. In this case, $\mathbf{H}_{oi} = \mathbf{I}_{6\times6}$.

The partial grasp matrix $\tilde{\mathbf{G}}_{oi}(\mathbf{x_o})$ is given by:

$$\tilde{\mathbf{G}}_{oi}(\mathbf{x_o}) = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{S}(\mathbf{p}_i^w) & \mathbf{I}_{3\times3}\end{bmatrix}, \tag{2.4}$$

where $\mathbf{S}(\mathbf{p}_i^w)$ is the skew-symmetric matrix with input vector $\mathbf{p}_i^w$ expressing the translation in the world frame to reach $\mathcal{F}_o$ from the end-effector i.

In Eq. (2.2), $\boldsymbol{\eta} = \begin{bmatrix}\boldsymbol{\eta}_1^T & \boldsymbol{\eta}_2^T\end{bmatrix}^T$ with $\boldsymbol{\eta}_i^T$ $(i \in \{1, 2\})$ the vector of contact force and moment components transmitted through contact i. Again, assuming that the object is rigidly maintained by the arms, all wrench components are transmitted through the contact, resulting in:

$$\boldsymbol{\eta}_i = \begin{bmatrix}\mathbf{F}_i\\ \mathbf{M}_i\end{bmatrix}. \tag{2.5}$$

In the context of physical-human robot interaction where the robot has to operate at low speed, the inertia terms are negligible ans the system can be considered as quasistatic. With this assumption, Eq. (2.2) is reduced to:

$$\mathbf{W}_{o,ext} = \mathbf{G}_o(\mathbf{x_o})\boldsymbol{\eta}, \tag{2.6}$$

Using Eq. (2.3), Eq. (2.4) and Eq. (2.5), the dynamic equation leads to:

$$\begin{bmatrix} \mathbf{F}_{o,ext} \\ \mathbf{M}_{o,ext} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{S}(\mathbf{p}_1^w) & \mathbf{I}_{3\times3} & \mathbf{S}(\mathbf{p}_2^w) & \mathbf{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{M}_1 \\ \mathbf{F}_2 \\ \mathbf{M}_2 \end{bmatrix}, \tag{2.7}$$

This equation allows to relate the resultant of external wrenches perceived at $\mathcal{F}_o$ with the wrenches measured at each end-effector. By definition, we can directly associate external wrenches with the absolute task. Hence, one can retrieve the resultant forces $\mathbf{F}_a$ and moments $\mathbf{M}_a$ applied at the absolute task frame $\mathcal{F}_a$ (and expressed in the frame $\mathcal{F}_w$):

$$\begin{cases} \mathbf{F}_a = \mathbf{F}_1^w + \mathbf{F}_2^w, \\ \mathbf{M}_a = \mathbf{M}_1^w + \mathbf{p}_1^w \times \mathbf{F}_1^w + \mathbf{M}_2^w + \mathbf{p}_2^w \times \mathbf{F}_2^w, \end{cases} \tag{2.8}$$

which is equivalent to the formulation introduced in (UD88), where $\mathbf{p}_1^w$, $\mathbf{p}_2^w$, are vectors representing the *virtual sticks*.

By taking a closer look to Eq. (2.6), we notice that the dimension of $\mathbf{W}_{o,ext}$ is 6 while the dimension of $\boldsymbol{\eta}$ is twelve. this means that the grasping system is undetermined (PT16). Let $N(\mathbf{G}_o)$ denote the null space projection of the grasping matrix, then from Eq. (2.6) one can write:

$$\boldsymbol{\eta} = \mathbf{G}_o(\mathbf{x_o})^+\mathbf{W}_{o,ext} + N(\mathbf{G}_o)\mathbf{W}_{int} \tag{2.9}$$

Wrenches $\mathbf{W}_{int}$ are referred to as *internal object forces*. These wrenches are internal because they do not contribute to the motion of the object, i.e. $\mathbf{G}_o(\mathbf{x_o})\mathbf{W}_{int} = 0$. By definition, internal wrenches can be directly associated with the relative task. There exists an infinite number of combinations of $\mathbf{W}_1$ and $\mathbf{W}_2$ that satisfy this condition. As in (UD88), we define the internal wrench vector $\mathbf{W}_r = \begin{bmatrix} \mathbf{F}_r & \mathbf{M}_r \end{bmatrix}^T$ (expressed in the end-effector frame of *Arm 1*):

$$\begin{cases} \mathbf{F}_r = \dfrac{1}{2}(\mathbf{F}_2^1 - \mathbf{F}_1^1), \\ \mathbf{M}_r = \dfrac{1}{2}(\mathbf{M}_2^1 - \mathbf{M}_1^1). \end{cases} \tag{2.10}$$

The internal wrenches are neither affected by the object's gravity nor by the interaction with the human. Thus, the vector $\bar{\mathbf{W}}_r$ computed at this point corresponds to the final internal wrenches, such as $\bar{\mathbf{W}}_r = \mathbf{W}_r$.

2.2.2 Identification and cancellation of the objects' gravity effects

When performing physical interaction with kinesthetic guidance through a commonly held object, its mass permanently applies external wrenches that should not be interpreted as a human action.

To overcome this undesired behavior, we propose a practical method to estimate and then cancel the effects of the payload.

Similarly to Eq. (2.8), the perceived external wrench in the object's frame $\mathcal{F}_o$ can be expressed with:

$$\mathbf{W}_o^w = \begin{bmatrix} \mathbf{F}_1^w + \mathbf{F}_2^w \\ \mathbf{M}_1^w + \mathbf{M}_2^w + \mathbf{F}_1^w \times \mathbf{p}_1^w + \mathbf{F}_2^w \times \mathbf{p}_2^w \end{bmatrix} \tag{2.11}$$

Where $\mathbf{p}_1^w$ and $\mathbf{p}_2^w$ denote the translation vectors going from each end-effector to the object's frame, expressed in the world's frame.

If we consider the manipulated object as a point mass under the sole influence of gravity (interaction being done at low speed/acceleration), this wrench can also be estimated as:

$$\widehat{\mathbf{W}}_o^w = \begin{bmatrix} m\mathbf{g} \\ \mathbf{R}_o^w \mathbf{c} \times m\mathbf{g} \end{bmatrix} \tag{2.12}$$

where m is the mass of the object, $\mathbf{c}$ the CoM, expressed in the object's frame, and $\mathbf{g}$ the gravity vector in the world frame.

When considering practical object manipulation, m and $\mathbf{c}$ are generally unknown or not well known. To estimate their parameters, one can collect data while manipulating the object and use them in a non-linear optimization problem. The problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \left\| \mathbf{W}_o^w - \widehat{\mathbf{W}}_o^w \right\|_2 \\ \text{s.t.} \quad & m > 0, \end{aligned} \tag{2.13}$$

where $\mathbf{z}$ is the vector of optimization variables:

$$\mathbf{z} = \begin{bmatrix} m \\ \mathbf{c} \end{bmatrix}. \tag{2.14}$$

Once $\mathbf{z}$ that satisfies Eq. (2.13) has been found, it can be used to cancel objects' gravity effects. This operation is performed by the *Cancel object's gravity* block from Fig. 2.4.

From the cooperative task perspective, the gravity effects of the manipulated object do not influence internal wrenches. They add however an additional component to the external wrenches, which should be subtracted to avoid interfering with the interaction.

Based on Eq. (2.8) and taking into account the objects' gravity effects, we can compute the external wrench vector $\tilde{\mathbf{W}}_a = \begin{bmatrix}\tilde{\mathbf{F}}_a & \tilde{\mathbf{M}}_a\end{bmatrix}^T$ from which objects' gravity effects have been removed:

$$\begin{cases} \tilde{\mathbf{F}}_a = \mathbf{F}_a - m\mathbf{g}, \\ \tilde{\mathbf{M}}_a = \mathbf{M}_a - \mathbf{R}_o^w \mathbf{c} \times m\mathbf{g}, \end{cases} \tag{2.15}$$

In Eq. (2.15), $\mathbf{R}_o^w$ is the rotation matrix expressing the center of mass $\mathbf{c}$ in the world frame.

The external force is not influenced by the location of the interaction during human-robot co-manipulation. The vector $\tilde{\mathbf{F}}_a$ computed at this point corresponds to the final external force, so that $\tilde{\mathbf{F}}_a = \mathbf{F}_a$.

2.2.3 Retrieve human interaction wrench

During human-robot co-manipulation, the location on the object where the human exerts a force (called interaction point) has an incidence on what is perceived by the sensors. Indeed, considering the principle of levers, a linear force applied at some point of the object may create a moment at the grasping points. The bigger the object is, the more the perception is disturbed.

In (DGB+12), Dumora et al. proposed a statistical model to identify the human intention during shared human-robot collaborative task, based on haptic measures only.

In this work, we assume that the interaction point is known (e.g. using visual monitoring). Considering that the weight of the object has been priorly canceled, it is possible to figure out the effort exerted by the human by removing undesired components of the external moment vector $\tilde{\mathbf{M}}_a$ coming from the lever action. In contrast with (DGB+12), this methods gives an accurate result.

Reusing the principle of *virtual sticks*, one can easily retrieve the external moments $\mathbf{M}_a$ applied at the interaction point from the following equation:

$$\mathbf{M}_a = \tilde{\mathbf{M}}_a + \mathbf{F}_a \times \mathbf{p}_{3,a}^w, \tag{2.16}$$

where $\mathbf{p}_3^w$ is the translation vectors going from the absolute task frame to the interaction point, as depicted in Fig. 2.6. Note that the computed wrench is still associated with the absolute task control frame defined by $\mathcal{F}_a$.

2.3 Task space control

The previous section was dedicated to the *Task Space Adapter* module from Fig. 2 which is in charge of converting the data coming from sensors into usable information in the task space.

The role of the *Dual-Arm Controller* block is to provide the joint velocity commands to the robot from task space information. The block is divided into two sequential

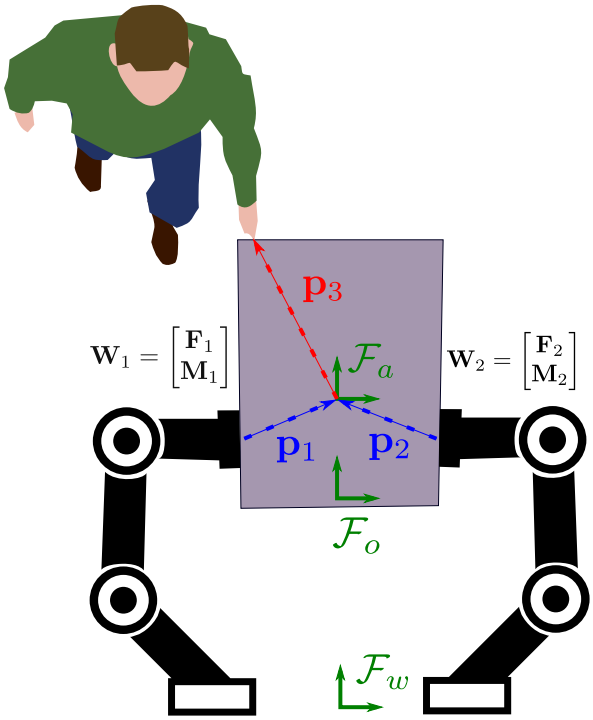


Figure 2.6 – Assuming a rigid grasp, wrenches applied at the absolute frame $\mathcal{F}_a$ are reconstructed considering the virtual sticks $\mathbf{p}_1$ and $\mathbf{p}_2$. Thereafter, external moments exerted by the human are retrieved from the virtual stick $\mathbf{p}_3$.

processes. In this section, we focus on the *Task Space Command* generation while the *Inverse Kinematics* resolution will be the subject of Section 3.1.

The task space command aims at achieving a cooperative kinematic task while interacting with the environment. For this, we implement the closed-loop admittance control law described in Section 2.3.1. However, for safety reasons, it is sometimes preferable to drift from the planned trajectory in order to get away from obstacles. To this end, we specify in Section 2.4.3 a complementary velocity command which is activated when approaching obstacles. When this happens, the weighted combination of the two tasks allows to avoid obstacles while still trying to reach the target.

2.3.1 Closed-loop admittance control for physical interactions

As presented in the previous section, the feedback data coming from the different sensors are interpreted in the task space. Here, the trajectory from the *Trajectory Generator* block is combined with this information to adapt the command to the actual state of the environment..

To let the robot interact with the environment, let us consider a virtual spring-damper mechanisms attached to each control frame, as illustrated in Fig. 2.7.

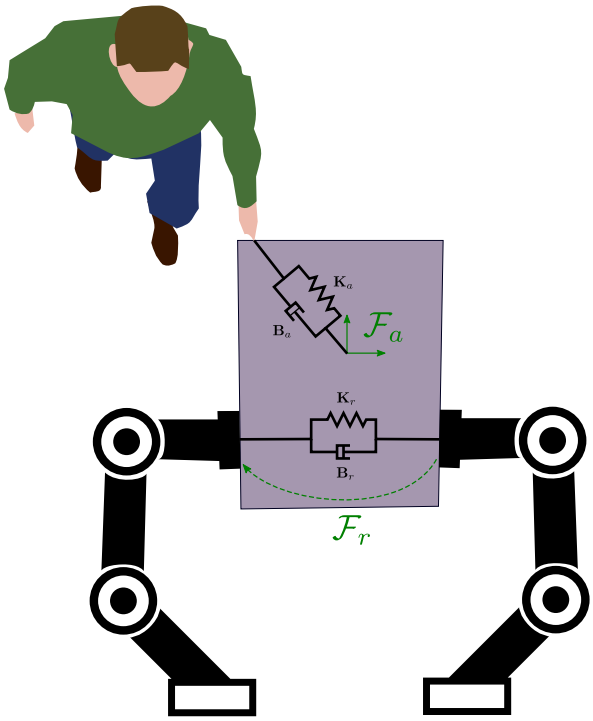


Figure 2.7 – Representation of the virtual spring-damper systems attached to the relative and absolute task control frames.

In the remainder of this section, without loss of generality, we will treat the general case of admittance control without distinguishing between the absolute and relative task, since the process is similar.

Applying Newton's law on the virtual spring-damper system leads to the dynamic equation of motion:

$$\mathbf{W} = \mathbf{K}\mathbf{x} + \mathbf{B}\dot{\mathbf{x}}. \tag{2.17}$$

This equation relates the wrench $\mathbf{W}$ applied by the spring-damper system on the control frame and the pose $\mathbf{x}$ of this control frame by means of a proportional-derivative controller. The stiffness $\mathbf{K}$ and the damping $\mathbf{B}$ are positive definite diagonal matrices representing the gains of the controller.

Let us now instantiate the previous differential equation at the desired values (superscript *):

$$\mathbf{W}^* = \mathbf{K}\mathbf{x}^* + \mathbf{B}\dot{\mathbf{x}}^*. \tag{2.18}$$

By subtracting Eq. (2.18) from Eq. (2.17), we obtain the closed-loop impedance control law:

$$\Delta\mathbf{W} = \mathbf{K}\Delta\mathbf{x} + \mathbf{B}\Delta\dot{\mathbf{x}}, \tag{2.19}$$

with $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$, $\Delta\dot{\mathbf{x}} = \dot{\mathbf{x}} - \dot{\mathbf{x}}^*$ being the errors between the current and desired task pose and velocity, respectively.

To realize admittance control, Eq. (2.19) can be rewritten in the following form:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}^* + \mathbf{B}^{-1}(\Delta\mathbf{W} - \mathbf{K}\Delta\mathbf{x}). \tag{2.20}$$

For each task variable, a specific behavior can be obtained. In particular, we can design three control modes from the admittance equation Eq. (2.20) (without loss of generality, every term is now expressed as a scalar value and index $i \in [1; 6]$ is used to designate a task space DoF):

- **Pose control**: the controller aims at following the desired pose x_i^* and velocity $\dot{x}_i^*$ delivered by the trajectory generator. Pose feedback is used to compensate for the error. Wrenches are not used in this control mode and the control goal is to track the desired x_i^*, $\dot{x}_i^*$. The resulting equation is

$$\dot{x}_i = \dot{x}_i^* - \frac{K_i}{B_i}\Delta x_i. \tag{2.21}$$

  with $\frac{K_i}{B_i} \geq 0$ to ensure proper trajectory tracking and stability of the system.

- **Force control**: the desired velocity is computed to regulate the applied wrench to some desired value W_i^*.

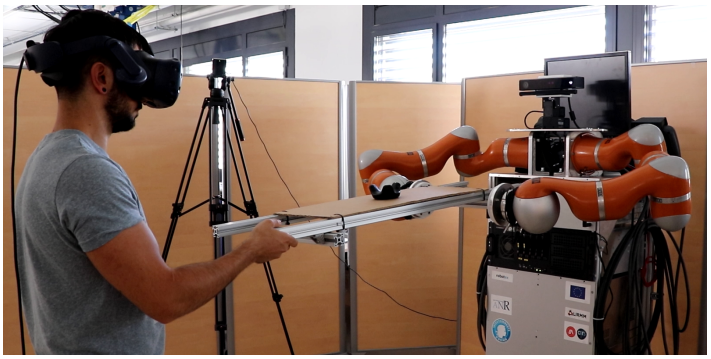
$$\dot{x}_i = \frac{\Delta W_i}{B_i}. \tag{2.22}$$

- **Damping control**: this is a particular case of the force control in which $W_i^* = 0$. The robot's motion is adapted according to the perceived external wrenches. This mode is particularly interesting to perform kinesthetic guidance (e.g. teaching by demonstration) where a human operator manually drives the robot by exerting efforts on it.

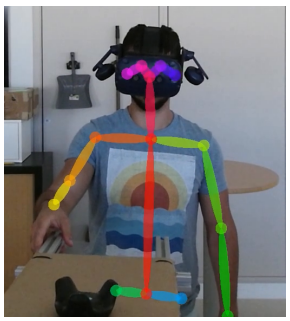
$$\dot{x}_i = \frac{W_i}{B_i}. \tag{2.23}$$

Except for these particular cases, the general admittance control equation Eq. (2.20) can be tuned to give to the robot the desired compliant behavior.

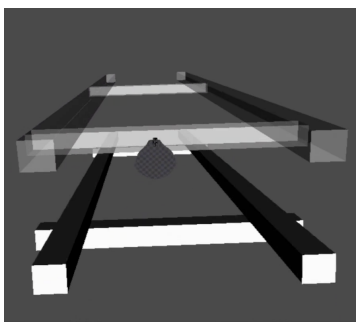
2.4 Application to human-robot collaborative transportation



(a) Experimental setup



(b) Model of the user skeleton.



(c) Virtual objects seen in the HMD.

Figure 2.8 – Human-robot co-manipulation of a large object using damping control. The interaction point is detected by the computer vision system. The current and target location of the object are displayed in the Head Mounted Display.

The proposed approach has been evaluated through an experimental study conducted on the dual-arm cobot BAZAR (CPN+19). To assess human intention consideration during cooperative manipulation, we set up an original simulated application

scenario: selected participants were asked to achieve interactive tasks with the robot in the form of a table-moving scenario. The dual-arm robot is in charge of maintaining the table from one side by regulating internal efforts while the human operator grasps it on the opposite side with only one hand. The objective is to move the object, with damping control, from an initial pose in the space to a final one. The human motion should remain as natural as possible (taking the shortest distance to reach the goal and applying reasonable efforts).

During the experiments, participants wear a Head-Mounted Display (HMD) in which the carried table is displayed both at the current (solid) and goal (transparency) location, as depicted in Fig. 2.8c. The participants' task is to bring the solid object on top of the transparent one. The current pose is tracked online and its visualization in the virtual environment is updated continuously. This way, the user determines the motion needed to complete the task. Participants randomly execute the operation with/without the interaction point consideration without knowing it. Screenshots of the experiment are given in Fig. 2.8a.

Three different operations have been elaborated for this study. In any case, the relative task is kept in a constant position on every component except for the z translational axis for which the force is regulated (to maintain the object properly). A damping term is added to the position control to give some compliance to the system. From a constant initial pose, a transform is applied to the absolute task for the three different cases:

1. *trany+*: translation of 25 cm along y axis. The user can move the object in the horizontal plane (in translation and rotation).

2. *tranz+*: translation of 25 cm along z axis. The user can move the object in the vertical plane in translation and rotate around y axis.

3. *rotz+*: rotation of 0.5 rad around z axis. The user can move the object in the horizontal plane (in translation and rotation).

Video of the experiments is available at http://bit.do/e3mxM.

2.4.1 Setup

The BAZAR robot is equipped with two 7-DoF Kuka LWR4 arms. All experiments are performed on a computer with an Intel(R) Xeon(R) E5-2620 v3 CPU running Linux with the PREEMPT-RT patch. Our approach has been implemented in C++ using the RKCL framework (presented in Chapter 4). The Fast Research Interface Library (FRI)[2] is used to communicate with the Kuka arms, and the controller sample time was set to $T = 5$ ms.

We tune the gains depending on the control mode: for compliant pose control, we set $B = 150$, $K = 250$ for forces and $B = 25$, $K = 40$ for torques; same gains are

[2] https://cs.stanford.edu/people/tkr/fri/html/

used for damping controlled variables but the stiffness term is removed ($K = 0$); for force-controlled variables, the stiffness gain is also $K = 0$ while the damping term is $B = 1000$ for forces and $B = 500$ for torques.

We use computer vision to estimate online the position of the interaction point. The algorithm is based on the OpenPose library (CHS+18) which extracts the set of 2D points composing the skeleton of the persons present in a given color image. By using a Microsoft Kinect V2 RGB-D camera, we can reproject these 2D points in 3D using the depth information provided by the sensor, as shown in Fig. 2.8b. We use the right-hand wrist as the interaction point since OpenPose doesn't provide a point for the hand itself. This allows us to position the point of interaction with an accuracy of a few centimeters[3], which is sufficient for our application. In the case of occlusions, the last interaction point, expressed in the absolute frame, is kept. This hypothesis holds as long as the operator does not reposition his/her hand while the occlusion occurs. By using an NVIDIA GTX 1080 Ti GPU, we achieve around 7 estimations per second.

To get reproducible and accurate results to validate the proposed approach, we set up a virtual reality (VR) system to instruct the operator on the task to achieve. This system is composed of an HTC Vive Pro HMD, an HTC Vive tracker attached to the transported object and two fixed Steam VR base stations. This VR system was only used for validation purposes and is not required in normal operations.

2.4.2 Results

Ten participants performed a total of 15 operations (five for each case, selected in random order). Several criteria have been evaluated and statistical results are given in Fig. 2.9. Instantaneous power P has been obtained using the relation $P = |\overrightarrow{\mathbf{W}} \cdot \overrightarrow{\dot{\mathbf{x}}}|$ and the average energy $\bar{E}$ expended during a time interval Δt arises from $\bar{E} = \bar{P}\Delta t$; $\bar{P}$ being the average power supplied during the same time interval.

A general observation is that the interaction point consideration greatly improves task performances. Indeed, the various metrics indicate that all the tasks can be completed effortlessly when the interaction point is known (orange bars) while it is much more difficult otherwise (blue bars).

A closer look reveals that the differences are more noticeable for the translation tasks for which the necessary energy is between 2.5 (for the trany+ task) and 7.9 (for the tranz+ task) times higher when the interaction point is unknown. By contrast, the gap is smaller for the rotation task with a ratio of 1.5 for the average energy. The power consumption is even equivalent in this case with around 0.3W on average. Referring to Eq. (2.16), this can be explained by the fact that the interaction point has an influence on the final wrench only in the presence of translational forces. In practice, it is almost impossible to apply perfect torque on the object without exerting any residual translation. Thus, the surplus energy supplied to complete the rotx task without interaction point may result from the greater effort generated to compensate

[3]Accuracy decreases with distance due to depth inacuraries

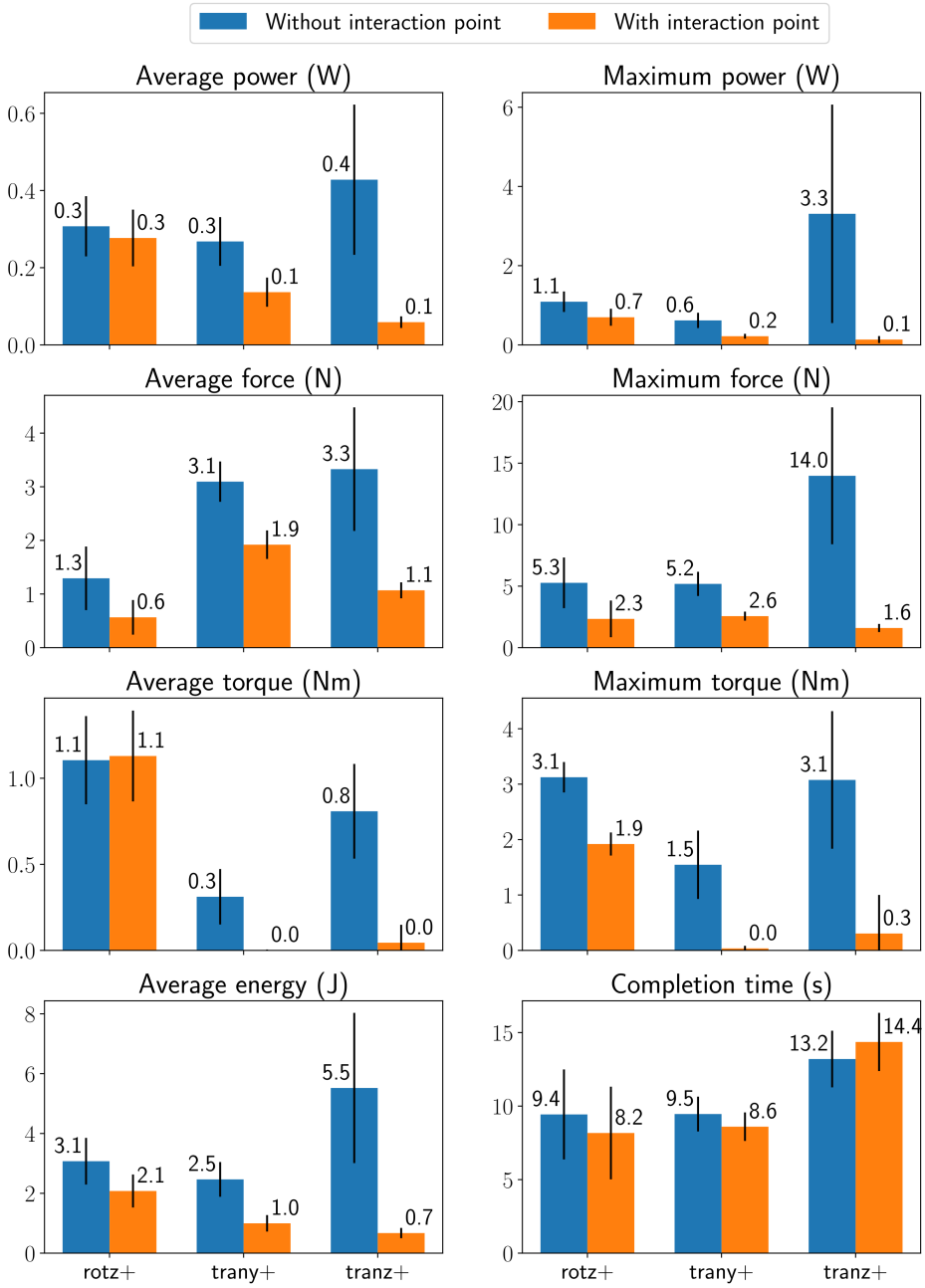


Figure 2.9 – Statistical data comparing several metrics for the three different tasks when the interaction point is taken into account (orange) or not (blue). Results showcase the benefits of our method since less effort is required by the participants to achieve the tasks.

for the translational error (1.3N against 0.6N), as the motion is less intuitive in this case.

