An admittance based hierarchical control framework for dual-arm cobots

Sonny Tarbouriech, Benjamin Navarro, Philippe Fraisse, André Crosnier, Andrea Cherubini, Damien Sallé

Abstract Dual-arm robotic platforms have solid arguments to match the growing need for versatility in the industry. Coupling the control of two manipulators for cooperative purposes enlarges the scope of feasible operations, while adding perception capabilities allows to navigate in dynamic environments. In this respect, we propose a complete online kinematic control framework for dual-arm robots operating in unstructured industrial settings. We base our approach on admittance control in the cooperative task space. Regulating internal and external efforts offer safe bimanual task execution and enables physical interaction. We implement a hierarchical quadratic programming architecture that applies a prioritization of tasks: most efforts are concentrated on the proper tracking of relative motions of the arms, which is the most critical for safety reasons. We demonstrate the performance of our framework through a "teaching-by-demonstration" experiment on the dual-arm mobile cobot BAZAR.

Keywords Dual-arm cooperation · Redundant manipulators · Compliant control

An admittance based hierarchical control framework for dual-arm cobots

Sonny Tarbouriech · Benjamin Navarro · Philippe Fraisse · André Crosnier · Andrea Cherubini · Damien Sallé

the date of receipt and acceptance should be inserted later

1 Introduction

The vision of industry in all 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 aims at reducing production lot sizes.

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

Dual-arm setups are attractive solutions to meet the growing need for versatility of production lines. Indeed, the cooperative motion of two manipulator arms expands robot capabilities to a wide range of operations including transportation of bulky objects [27] and realization of complex assembly tasks [10][21].

However, despite their undeniable usefulness, systems with a large number of degrees of freedom (DOF) require an advanced control architecture to make adequate use of their potential. In particular, dual-arm cooperation leads to some specific considerations:

1. Synchronization of the arms has to be precisely handled to provide coordinated motions.
2. When manipulating a single object with the two arms, a closed kinematic chain is formed. Then, internal wrenches appear in the system and may create damages if not correctly managed.

While still topical, these issues have been addressed in the literature for many years [26]. A key element which emerges from previous research is that an appropriate task description is required to allow dual-arm collaboration. Initially

Sonny Tarbouriech · Benjamin Navarro · Philippe Fraisse · André Crosnier · Andrea Cherubini
LIRMM, Université de Montpellier, CNRS, Montpellier, France.
E-mail: firstname.lastname@lirmm.fr

Damien Sallé
Tecnalia Research and Innovation, Industry and Transport Division.
E-mail: damien.salle@tecnalia.com

introduced in [6], the cooperative task-space representation has been commonly adopted when dealing with dual-arm robots. Based on this approach, it is possible to define a task by way of meaningful variables and kinematically solve it using a dedicated Jacobian matrix [15].

Various control frameworks have derived from this paradigm. Indeed, high redundancy of dual-arm systems leaves room for optimization of any arbitrary criteria. For instance, the remaining DOF have been exploited to increase manipulability in [9] or to satisfy joint constraints in [14][20]. These approaches establish a strict hierarchy between tasks, i.e. tasks of lower priority do not affect the performance of the highest priority task, since they are projected in the null space of the kinematic Jacobian [7]. However, hard constraints (e.g. physical limits of the robot) cannot be explicitly handled.

A solution to the inverse kinematics problem for redundant robots under hard bounds has been formulated in [12][11] by introducing the Saturation in the Null Space (SNS) algorithm. The idea is to disable the most critical joints (according to some criterion) and to redistribute the saturated contribution of these joints to ensure the satisfaction of all joint constraints. When it is not possible to fully achieve the task, some scaling is applied on the cost function to keep following the desired path (at reduced speed). However, this algorithm considers inequality constraints at the joint level only.

To deal with hard constraints at both the joint and task level, a convenient and powerful approach, which has been widely adopted in recent works, consists in solving a Quadratic Programming (QP) optimization problem. QP solvers allow to find the extrema of a convex cost function while searching in an admissible solution space defined by a set of linear constraints. Based on QP solvers, it is possible to solve a sequence of prioritized tasks using a hierarchical quadratic programming (HQP) architecture [16]. Similar to the projection on the null space of the Jacobian [7], this method applies a strict hierarchy of tasks. In concrete terms, the process ensures that tasks of lower priority do not affect the performance of the highest priority task. The HQP algorithm has proved effective in solving problems of high complexity such as inverse dynamics for anthropomorphic systems [23] or whole-body dynamic motion control of humanoid robots [24].