The task tranz+ is the one that shows the strongest contrast when comparing the metrics. In particular, the maximum values are extremely higher when the interaction point is not taken into account. This is also the task in which the data are the most irregular (highlighted by standard deviation significant values). It seems that, depending on the participant, the unwanted torque generated when the interaction point is omitted creates some orientation error that is not trivial to compensate. It would appear that a task on the horizontal plane is easier to get under control when some error occurs.

The completion time is not significantly influenced by the method that is used and is more dependent on the person realizing the task.

2.4.3 Repulsive action for collision avoidance

In the light of factories of the future, environments in which robots operate tend to be less and less structured. Unexpected events are likely to occur while machines are doing their job, sometimes leading to dangerous situations. A concrete example of this is the increasing proximity with human operators. During non-collaborative operations, they should be treated as obstacles to avoid. There exist different kinds of reactive control strategies for avoiding collisions, some being more restrictive than others.

One option is to implement the Velocity Damper method introduced in (FT87) as linear constraints of the inverse kinematics optimization problem as described in Section 3.2.3. This allows to ensure that a strict minimum distance is always guaranteed between moving parts of the robot and obstacles. However, using such method, the robot just reduces its speed until the critical distance is reached and then stops, so the task cannot be completed. In some situations, it is possible to find another valid path in the task space by trying to bypass obstacles. Our idea is to combine the use of hard constraints (see Section 3.2) with a repulsive action included in the cost function of the absolute task. We describe the latter approach in what follows.

Let us start by defining a repulsive velocity vector using the method presented in (FKDLK12): let $\mathbf{p}_a$ be the position vector of the absolute task frame $\mathcal{F}_a$ and $\mathbf{p}_o$ the position of the nearest obstacle point from $\mathbf{p}_a$. A repulsive velocity vector $\dot{\mathbf{x}}$ between these two points is defined as:

$$\dot{\mathbf{x}} = v(\mathbf{p}_a, \mathbf{p}_o) \frac{\mathbf{D}(\mathbf{p}_a, \mathbf{p}_o)}{\|\mathbf{D}(\mathbf{p}_a, \mathbf{p}_o)\|}, \tag{2.24}$$

where $\mathbf{D}(\mathbf{p}_a, \mathbf{p}_o)$ is the vector joining $\mathbf{p}_o$ from $\mathbf{p}_a$, and $v(\mathbf{p}_a, \mathbf{p}_o)$ refers to the magnitude of the repulsive vector, such as:

$$v(\mathbf{p}_a, \mathbf{p}_o) = \frac{V_{max}}{1 + e^{(\|\mathbf{D}(\mathbf{p}_a, \mathbf{p}_o)\|(2/\rho) - 1)\alpha}}, \tag{2.25}$$

V_{max} being the maximum admissible magnitude, α a shape factor, and ρ the separation distance from which the repulsion is activated.

In this work, only the nearest obstacle from the point of interest is kept to evaluate the repulsive action.

We chose to take into account the repulsive action by performing a weighted combination of the two velocity command vectors applied at the absolute task frame. For this, we define the weight variable $w \in [0; 1]$ to adjust the influence of each task. In the following equation, we call $\dot{\mathbf{x}}_{a1}$ the admittance velocity command of the absolute task resulting from Eq. (2.20) and $\dot{\mathbf{x}}_{a2}$ the repulsive action. The final absolute task velocity command $\dot{\mathbf{x}}_a$ is obtained with:

$$\dot{\mathbf{x}}_a = w\dot{\mathbf{x}}_{a1} + (1 - w)\dot{\mathbf{x}}_{a2}. \tag{2.26}$$

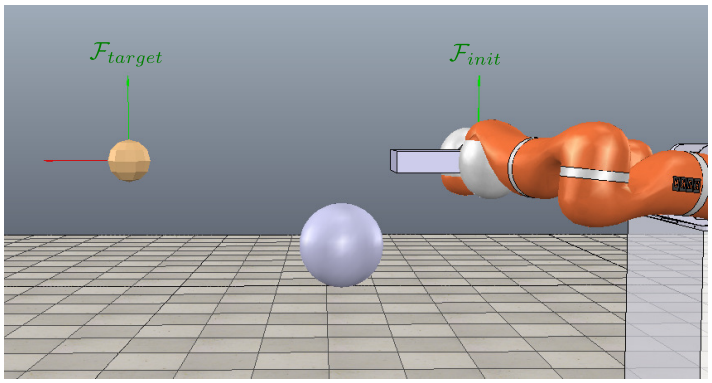
2.4.4 Simulation experiments

We evaluate the behavior of the task space command by performing a simple simulation using the Virtual Robot Experimentation Platform (V-REP)[4]. The task consists in moving an object with a mobile dual-arm robot between two poses of the workspace, while a spherical object obstructs the path, as shown in Fig. 2.10. For the absolute task space variables, the admittance control law is reduced to its pose control version Eq. (2.21) as wrenches are not taken into account. The goal position to reach in the absolute task coordinates is at $x = +0.8$m from the initial state while the center of the obstacle is located at a position $\mathbf{p}_{obs} = [0.3, 0, -0.2]^T$ from this same state.

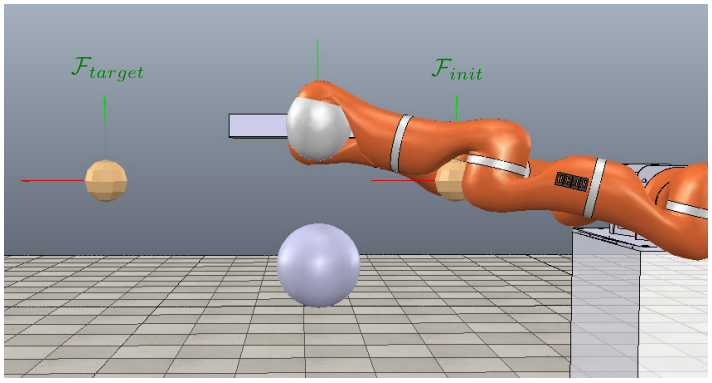
We set a gain of $B^{-1}K = 1$ to compensate the position error. Parameters for the repulsive action are $V_{max} = 1\,\mathrm{m\,s^{-1}}$, $\alpha = 6$ and $\rho = 0.2$m. The weight variable from Eq. (2.26) is set to $w = 0.5$, meaning that both tasks have the same impact on the final solution.

The results of this experiment are given in Fig. 2.11. From the beginning of the task to time $t = 1$s, only the translational twist from the admittance command is effective. As long as the absolute task frame is located at a distance $d > \rho$ from the obstacle, no repulsive action is generated. From $t = 1$s, the distance is no longer safe and the collision avoidance strategy starts to be activated. We observe that the repulsive vector drives the control frame towards the $x-$ and $z+$ direction, which makes the robot slow its progression towards the goal and try to pass over the object. This creates some error on the planned trajectory that the admittance controller starts compensating by increasing the velocity commands in $x+$ and $z-$ directions. Thanks to this, the robot is still able to continue its progress towards the target location, at a slower velocity and getting around the obstacle. From $t = 6.65$s, the obstacle has been passed and the robot finishes the task at time $t = 9.14$s.

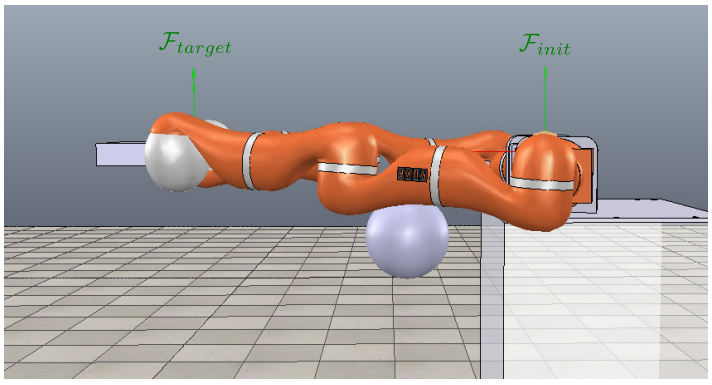
[4] http://www.coppeliarobotics.com/



(a) Initial state



(b) intermediate state



(c) Final state

Figure 2.10 – Snapshots of the simulated experiment. The dual-arm robot has to bring the object maintained with its two arms from an initial pose $\mathcal{F}_{init}$ to a target pose $\mathcal{F}_{target}$. The operation is characterized by a simple translation along the x axis in the operational space, with respect to the world frame. The spherical obstacle located along the way produces a repulsive action which makes the robot deviate from its initial direction but still reach the target location.

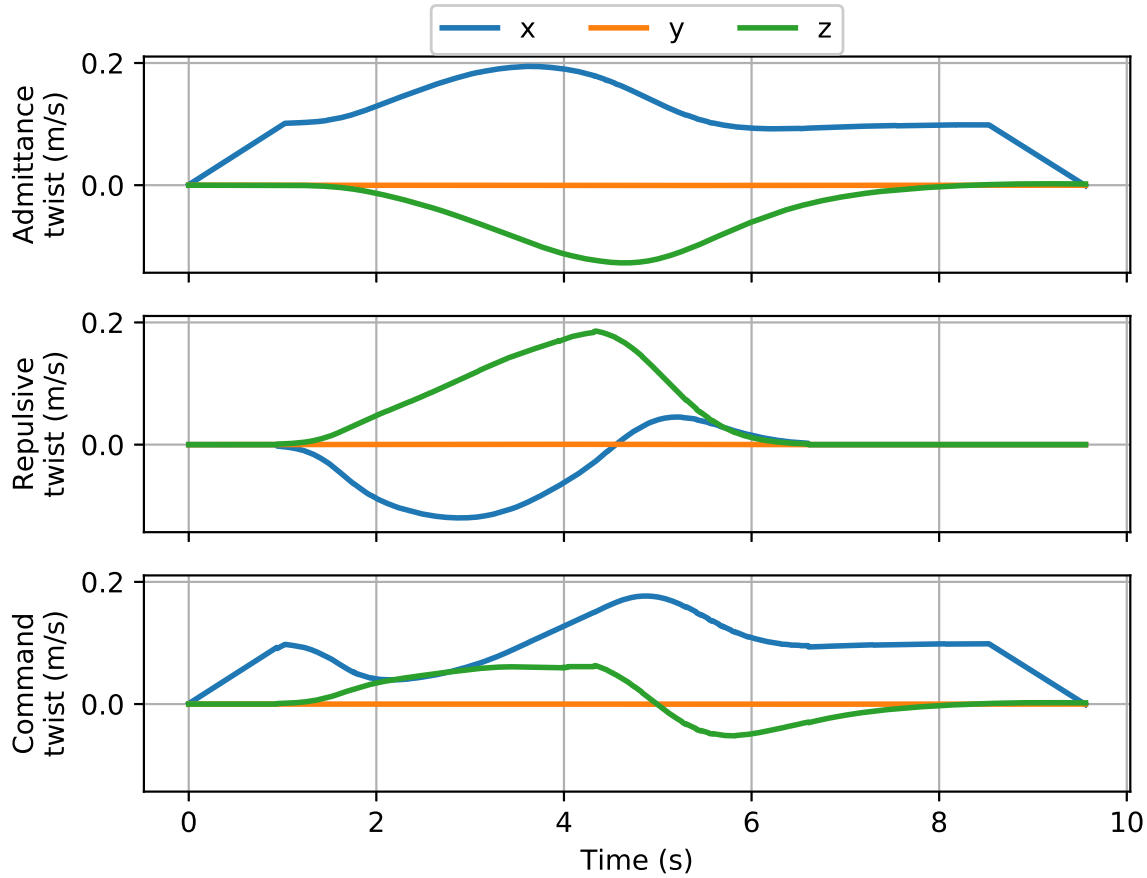


Figure 2.11 – Absolute task translational twist command generation during the simulated experiment shown in Fig. 2.10. The top graph shows the admittance command $\dot{\mathbf{x}}_{a1}$ resulting from Eq. (2.20). It follows the trajectory sent by the trajectory generator while compensating for the position error. Just below is represented the repulsive velocity action $\dot{\mathbf{x}}_{a2}$ for collision avoidance. The generated vector points towards the direction opposite the object and its magnitude increases as it gets closer. Finally, the bottom graph is the translational twist $\dot{\mathbf{x}}_a$ sent to the IK process, resulting from the combination of the two previous commands via Eq. (2.26).

2.5 Conclusion

This section addressed the problem of controlling a dual-arm robot at task level. By adopting and extending the cooperative task space description, we specify bimanual operations in a straightforward and robust way. Thanks to the wrench feedback conversion from the tip of each arm to the cooperative task space, interactions with the environment are properly interpreted. An admittance based control law uses this information to generate desired velocity commands. This allows us to propose a hybrid position/force control strategy by letting the user select the desired control modes for each task variable.

To allow natural and efficient human-robot collaboration for industrial purposes, we gave a special focus on object co-manipulation with a dual-arm robot. Notably, we developed an online method for estimating gravity parameters of the manipulated object. Then, by applying some treatments on the wrench feedback data during collaborative operations, we can compensate the load exerted on the robotic arms to keep only external efforts applied by the human.

Furthermore, the human's contact location with the object is crucial information allowing to retrieve his intention from wrench measured at the arms' wrist. An experimental study has been elaborated to simulate a table-moving scenario using a virtual reality system and computer vision to detect the participant's hands during co-manipulation. The results clearly indicate that considering the interaction point is beneficial for co-manipulation tasks as it is more intuitive and thus requires less energy to be completed.

Finally, The absolute task admittance command can be complemented with a repulsive action vector to perform collision avoidance. The detection of a close obstacle leads to the generation of a complementary velocity vector oriented in the opposite direction and added to the original command with a weighting factor.

In the next chapter, we will focus on solving the inverse kinematics problem to obtain the command in the joint space, where actuation takes place.

CHAPTER 3

Dual-arm joint motion control

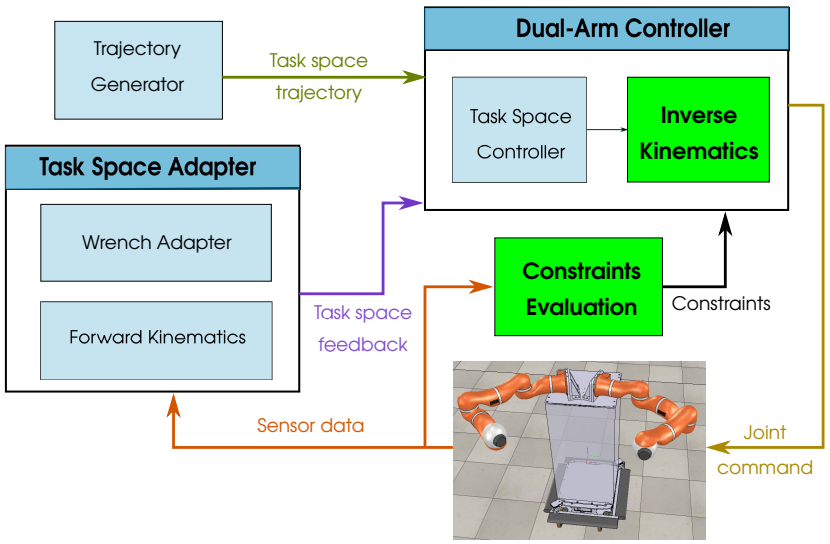


Figure 3.1 – Components of the dual-arm kinematic control scheme tackled in this chapter.

The final process of the dual-arm controller is to solve the inverse kinematics problem stated in Eq. (1.6), whose aim is to convert the specification of the task in the operational space into the joint space. This operation is performed by the *Inverse Kinematics* block depicted in Fig. 3.1.

In this chapter, we propose several strategies to manage the high number of DoF of dual-arm robots. The objective is to comply as much as possible with the task control while ensuring that the robot motion is safe with respect to itself and the environment. Additionally, redundancy of bimanual platforms leaves room for optimization of any arbitrary criteria. Considering a dynamic environment populated with humans, we

chose to dedicate remaining DoF to reduce the overall displacements in the workspace, thus limiting the risk of hazardous situation.

It is usual that dual-arm platforms embed robotic extensions such as a mobile base or an articulated torso. However, the characteristics of such robots generally differ from the arms. We present in this chapter an original inverse kinematics approach which deals with extra joints in an appropriate way.

Finally, we give a thorough analysis of the different constraints that should be taken into account and their formulation in the *Constraints Evaluation* block of Fig. 3.1.

3.1 Inverse kinematics resolution

In Section 1.2, we reviewed existing methods for solving the inverse kinematics of redundant robots. Although easily implementable and computationally efficient, we showed the limits of analytical solutions, especially when dealing with highly constrained systems. In this section, we present several task-solving strategies for dual-arm robots performing cooperative tasks under hard constraints.

We base our methods on QP/HQP resolution following the general QP equation given in Eq. (1.10) and the hierarchical extension proposed in Algorithm (1).

In the remainder of the paper, we denote by $\boldsymbol{\epsilon}$ the residual error on the kinematic task, such as:

$$\boldsymbol{\epsilon} = \dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}} \tag{3.1}$$

When $\|\boldsymbol{\epsilon}\| = 0$, the task is tracked without any error.

3.1.1 Solving the cooperative tasks with a unique QP

Considering the cooperative dual-arm task space, both the relative and absolute tasks have to be satisfied at the same time. A natural approach to solve the problem in one go is to merge the variables associated with each task to formulate the optimization problem as in Eq. (1.10).

Let $\mathbf{J}_a$, $\dot{\mathbf{x}}_a$ and $\mathbf{J}_r$, $\dot{\mathbf{x}}_r$ be the Jacobian matrices and task space command vectors for the absolute and the relative tasks, respectively. We gather the dual space information in the Jacobian matrix $\mathbf{J}_{coop}$ and task space command vector $\dot{\mathbf{x}}_{coop}$ by performing the following concatenation operations:

$$\begin{cases} \mathbf{J}_{coop} = \begin{bmatrix} \mathbf{J}_a^T & \mathbf{J}_r^T \end{bmatrix}^T, \\ \dot{\mathbf{x}}_{coop} = \begin{bmatrix} \dot{\mathbf{x}}_a^T & \dot{\mathbf{x}}_r^T \end{bmatrix}^T. \end{cases} \tag{3.2}$$

The QP optimization problem can now be rewritten as:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_{coop}\dot{\mathbf{q}} - \dot{\mathbf{x}}_{coop}\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \quad \mathbf{C}\dot{\mathbf{q}} = \mathbf{d} \end{aligned} \tag{3.3}$$

The solution vector $\dot{\mathbf{q}}$ of Eq. (3.3) is one that minimizes the quadratic error on the mixed cooperative task space variables. It contains the joint velocity commands for the whole dual-arm robot. This result is then split to be transmitted to the appropriate hardware parts. The joint order should correspond to the one used to define the Jacobian matrices.

The equality and inequality constraints are identical for the absolute and relative task subproblems, they are reused here to fully specify the admissible solution space.

3.1.2 Parsimonious task-solving approach

Despite an advanced state of the art in the field, one drawback with Eq. (1.10) is the generation of joint velocities for which every component is non-null, regardless of the number of DoF controlled in the task space. This minimizes the energy dissipated by the system but not the robot overall displacement.

In non-rigid industrial settings where the robot's free space can change, a parsimonious resolution would be more appropriate (GFCA16). Here, we aim at minimizing the number of joints involved in the task, thus reducing the robot motion.

The corresponding optimization problem is:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\dot{\mathbf{q}}\|_0 \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}, \end{aligned} \tag{3.4}$$

with $\|\dot{\mathbf{q}}\|_0$ the number of nonzero components in $\dot{\mathbf{q}}$. This is a NP-hard problem that requires a combinatorial approach and dual-arm complexity does not allow to solve it on line. It has been shown (EB02) that the sparsest solution can also be obtained by solving the following equivalent problem:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\dot{\mathbf{q}}\|_1 \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}. \end{aligned} \tag{3.5}$$

By easily transforming Eq. (3.5) into a linear program, a sparse solution can be efficiently generated.

Fuchs also proved (Fuc04) that a parametrized quadratic program, known in statistics as the lasso (Least Absolute Shrinkage and Selection Operator), and expressed as:

$$\min_{\dot{\mathbf{q}}} \quad \frac{1}{2}\|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|_2^2 + \beta\|\dot{\mathbf{q}}\|_1, \quad \beta \in \left[0^+; \left\|\mathbf{J}^T\dot{\mathbf{x}}\right\|_\infty\right] \tag{3.6}$$

converges to the same minimum point solution, i.e. having least l_1-norm, when $\beta = 0^+$. The advantage of using Eq. (3.6) instead of Eq. (3.5) is that the kinematic problem is formulated as a cost function and not as an equality constraint. This leads to a softer behavior, since a solution is always found. In fact, the optimization process tries to minimize the tracking error of the task. If the kinematic problem is undetermined and

several solutions exist, then the sparsest one is returned. Instead, with Eq. (3.5), no valid solution is computed when $\|\boldsymbol{\epsilon}\| > 0$.

An algorithm for solving this optimization problem, while also including both equality and inequality constraints, has been proposed in (GZ16). The objective is to find a solution to the constrained lasso problem:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \frac{1}{2}\left\|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\right\|_2^2 + \beta\left\|\dot{\mathbf{q}}\right\|_1, \quad \beta \in \left[0^+; \left\|\mathbf{J}^T\dot{\mathbf{x}}\right\|_\infty\right] \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \quad \mathbf{C}\dot{\mathbf{q}} = \mathbf{d} \end{aligned} \tag{3.7}$$

In (GZ16), the problem is reformulated as a standard QP: the l_1 penalty term $\|\dot{\mathbf{q}}\|_1$ can be handled by decomposing $\dot{\mathbf{q}}$ into its positive and negative parts, such as $\dot{\mathbf{q}} = \dot{\mathbf{q}}^+ - \dot{\mathbf{q}}^-$ with $\dot{\mathbf{q}}^+ \geq 0$, $\dot{\mathbf{q}}^- \geq 0$. Then, plugging $\|\dot{\mathbf{q}}\|_1 = \dot{\mathbf{q}}^+ + \dot{\mathbf{q}}^-$ in Eq. (3.7):

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \frac{1}{2}\begin{pmatrix}\dot{\mathbf{q}}^+\\ \dot{\mathbf{q}}^-\end{pmatrix}^T \begin{pmatrix}\mathbf{J}^T\mathbf{J} & -\mathbf{J}^T\mathbf{J}\\ -\mathbf{J}^T\mathbf{J} & \mathbf{J}^T\mathbf{J}\end{pmatrix}\begin{pmatrix}\dot{\mathbf{q}}^+\\ \dot{\mathbf{q}}^-\end{pmatrix} \\ & + \left(\beta\mathbf{1}_{2N_{\text{dof}}} - \begin{pmatrix}\mathbf{J}^T\dot{\mathbf{x}}\\ -\mathbf{J}^T\dot{\mathbf{x}}\end{pmatrix}\right)^T \begin{pmatrix}\dot{\mathbf{q}}^+\\ \dot{\mathbf{q}}^-\end{pmatrix} \\ \text{s.t.} \quad & \begin{pmatrix}\mathbf{A} & -\mathbf{A}\end{pmatrix}\begin{pmatrix}\dot{\mathbf{q}}^+\\ \dot{\mathbf{q}}^-\end{pmatrix} \leq \mathbf{b}, \\ & \begin{pmatrix}\mathbf{C} & -\mathbf{C}\end{pmatrix}\begin{pmatrix}\dot{\mathbf{q}}^+\\ \dot{\mathbf{q}}^-\end{pmatrix} = \mathbf{d}, \\ & \dot{\mathbf{q}}^+ \geq 0, \qquad\qquad \dot{\mathbf{q}}^- \geq 0 \end{aligned} \tag{3.8}$$

which is the formulation of a standard QP of $2N_{\text{dof}}$ variables, N_{dof} being the dimension of $\mathbf{q}$.

3.1.3 Hierarchical inverse kinematics strategy

In Section 1.2.2, we introduced the hierarchical resolution of the inverse kinematics problem. It consists in solving a sequence of QP for which the solution space of a given task is restricted to the null-space of higher priority tasks. This way, tasks that have less priority do not interfere with the others.

We recall the general formulation of the prioritized inverse kinematics problem based on HQP. To do so, let us consider a set of N tasks sorted by decreasing order of priority. The solution vector $\dot{\mathbf{q}}_i$ which allows the ith ($i \in [2; N]$) task to be executed with lower priority with respect to the previous $i-1$ tasks is given by:

$$\begin{aligned} \dot{\mathbf{q}}_i \in \min_{\dot{\mathbf{q}}} \quad & f_i(\dot{\mathbf{q}}) \\ \text{s.t.} \quad & \underline{\dot{\mathbf{Q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{Q}}}, \\ & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \\ & f_{i-1}(\dot{\mathbf{q}}) = f_{i-1}(\dot{\mathbf{q}}_{i-1}), \\ & ..., \\ & f_1(\dot{\mathbf{q}}) = f_1(\dot{\mathbf{q}}_1), \end{aligned} \tag{3.9}$$

where f_i is the cost function associated with task i,$\underline{\dot{\mathbf{Q}}}$ and $\overline{\dot{\mathbf{Q}}}$ are respectively the lower and upper bounds of the joint velocity vector (details are given in the next section). The equality constraints are used to restrict the solution space to the null space of higher priority tasks.

3.1.3.1 HQP for dual-arm robots performing cooperative tasks

Considering the cooperative dual-arm task space, both the relative and absolute tasks have to be satisfied at the same time (see Section 2.1). However, a relevant choice is to prioritize the relative task over the absolute one. Indeed, it is critical to ensure the relative task is fulfilled during bimanual manipulation of an object, to avoid the generation of undesired internal stress and damage the object and/or the robot.

Applied to the cooperative dual-arm task space and taking into account joint limits and collision avoidance, the highest priority task is solved through the following optimization problem:

$$\begin{aligned} \dot{\mathbf{q}}_1 \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_r\dot{\mathbf{q}} - \dot{\mathbf{x}}_r\|_2 \\ \text{s.t.} \quad & \underline{\dot{\mathbf{Q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{Q}}}, \\ & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \end{aligned} \tag{3.10}$$

where $\mathbf{J}_r$, $\dot{\mathbf{x}}_r$, are the Jacobian matrix and task space command vector associated with the relative task.

The obtained solution vector $\dot{\mathbf{q}}_1$ is then used to provide the null-space condition to the second task, that is expressed as:

$$\begin{aligned} \dot{\mathbf{q}}_2 \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_a\dot{\mathbf{q}} - \dot{\mathbf{x}}_a\|_2 \\ \text{s.t.} \quad & \underline{\dot{\mathbf{Q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{Q}}}, \\ & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \\ & \mathbf{J}_r\dot{\mathbf{q}} = \mathbf{J}_r\dot{\mathbf{q}}_1, \end{aligned} \tag{3.11}$$

where $\mathbf{J}_a$, $\dot{\mathbf{x}}_a$, are the Jacobian matrix and task space command vector associated with the absolute task. The relative task solution has been added as a constraint to the

problem to avoid interfering with it. Using this formulation, the absolute task error is minimized as long as the resulting joint velocity vector provides the best solution for Eq. (3.10). If no more DoF are available after solving the relative task, this process will have no impact on the final solution, i.e. $\dot{\mathbf{q}}_2 = \dot{\mathbf{q}}_1$ will be obtained from Eq. (3.11).

3.1.3.2 Adding tunable parsimonious control

In some situations, redundancy is still available after solving all the tasks. It happens for instance when the dual-arm robot has a high overall number of joints (directly from the arms or coming from additional joints in the torso). It is also the case when the cooperative task is not fully specified. In particular, many industrial applications can be performed by only controlling the coordination of the arms' relative motions (see Section 3.1.4).

In that case, it would be interesting to adopt the parsimonious approach presented in Section 3.1.2.

However, taking the sparsest solution, in the sense of minimizing the l_1-norm of joint velocities, sometimes leads to undesired behavior on dual-arm robots. Indeed, due to the geometric symmetry of such systems (the two arms are generally identical) and to the local property of the controller, several solutions that activate completely different joints may have equivalent cost. Consequently, from one time step to the other, the set of joints that are actuated might switch and create chattering phenomena.

To overcome this issue, we decided to implement a third task in the hierarchy that is in charge of managing the level of sparsity of the solution. Instead of choosing between the standard least l_2-norm solution, that results in the permanent activation of almost every joint, and the fully sparse least l_1-norm solution, that creates violent changes in joints actuation, we propose an alternative approach that consists in regulating the proportion of the norms that are minimized.

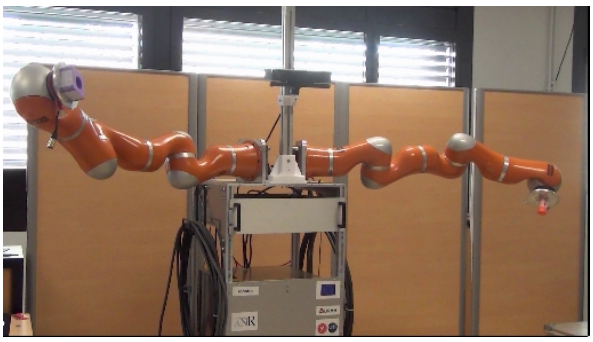
Practically, the task with the least priority is defined by the following optimization problem:

$$\begin{aligned} \dot{\mathbf{q}}_3 \in \min_{\dot{\mathbf{q}}} \quad & (1-\lambda)\left\|\dot{\mathbf{q}}\right\|_2^2 + \lambda\left\|\dot{\mathbf{q}}\right\|_1, \quad 0 \leq \lambda \leq 1 \\ \text{s.t.} \quad & \underline{\dot{\mathbf{Q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{Q}}}, \\ & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \\ & \mathbf{J}_r\dot{\mathbf{q}} = \mathbf{J}_r\dot{\mathbf{q}}_1, \\ & \mathbf{J}_a\dot{\mathbf{q}} = \mathbf{J}_a\dot{\mathbf{q}}_2, \end{aligned} \tag{3.12}$$

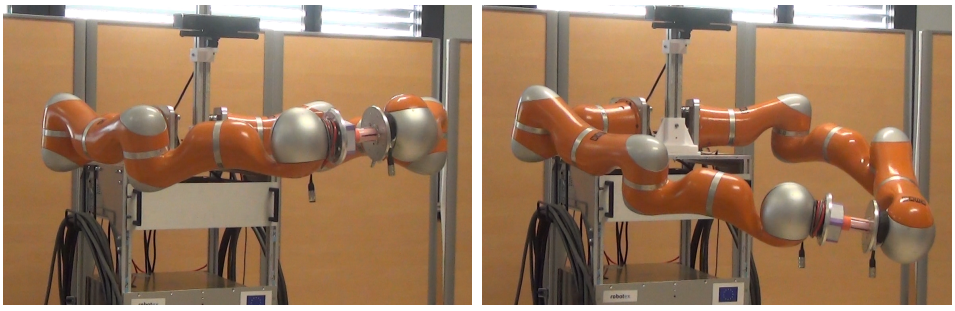
The tunable parameter $\lambda \in [0;1]$ in the cost function is used to distribute the weight assigned to each norm. Setting $\lambda = 0$ is equivalent to performing standard l_2-norm minimization while $\lambda = 1$ provides the sparsest solution, which is identical to the solution of Eq. (3.7). An interesting trade-off is to take a low value of λ (e.g. $\lambda = 0.1$) to take advantage of parsimony while avoiding its side effect.

The equality constraints enforce the solution to be in the null-space of the absolute task which is itself in the null-space of the relative task, thus respecting the hierarchy. If no more redundancy is available after solving the kinematic tasks, this process would have no impact on the final solution, i.e. $\dot{\mathbf{q}}_3 = \dot{\mathbf{q}}_2$ would be obtained from Eq. (3.12).

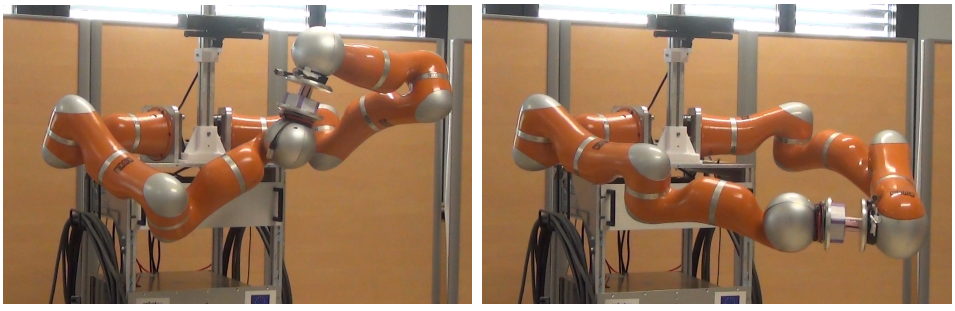
3.1.4 Application to relative tasks



(a) Initial robot configuration



(b) First relative pose target reached



(c) Final relative pose target reached

Figure 3.2 – Initial, middle and final configuration reached during the simulated assembly task. Pictures on the left are taken when the task are solved with the standard inverse kinematics method (no sparsity) while those on the right result from the fully parsimonious task-solving strategy.

Despite the high number of DoF, a dual-arm system is generally "not very redundant". In fact, the task is often specified by 12 parameters (6 for the pose of each

end-effector). Manipulators commonly used in the industry have 6 or 7 DoF, meaning that the overall dual-arm platform will have 12 or 14 DoF. In such cases, the robot is not (or slightly) redundant with respect to the task as all (or almost all) the joints are required to achieve it.

Many industrial bimanual applications require the coordination of the arms' relative motions without having to specify the task in the workspace. Notably, this happens when an object is held by one manipulator while the other realizes a manufacturing operation on it with a tool. Examples of such tasks are assembling (AVK16), sculpting (OCB08) or welding (AL89). For this category of operations, the absolute task is free, meaning that there is no control on the location in the workspace where the robot performs. This increases the redundancy with respect to the task as 6 DoF are added to its null space.

We demonstrate the performance of our hierarchical inverse kinematics strategy through an assembly task scenario involving the control of the relative task only.

We perform the operation on the real dual-arm cobot BAZAR. The objective is to simulate a peg-in-hole insertion: a peg is attached to one arm while a shape with the corresponding hole is fixed to the other. Screwing motion is also required to realize the insertion. The task is divided into two parts:

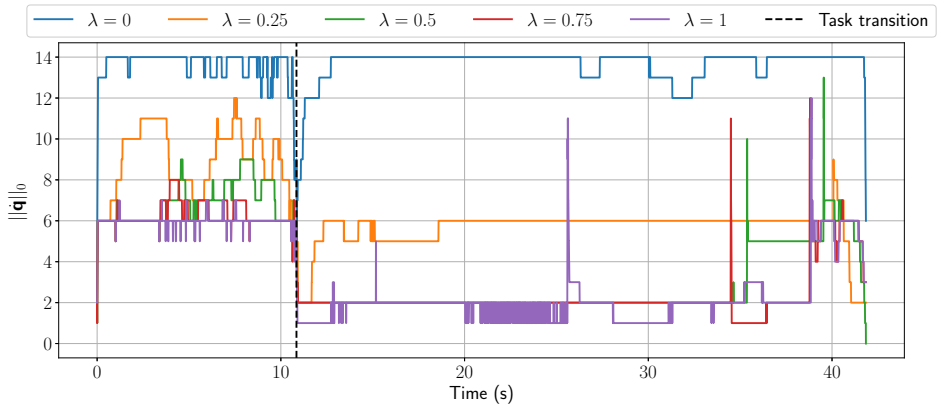
- At the beginning, the two arms are located far from each other as in Fig. 3.2a. The first relative motion aims at bringing the objects closer and aligning the peg with the hole, as shown in the left pictures of Fig. 3.2b.
- Then, the screwing operation is performed to insert the peg in the hole. The robot has to rotate one full turn to accomplish the task. Final configurations reached are shown in Fig. 3.2c.

To evaluate the effect of the sparsity parameter λ from Eq. (3.12), the scenario is performed with five different task-solving configurations: $\lambda = 0$ (regular QP, no sparsity), $\lambda = 0.25$, $\lambda = 0.5$, $\lambda = 0.75$ and $\lambda = 1$ (fully sparse). Hard constraints detailed in Section 3.2 are integrated in any case.

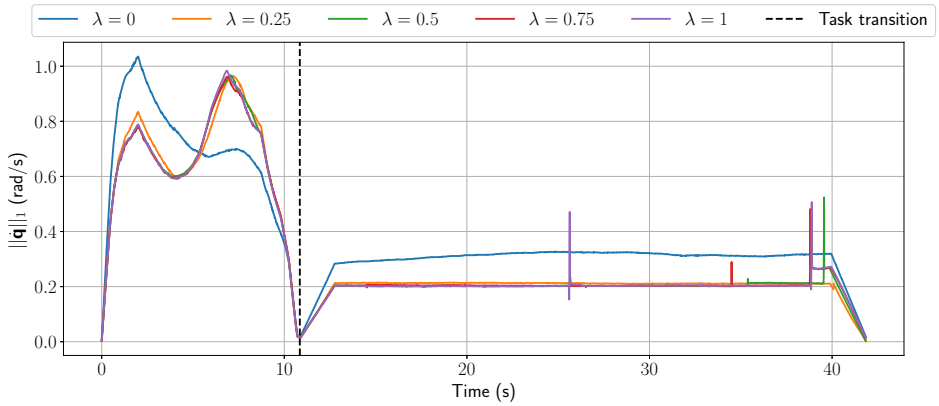
Table 3.1 – Comparison of the methods during the assembly tasks using different metrics.

Metric	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\int_0^\infty \|\|\dot{\mathbf{q}}\|\|_0 dt$	13.64	6.48	3.77	3.33	3.14
$\int_0^\infty \|\|\dot{\mathbf{q}}\|\|_1 dt$	0.392	0.319	0.314	0.314	0.314
$\int_0^\infty \|\|\dot{\mathbf{q}}\|\|_2 dt$	0.023	0.037	0.044	0.047	0.058

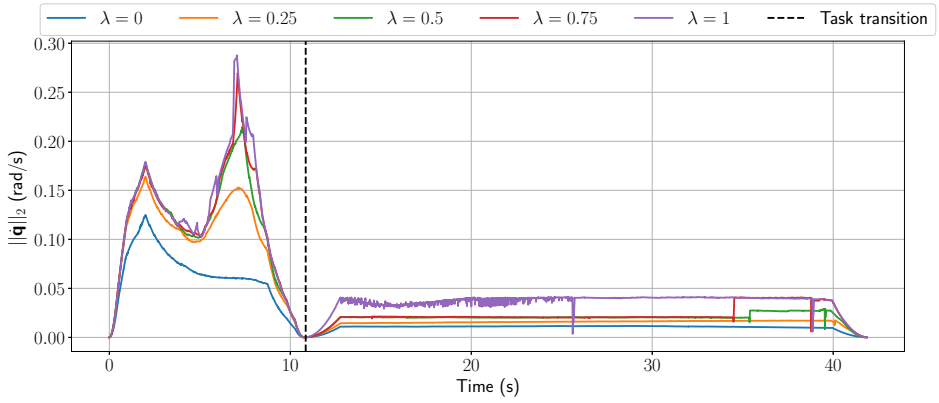
From a high-level point of view, Fig. 3.2 shows initial and final configurations obtained for the screwing task with extreme values of λ. This highlights the behavioral differences between the l_1-norm and l_2-norm minimization methods in terms of generated motions. Indeed, the screwing operation, that could be intuitively performed by



(a) l_0-norm



(b) l_1-norm



(c) l_2-norm

Figure 3.3 – Comparison of joint velocity norms during the assembly task execution for varying values of λ. A threshold of $1 \times 10^{-3}\,\mathrm{rad\,s^{-1}}$ is taken to differentiate active from inactive joints (l_0-norm).

a simple action, generated a high occupancy of the workspace using the standard QP with least square error minimization. Conversely, the parsimonious solution has re-

sulted in local and economic movements as demonstrated by the right side of Fig. 3.2b and Fig. 3.2c that shows similar intermediate and final configurations, except for the grippers' orientations.

Numerical results from Fig. 3.3 and Table (3.1) confirm this observation. The overall number of actuated joints as well as the sum of absolute joint velocities are reduced when λ increases. Indeed, aligning both end-effectors from an arbitrary configuration required the use of six joints for the most part with $\lambda = 1$, corresponding to the number of controlled task variables. Giving more weight to the l_2-norm minimization leads to the progressive activation of extra joints.

Even fewer joints are needed to execute the screwing movement, because of the geometric properties of the operation. With $\lambda > 0.25$ the task can be achieved using only two joints on average. As expected, the standard QP approach with $\lambda = 0$ minimizes the energy consumption during the task. However, this resolution involves the use of almost every joint, regardless of the geometric complexity of the task.

We notice close results for the different metrics with $\lambda > 0$ and particularly for the l_1-norm. In fact, the influence of the weighting factor is not proportionnal to its value because the order of magnitudes of $\|\dot{\mathbf{q}}\|_2$ and $\|\dot{\mathbf{q}}\|_1$ in Eq. (3.12) are not the same. Referring to Table (3.1), there is a factor of about 10 between them, meaning that even a low value of λ strongly influences the task resolution in favor of sparsity.

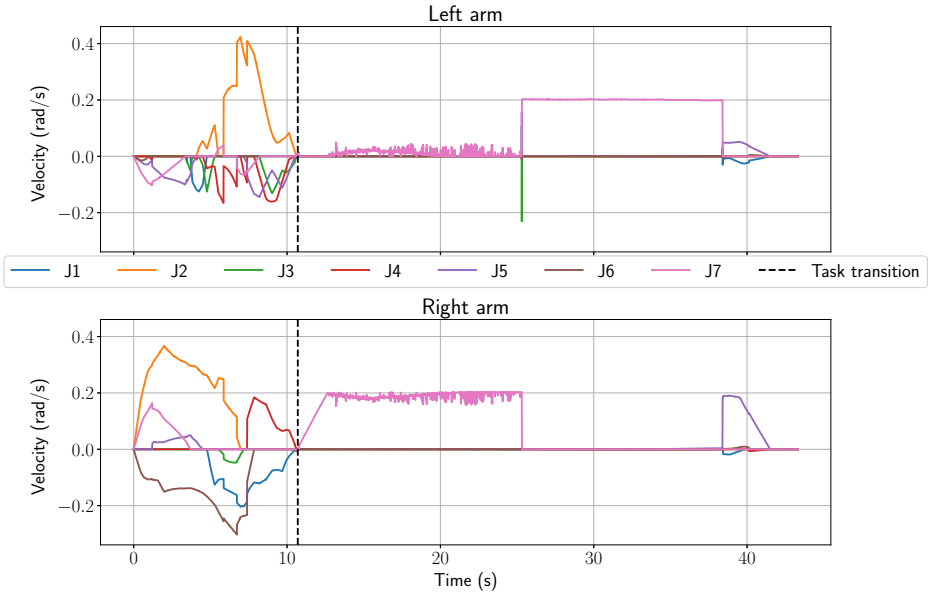
We observe some sudden changes in the metric magnitudes during the task execution. For instance, at time $t = 26$s, the number of active joints for the case $\lambda = 1$ quickly raises from 1 to 11 and the l_1-norm increases by a factor 2.5 in a very short time. This is due to the algorithms which are local and do not ensure global optimality. For some constrained joint configurations, the robot loses manipulability and additional joints should be enabled to fulfill the task.

The fully sparse approach performs slightly better than hybrid l_1-norm/l_2-norm in terms of reduction of the number of activated joints. However, optimizing the l_1-norm only can potentially induce undesired behavior. Indeed, let us focus on the joint velocity profiles depicted in Fig. 3.4. For the sparse approach (Fig. 3.4a), we observe some chattering effect on the last joints of each arm between $t = 12$s and $t = 26$s. In fact, there is no best solution, regarding the l_1-norm minimization, between using one joint or another. Hence, the optimization process variably distributes the efforts between the joints which causes unnecessary velocity variations. Based on Fig. 3.4b, we show that even a small influence of the l_2-norm minimization allows to prevent this phenomenon from happening.

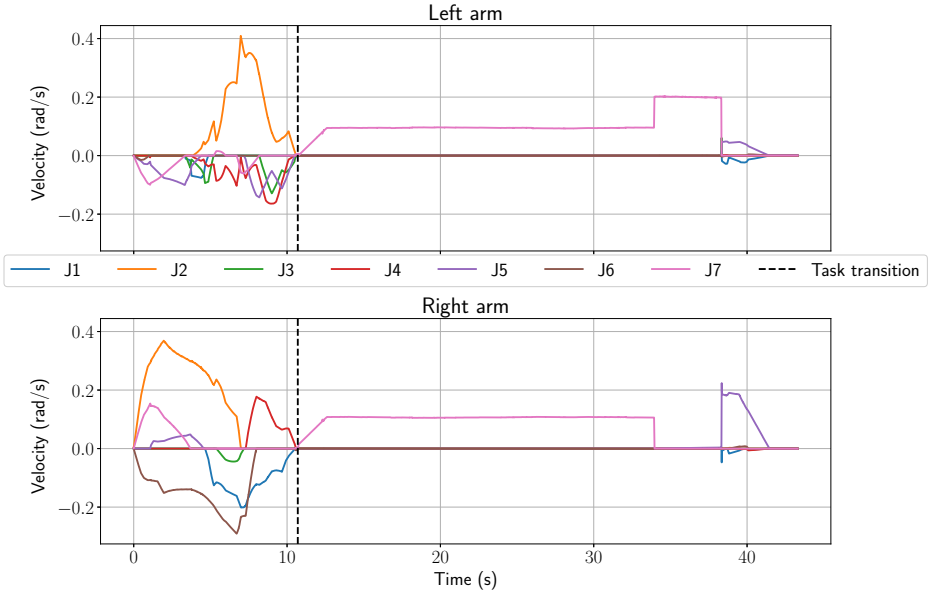
3.1.5 Extension to other robots

Flexibility of production lines is a crucial aspect towards the development of the "factory of the future". Endowing dual-arm robots with mobile capabilities or moving torso enhances the versatility of such systems by widening the accessible area in which they can operate.

Several variations for extending a dual-arm robot are possible. For instance, there



(a) Robot joint velocity with $\lambda = 1$



(b) Robot joint velocity with $\lambda = 0.75$

Figure 3.4 – Comparison of joint velocity profiles between full parsimonious IK resolution ($\lambda = 1$) and high parsimonious with some l_2-norm consideration ($\lambda = 0.75$) during the assembly task execution. Minimizing the l_1-norm only creates some chattering effect. This phenomena disappears when switching to a hybrid l_1-norm/l_2-norm task solving strategy.

exist many kinds of wheeled platforms (e.g. steerable wheeled, Swedish wheels, etc ...) for which features (holonomic, omnidirectional) and control strategies may vary. Also, it might be possible to mount specific articulated tools to expand the arms (e.g robotic

hands) which will provide additional joints at the top of the kinematic system.

3.1.5.1 Control strategy

A naive approach would be to simply consider the additional DoF coming from the additional robot as an extension of the arms and to solve the inverse kinematics problem of the whole system in one go. However, this method is not appropriate as the extended robot and the manipulators may behave differently. In particular, this might be due to:

- their responsiveness. For instance, due to the high inertia of the platform (that supports the entire weight of the dual-arm robot plus generally some embedded equipment), the base cannot provide high accelerations and thus it cannot be used to react promptly to sudden change as the arms do.
- their accuracy. For example, mobile platforms that use odometry data to estimate their precision are generally less precise than robotic arms.

One solution would be to weight the joint contribution in the the QP cost function in order to penalize the use of joints which are less adapted to reactive control. However, when there are more than two groups, it becomes difficult to precisely evaluate the contribution of each group using a weighting strategy.

Instead, we propose a hierarchical control structure at the joint level by setting priority between joint groups. Basically, each task is first solved with the joint group of highest priority. If the vector of residual error $\boldsymbol{\epsilon}$ is non-null (see Eq. (3.1)), then it is forwarded to another optimization process whose decision variables refer to the next joint group regarding priorities. The procedure is repeated until the task is solved without any error or all the joint groups have been involved in the resolution.

Let us suppose that the robot is made of N_{group} joint groups, going from priority 1 to N_{group}, and only one task has to be performed. The velocity vector command $\dot{\mathbf{q}}_i$ for the joint group of priority i ($0 \leq i \leq N_{\text{group}}$) is computed as follows:

$$\begin{aligned} \dot{\mathbf{q}}_i \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_i\dot{\mathbf{q}} - \boldsymbol{\epsilon}_{i\text{-}1}\|_2 \\ \text{s.t.} \quad & \mathbf{A}_i\dot{\mathbf{q}} \leq \mathbf{b}_{i\text{-}1}, \end{aligned} \tag{3.13}$$

where $\mathbf{J}_i \in \mathbb{R}^{6\times N_{\text{dof}}(i)}$ ($N_{\text{dof}}(i)$ being the number of DoF for the joint group i) is the Jacobian matrix associated with the task and for which only the columns related to the joint group i have been kept, i.e. $\mathbf{J}_i = \frac{\partial \dot{\mathbf{q}}_i}{\partial \dot{\mathbf{x}}}$, $\boldsymbol{\epsilon}_i$ is the residual error remaining on the task after solving the problem with the joint group i:

$$\boldsymbol{\epsilon}_i = \dot{\mathbf{x}} - \sum_{k=1}^{i} \mathbf{J}_k\dot{\mathbf{q}}_k, \tag{3.14}$$

$\mathbf{A}_i \in \mathbb{R}^{N_{\text{ineq}}\times N_{\text{dof}}(i)}$ is the linear coefficients matrix of the inequality constraints for which only the columns related to the joint group i have been kept, and $\mathbf{b}_i$ is the

constant vector of the inequality constraint after solving the problem with the joint group i:

$$\mathbf{b}_i = \mathbf{b} - \sum_{k=1}^{i} \mathbf{A}_k \dot{\mathbf{q}}_k. \tag{3.15}$$

Hence, both the cost function and the inequality constraint take into account the impact of the joint velocities which have been already computed (because of higher priority group).