In [25], the authors proposed a HQP framework for dual-arm robots performing relative tasks. The proposed framework allows to optimize several criteria 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 the work difficult to evaluate.

Letting robots physically interact with humans for industrial purposes is a significant asset [4]. Combining the strength of robots (accuracy, repeatability, efficiency) with human intelligence allows to rapidly adapt to changes on production lines.

Recently, the cooperative task space representation has been used to perform physical human-robot collaboration. In [2], Compliant Movement primitives [8] 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 [18][19], a cooperative control scheme based on an impedance law allows to perform kinesthetic guiding operations. Adaptation of the stiffness along the trajectory provides more accuracy to the human co-worker during critical parts of the task.

In this paper, we propose a complete framework for programming collaborative tasks with dual-arm robots, which is summarized in Section 2. The main contributions of this work are:

- An improved kinematic representation for cooperative dual-arm robot. It consists of a robust and straightforward description of kinematic tasks and of a wrench interpretation at the cooperative task space level, as presented in Section 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 4.
- An inverse kinematics strategy for solving cooperative tasks performed by dual-arm robots. Based on a hierarchical Quadratic Programming (HQP) architecture, our strategy gives more strenght to critical aspects of bimanual operations. Details are given in Section 5

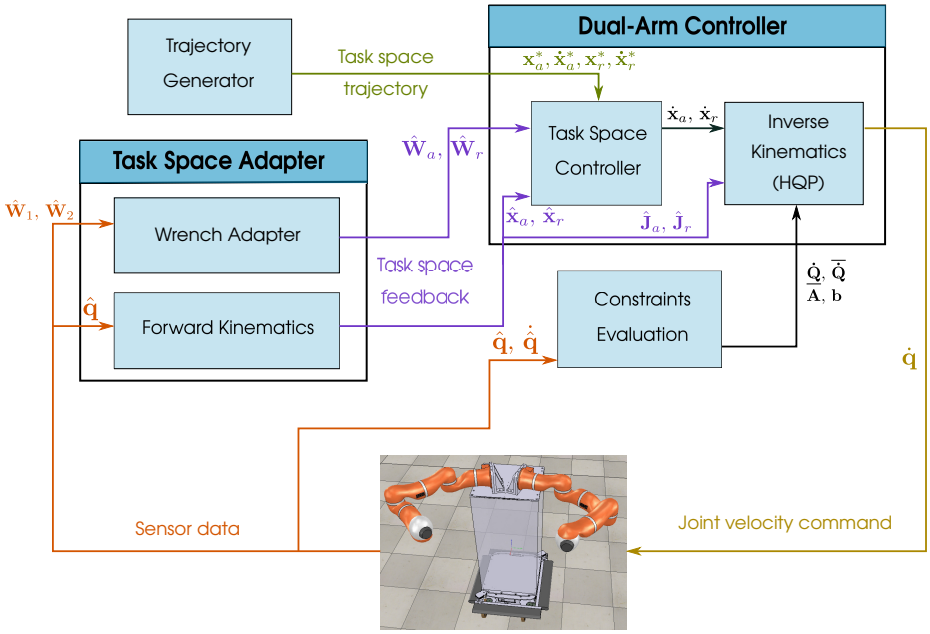


Fig. 1: Dual-arm kinematic control scheme. Based on the cooperative task representation [6], the absolute (subscript 'a') and relative (subscript 'r') tasks are controlled. The *Task Space Adapter* converts measurements (designated with hat symbol) of joint positions $\hat{\mathbf{q}}$ and wrench $\hat{\mathbf{W}}_1$, $\hat{\mathbf{W}}_2$ at the wrist of each arm into task space poses $\hat{\mathbf{x}}_a$, $\hat{\mathbf{x}}_r$, wrenches $\hat{\mathbf{W}}_a$, $\hat{\mathbf{W}}_r$ and extracts task space Jacobian matrices $\hat{\mathbf{J}}_a$, $\hat{\mathbf{J}}_r$. The *Dual-arm Controller* then generates the command at the task space level $\dot{\mathbf{x}}_a$, $\dot{\mathbf{x}}_r$ from these feedback values and the desired (superscript '*') task space trajectory. Finally, the *Inverse Kinematics* block provides the joint velocity command $\dot{\mathbf{q}}$ sent to the robot.