Algorithm 2 Double-layer Hierarchical Inverse Kinematics

Input: $\mathbf{J}, \dot{\mathbf{x}}, \mathbf{A}, \mathbf{b}, \underline{\dot{Q}}, \overline{\dot{Q}}$
Output: $\dot{\mathbf{q}}$
  procedure HierarchicalInverseKinematics
    $\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_1 & \dots & \dot{\mathbf{q}}_{N_{\text{group}}} \end{bmatrix}^T$
    for $i \leftarrow 1 : N_{\text{group}}$ **do**
      $\dot{\mathbf{q}}_i \leftarrow \text{MinimizeJointVelocity}(\underline{\dot{Q}}, \overline{\dot{Q}})$
      $\mathbf{C}_{i,0} \leftarrow [\,]$
      $\mathbf{d}_{i,0} \leftarrow [\,]$
    end for
    for $j \leftarrow 1 : N_{\text{task}}$ **do**
      $\boldsymbol{\epsilon}_{0,j} \leftarrow \dot{\mathbf{x}}_j$
      $\mathbf{b}_{0,j} \leftarrow \mathbf{b}$
      $i \leftarrow 1$
      while $i \leq N_{\text{group}}$ AND $\|\boldsymbol{\epsilon}_{i,j}\| > 0$ **do**
        $\dot{\mathbf{q}}_i \leftarrow \text{QP}(\mathbf{J}_{i,j}, \boldsymbol{\epsilon}_{i\text{-}1,j}, \mathbf{A}_i, \mathbf{b}_{i\text{-}1,j}, \mathbf{C}_{i\text{-}1,j}, \mathbf{d}_{i\text{-}1,j})$
        $\boldsymbol{\epsilon}_{i,j} \leftarrow \boldsymbol{\epsilon}_{i\text{-}1,j} - \mathbf{J}_i \dot{\mathbf{q}}_i$
        $\mathbf{b}_{i,j} \leftarrow \mathbf{b}_{i\text{-}1,j} - \mathbf{A}_i \dot{\mathbf{q}}_i$
        $\mathbf{C}_{i,j} \leftarrow \begin{bmatrix} \mathbf{C}_{i,j\text{-}1} & \mathbf{J}_{i,j} \end{bmatrix}^T$
        $\mathbf{d}_{i,j} \leftarrow \begin{bmatrix} \mathbf{d}_{i,j\text{-}1} & \mathbf{J}_{i,j} \dot{\mathbf{q}}_i \end{bmatrix}^T$
        $i \leftarrow i + 1$
      end while
    end for
  end procedure

Let us now consider N_{task} tasks to be solved simultaneously, going from priority 1 to N_{task}. Our inverse kinematics approach allows to set up a double-layer hierarchical approach: the first one being at the task level and the second one at the joint group level. The velocity vector command $\dot{\mathbf{q}}_{i,j}$ for the joint of priority i ($0 \leq i \leq N_{\text{group}}$) regarding the task of priority j ($0 \leq j \leq N_{\text{task}}$) is computed as follows:

$$\begin{aligned} \dot{\mathbf{q}}_{i,j} \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_{i,j}\dot{\mathbf{q}} - \boldsymbol{\epsilon}_{i\text{-}1,j}\|_2 \\ \text{s.t.} \quad & \mathbf{A}_i\dot{\mathbf{q}} \leq \mathbf{b}_{i\text{-}1}, \\ & \mathbf{J}_{i,j\text{-}1}\dot{\mathbf{q}} = \mathbf{J}_{i,j\text{-}1}\dot{\mathbf{q}}_{i,j\text{-}1}, \\ & \ldots, \\ & \mathbf{J}_{i,1}\dot{\mathbf{q}} = \mathbf{J}_{i,1}\dot{\mathbf{q}}_{i,1}, \end{aligned} \tag{3.16}$$

where $\mathbf{J}_{i,j} \in \mathbb{R}^{6\times N_{\text{dof}(i)}}$ is the Jacobian matrix associated with task j and for which only the columns related to the joint group i have been kept, i.e. $\mathbf{J}_{i,j} = \frac{\partial \dot{\mathbf{q}}_i}{\partial \dot{\mathbf{x}}_j}$ and $\boldsymbol{\epsilon}_{i,j}$ is the residual error remaining on task j after solving the problem with the joint group i:

$$\boldsymbol{\epsilon}_{i,j} = \dot{\mathbf{x}}_j - \sum_{k=1}^{i} \mathbf{J}_{k,j}\dot{\mathbf{q}}_k. \tag{3.17}$$

Thus, using the double-layer hierarchical structure, the final joint velocity command sent to the joint group i is obtained after solving the lowest priority task. The corresponding vector is given by $\dot{\mathbf{q}}_{i,N_{\text{task}}}$.

Note that if the task j can be satisfied after solving the problem with the joint group $i < N_{\text{group}}$, it is useless to continue the resolution procedure for the successive groups regarding this task as it will have no effect. Indeed, the computation of the joint velocity vector for the group $i < k \leq N_{\text{group}}$ will necessarily end up to $\dot{\mathbf{q}}_{k,j} = 0^{N_{\text{dof}(i)}}$ for $j = 1$ or $\dot{\mathbf{q}}_{k,j} = \dot{\mathbf{q}}_{k,j\text{-}1}$ for $j > 1$.

The complete hierarchical inverse kinematics process is described in Algorithm (2). The QP optimization process is referred to as the function

$$\dot{\mathbf{q}} = \text{QP}(\mathbf{J}, \dot{\mathbf{x}}, \mathbf{A}, \mathbf{b}, \mathbf{C}, \mathbf{d}), \tag{3.18}$$

for which the parameters correspond to the variables defined in Eq. (1.10).

It must be pointed out that the input data $\mathbf{J}$ and $\dot{\mathbf{x}}$ in Algorithm (2) are containers vectors of Jacobian matrices and task space command vectors, respectively, such as $\mathbf{J} = \begin{bmatrix}\mathbf{J}_1 & \ldots & \mathbf{J}_{N_{\text{task}}}\end{bmatrix}^T$ and $\dot{\mathbf{x}} = \begin{bmatrix}\dot{\mathbf{x}}_1 & \ldots & \dot{\mathbf{x}}_{N_{\text{task}}}\end{bmatrix}^T$, each element being associated with the task of corresponding priority. Similarly, the output data $\dot{\mathbf{q}}$ is the container vector of the joint velocity vector for each joint group, such as $\dot{\mathbf{q}} = \begin{bmatrix}\dot{\mathbf{q}}_1 & \ldots & \dot{\mathbf{q}}_{N_{\text{group}}}\end{bmatrix}^T$. When two subscripts are used, e.g $\mathbf{J}_{i,j}$, this means that only the components belonging to a specific joint are extracted from the variable. In this example, only the elements associated with joint group i are extracted from the matrix $\mathbf{J}_j$ related to task j.

When entering the double-layer hierarchical inverse kinematics process, the first step consists in precomputing the velocity vector for each joint group with the minimum norm allowed by the constraints. In other words, it is desired to reset every joint group to have zero velocity, but due to the deceleration constraint, this might not be possible. This operation is performed by the procedure *MinimizeJointVelocity* detailed in Algorithm (3).

This initialization process is required to ensure that the hierarchy between groups is respected. Indeed, the joint group of highest priority should solve the tasks first, with minimal contribution of other groups. This allows to slow down and then stop the joint groups that are no longer needed at some point.

Algorithm 3 Minimize joint velocity

Input: $\underline{\dot{Q}}, \overline{\dot{Q}}$
Output: $\dot{\mathbf{q}}$
  procedure MinimizeJointVelocity
    $\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_1 & \dots & \dot{\mathbf{q}}_{N_{\text{dof}}} \end{bmatrix}^T$
    for $i \leftarrow 1 : N_{\text{dof}}$ **do**
      if $\underline{\dot{Q}} > 0$ **then**
        $\dot{\mathbf{q}}_i \leftarrow \underline{\dot{Q}}$
      else if $\overline{\dot{Q}} < 0$ **then**
        $\dot{\mathbf{q}}_i \leftarrow \overline{\dot{Q}}$
      else
        $\dot{\mathbf{q}}_i \leftarrow 0$
      end if
    end for
    return $\dot{\mathbf{q}}$
  end procedure

As presented, the Double-layer Hierarchical Inverse Kinematics algorithm is fully generic as it can be applied to any type of robot performing any kind of task.

3.1.5.2 Application to mobile dual-arm cooperative task control

In this work, we will treat the case of a dual manipulator mounted on a steerable wheeled mobile robot. This corresponds to a non-holonomic omnidirectional robot, meaning that it requires initializing the steer joint configuration to perform arbitrarily three-dimensional trajectories in the plane of motion. In this chapter, the kinematic representation of the mobile platform is restricted to 3 DoF in Cartesian space. The specific management of the wheels for reactive control purposes is based on the motion discontinuity-robust controller for steerable mobile robots, introduced in (SCFP17).

The mobile robot is defined by two successive prismatic joints (motion along x, y in the world frame) followed by one revolute joint (rotation around z in the world frame) that are added at the root of the dual-arm robot kinematic chain.

From the cooperative task space perspective, the integration of the mobile base in the representation is straightforward. The extension of the kinematic chain has obviously no effect on the relative task since the motion generated by the wheels leads to identical displacements of the arms. However, it affects the absolute task: the 3

DoF added by the mobile base in the Cartesian space directly modify the pose of the absolute task frame, as shown in Fig. 3.5.

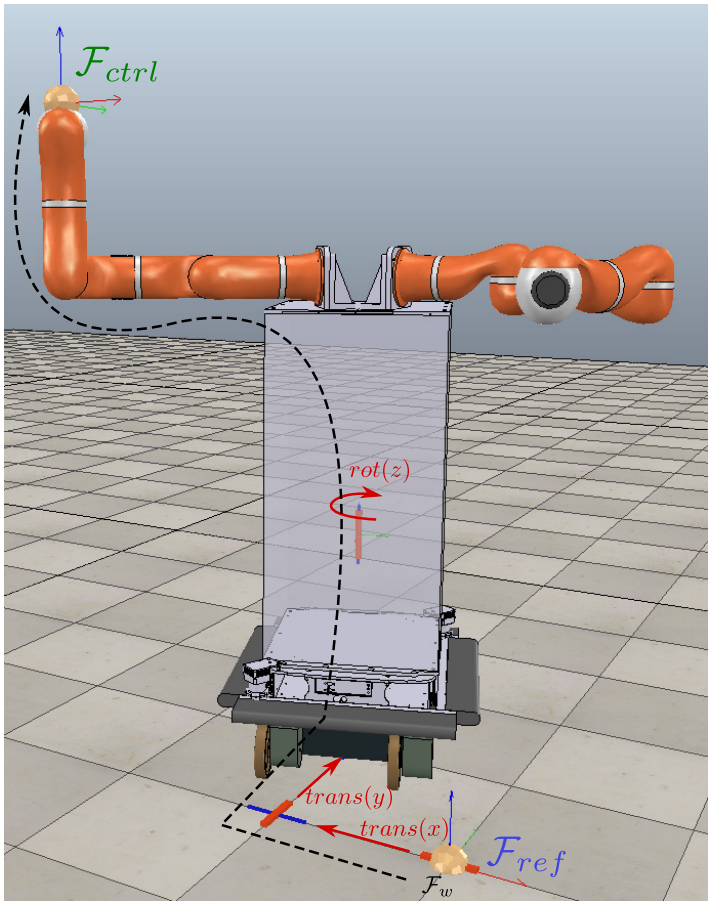


Figure 3.5 – Representation of the absolute task with a mobile base. Three Cartesian joints are added at the root of the mobile robot's kinematic structure and influence this task.

Taking into account the relatively low reactivity of the mobile part, we establish a relevant hierarchical control strategy: only the arms are engaged as long as they are sufficient to fulfill the task. Whenever some error $\boldsymbol{\epsilon}$ remains on the absolute task after solving Eq. (3.11) with the arms, the mobile base is activated to compensate it. This strategy is particularly interesting as it lets the arms operate locally and it switches to the wheels when the target exits the arms' reachable space.

The mobile base joint velocity vector $\dot{\mathbf{q}}_{mob}$ that follows this strategy is obtained by solving the optimization problem:

$$\begin{aligned} \dot{\mathbf{q}}_{mob} \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_{a,mob}\dot{\mathbf{q}} - \boldsymbol{\epsilon}_{a,arms}\|_2 \\ \text{s.t.} \quad & \mathbf{A}_{mob}\dot{\mathbf{q}} \leq \mathbf{b} - \mathbf{A}_{arms}\dot{\mathbf{q}}_{arms}, \end{aligned} \tag{3.19}$$

where $\dot{\mathbf{q}}_{arms}$, $\mathbf{A}_{arms}$ are the joint velocity vector and inequality linear coefficients, respectively, resulting from Eq. (3.11). The residual error on the absolute task $\boldsymbol{\epsilon}_{a,arms}$

is obtained after solving the inverse kinematics problem with the arms (see Eq. (3.1)). Subscripts mob/arms are used here to differentiate the joint groups related to the corresponding variable.

Note that the cost function and inequality constraint take into account the effect of the computed arms' joint velocities on the current task and obstacle avoidance constraint, respectively. In fact, task space constraints represented by the linear inequality are used to restrict the motion of some arbitrary Cartesian points in space. To make it possible, these points have to be controllable (i.e. affected by the robot movements). Yet, some of these points may be influenced by the motion of both the arms and the mobile base. In this case, since the arms' joint velocity command has already been computed at this control step, the anticipated effects on the motion of these points are used to update the constraint.

3.1.5.3 Simulations

Using the same simulated scenario as in Section 2.4.4, we validate our joint group prioritization strategy. We recall that the task specifications for this scenario are to move an object grasped with the two arms to a target frame located at a distance $d = 0.8$m on the world frame $x+$ axis (in front of the robot). This target location is unreachable with the arms only and requires the motion of the base.

Fig. 3.6 shows the role of each joint group in the task completion. We quantify the contribution γ_k of joint group k ($k \in [1;3]$) as a ratio of the overall generated velocities:

$$\gamma_k = \frac{\|\dot{\mathbf{q}}_k\|}{\|\dot{\mathbf{q}}_1\| + \|\dot{\mathbf{q}}_2\| + \|\dot{\mathbf{q}}_3\| + \epsilon_0}, \quad \text{with } \epsilon_0 = 0^+ \tag{3.20}$$

In a first stage (from time $t = 0$s to $t = 9$s), the use of the arms is sufficient to track the absolute task space trajectory without any error. During this period, the mobile base does not provide any contribution. Then, from $t = 9$s, the constraints brought by the relative task, to keep maintaining the object, do not allow to perfectly follow the trajectory generated for the absolute task frame. After solving the inverse kinematics optimization problem with the arms only, some residual error $\boldsymbol{\epsilon}_{a,arms}$ starts to appear and the cost function in (3.19) requires the mobile base actuation to be minimized. From then on, the contribution of the arms decreases until it becomes null at $t = 10.2$s. In the subsequent phase, the work becomes progressively balanced between the joint groups as the object approaches the spherical obstacle: the mobile base keeps going towards the final destination on the $x+$ axis while the arms are used again to bypass the object (see Fig. 2.10). The evolution of the residual error on the absolute task positions shown in Fig. 3.7 reveals that the mobile base is able to fully eliminate the remaining error on the x and y axes while the error on the z axis cannot be corrected.

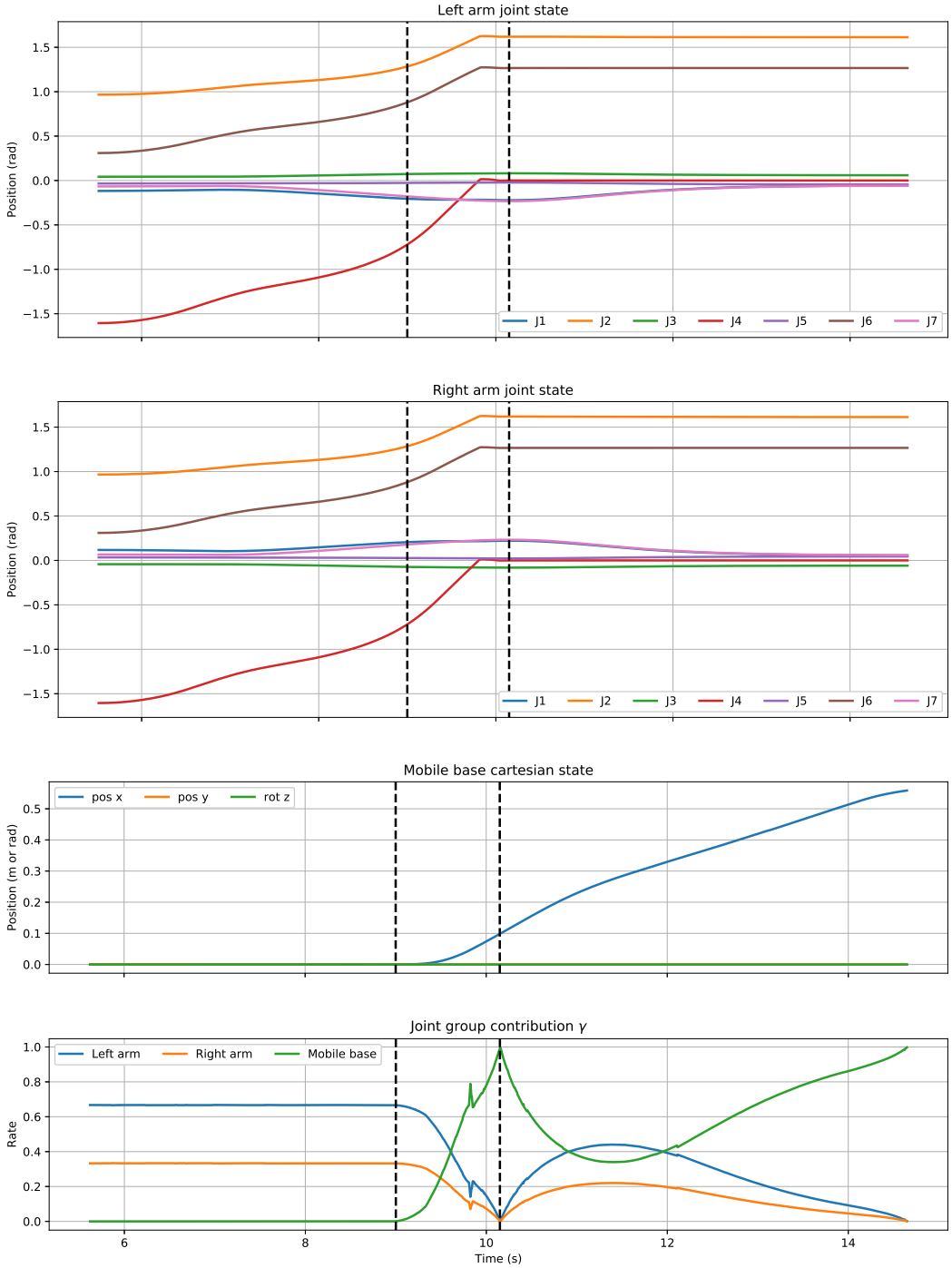


Figure 3.6 – Evolution of joint group states during the simulated experiment. Since the left and right arms have the highest priority in the task solving process, in the first stage they fully contribute to move the control frame towards the target. Once the workspace limits for the arms approach, the mobile base is activated to compensate for the tracking error on the trajectory. Arms and mobile base work jointly during the second phase of the experiment to achieve the task while avoiding an obstacle.

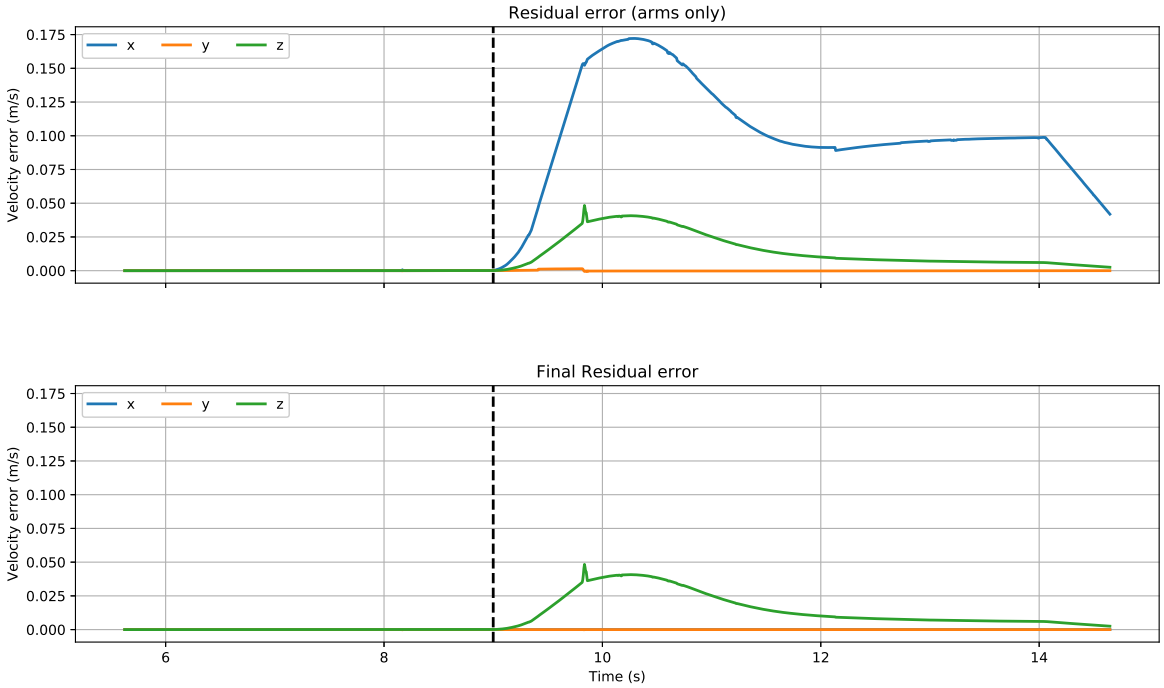


Figure 3.7 – Evolution of the residual error obtained in the absolute task positions during the simulated experiment. The top figure shows the residual error $\epsilon_{a,arms}$ after solving the inverse kinematics problem with the arms only, while the bottom figure is the final residual error after involving the base.

3.2 Hard constraints consideration

We distinguish two types of constraints: first, the physical limits of robots lead to the specification of admissible ranges of joint solutions. It applies at the position, velocity and acceleration level. Second, constraints also appear in the Cartesian space. To avoid hazardous motion in the workspace, one might set velocity and/or acceleration limits on some specific frames. Furthermore, obstacles of the environment can be considered under the form of collision avoidance constraints. In the following section, we detail each of these.

3.2.1 Joint limits

Let us define the following bounds on the joint positions, velocities and accelerations:

$$\begin{aligned} \underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}}, \\ \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{q}}}, \\ \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}} \leq \overline{\ddot{\mathbf{q}}}. \end{aligned} \tag{3.21}$$

Note that the acceleration constraint is not constant over the state of the robot as

it depends on the dynamic of the system. However, we use constant bounds because we intentionally do not integrate dynamic considerations in our control strategy.

Since the control variables are joint velocities, we want to express the position and acceleration constraints as velocity constraints.

Considering a discrete control loop at iteration k and of sampling time $T > 0$, assuming that the joint velocity and acceleration remain constant between two iterations, joint constraints that have to be satisfied at $k+1$ can be rewritten as:

$$\begin{aligned} \frac{\underline{\mathbf{q}} - \mathbf{q}_k}{T} \leq \dot{\mathbf{q}} \leq \frac{\overline{\mathbf{q}} - \mathbf{q}_k}{T}, \\ \underline{\dot{\mathbf{q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{q}}}, \\ \underline{\ddot{\mathbf{q}}} T + \dot{\mathbf{q}}_k \leq \dot{\mathbf{q}} \leq \overline{\ddot{\mathbf{q}}} T + \dot{\mathbf{q}}_k. \end{aligned} \tag{3.22}$$

An additional constraint is needed to ensure consistency between the position and acceleration limits. In fact, if one of the joints has to stop promptly due to position bound proximity, the acceleration constraint might be violated.

In (FDLK12), Flacco et al. showed that to avoid conflicting constraints, the following bounds should be used:

$$-\sqrt{2\underline{\ddot{\mathbf{q}}}(\underline{\mathbf{q}} - \mathbf{q}_k)} \leq \dot{\mathbf{q}} \leq \sqrt{2\overline{\ddot{\mathbf{q}}}(\overline{\mathbf{q}} - \mathbf{q}_k)}. \tag{3.23}$$

Contrary to (FDLK12), we decided not to replace the original acceleration constraint by Eq. (3.23) because in this case acceleration bounds are only active close to the limits, possibly leading to excessive torque generation in the middle of the range. However, the position constraint becomes unnecessary as Eq. (3.23) defines a subspace of it.

Hence, from Eq. (3.22) and Eq. (3.23), we keep the most restrictive condition for each joint and we define the final lower and upper bounds, respectively $\underline{\dot{\mathbf{Q}}}$ and $\overline{\dot{\mathbf{Q}}}$:

$$\begin{aligned} \underline{\dot{\mathbf{Q}}} &= \max\left\{-\sqrt{2\underline{\ddot{\mathbf{q}}}(\underline{\mathbf{q}} - \mathbf{q}_k)}\ ;\ \underline{\dot{\mathbf{q}}}\ ;\ \underline{\ddot{\mathbf{q}}} T + \dot{\mathbf{q}}_k\right\} \\ \overline{\dot{\mathbf{Q}}} &= \min\left\{\sqrt{2\overline{\ddot{\mathbf{q}}}(\overline{\mathbf{q}} - \mathbf{q}_k)}\ ;\ \overline{\dot{\mathbf{q}}}\ ;\ \overline{\ddot{\mathbf{q}}} T + \dot{\mathbf{q}}_k\right\}, \end{aligned} \tag{3.24}$$

so that Eq. (3.21) breaks down to:

$$\underline{\dot{\mathbf{Q}}} \leq \dot{\mathbf{q}} \leq \overline{\dot{\mathbf{Q}}}. \tag{3.25}$$

3.2.2 Task space limits

Setting bounds at the joint level is crucial for ensuring the safety of hardware components. However, the robot motion in the task space should also be monitored when

dealing with shared human-robot environments. Indeed, for high DoF robotic arms, the motion generated at the tip may exceed a tolerable safety threshold due to the cumulative contribution of every joint.

One initial action to alleviate this effect is to tackle the problem from the task space command either by saturating or by scaling the task space command velocity vectors to fit inside the delimited bounds. The problems with this method are:

1. by integrating the limits in the cost function of an optimization problem (see Section 3.1), there is no certainty that the computed solution fulfills the constraints.

2. considering the cooperative task space, setting the constraints on the absolute and relative control frames is probably not the best choice as it does not directly reflect the actual motion of each manipulator.

Instead, we propose to define velocity and acceleration constraints at the tip of the two arms:

$$\begin{aligned} \underline{\dot{\mathbf{x}}} \leq \dot{\mathbf{x}} \leq \overline{\dot{\mathbf{x}}}, \\ \underline{\ddot{\mathbf{x}}} \leq \ddot{\mathbf{x}} \leq \overline{\ddot{\mathbf{x}}}, \end{aligned} \tag{3.26}$$

where $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$ are the 6D twist and acceleration of one end-effector (the same process applies for the other arm), T the sampling period, $\underline{\dot{\mathbf{x}}}$, $\overline{\dot{\mathbf{x}}}$ the lower and upper velocity bounds and $\underline{\ddot{\mathbf{x}}}$, $\overline{\ddot{\mathbf{x}}}$ the lower and upper acceleration bounds.

From Eq. (1.6), we can express the twist and acceleration vectors with respect to the arm joint velocity $\dot{\mathbf{q}}$ and acceleration $\ddot{\mathbf{q}}$:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{J}\dot{\mathbf{q}}, \\ \ddot{\mathbf{x}} &= \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}, \end{aligned} \tag{3.27}$$

where $\mathbf{J}_k$ is the Jacobian matrix computed at the tip of the arm and $\dot{\mathbf{J}}$ its time derivative. Similar to the joint case, we want to express the task space bounds as joint velocity constraints. Using Eq. (3.27), Eq. (3.26) can be rewritten as:

$$\begin{aligned} \underline{\dot{\mathbf{x}}} \leq \mathbf{J}\dot{\mathbf{q}} \leq \overline{\dot{\mathbf{x}}}, \\ \underline{\ddot{\mathbf{x}}} \leq \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} \leq \overline{\ddot{\mathbf{x}}}, \end{aligned} \tag{3.28}$$

Considering a discrete control loop at iteration k and of sampling time $T > 0$, assuming that the joint acceleration remains constant between two iterations, the task acceleration constraint that have to be satisfied at $k+1$ can be rewritten as:

$$\underline{\ddot{\mathbf{x}}} \leq \frac{1}{T}\mathbf{J}(\dot{\mathbf{q}} - \dot{\mathbf{q}}_k) + \dot{\mathbf{J}}\dot{\mathbf{q}} \leq \overline{\ddot{\mathbf{x}}}, \tag{3.29}$$

Finally, by reorganizing Eq. (3.29), the task space acceleration constraint is equivalent to the following joint velocity constraint:

$$\underline{\ddot{\mathbf{x}}} + \frac{1}{T}\mathbf{J}\dot{\mathbf{q}}_k \leq \left(\frac{1}{T}\mathbf{J} + \dot{\mathbf{J}}\right)\dot{\mathbf{q}} \leq \overline{\ddot{\mathbf{x}}} + \frac{1}{T}\mathbf{J}\dot{\mathbf{q}}_k, \tag{3.30}$$

3.2.3 Collision avoidance hard constraints

For a dual-arm system operating in dynamic cluttered environments, both collision with objects of the world and self-collision may occur. Many research activities have focused on tackling the collision avoidance issue at the planning level (LaV06). Resulting algorithms aim at finding a collision-free path to join a final state from an initial one. Although effective, these approaches are time-consuming. It makes them unsuitable for changing environments that require online computation.

A real-time obstacle avoidance method has been introduced by Khatib in (Kha86). The concept of artificial potential fields is presented in this paper: these are virtual repulsive forces emanating from obstacles and input to the task space command, to remain far from obstacles.

Another local approach has been proposed in (FT87). The main difference with the artificial potential field method is that collision avoidance constraints are separated from the task description. This way, it is possible to precisely control the different aspects of the problem. The obstacle avoidance strategy is expressed as linear constraints using Velocity Damper. This concept has recently been applied to dual-arm manipulators performing relative tasks (SD18). Avoiding Collisions with objects of the workspace and self-collisions is possible by defining constraints in an optimization framework.

In our work, we extend the method of Velocity damper to the general case of kinematic chains having a tree structure. Thus, it can be applied to standard dual-arm robots, but also to the ones equipped with joints on the torso, a mobile base, and so on.

Let us consider one collision evaluation (either with an obstacle of the world or between the robot's parts). The current distance d that separates the bodies is measured between the critical points $\mathbf{p_1}$ and $\mathbf{p_2}$.

We recall the definition of the Velocity Damper $\dot{d}^*$:

$$\dot{d}^* = -\xi\frac{d - d_s}{d_i - d_s}, \quad d \leq d_i, \tag{3.31}$$

with d_i the influence distance from which the constraint is activated, d_s the safety distance that must separate two bodies at any time, d the current distance between the bodies, and ξ a positive gain for tuning the convergence speed.

The derived inequality constraint is simply given by:

$$\dot{d} \geq \dot{d}^*, \tag{3.32}$$

with $\dot{d}$ being the current velocity between the bodies.

In concrete terms, when two bodies are separated by a distance smaller than d_i, their relative speed becomes bounded. If they keep getting closer, the motion is slowed down until the safety distance is reached. Then, when $d = d_s$ the constraint Eq. (3.32) becomes $\dot{d} \geq 0$ meaning that the only admissible motion between the bodies is one that maintains or increases their relative distance.

To apply such method to a robot, the relative velocity $\dot{d}$ should be controllable. The generic expression to formulate it in function of of joint velocities is:

$$\dot{d} = \mathbf{n}^T \mathbf{J}_p \dot{\mathbf{q}}_d, \tag{3.33}$$

$\mathbf{n}$ being the normalized vector joining the critical points $\mathbf{p_1}$ and $\mathbf{p_2}$ on the objects, as illustrated in Fig. 3.8, such as:

$$\mathbf{n} = \frac{\mathbf{p_1} - \mathbf{p_2}}{\|\mathbf{p_1} - \mathbf{p_2}\|}, \tag{3.34}$$

The Jacobian matrix $\mathbf{J}_p \in \mathbb{R}^{3\times N_s}$ is computed at the point $\mathbf{p_1}$ and takes as reference the point $\mathbf{p_2}$. It takes only the position part of the Jacobian. It includes all the N_s joints playing a role in the variation of the distance between these two points. These also correspond to the ones selected in the joint velocity vector $\dot{\mathbf{q}}_d$.

From Eq. (3.33) can be derived the Jacobian matrix $\mathbf{J}_d \in \mathbb{R}^{1\times N_s}$ expressing the influence of the selected joints on the distance variation, such as $\mathbf{J}_d = \mathbf{n}^T \mathbf{J}_p$, leading to:

$$\dot{d} = \mathbf{J}_d \dot{\mathbf{q}}_d. \tag{3.35}$$

To illustrate this, we present two different types of collision avoidance evaluations for which the computation of $\dot{d}$ and $\dot{d}^*$ is detailed.

To do so, let us consider a mobile dual-arm robot equipped with two n-DoF arms, and for which m-DoF are added by the mobile base.

We denote as $\mathbf{q}_b = \begin{bmatrix} q_b(1) & \dots & q_b(m) \end{bmatrix}^T$, $\mathbf{q}_1 = \begin{bmatrix} q_1(1) & \dots & q_1(n) \end{bmatrix}^T$, and $\mathbf{q}_2 = \begin{bmatrix} q_2(1) & \dots & q_2(n) \end{bmatrix}^T$ the vector of joint position for the mobile base, the first arm (*Arm 1*), and the second arm (*Arm 2*), respectively.

Collision with an obstacle: the first kind of collision that may happen involves a robot part and an obstacle of the world. An example is given in Fig. 3.8a in which one robot's arm is close to a table.

It is assumed that the point $\mathbf{p_1}$ belongs to the link from *Arm 1* preceded by joint i ($1 \leq i \leq n$). The point $\mathbf{p_2}$ is located on the surface of the table.

In this case, all the joints from the mobile base and the first i joints from *Arm 1* influence the distance d between the objects.

The joint velocity vector $\dot{\mathbf{q}}_d$ is thus define as :

$$\dot{\mathbf{q}}_d = \begin{bmatrix} \dot{q}_b(1) & \dots & \dot{q}_b(m) & \dot{q}_1(1) & \dots & \dot{q}_1(i) \end{bmatrix}^T, \tag{3.36}$$

and the Jacobian matrix $\mathbf{J}_p$ as :

$$\mathbf{J}_p = \begin{bmatrix} \mathbf{J}_b(1) & \dots & \mathbf{J}_b(m) & \mathbf{J}_1(1) & \dots & \mathbf{J}_1(i) \end{bmatrix}^T, \tag{3.37}$$

where $\mathbf{J}_b(x)$ $(1 \leq x \leq m)$ and $\mathbf{J}_1(y)$ $(1 \leq y \leq n)$ are 3D vectors expressing the influence of joint x from the mobile base (resp. joint y from *Arm 1*) on the position of point $\mathbf{p_1}$. The Jacobians are expressed with respect to a fixed world frame.

Self-collision: other than hitting an obstacle from the environment, there is also a risk that the robot collides with itself. In particular, collision between the arms is likely to occur. An example of self-collision evaluation is given in Fig. 3.8b, in which the critical points are located on the wrist of each arm.

It is assumed that the point $\mathbf{p_1}$ belongs to the link from *Arm 1* preceded by joint i $(1 \leq i \leq n)$ and the point $\mathbf{p_2}$ belongs to the link from *Arm 2* preceded by joint j $(1 \leq j \leq n)$.

In this case, the first i joints from *Arm 1* and the first j joints from *Arm 2* influence the distance d between the objects.

Note that the mobile base does not have any effect on the variation of the distance and can be omitted for self-collision evaluations.

In this example, the joint velocity vector $\dot{\mathbf{q}}_d$ is define as :

$$\dot{\mathbf{q}}_d = \begin{bmatrix} \dot{q}_2(j) & \dots & \dot{q}_2(1) & \dot{q}_1(1) & \dots & \dot{q}_1(i) \end{bmatrix}^T, \tag{3.38}$$

and the Jacobian matrix $\mathbf{J}_p$ as :

$$\mathbf{J}_p = \begin{bmatrix} \mathbf{J}_2(j) & \dots & \mathbf{J}_2(1) & \mathbf{J}_1(1) & \dots & \mathbf{J}_1(i) \end{bmatrix}^T, \tag{3.39}$$

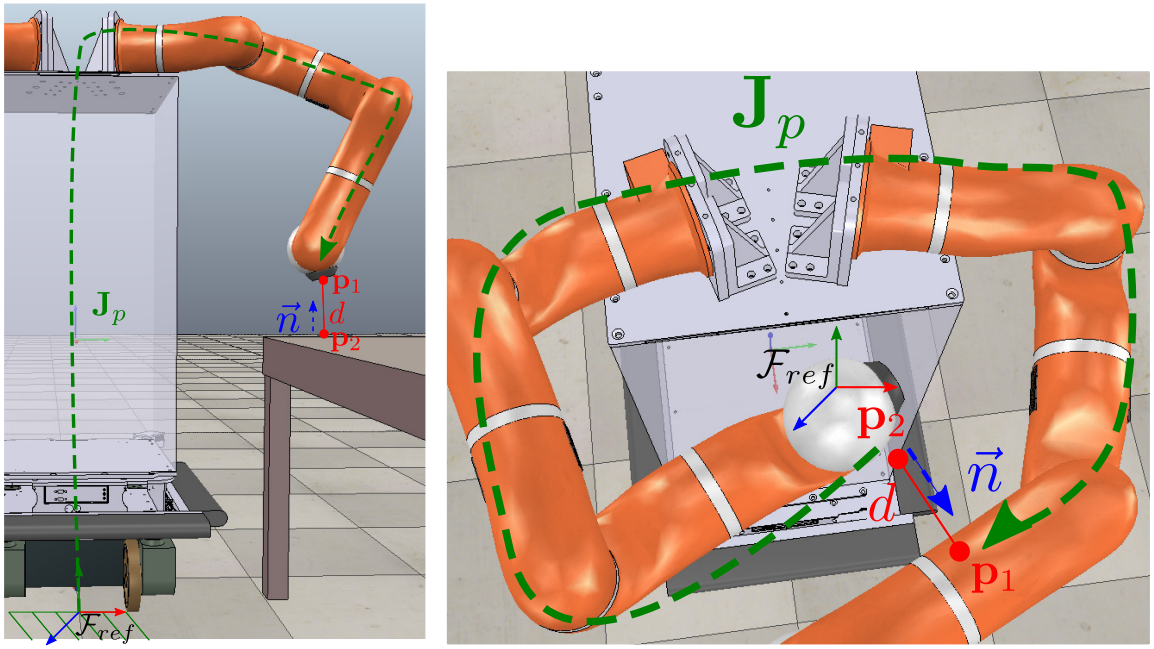
where $\mathbf{J}_1(x)$ $(1 \leq x \leq n)$ and $\mathbf{J}_2(y)$ $(1 \leq y \leq n)$ are 3D vectors expressing the influence of joint x from *Arm 1* (resp. joint y from *Arm 2*) on the position of point $\mathbf{p_1}$ with respect to a reference frame $\mathcal{F}_{ref}$ arbitrarily attached to the link holding point $\mathbf{p_2}$. Details on how to compute $\mathbf{J}_p$ are given in Appendix A.

In this case, vector $\mathbf{n}$ defined in Eq. (3.34) should also be expressed with respect to $\mathcal{F}_{ref}$ for consistency.

At each time step, a collision avoidance constraint is generated for all pairs of potentially colliding parts (including collisions with an obstacle and self-collisions) which are separated by a distance $d \leq d_i$. Referring to Eq. (3.32), Eq. (3.35), we can specify a constraint in function of joint velocities under the standard form of linear inequality equations:

$$\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \tag{3.40}$$

with $\mathbf{A} = -\mathbf{J}_d$ and $\mathbf{b} = -\dot{d}^*$.



(a) Collision with an object of the workspace

(b) Self-collision between the arms

Figure 3.8 – Representation of collision avoidance evaluations. On the left, the robot's arm is close to a table while on the right self-collision is assessed. Points $\mathbf{p_1}$ *and* $\mathbf{p_2}$ *are the critical points defining the minimal distance* d *between the parts. The Jacobian matrix* $\mathbf{J}_d$ *reveals the influence of the joints involved on the distance variation.*

3.3 Conclusion

We proposed in this chapter to tackle the inverse kinematics problem for a dual-arm robot performing cooperative tasks. We suggested different methods for solving the inverse kinematics. All are based on QP optimization to find solutions that satisfy the constrained listed above. First, we proposed to gather the resolution of the two cooperative tasks in one optimization problem.

Because relative motions are generally more critical than absolute ones, we developed a HQP strategy that gives priority to the relative task. After introducing the notion of parsimony, we integrated an additional task which aims at tuning the level of sparsity that we want the robot to adopt. When the robot is highly redundant with respect to the task, we showed how this parameter influences the number of activated joints.

We then tackled the issue of using additional robots to extend the dual-arm platform. Because they usually behave differently than the arms, we believe that they should not be used the same way. Instead, we proposed to classify the extra joints brought by the additive robot in a second joint group and adopt a hierarchical strategy at the joint group level for the task-solving process: one can choose to solve the tasks with a selected set of joints and only use secondary joints when necessary. This method is especially interesting for controlling dual-arm robots endowed with mobile capabilities as it lets the arms operate locally and activates the wheeled system for large motions only.

To ensure the proper and safe motion of the joints, we stated a set of hard constraints that defines the solution space. First, position, velocity and acceleration limits of the joints were combined to determine the velocity bounds. This allows to cope with the physical capabilities of the hardware components. Then, constraints were specified at the task level to provide safe motions of the tools and prevent hazardous situations for human operators working around the robot. We strengthened the safety aspect by adding constraints for collision avoidance. Using a velocity damper, the constraint forces the robot to slow down then stop when getting too close from an obstacle. This applies to avoiding collisions with an obstacle of the workspace as well as self-collision between robot parts.

The next chapter will be dedicated to the software developments carried out during this thesis. They have led to the creation of a complete software library intended for the kinematic control of redundant robots.

CHAPTER 4

Kinematic control framework: RKCL library development

In this chapter, we present the software developments implementing the complete online kinematic controller for dual-arm collaborative tasks in unstructured industrial environments.

Let us first summarize how the different elements seen so far are combined together to make the whole control process work, by referring to the closed-loop control scheme with input/output data depicted in Fig. 4.1.

In Chapter 2, we explained how the information is converted and then exploited in the cooperative task space. Joint position ${}^{m}\mathbf{q}$ coming from the robot encoders is processed by the *Forward Kinematics* block to get the poses ${}^{m}\mathbf{x}_a$, ${}^{m}\mathbf{x}_r$ and Jacobian matrices ${}^{m}\mathbf{J}_a$, ${}^{m}\mathbf{J}_r$ in the cooperative space. Wrench feedback ${}^{m}\mathbf{W}_1$, ${}^{m}\mathbf{W}_2$ coming from F/T sensors mounted at the arms' wrist are transformed by the *Wrench Adapter* block into cooperative task space vectors ${}^{m}\mathbf{W}_a$, ${}^{m}\mathbf{W}_r$. As seen in Section 2.2, additional treatment can be applied on the wrench when dealing with human-robot comanipulation, to enhance the quality of the interaction. Once all the feedback data has been interpreted in the task space, it is send to the *Task Space Command* block. This block also receives the feedforward task space trajectory consisting of the absolute $\mathbf{x}_a^*$ and relative $\mathbf{x}_r^*$ desired poses as well as the absolute $\dot{\mathbf{x}}_a^*$ and relative $\dot{\mathbf{x}}_r^*$ desired velocities. Then, it generates the task space command ${}^{c}\dot{\mathbf{x}}_a$ and ${}^{c}\dot{\mathbf{x}}_r$ as detailed in Section 2.3.

Chapter 3 was dedicated to the generation of joint motion command ${}^{c}\dot{\mathbf{q}}$ from the task space command ${}^{c}\dot{\mathbf{x}}_a$ and ${}^{c}\dot{\mathbf{x}}_r$, taking into account the admissible range on the joint velocities, bounded by $\underline{\dot{\mathbf{Q}}}$ and $\overline{\dot{\mathbf{Q}}}$, and the collision avoidance inequality constraint defined by $\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}$.

Our control framework is intended to be generic in the sense that the method can be applied to every type of dual-arm robots: the two arms might be different and with any

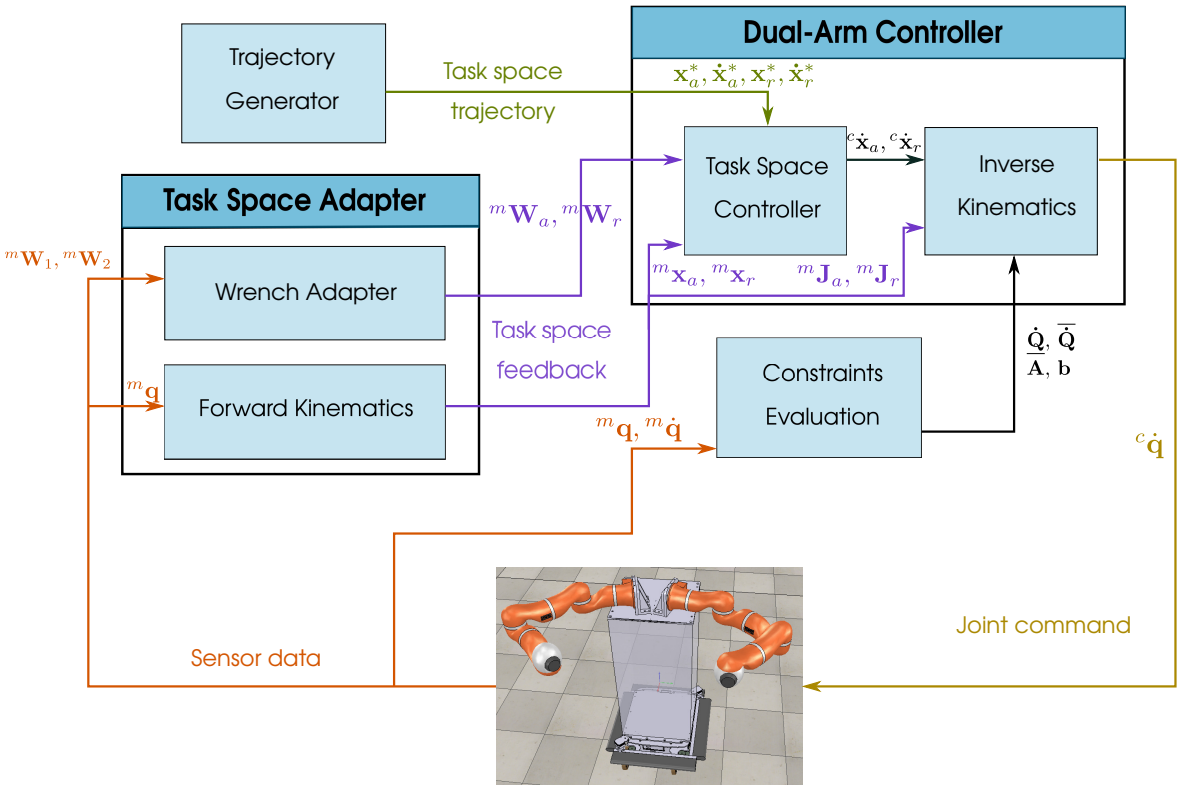


Figure 4.1 – Dual-arm kinematic control scheme with transmitted variables. The absolute (subscript 'a') and relative (subscript 'r') tasks are controlled. The Task Space Adapter converts measurements (superscipt 'm') of joint positions ${}^m\mathbf{q}$ and wrench ${}^m\mathbf{W}_1$, ${}^m\mathbf{W}_2$ at the wrist of each arm into task space poses ${}^m\mathbf{x}_a$, ${}^m\mathbf{x}_r$, wrenches ${}^m\mathbf{W}_a$, ${}^m\mathbf{W}_r$ and extracts task space Jacobian matrices ${}^m\mathbf{J}_a$, ${}^m\mathbf{J}_r$. The Dual-arm Controller then generates the command (superscipt 'c') at the task space level ${}^c\dot{\mathbf{x}}_a$, ${}^c\dot{\mathbf{x}}_r$ from these feedback values and the desired (superscript '') task space trajectory. Finally, the inverse kinematics provides the joint velocity command ${}^c\dot{\mathbf{q}}$ sent to the robot.*

number of DoF (note however that the system will perform properly only if an overall minimum number of DoF is respected). Besides the two arms, dual-arm platforms may be extended with additional joints, usually provided by the use of a mobile base or an articulated torso. In Section 3.1.5, we discussed the approach we have developed to incorporate extra DoF in the kinematic model of the robot and the control strategy we have adopted to make adequate use of them.

4.1 Software developments in RKCL

To perform dual-arm operations on real and simulated environments, a dedicated kinematic controller library called RKCL has been developed during this thesis. Originally specific to dual-arm robots performing cooperative tasks, a complete refactoring of the library has been done to be more flexible regarding the type of robots which can be controlled. Indeed, as we discussed in the previous sections, dual-arm robotic platforms might be made of several components extending the two manipulators, such as mobile base or articulated torsos. In fact, there are plenty of combinations that can be used to compose those robots. At some point, we realized that it was not appropriate to deal with particular cases but rather to adopt a generic approach that covers every system. The common thread between all those platforms lies in their kinematic representation: the kinematic chain created by the joints' path form a tree structure with at least two branches (the arms) and with a shared root (the world frame). In that respect, we designed the library to be suitable for any robot with a kinematic tree structure, dual-arm robots being one particular case.

4.1.1 RKCL main concepts

The philosophy behind RKCL is to provide an extensible library that can easily integrate additional features and additional robots over time. To do so, we decided to organize the project as a framework of packages, each of them relying on a core component. The architecture of the library is depicted in Fig. 4.2.

Most of the library is freely available online under the GNU LGPL license [1]. Only some specific packages are the property of Tecnalia and cannot be publicly accessed. The library is written in C++ to maximize application performance and flexibility by taking advantage of object-oriented design benefits.

The PID[2] projects development methodology is used to manage packages. Based on CMake and Git, it provides:

- Common and formal structure for projects.
- Uniform project life-cycle.

[1] https://www.gnu.org/licenses/lgpl-3.0.en.html
[2] http://pid.lirmm.net/pid-framework/

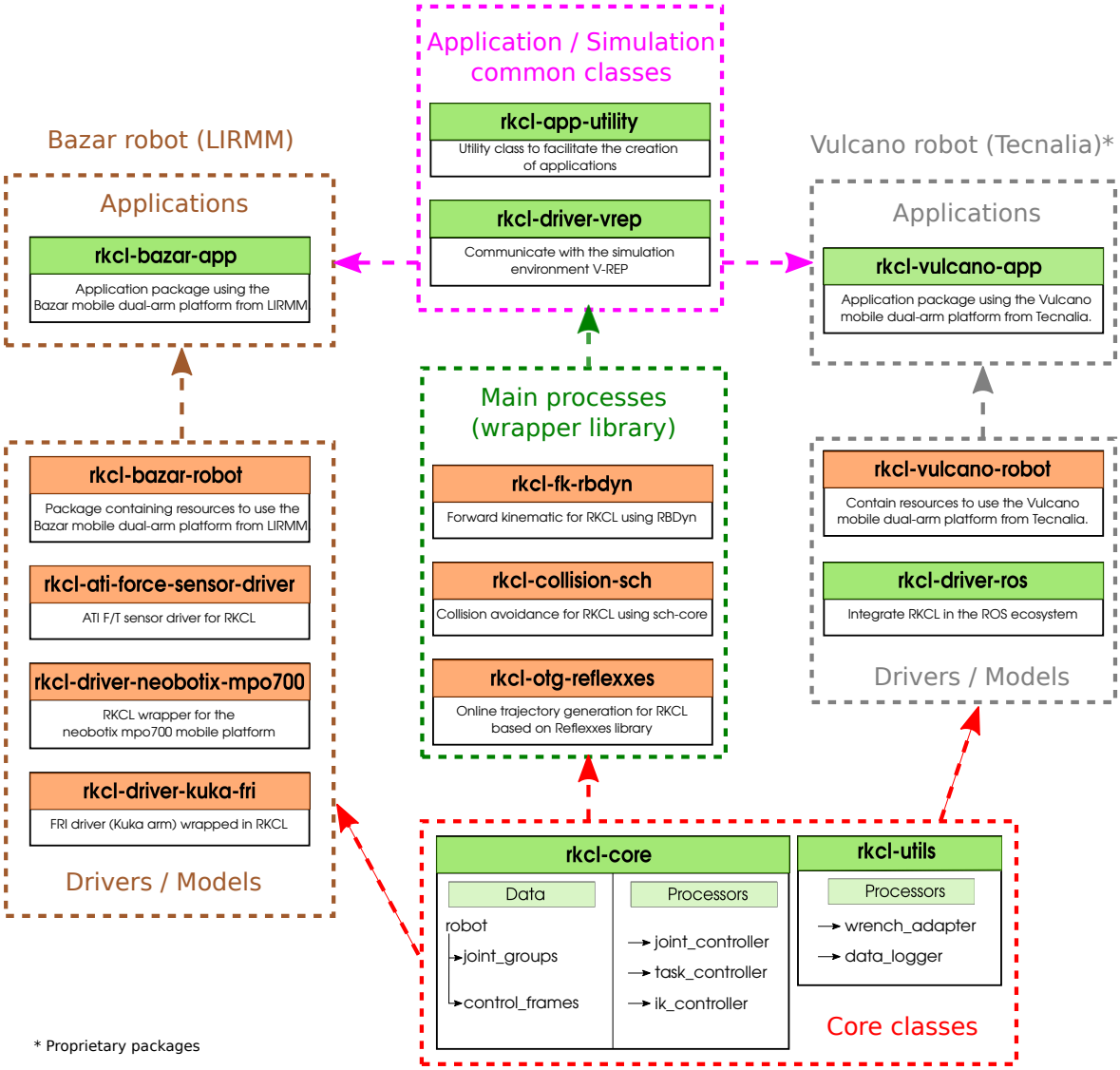


Figure 4.2 – Package architecture of the RKCL framework. Packages in green have been developed from scratch and are part of our contributions while packages in orange are based on external dependencies with integration in RKCL. The core classes contain the elementary data and processors to run the control loop. External libraries are also wrapped in RKCL to provide the main features (forward kinematics, trajectory generation, collision avoidance) to the controller. Several packages have been created for each dual-arm platform (BAZAR and Vulcano) gathering the robots' models but also the drivers for input/output data transmission. Applications packages for these platforms are used to perform different scenarios through simulation and real experiments. Dashed arrows represent direct dependencies between components (the targeted one depending on the other).

- Automating build/deployment/publishing processes.

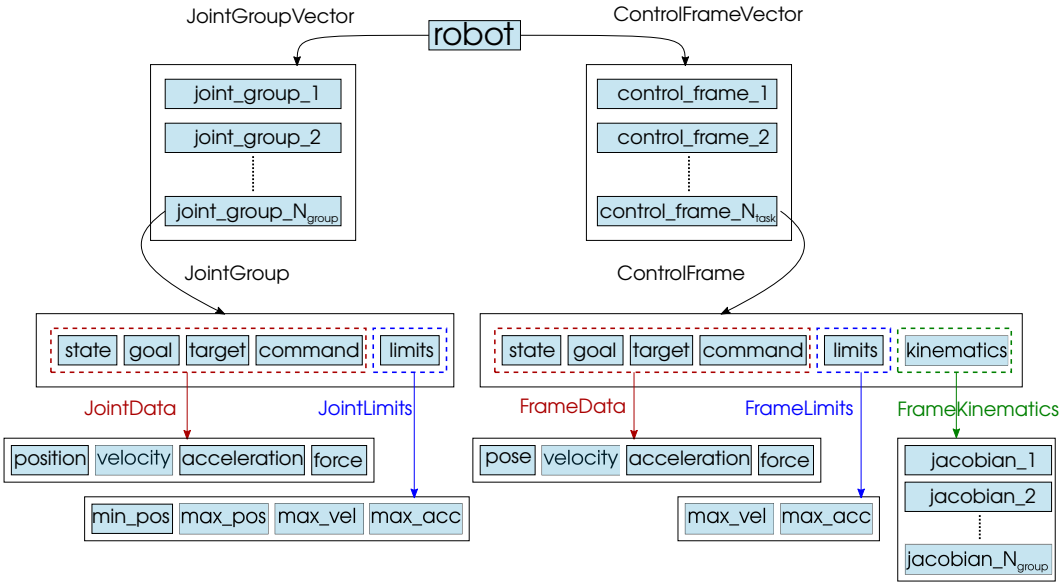


Figure 4.3 – RKCL data tree structure. At the root, the robot consists of a variable number of joint_group and control_frame, each of which contains the variables used by the controller.

The *rkcl-core* package is the foundation of the kinematic control library. It contains the fundamental data and processor classes used in the control loop, as listed in Fig. 4.2. In RKCL, the core object is a *robot*. It gathers the essential data needed to execute the control loop. It is made of the tree data structure shown in Fig. 4.3.

The *robot* component is initialized with a description file that provides the model of the robot. Then, it is possible to classify the joints extracted from the kinematic model into several *joint_group*. This categorization allows assigning a priority to each *joint_group* to perform the double-layer hierarchical inverse kinematics process presented in the previous section. The classification is arbitrary and is left free to the user. For instance, the two arms might be treated as a unique *joint_group* if they are identical but could also be split into two groups if they have different properties (e.g. control time step) or if it is desired to favor the use of one arm over the other.

A *control_frame* object is added to the *robot* each time a task is defined. It holds the information related to the current state of control. It also carries the limits of the control frame in the Cartesian space and the Jacobian matrices for every *joint_group*.

Referring to Fig. 4.2, the basics processors provided by the *rkcl-core* package are the *joint_controller*, the *task_controller* and the *ik_controller*. The *joint_controller* is used in a dedicated control loop to reach a specific joint configuration for the robot (e.g. at initialization), given the joint position constraints and maximum admitted velocity/acceleration. Complementary but not necessary features are implemented in the *rkcl-utils* library that is also part of the core package. In particular, the *wrench_adapter* process, whose actions are depicted in Fig. 2.4, is required only for cooperative tasks with wrench control.

External open-source libraries have also been wrapped in RKCL providing the main features to the controller:

- Forward kinematics adapted from the RBDyn library[3] which provides a set of class and function to model the dynamics of rigid body systems. The implementation is based on Roy Featherstone Rigid Body Dynamics Algorithms book and other state of the art publications.
- Online trajectory generation based on Reflexxes[4] (Krö11).
- Collision avoidance evaluating distance with sch-core[5], an efficient implementation of GJK algorithm for proximity queries between convex shapes. Collision objects are represented with sphere or superellipsoid (Bar81) geometric shapes. The former requires less computational efforts to perform proximity queries but the later gives better approximations of arbitrary objects.

To facilitate the creation of applications, we have also developed the *app_utility* class. It allows us to manage efficiently the different processors in the control loop and offers a quick and easy way to configure a new application.

An interface for the robotics simulator V-REP[6] is available in RKCL. Thus, the user can choose between launching an application on a real platform or on simulation.

4.1.2 Synchronization issues

The generic kinematic control approach we present in this chapter, allowing to deal with multiple robots in a unique control loop, leads to synchronization issues. Indeed, different types of machines with varying control time steps may constitute the whole robotic platform. Also, it is often not possible to use a common synchronization signal between the robots, which means that they usually operate at different times. There is no general way of solving this problem as communication aspects depend on robot manufacturers.

Instead, in keeping with the focus of making RKCL a generic and easy-to-use solution for kinematic control of various robots, we accepted the fact that robots may be unsynchronized and decided to deal with this in such a way that it does not noticeably affect the performances of the system.

In this regard, we propose to separate the execution of communication drivers from the controller by adopting a parallelization strategy. The case of BAZAR robot, equipped with two arms, a mobile base, and two force sensors at the arms' wrist is illustrated in Fig. 4.5.

[3] https://github.com/jrl-umi3218/RBDyn
[4] http://www.reflexxes.ws/
[5] https://github.com/jrl-umi3218/sch-core
[6] http://www.coppeliarobotics.com/

The ecosystem is composed of a unique kinematic control loop that executes the different processors sequentially and several drivers in charge of exchanging data between the hardware components and the controller. Each thread runs independently and at varying frequency: the drivers are triggered by the reception of new data coming from sensors, meaning that the time rate is fixed by the associated component.

The setting of the time step for the kinematic control loop, however, is arbitrary and left free to the user. Nevertheless, this choice will strongly impact the accuracy by influencing the tracking error of the task. The best solution is to align the control loop time step value with the highest frequency driver. Indeed, increasing the frequency has no positive effect and just induces unnecessary computations. Decreasing it degrades the performance of the controller for two reasons:

1. There is an interest in executing a new control step as soon as new input data are available. If at least one of the drivers is updated more often than the kinematic control loop, this means that some input information might not be treated by the controller which results in suboptimal behavior.

2. The joint limits constraints addressed in Section 3.2 take into account the position boundaries and maximum admissible acceleration/deceleration under the form of velocity bounds. However, this formulation is no longer valid when joint drivers are not synchronized with the controller.

In fact, the synchronization issue prevents from precisely managing the joint constraints and impels us to adopt a conservative approach. To do so, let us consider the simple case of a unique joint driver thread which time step is T_{joint} and the kinematic control loop running every T_{task}. The joint driver process takes Δt_{joint} to be completed while the control loop computation time is Δt_{task}. We assume that input data are read at the beginning of a process and changes become effective once it is ended. The length of time during which a joint command is being executed is denoted by $\Delta t_{\dot{\mathbf{q}}}$.

As depicted in Fig. 4.4 the command computed at time t by the kinematic controller is executed during $T_{joint}+T_{task}$ in the worst case, whereas it was supposed to be effective for T_{joint} only. In fact, this is the control loop that induces modifications in the joint command. Hence, the span $\Delta t_{\dot{\mathbf{q}}}$ is lower bounded by $\Delta t^{-}_{\dot{\mathbf{q}}}$ as follows:

$$\Delta t^{-}_{\dot{\mathbf{q}}} = \begin{cases} T_{task} & \text{if } T_{joint} \leq T_{task}, \\ T_{joint} & \text{otherwise.} \end{cases} \tag{4.1}$$

Depending on the synchronization between the control thread and the driver thread, some additional delay may arise and the command might be extended even longer than expected. In the worst case, the command computed by the control is made available just after the driver loop has begun, so the old data is considered. In this situation, a delay of T_{joint} is added to the system. This allows to define the upper bound $\Delta t^{+}_{\dot{\mathbf{q}}}$ of $\Delta t_{\dot{\mathbf{q}}}$:

$$\Delta t^{+}_{\dot{\mathbf{q}}} = \begin{cases} T_{task} + T_{joint} & \text{if } T_{joint} \leq T_{task}, \\ T_{joint} & \text{otherwise.} \end{cases} \tag{4.2}$$

Logically, setting $T_{joint} \geq T_{task}$ does not cause any perturbation in terms of command duration because it ensures that a new command has been generated between two driver time steps.

In what follows, we will see how we take into account synchronization issues in the joint constraints determination.

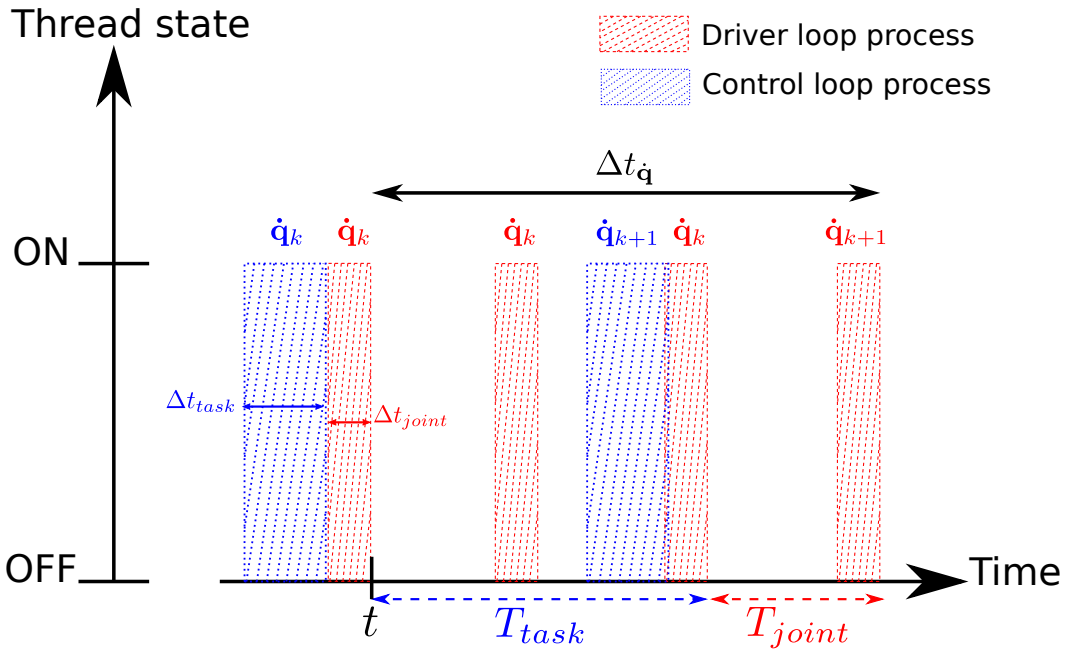


Figure 4.4 – Synchronization issue example when the control loop and one driver thread are running. This figure illustrates the worst case scenario where the velocity command $\dot{\mathbf{q}}_k$ is effective for $\Delta t_{\dot{\mathbf{q}}} = T_{joint} + T_{task}$ instead of T_{joint}. The fact that the control rate is slower that the driver rate implies that the same joint command is executed for at least T_{task}. Moreover, when the control thread and the driver thread are totally unsynchronized (the command computed by the control is published just after the driver loop has begun), the same command is still maintained for T_{joint}.

First, let us focus on the acceleration constraint. In Eq. (3.22), we expressed this constraint as joint velocity boundaries by performing a simple discrete-time integration.

The more restrictive bounds for this constraint are obtained for $\Delta t_{\dot{\mathbf{q}}} = \Delta t^{-}_{\dot{\mathbf{q}}}$. Hence, we ensure that the constraint is never violated by setting:

$$\underline{\ddot{\mathbf{q}}} \Delta t^{-}_{\dot{\mathbf{q}}} + \dot{\mathbf{q}}_k \leq \dot{\mathbf{q}} \leq \overline{\ddot{\mathbf{q}}} \Delta t^{-}_{\dot{\mathbf{q}}} + \dot{\mathbf{q}}_k. \tag{4.3}$$

For the deceleration/position constraint, we recall from Eq. (3.23) that the bounds on the joint velocity command are not time-dependent. In fact, they are expressed as a function of the current state $\mathbf{q}_k$. However, to ensure that it is always possible to decelerate and stop before reaching the position limit, we do not rely on the current

but on the future state. In this case, we have to consider the maximum duration $\Delta t_{\dot{\mathbf{q}}} = \Delta t^+_{\dot{\mathbf{q}}}$ for which the joint command is executed to compute the future minimum $\underline{\mathbf{q}}_{k+1}$ and maximum $\overline{\mathbf{q}}_{k+1}$ state that might be reached at the next iteration:

$$\begin{aligned} \underline{\mathbf{q}}_{k+1} &= \mathbf{q}_k + \max\left\{\underline{\dot{\mathbf{q}}}\ ;\ \underline{\ddot{\mathbf{q}}}T + \dot{\mathbf{q}}_k\right\} * \Delta t^+_{\dot{\mathbf{q}}}, \\ \overline{\mathbf{q}}_{k+1} &= \mathbf{q}_k + \min\left\{\overline{\dot{\mathbf{q}}}\ ;\ \overline{\ddot{\mathbf{q}}}T + \dot{\mathbf{q}}_k\right\} * \Delta t^+_{\dot{\mathbf{q}}}. \end{aligned} \tag{4.4}$$

The deceleration/position constraint which should be verified at any time, taking into account synchronization issues, becomes:

$$-\sqrt{2\underline{\ddot{\mathbf{q}}}(\underline{\mathbf{q}} - \underline{\mathbf{q}}_{k+1})} \leq \dot{\mathbf{q}} \leq \sqrt{2\overline{\ddot{\mathbf{q}}}(\overline{\mathbf{q}} - \overline{\mathbf{q}}_{k+1})}. \tag{4.5}$$

Hence, from Eq. (4.3) and Eq. (4.5), we keep the most restrictive condition for each joint and we replace the final lower and upper bounds defined in Eq. (3.24) by:

$$\begin{aligned} \underline{\dot{\mathbf{Q}}} &= \max\left\{-\sqrt{2\underline{\ddot{\mathbf{q}}}(\underline{\mathbf{q}} - \underline{\mathbf{q}}_{k+1})}\ ;\ \underline{\dot{\mathbf{q}}}\ ;\ \underline{\ddot{\mathbf{q}}}\Delta t^-_{\dot{\mathbf{q}}} + \dot{\mathbf{q}}_k\right\} \\ \overline{\dot{\mathbf{Q}}} &= \min\left\{\sqrt{2\overline{\ddot{\mathbf{q}}}(\overline{\mathbf{q}} - \overline{\mathbf{q}}_{k+1})}\ ;\ \overline{\dot{\mathbf{q}}}\ ;\ \overline{\ddot{\mathbf{q}}}\Delta t^-_{\dot{\mathbf{q}}} + \dot{\mathbf{q}}_k\right\}, \end{aligned} \tag{4.6}$$

4.1.3 Example

We present here a short application example setting up a dual-arm robot scenario using RKCL. The code is split into three different files: the main file which is executed, the application utility file which manages the execution of the controller, and the configuration file where the user can parametrize everything related to the application (configure the robot, tune the various gains, specify the tasks...).

In this example, the dual-arm platform BAZAR is used to perform collaborative tasks. An initialization phase consists in reaching a starting joints configuration. After that, the robot should execute a sequence of tasks in the Cartesian space and stop when they have all been completed.

The main file is given below. Comments have been added to make it self-explanatory. One may notice that the file contains only a few lines of code. In fact, all the parametrization and call to functions are done in the background thanks to the application utility object and configuration file.

```
1 //Include all needed dependencies to use the BAZAR robot
2  #include <rkcl/robots/bazar.h>
3
4
5 int main()
6 {
7     //Load the configuration file (further described)
```

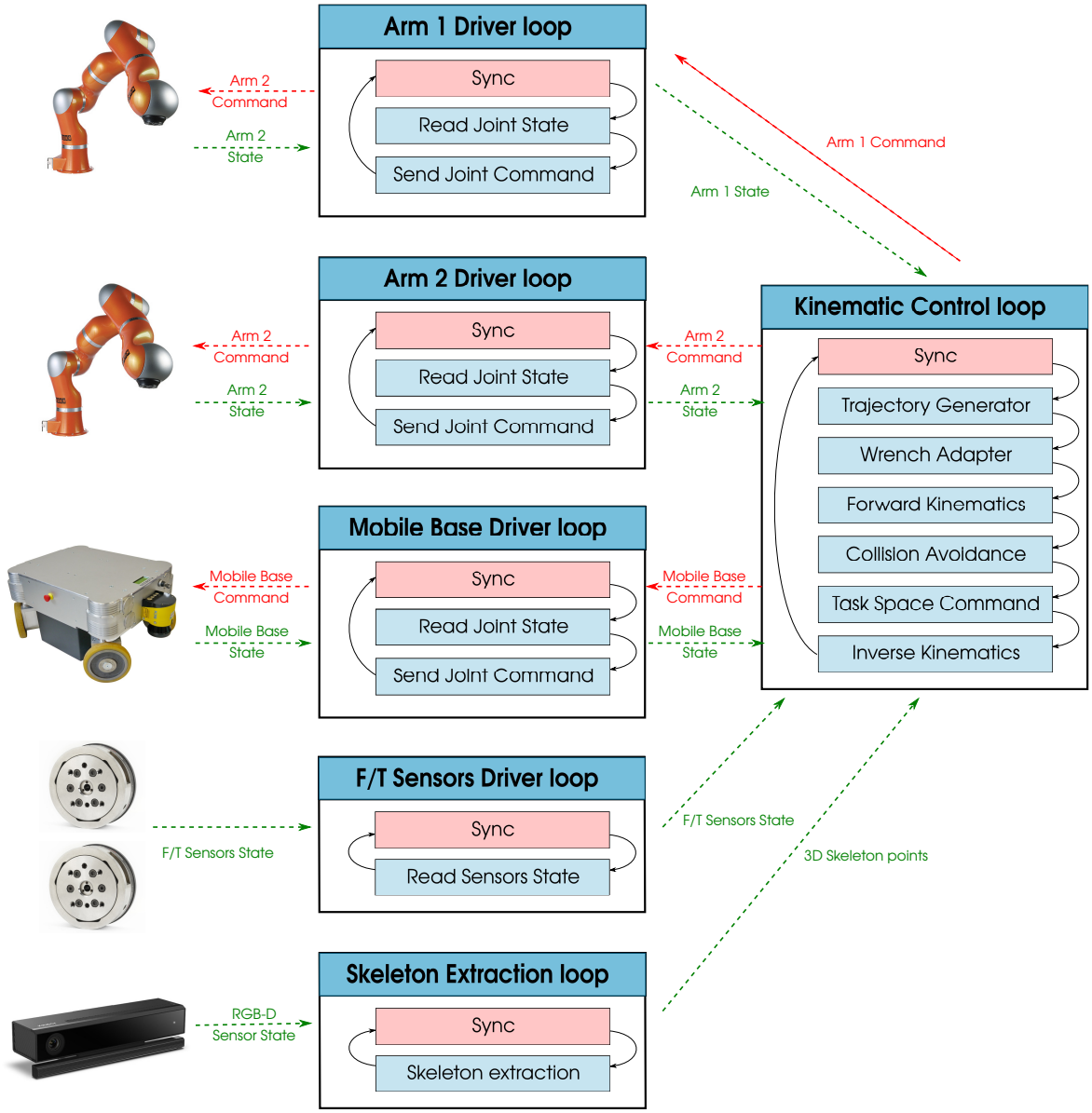


Figure 4.5 – Parallelization of driver/controller processors in RKCL for dual-arm mobile robot BAZAR. Each thread is executed independently and at its proper time rate, managed by the Sync block. When new data are available (either state for the controller or command for the drivers) they are sent to the concerned processors which will put them into effect at the next iteration.

```
8     auto conf = YAML::LoadFile("bazar_config.yaml");
9
10    //Create an application utility object (further described)
11    auto app = rkcl::AppUtility::create(conf);
12
13    //Starting joint control by loading the initial task
14    app.configureTask(0);
15
16    //Variable indicating if the current task has been completed
17    bool done = false;
18    while (not done)
19    {
20        //Execute one step of the joint control loop
21        app.runJointSpaceLoop();
22
23        //Keep looping until all joint groups have reached the desired
              configuration
24        done = true;
25        for (auto joint_controller : app.getJointControllers())
26            done &= joint_controller->isTaskCompleted();
27    }
28
29    //Load the first task in the Cartesian space
30    app.nextTask();
31    done = false;
32    while(not done)
33    {
34        //Execute one step of the task space control loop
35        app.runTaskSpaceLoop();
36
37        //Keep looping until the goal pose has been reached for all tasks
38        done &= app.getTaskSpaceController()->areTasksCompleted();
39
40        //If the current task is completed, move on to the next one.
              Return false if no more task remains
41        if(done)
42            done = not app.nextTask();
43    }
44    //End the application (stop the processors, close the communication
          with drivers, ...)
45    app.end();
46 }
```

The configuration file loaded at the beginning of the application is written in YAML data-serialization language[7]. It allows to get an ordered structure of parameters from the program side and dispatch the information to the concerned data or processors during initialization.

Here is an extract taken from the configuration file referred to as "bazar_config.yaml" in the main program, which aims at setting up the robot and various classes:

[7] https://yaml.org/

```yaml
robot:
    joint_groups:
      - name: left_arm
        priority: 1
        joints: [joint1_left, joint2_left, joint3_left, joint4_left,
            joint5_left, joint6_left, joint7_left]
        control_time_step: 0.001

      - name: right_arm
        priority: 1
        joints: [joint1_right, joint2_right, joint3_right, joint4_right
            , joint5_right, joint6_right, joint7_right]
        control_time_step: 0.001

      - name: mobile_base
        priority: 2
        joints: [joint1_base, joint2_base, joint3_base]
        control_time_step: 0.02

    control_frames:
      - name: relative_task
        task_priority: 1 //Decreasing priority level
        body_name: end-effector_right
        ref_body_name: end-effector_left
        gains:
            proportional: [10, 10, 10, 10, 10, 10]

      - name: absolute_task
        task_priority: 2
        body_name: absolute_frame
      ref_body_name: world
        gains:
            proportional: [10, 10, 10, 10, 10, 10]

model:
    path: bazar_model.yaml

app:
    task_config_files:
      - name: joint_init.yaml
      - name: task_1.yaml
      - name: task_2.yaml
```

Following this configuration file example, the *robot* structure represented in Fig. 4.3 is assigned three *joint_group*: the two arms corresponding to the manipulator robot Kuka LWR4+ with a control time step of 1ms and the Neobotix MPO-700 mobile base which runs at 20ms. The two arms have maximum priority due to their high responsiveness and accuracy, while the mobile base with more inertia and lower control frequency has the least priority. It is used to assist the arms in case of large motions in the workspace. Each *joint_group* is associated with a vector of joints, defined in the

robot model "bazar_model.yaml".

The *robot* holds two control frames referring to the cooperative tasks. As explained in Chapter 3, the priority is given to the relative task over the absolute task. A control frame is characterized by the link that is controlled ("body_name") and the link whose attached frame serves as a reference ("ref_body_name"). This information is also extracted from the model file.

The scenario for the application is configured at the end. The list of YAML file which is enumerated using "task_config_files" is stored at the beginning of the application. In the main file, the methods *configureTask()* (line 14) or *nextTask()* (line 42) are called to load a desired task. A task is simply updated by setting the goal data associated with the control frame, as in the following example:

```
1  robot:
2      control_frames:
3        -   name: relative_task
4            // Possible values : 'none', 'pos', 'force', 'damp' or 'adm'
5            control_mode: [pos, pos, pos, pos, pos, pos]
6            goal:
7                pose: [0, 0, 0.2, -3.14, 0, 0]
8
9        -   name: absolute_task
10           control_mode: [pos, pos, pos, pos, pos, pos]
11           goal:
12               pose: [0.5, 0, 1.5, 1.57,    0, 0]
```

Finally, let us briefly describe the role of the *AppUtility* class. As its name implies, it has been implemented to ease the development of applications by handling the control loops and processors that should be executed inside them. In fact, no matter what the scenario is, there exists a common structure between every application. In the previous subsection, we mentioned how we deal with synchronization issues. Here, the *AppUtility* is in charge of managing the parallelization by adopting a multithreading approach: a dedicated communication loop is created for each joint group while a unique task space control loop refreshes the joint command for all the groups. The content of the control loop is reduced to a sequential call to the various processors presented throughout this thesis:

```
1  bool AppUtility::runTaskSpaceLoop()
2  {
3      // Wait for the cycle time duration, from last call
4      waitNextCycle();
5
6      bool all_ok = true;
7
8      all_ok &= task_space_otg();
9
10     all_ok &= wrench_adapter();
11     all_ok &= forward_kinematics();
12     all_ok &= collision_avoidance();
13
```

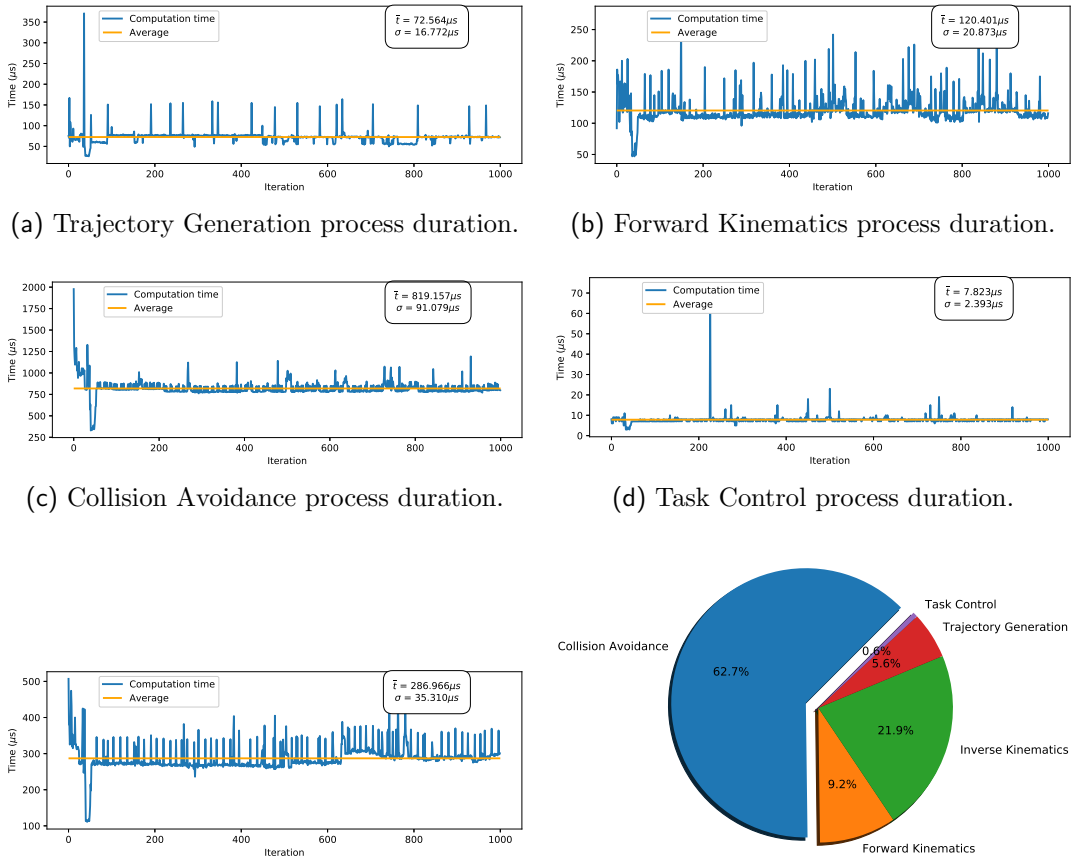
```
14      all_ok &= task_space_controller();
15      all_ok &= ik_controller();
16
17      return all_ok;
18  }
```

All the processors listed above have been given a pointer to the *robot* element, which makes them able to modify the data internally without passing parameters. At the end of the task space control loop, a new vector of joint velocity command is available for every joint group and will be applied at the next iteration of the corresponding joint communication loop.

4.1.4 Benchmarks

To perform online control in changing industrial environments where safety of individuals is engaged, responsiveness of robots is a key parameter. In case of unforeseen event, the robot should be able to react promptly to avoid collisions or reduce damages if an impact is inevitable. Furthermore, performing pHRI require a high control frequency to ensure smooth and appropriate motions of the robot. Thus, it is crucial that the controller presented here and its implementation in RKCL are fast enough to comply with timing constraints.

We assessed the performance of the library by running some benchmarks on a computer with Intel(R) Core(TM) i7-6600U CPU running Linux. In this experiment, we performed a simple pose control with trajectory tracking on the dual-arm robot BAZAR. We placed five obstacles at random locations in the workspace to evaluate the process in a cluttered environment. In Fig. 4.6, we present the results of the benchmarks for the most computationally demanding processes of the kinematic control loop. We selected randomly 1000 time steps among the whole task execution and evaluated the computation time for each process. On Fig. 4.6a – 4.6e, we indicate the average computation time $\bar{t}$ and the standard deviation σ recorded over the 1000 iterations. The first thing to be pointed out is that we notice significant disparities among the processes in terms of computation time. This observation is highlighted by Fig. 4.6f which shows the computing-time distribution. The collision avoidance process is by far the most computationally intensive since it is responsible for around 63% of the total calculation time. On average, the collision avoidance process requires $\bar{t} = 0.8$ms which is substantial in the context of reactive control. However, we could easily mitigate this problem by executing this process in a separate thread. In any case, vision systems or other sensory equipments that are used to locate obstacles are generally slow, meaning that it would be useless to perform this operation at higher frequency. Apart from the collision avoidance process, the other units of the controller are executed within a reasonable time. As expected, the inverse kinematics resolution, including the HQP architecture, takes the most time among the remaining processes. With a computation time of less than $\bar{t} = 0.3$ms on average and $\sigma = 35$µs, it allows to let the whole control loop run at 1kHz, which is suitable for pHRI under proper conditions.



(a) Trajectory Generation process duration.

(b) Forward Kinematics process duration.

(c) Collision Avoidance process duration.

(d) Task Control process duration.

(e) Inverse Kinematics process duration.

(f) Distribution of the computational burden between the processes.

Figure 4.6 – Benchmarks of the different processes running sequentially in the kinematic control loop.

4.2 Application to *teaching-by-demonstration*

In this section, we elaborate an industrial application fully illustrating the different features of our collaborative framework. *Teaching-by-demonstration* is a good technique to promote greater flexibility and agility in the industry. Instead of spending time and money in reprogramming robots offline, *teaching-by-demonstration* allows to easily and intuitively reconfigure the tasks when changing production lines.

4.2.1 Description of the scenario

The objective of this experiment is to quickly configure the robot so that it can repeatedly move boxes with its two arms from an initial location to a desired destination. A video of the experiments is available at https://youtu.be/2ihgqm4MCEQ. The scenario is divided into two parts :

1. **the teaching phase** - the robot is in compliant mode. A human operator can physically interact with it to teach the successive sequences to perform to accomplish the whole task. To manage the consecutive operations, we design a set of tasks to be loaded sequentially. This is represented in Fig. 4.7 using snapshots of the experiments. Thanks to our framework, we develop a hybrid position/force learning strategy in the cooperative task space: target waypoints are recorded for the absolute task frame in order to reach and then move the object in the workspace while wrench measurements on the relative tasks are stored to apply desired internal wrench on the box during the manipulation. Besides providing safe manipulation, this allows repeating the operation properly without the need for high accuracy. In particular, the robot can complete the task even if the object is not precisely located at the same place as it was during the teaching phase, or if the object's dimensions have (slightly) changed.

2. **the replay phase** - the robot autonomously reproduces the set of subtasks learned during the teaching phase. Online trajectory generation is performed to reach the successive waypoints for the absolute task frame and internal wrench regulation allows to grasp/release the object. The human operator, previously seen as a collaborator, is now considered as an obstacle. Thus, we use computer vision to detect and locate the human skeleton in the 3D Cartesian space. We apply the repulsive action on the absolute task to prevent from being too close to unpredictable obstacles without interrupting the task.

4.2.2 Setup

For this experiment, we use the same setup as the one presented in Section 2.4.1, except that here the Neobotix MPO-700 omnidirectional mobile base of the dual-arm cobot BAZAR is enabled with a time rate $T_{mob} = 20\,\text{ms}$. As a reminder, the Kuka arms run at $T_{arms} = 5\,\text{ms}$, which is also the time step used for the main control loop.

During the replay phase, the Reflexxes Motion Library is used to generate trajectories between the stored waypoints. To set up the trajectory profile, we specify the following parameters: maximum translational veclocity $0.5\text{m}\,\text{s}^{-1}$, maximum rotational veclocity $0.5\text{rad}\,\text{s}^{-1}$, maximum translational acceleration $0.2\text{m}\,\text{s}^{-2}$, and maximum rotational acceleration $0.2\text{rad}\,\text{s}^{-2}$.

As in Section 2.4.1, computer vision is used to estimate online the location of the human operator, as shown in Fig. 4.8.

4.2.3 Results

To assess the performance of our mobile dual-arm collaborative framework RKCL and demonstrate that it provides a suitable solution for flexible industrial applications, let us first focus on the reproducibility of the taught operations. Fig. 4.9 shows the logged error for both the relative and absolute tasks during the whole replay phase. For the

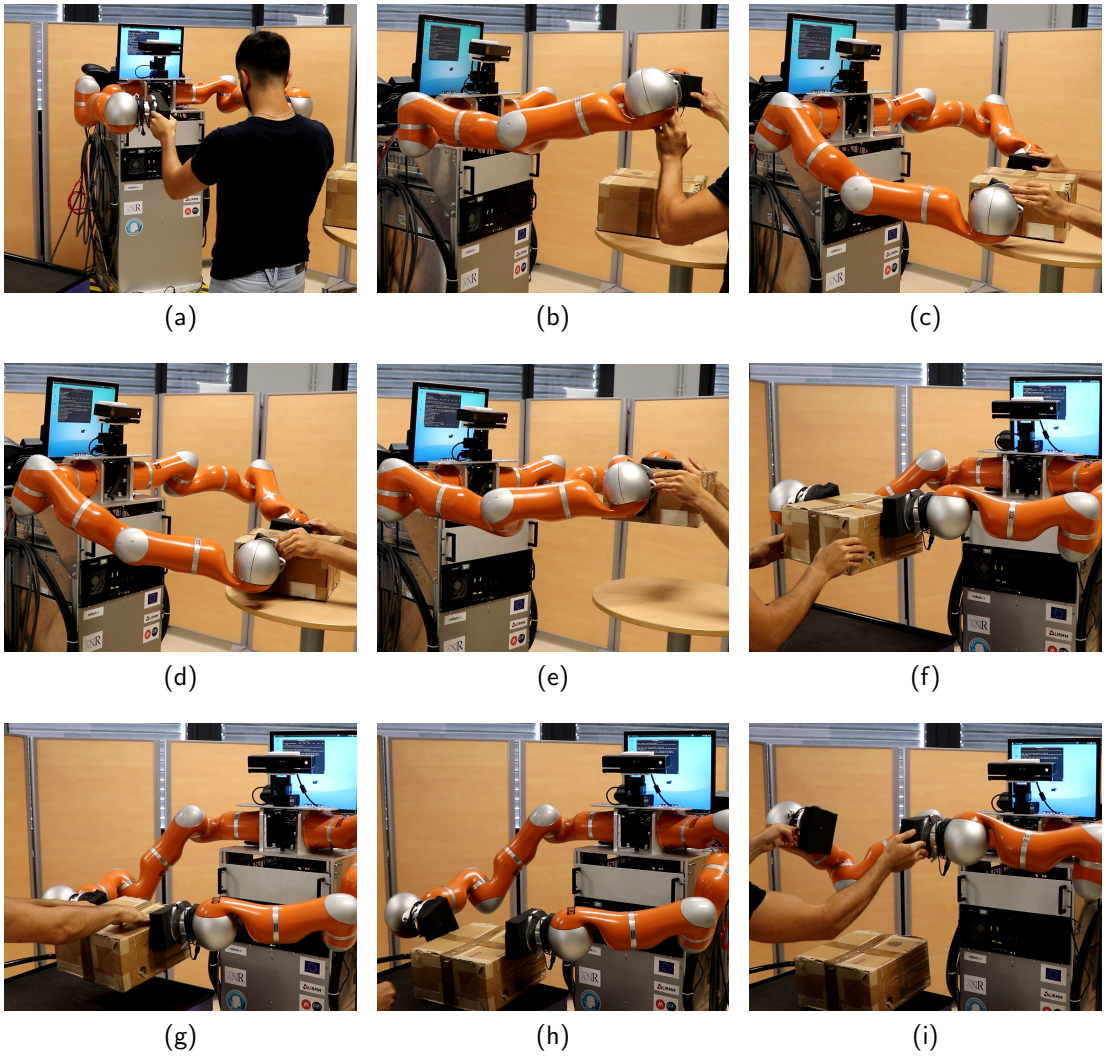


Figure 4.7 – Some snapshots taken during the teaching phase of the experiment: (a) The operator moves the whole dual-arm platform to a workable location using the mobile base. (b) The operator brings the robot to the first waypoint. (c) The robot is ready to grasp the box. (d) The operator teaches the internal force required to maintain the object. (e) The robot lifts the box to the first waypoint shown by the user. (f) The operator guides the robot towards the deposit station. (g) The operator shows where to release the box. (h) The robot releases the object. (i) The robot is taught how to move its end-effectors away from the box.

relative task, the *pos z* and *rot z* error on the pose tracking have been omitted, because *pos z* is force-controlled and *rot z* has been left free to give more redundancy to the system without compromising the task.

Since we assign maximum priority to the relative task, its error is negligible: the average error is 2×10^{-4}m for translation and 4×10^{-4}rad for orientation. More importantly, the error magnitude never exceeds 2×10^{-3}m for the position variables and 4×10^{-3}rad for the orientation, which means that the object is always held firmly. The

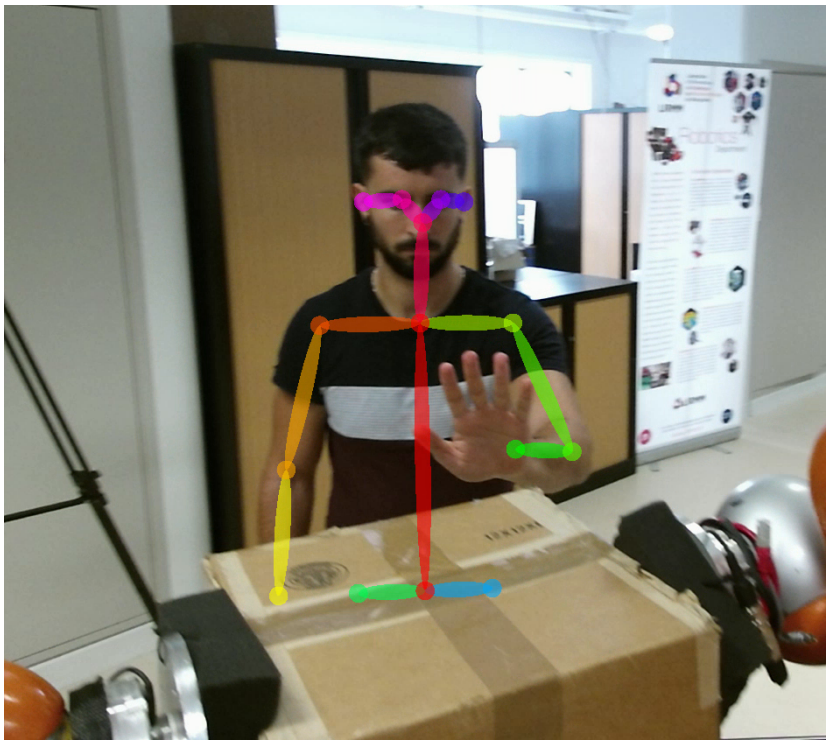


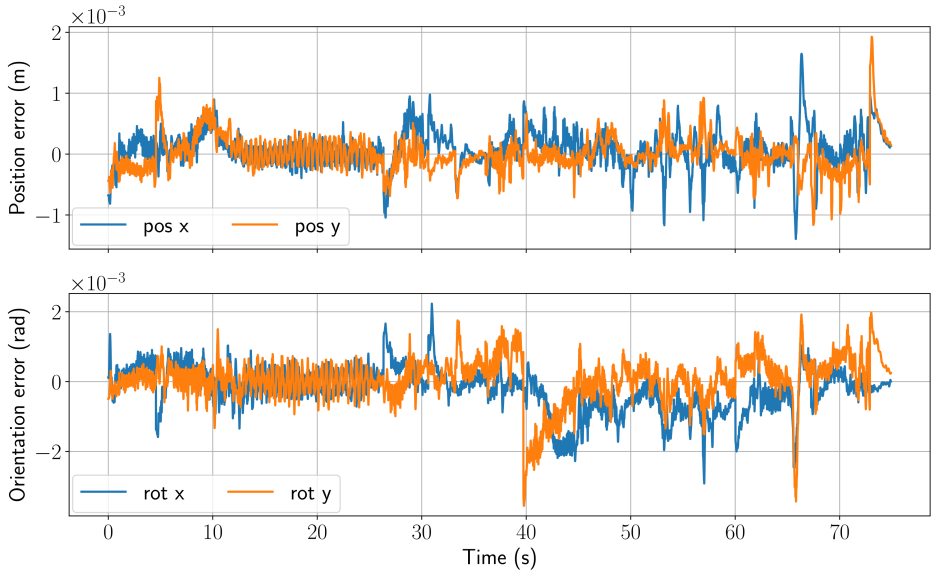
Figure 4.8 – View from the BAZAR Microsoft Kinect. The human is detected and his skeleton projected in the 3D space using depth. During the replay phase, people entering the robot's workspace are treated as obstacles to avoid.

low error comes from the damping term used to make the system compliant.

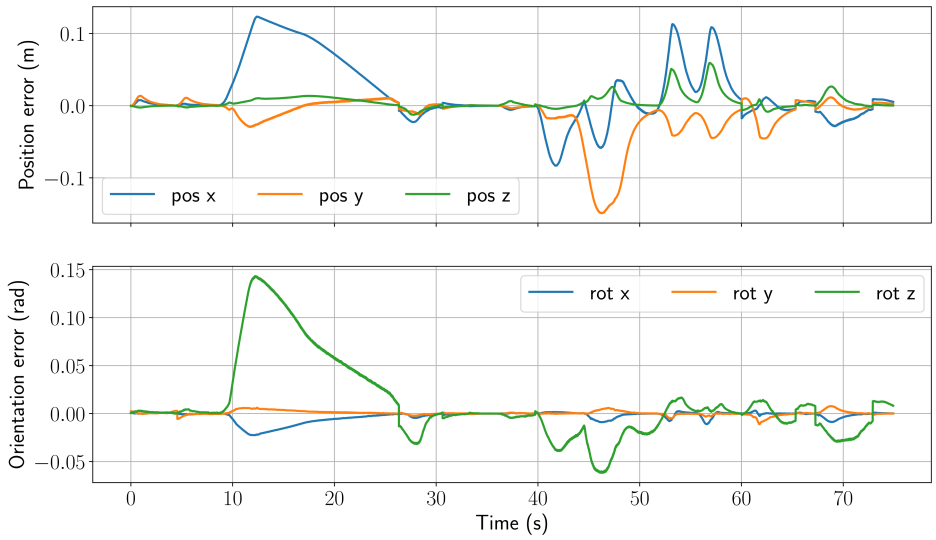
The absolute task error, i.e. the error with respect to the planned trajectories generated with the stored waypoints, is much higher: the average is 2×10^{-2}m for the position and 1×10^{-2}rad for the orientation. The maxima almost reach 0.15m and 0.15rad regarding respectively the position and orientation components. There are several reasons for this: on one hand, the absolute task is set as secondary in the optimization problem. Hence, when the robot is in a low manipulability configuration and loses some DoF, the efforts turn in favor of the relative task. On the other hand, the collision avoidance strategy adopted during the replay phase leads to some deviations from the initial trajectory. In particular, this explains the important error observed between time $t = 10$s and $t = 28$s and then between time $t = 40$s and $t = 60$s. It is important to note that despite these significant errors, the robot is able to complete the task safely (without hitting the operator nor other obstacles).

We also evaluate safety by focusing on internal constraints arising during object transportation. In Fig. 4.10, we show the evolution of the state and target wrench values for the z component of the relative task (in charge of tightening the arms) during both the teaching and replay phases. At time $t = 128$s, the human operator teaches the reference force to the robot by applying the desired pressure on the object with the tools, as shown in Fig. 4.7d. The reference force of 17.5N is then used during the replay phase to grasp the box and to regulate internal stress. As we can see, there is a delay of a few seconds between the time when the reference force is set and the time it is reached. During this phase, the robot slowly tightens the arms before going in contact with the object. The object seizure creates a slight overshoot of the reference (around 5N) which can be alleviated by adding a derivative term on the command law.

Note that the undesired variations from zero observed during the teaching phase are caused by the operator which guides the robot. Indeed, it sometimes happens that he applies unbalanced wrenches on the two arms to move the robot in the workspace. This emulates internal wrenches but does not affect the teaching process.



(a) Relative task pose error



(b) Absolute task pose error

Figure 4.9 – Evolution of the tracking error for the cooperative task variables. Thanks to the hierarchical inverse kinematics strategy, the relative task has negligible error throughout the whole operation ensuring the safe manipulation of the object. Due to the lower priority assigned to the absolute task, we naturally record greater deviations with regards to the initially planned trajectory. In addition to that, the repulsive effect generated by surrounding obstacles temporarily pulls the robot away from the straight path.

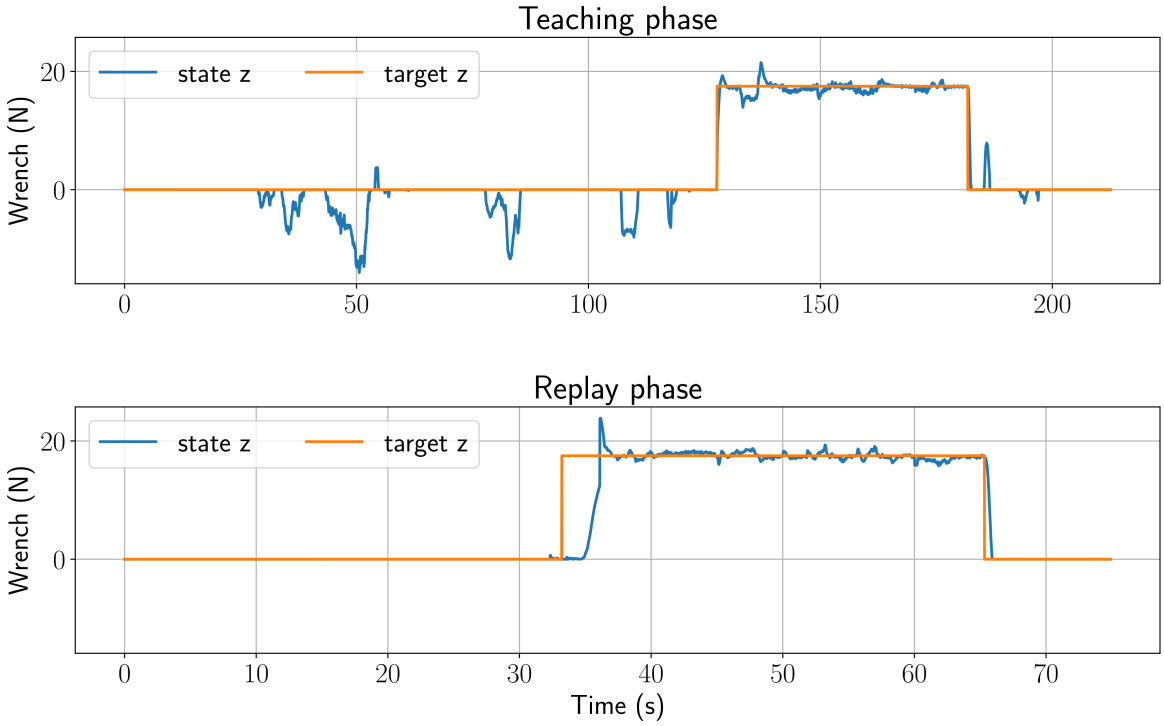


Figure 4.10 – Evolution of the relative task state and target wrench (z component only). The reference value stored during the teaching phase is properly replicated during the replay phase, to safely manipulate the box.

4.3 Conclusion

This chapter gave an overlook of the dual-arm kinematic control framework developed in the scope of this thesis. After summarizing the overall functioning of the control loop, We introduced RKCL, the kinematic control library which implements all the features presented in this thesis. Based on a generic core component, many functionalities are available through the inclusion of additional packages. In particular, we implemented the drivers to communicate with the different parts of the robot. The library can be easily completed with new robots and features. The multi-robot approach requires to take into account the various control frequencies of the independent components. To do so, a unique control loop manages the execution of the different processes and establishes an unsynchronized communication with every hardware part to get updated states and send the computed command.

Finally, we proposed to spotlight the dual-arm collaborative framework by implementing a *teaching-by-demonstration* application on the BAZAR cobot. The results show that a human operator with no specific knowledge in robotics can configure new operations quickly and easily.

Conclusion

In this thesis, we addressed the control problem of dual-arm collaborative robots intended to be used in modern industrial setups. This type of platforms benefits from the dexterity and accuracy of the two arms which, used cooperatively, can achieve a wide range of complex operations. Our work aims at providing a generic control strategy which takes full advantage of dual-arm robot capabilities.

We first tackled the control issues by stating a relevant representation of cooperative tasks. We extended the so-called cooperative task representation which fully characterizes the operational space for bi-manual cooperative control with geometrically meaningful motion variables. Compared to the original definition, our version avoids representation singularity and offers a simple and intuitive manner to describe most manufacturing operations. We use wrench feedback measured at the tip of each arm to manage both internal constraints arising during dual-arm manipulation of objects and perceive external forces which makes the robot able to interact with the environment. Additionally, we include new considerations from wrench feedback to enhance the quality of pHRI. In particular, the proposed method intends to make cooperative object transportation more natural and efficient by acting on two levels: estimating and compensating gravity effects on one side, and retrieving human interaction wrench on the other. We designed a closed-loop admittance controller which can adopt a specific behavior for each task variable to safely interact with the environment.

Then, we dealt with the issue of how to transfer the task space command to the joint space, to enable the robot actuation. We proposed several task-solving strategies based on QP optimization which includes this set of hard constraints. Among the alternatives, we want to emphasize the parsimonious approach, that consist in minimizing the number of joints actuated to complete a given task. However, we showed that the sparsest solutions to IK problems often leads to chattering effects. To overcome this, we implemented a HQP architecture in which the last resolution process allows to tune the desired level of sparsity. During experiments, we observed that a minor reduction of parsimony is sufficient to solve the problem. Beyond that, our HQP implementation allows to establish a task prioritization strategy between the cooperative tasks with the aim of devoting maximum efforts to satisfy the relative task requirements; in particular, this ensures safe manipulation of objects. We expanded the scope of our kinematic

control strategy to any dual-arm platform with additional actuation capabilities. To do so, we developed an original approach in which robot joints are classified in several groups with priority assigned to them. The group of highest priority first tries to solve the tasks, and other groups are then involve only if residual error remains. This strategy is applied to dual-arm mobile robots where extra joints are engaged only for large motions. In unstructured workspace populated by humans, it is imperative to avoid brutal motions of the robot and anticipate dangerous situations. Hence, we gave a thorough analysis of constraints that must be respected at all time to ensure proper performance of robots. The management of admissible solutions is done both at joint and task levels.

The last chapter was dedicated focused on how all the components of the control framework work together. All the developments have been implemented in the kinematic control library RKCL. To deal with different attributes of hardware components, the main control loop establishes unsynchronized communications with every part of the robot. The resulting delays are adequately handled to prevent from violating hard constraints. We spotlight our control framework with a "teaching-by-demonstration" application on the BAZAR cobot. The results show that a human operator with no specific knowledge in robotics can configure new operations in a quick and easy manner.

The solutions provided in this thesis attempt to facilitate and make safer the deployment of autonomous dual-arm robots in assembly units. However, there is still plenty of room for improvements as our method presents some disadvantages. The main one lies in the fact that our controller is purely local. Indeed, it considers only the current state of the robot and the environment to compute the command for the next step. Without a global view of the task, there is no guarantee that a feasible solution can be found. In addition, dual-arm robots are subject to a large variety of constraints and, as a result, are likely to get stuck in local minima. An interesting approach would be to combine our reactive controller with global resolution techniques, such as motion planning. For instance, the sampling-based path planner proposed in (GCS08) could be processed offline to provide an initial task-space solution that would be adapted online by the reactive controller, taking into account changes of the environment. An other idea is to explore predictive strategies for the online redundancy resolution. Indeed, using a predictive model would result in a more efficient handling of constraints and would probably provide overall better performances. Existing works on model predictive control of redundant manipulators (SBB+14; Zub15; FBTV17) have demonstrated their efficiency and could be extended to the dual-arm case.

Finally, being exclusively based on kinematics, our framework cannot include dynamics constraints, such as torque limitations, or incorporate robots that rely on a dynamic model and on torque control. These issues will be investigated for future versions of RKCL, to make the library usable in more cases.

APPENDIX A

Jacobian computation

The notion of kinematic Jacobian is at the heart of our controller since it allows to relate Cartesian motions of some frames of interest attached to the robot with its joint displacements. There exists a simple and computationally efficient way of computing the Jacobian matrix through the geometric method.

Let us consider a manipulator arm with n DoF. The geometric Jacobian $\mathbf{J} \in \mathbb{R}^{6\times n}$ associated with its end-effector (subscript 'e') is divided into a position part $\mathbf{J}_P$ and an orientation part $\mathbf{J}_O$, such as:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{P1} & & \mathbf{J}_{Pn} \\ & \dots & \\ \mathbf{J}_{O1} & & \mathbf{J}_{On} \end{bmatrix}, \tag{A.1}$$

where $\mathbf{J}_{Pi}$, $\mathbf{J}_{Oi}$ are the (3×1) column vectors expressing the relation between the velocity of joint i and the translational and rotational velocity of the control frame, respectively.

Based on (SSVO10), the geometric Jacobian can be simply computed, depending on the type of joints, using the direct kinematics relations:

$$\begin{bmatrix} \mathbf{J}_{Pi} \\ \mathbf{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ 0 \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} & \text{for a revolute joint} \end{cases} \tag{A.2}$$

where $\mathbf{z}_{i-1}$ is the unit vector of Joint i axis, $\mathbf{p}_e$ is the position of the end-effector and $\mathbf{p}_{i-1}$ the position of the origin of joint i, all expressed in a fixed reference frame.

As shown in Section 1.1.3, the Jacobian matrices associated with a dual-arm robot performing absolute and relative tasks can be directly expressed from the individual Jacobian of the manipulators.

However, with the aim of providing a flexible framework (see Chapter 4), we want to be capable of defining and solving any kinematic task. Additionally, to evaluate the collision avoidance inequality constraint defined in Eq. (3.40), we should be able to compute the Jacobian matrix between different parts of the robot.

We addressed these issues by proposing a generic approach to formulate the Jacobian matrix between any points of a kinematic chain, and expressed in an arbitrary reference frame, as long as they belong to the same kinematic tree structure.

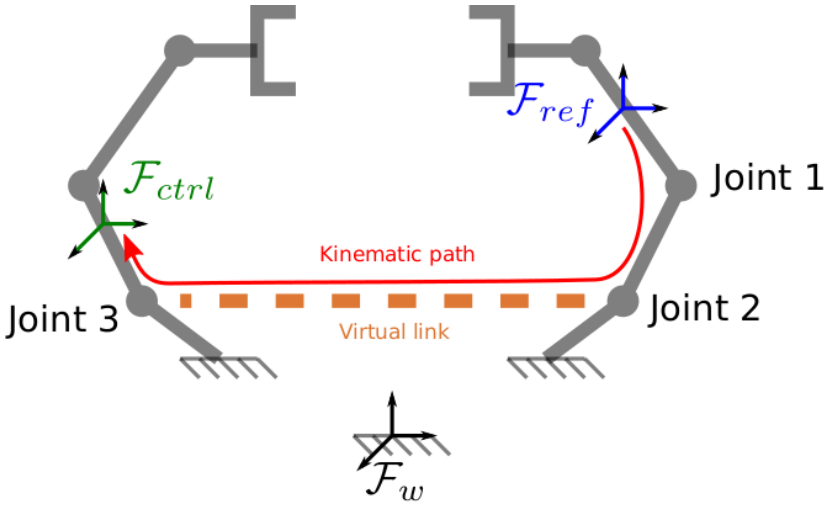


Figure A.1 – Kinematic path associated with the task consisting of the control frame $\mathcal{F}_{ctrl}$ and the reference frame $\mathcal{F}_{ref}$. The geometric Jacobian computation from (SSVO10) is simply adapted to obtain the Jacobian matrix related to any arbitrary task.

Let us illustrate the method by referring to the dual-arm case depicted in Fig. A.1. In this example, we want to compute the Jacobian matrix associated with the control frame $\mathcal{F}_{ctrl}$ and the reference frame $\mathcal{F}_{ref}$ located on each arm of a bimanual robot. First, let us create a virtual link that joins the initial joint of the two arms. This allows to clearly identify the kinematic path which connects $\mathcal{F}_{ref}$ to $\mathcal{F}_{ctrl}$.

We keep the same convention as in Eq. (A.2) to be homogeneous with the formulation used in (SSVO10). In our case, joint indices are incremented according to the kinematic path as shown in Fig. A.1. We differentiate the computation of the geometric Jacobian vectors $\mathbf{J}_{Pi}$ and $\mathbf{J}_{Oi}$ defined in Eq. (A.2) depending on whether the kinematic path at Joint i points towards the root of the tree or a leaf. In our example, Joint 3 is in the "good" direction (the path points towards an end-effector) while Joint 1 and 2 are directed towards the root (the path points towards $\mathcal{F}_w$). In the first case, Eq. (A.2) is reformulated as follows:

$$\begin{bmatrix}\mathbf{J}_{Pi}\\ \mathbf{J}_{Oi}\end{bmatrix} = \begin{cases} \begin{bmatrix}\mathbf{z}_{i-1}^{ref}\\ 0\end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix}\mathbf{z}_{i-1}^{ref} \times (\mathbf{p}_{ctrl}^{ref} - \mathbf{p}_{i-1}^{ref})\\ \mathbf{z}_{i-1}^{ref}\end{bmatrix} & \text{for a revolute joint} \end{cases} \tag{A.3}$$

where $\mathbf{p}_{ctrl}^{ref}$ expresses the position of frame $\mathcal{F}_{ctrl}$ in $\mathcal{F}_{ref}$ and $\mathbf{p}_{i-1}$ the position of the origin of joint i in $\mathcal{F}_{ref}$. A particular attention should be given to $\mathbf{z}_{i-1}^{ref}$, which corresponds to the unit vector of Joint i axis with respect to $\mathcal{F}_{ref}$: the joint actuation will have the reverse effect if, at its location, the kinematic path does not follow the direction of the tree (Joint 1 and 2 in Fig. A.1). In this case, the geometric Jacobian vectors are obtained through:

$$\begin{bmatrix} \mathbf{J}_{Pi} \\ \mathbf{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} -\mathbf{z}_{i-1}^{ref} \\ 0 \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} -\mathbf{z}_{i-1}^{ref} \times (\mathbf{p}_{ctrl}^{ref} - \mathbf{p}_{i-1}^{ref}) \\ -\mathbf{z}_{i-1}^{ref} \end{bmatrix} & \text{for a revolute joint} \end{cases} \tag{A.4}$$

Bibliography

[15016] "Robots and robotic devices–collaborative robots," International Organization for Standardization, Geneva, CH, Standard, 2016.

[ACB+14] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative human-humanoid carrying using vision and haptic sensing," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 607–612.

[AFD10] B. V. Adorno, P. Fraisse, and S. Druon, "Dual position control strategies using the cooperative dual task-space framework," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2010, pp. 3955–3960.

[AIC09] G. Antonelli, G. Indiveri, and S. Chiaverini, "Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems with velocity saturations," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2009, pp. 5892–5897.

[AL89] S. Ahmad and S. Luo, "Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in cartesian space," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 4, pp. 409–417, 1989.

[AVK16] D. Almeida, F. E. Vina, and Y. Karayiannidis, "Bimanual folding assembly: Switched control and contact point estimation," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2016.

[Bar81] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

[BK11] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2011, pp. 4414–4419.

[BKCK12] A. Bussy, A. Kheddar, A. Crosnier, and F. Keith, "Human-humanoid haptic joint object transportation case study," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2012, pp. 3633–3638.

[BNU+17] A. Batinica, B. Nemec, A. Ude, M. Raković, and A. Gams, "Compliant movement primitives in a bimanual setting," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2017, pp. 365–371.

[CAB+07] B. Corteville, E. Aertbeliën, H. Bruyninckx, J. De Schutter, and H. Van Brussel, "Human-inspired robot assistant for fast point-to-point movements," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3639–3644.

[CCC00] F. Caccavale, P. Chiacchio, and S. Chiaverini, "Task-space regulation of cooperative manipulators," *Automatica*, vol. 36, no. 6, pp. 879–887, 2000.

[CCS96] P. Chiacchio, S. Chiaverini, and B. Siciliano, "Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system," *Journal of dynamic systems, measurement, and control*, vol. 118, no. 4, pp. 691–697, 1996.

[CHS+18] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[COW08] S. Chiaverini, G. Oriolo, and I. D. Walker, "Kinematically redundant manipulators," *Springer handbook of robotics*, pp. 245–268, 2008.

[CPC+16] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse, "Collaborative manufacturing with physical human–robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 2016.

[CPN+19] A. Cherubini, R. Passama, B. Navarro, M. Sorour, A. Khelloufi, O. Mazhar, S. Tarbouriech, J. Zhu, O. Tempier, A. Crosnier *et al.*, "A collaborative robot for the factory of the future: Bazar," *The Int. Journal of Advanced Manufacturing Technology*, pp. 1–17, 2019.

[DG07] V. Duchaine and C. M. Gosselin, "General model of human-robot cooperation using a novel velocity based variable impedance control," in *IEEE Conf. and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2007, pp. 446–451.

[DGB+12] J. Dumora, F. Geffard, C. Bidard, T. Brouillet, and P. Fraisse, "Experimental study on haptic communication of a human in a shared human-robot collaborative task," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2012, pp. 5137–5144.

[DGUP15] M. Deniša, A. Gams, A. Ude, and T. Petrič, "Learning compliant movement primitives through demonstration and statistical generalization," *IEEE/ASME transactions on mechatronics*, vol. 21, no. 5, pp. 2581–2594, 2015.

[DSBDS09] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter, "Extending itasc to support inequality constraints and non-instantaneous task specification," in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 964–971.

[EB02] M. Elad and A. M. Bruckstein, "A generalized uncertainty principle and sparse representation in pairs of bases," *IEEE Trans. on Information Theory*, vol. 48, no. 9, pp. 2558–2567, 2002.

[EMW] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. Journal of Robotics Research*.

[EMW10] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 3733–3738.

[ESH13] S. Erhart, D. Sieber, and S. Hirche, "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2013, pp. 315–322.

[FBTV17] M. Faroni, M. Beschi, L. M. Tosatti, and A. Visioli, "A predictive approach to redundancy resolution for robot manipulators," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8975–8980, 2017.

[FBVT16] M. Faroni, M. Beschi, A. Visioli, and L. M. Tosatti, "A global approach to manipulability optimisation for a dual-arm manipulator," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, 2016.

[FDLK12] F. Flacco, A. De Luca, and O. Khatib, "Motion control of redundant robots under joint constraints: Saturation in the null space," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 285–292.

[FKDLK12] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 338–345.

[FLMO16] A. Freddi, S. Longhi, A. Monteriù, and D. Ortenzi, "Redundancy analysis of cooperative dual-arm manipulators," *Int. Journal of Advanced Robotic Systems*, vol. 13, no. 5, 2016.

[FT87] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," Ph.D. dissertation, INRIA, 1987.

[Fuc04] J.-J. Fuchs, "On sparse representations in arbitrary redundant bases," *IEEE trans. on Information theory*, vol. 50, no. 6, pp. 1341–1344, 2004.

[GCS08] M. Gharbi, J. Cortés, and T. Simeon, "A sampling-based path planner for dual-arm manipulation," in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2008, pp. 383–388.

[GFCA16] V. M. Gonçalves, P. Fraisse, A. Crosnier, and B. V. Adorno, "Parsimonious kinematic control of highly redundant robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 65–72, 2016.

[GKB11] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 4326–4332.

[GZ16] B. R. Gaines and H. Zhou, "Algorithms for fitting the constrained lasso," *arXiv preprint arXiv:1611.01511*, 2016.

[HHY15] Y. Hu, B. Huang, and G.-Z. Yang, "Task-priority redundancy resolution for co-operative control under task conflicts and joint constraints," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2015.

[Hog84] N. Hogan, "Impedance control: An approach to manipulation," in *IEEE American control conference*, 1984, pp. 304–313.

[HYN81] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and control of articulated robot arms with redundancy," *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927–1932, 1981.

[II95] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, 1995, pp. 3097–3102.

[IMM02] R. Ikeura, T. Moriguchi, and K. Mizutani, "Optimal variable impedance control for a robot and its application to lifting an object with a human," in *IEEE Int. Workshop on Robot and Human Interactive Communication*, 2002, pp. 500–505.

[ISO11] "Robot for industrial environments - safety requirements - part 1: Robot," International Organization for Standardization, Geneva, CH, Standard, 2011.

[JR15] R. S. Jamisola and R. G. Roberts, "A more compact expression of relative jacobian based on individual manipulator jacobians," *Robotics and Autonomous Systems*, vol. 63, no. P1, pp. 158–164, 2015.

[KD04] W. Khalil and E. Dombre, *Modeling, identification and control of robots.* Butterworth-Heinemann, 2004.

[Kha86] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles.* Springer, 1986, pp. 396–404.

[Kha87] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[Kha88] O. Khatib, "Object manipulation in a multi-effector robot system," in *Proceedings of the 4th international symposium on Robotics Research.* MIT Press, 1988, pp. 137–144.

[KLW+09] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 2939–2944.

[KLW11] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[Krö11] T. Kröger, "Opening the door to new sensor-based robot applications—the reflexxes motion libraries," in *IEEE Int. Conf. on Robotics and Automation*, 2011.

[KSK00] K. Kosuge, M. Sato, and N. Kazamura, "Mobile robot helper," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2000, pp. 583–588.

[KYT+94] K. Kosuge, H. Yoshida, D. Taguchi, T. Fukuda, K. Hariki, K. Kanitani, and M. Sakai, "Robot-human collaboration for new robotic applications," in *Proceedings of IECON'94-20th Annual Conf. of IEEE Industrial Electronics*, vol. 2. IEEE, 1994, pp. 713–718.

[LaV06] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[LMH10] M. Lawitzky, A. Mörtl, and S. Hirche, "Load sharing in human-robot cooperative manipulation," in *IEEE Int. Symposium on Robot and Human Interactive Communication*, 2010, pp. 185–191.

[Nic18] J. Nicholas, *Lean production for competitive advantage: a comprehensive guide to lean methodologies and management practices.* Productivity Press, 2018.

[NLGU16] B. Nemec, N. Likar, A. Gams, and A. Ude, "Bimanual human robot cooperation with adaptive stiffness control," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, pp. 607–613.

[NLGU18] B. Nemec, N. Likar, A. Gams, and A. Ude, "Adaptive human robot cooperation scheme for bimanual robots," in *Advances in Robot Kinematics*. Springer, 2018, pp. 371–380.

[OCB08] W. Owen, E. Croft, and B. Benhabib, "Stiffness optimization for two-armed robotic sculpting," *Industrial Robot: An International Journal*, vol. 35, no. 1, pp. 46–57, 2008.

[OMF+18] D. Ortenzi, R. Muthusamy, A. Freddi, A. Monteriù, and V. Kyrki, "Dual-arm cooperative manipulation under joint limit constraints," *Robotics and Autonomous Systems*, vol. 99, pp. 110–120, 2018.

[ÖSKK12] P. Ögren, C. Smith, Y. Karayiannidis, and D. Kragic, "A multi objective control approach to online dual arm manipulation1," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 747–752, 2012.

[PA93] Z.-X. Peng and N. Adachi, "Compliant motion control of kinematically redundant manipulators," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 6, pp. 831–836, 1993.

[PT16] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer handbook of robotics*. Springer, 2016, pp. 955–988.

[SBB+14] C. Schuetz, T. Buschmann, J. Baur, J. Pfaff, and H. Ulbrich, "Predictive online inverse kinematics for redundant manipulators," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 5056–5061.

[SCFP17] M. Sorour, A. Cherubini, P. Fraisse, and R. Passama, "Motion discontinuity-robust controller for steerable mobile robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 452–459, 2017.

[SD18] S. Stavridis and Z. Doulgeri, "Bimanual assembly of two parts with relative motion generation and task related optimization," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2018, pp. 7131–7136.

[Ser89] H. Seraji, "Configuration control of redundant manipulators: Theory and implementation," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 4, pp. 472–490, 1989.

[SKN+12] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[SPB11] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1283–1290.

[SPK10] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 483–501, 2010.

[SS91] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *proceeding of 5th International Conference on Advanced Robotics*, vol. 2, 1991, pp. 1211–1216.

[SS12] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[SSVO10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[TAHT02] T. Takubo, H. Arai, Y. Hayashibara, and K. Tanie, "Human-robot cooperative manipulation using a virtual nonholonomic constraint," *The Int. Journal of Robotics Research*, vol. 21, no. 5-6, pp. 541–553, 2002.

[TBBD09] E. Tatlicioglu, D. Braganza, T. C. Burg, and D. M. Dawson, "Adaptive control of redundant robot manipulators with sub-task objectives," *Robotica*, vol. 27, no. 6, pp. 873–881, 2009.

[TNF+18] S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Dual-arm relative tasks performance using sparse kinematic control," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2018.

[TNF+19] S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini, and D. Sallé, "Admittance control for collaborative dual-arm manipulation," in *Int. Conf. on Advanced Robotics, ICAR*, 2019.

[Tsa99] L.-W. Tsai, *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons, 1999.

[UD88] M. Uchiyama and P. Dauchez, "A symmetric hybrid position/force control scheme for the coordination of two robots," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 1988, pp. 350–356.

[WK93] D. Williams and O. Khatib, "The virtual linkage: A model for internal forces in multi-grasp manipulation," in *IEEE Int. Conf. on Robotics and Automation*, 1993, pp. 1025–1030.

[WSKÖ15] Y. Wang, C. Smith, Y. Karayiannidis, and P. Ögren, "Cooperative control of a serial-to-parallel structure using a virtual kinematic chain in a mobile dual-arm manipulation application," in *IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*, 2015, pp. 2372–2379.

[WVK+14] Y. Wang, F. Vina, Y. Karayiannidis, C. Smith, and P. Ogren, "Dual arm manipulation using constraint based programming," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 311–319, 2014.

[ZM07] Y. Zhang and S. Ma, "Minimum-energy redundancy resolution of robot manipulators unified by quadratic programming and its online solution," in *IEEE Int. Conf. on Mechatronics and Automation*, 2007, pp. 3232–3237.

[Zub15] A. Zube, "Cartesian nonlinear model predictive control of redundant manipulators considering obstacles," in *IEEE Int. Conf. on Industrial Technology (ICIT)*, 2015, pp. 137–142.