2 Overview of the approach

The framework relies on the online closed-loop control scheme depicted in Fig. 1.

We have adopted the following notations in this document: $\mathbf{x}$ is a vector designating the pose (position and orientation) of a control point. Depending on the convention used to express orientations, $\mathbf{x}$ is usually a 6D or 7D vector. The twist (translational and rotational velocity) of a control point is referred as $\dot{\mathbf{x}}$ and is a 6D vector. Dual-arm joint positions and velocities are expressed through vectors $\mathbf{q}$ and $\dot{\mathbf{q}}$, respectively. These vectors concatenate joint values of the two arms, which can be of different size, in the following way: let *Arm 1* have M DOF and *Arm 2* have N DOF, then the corresponding joint position and velocity vectors are respectively defined as $\mathbf{q} = [q_{11}, \ldots, q_{1M}, q_{21}, \ldots, q_{2N}]^T$ and $\dot{\mathbf{q}} = [\dot{q}_{11}, \ldots, \dot{q}_{1M}, \dot{q}_{21}, \ldots, \dot{q}_{2N}]^T$ which results in $M + N$ dimensional vectors. Note that the choice of *Arm 1* and *Arm 2* is arbitrary.

The presented strategy is based on admittance control in the operational space. Wrench measurements at the tip of each arm are input into the dual-arm controller. It allows 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. Effort management is essential for industrial operations with high payloads, for which the risk of breakage is not negligible. Also, this sensor perception opens the way to physical human-robot interaction.

We use the so-called cooperative task representation[6] which fully characterizes the operational space for bi-manual cooperative control. It allows to specify the operations in terms of absolute task (i.e. expressing the pose of any frame in space with respect to a fixed world frame) and relative task (i.e. expressing the pose of one end-effector with respect to the other), which are geometrically meaningful motion variables in this context. Thanks to this formalism, it is easy to adapt control algorithms originally designed for single manipulators, to dual-arm systems.

From the task specification (e.g. waypoints to reach), a trajectory is generated and corrected with respect to task space feedback to compute the task space control law.

To figure out the desired joint space command to be sent to the dual-arm platform, we implement an inverse kinematics process based on Hierarchical Quadratic Programming (HQP). Priority is given to the relative task (to ensure the proper manipulation of objects) while remaining DOF solve the absolute task. The HQP includes a set of hard constraints to be fulfilled at all times, and defined both at joint space level (joint limits) and task space level (collision avoidance).

Referring to Fig. 1, we detail the role of each component. First, the *Trajectory Generator* provides the desired task space poses $\mathbf{x}_a^*$, $\mathbf{x}_r^*$ and velocities $\dot{\mathbf{x}}_a^*$, $\dot{\mathbf{x}}_r^*$. Concurrently, the *Task Space Adapter* is in charge of converting feedback data coming from the robot into relevant information in the task space. This involves two computations which are treated in parallel:

- Joint position feedback $\hat{\mathbf{q}}$ is interpreted by the *Forward Kinematics* process to evaluate the current state in the task space: it outputs the poses $\hat{\mathbf{x}}_a$, $\hat{\mathbf{x}}_r$ and Jacobian matrices $\hat{\mathbf{J}}_a$, $\hat{\mathbf{J}}_r$ associated with the absolute and relative tasks, respectively.

– Wrenches $\hat{\mathbf{W}}_1$, $\hat{\mathbf{W}}_2$ (i.e., force and torques) measured at the tip of each end-effector are transformed by the *Wrench Adapter* block from their specific sensor reference frame to the task space through vectors $\hat{\mathbf{W}}_a$, $\hat{\mathbf{W}}_r$.

The task space trajectory and the task space feedback (from the *Task Space Adapter*) are both sent to the *Dual-arm Controller* which delivers the desired joint velocity to the robot. This operation is decomposed in two successive steps:

– The *Task Space Command* block delivers the velocity commands for the absolute $\dot{\mathbf{x}}_a$ and relative $\dot{\mathbf{x}}_r$ tasks based on the admittance control law (presented in Section 4). The process can be tuned for each task variable to choose between admittance, position, force, or damping control modes.
– Finally, the *Inverse Kinematics* block outputs the joint velocity vector $\dot{\mathbf{q}}$ sent to the robot. The cost function of the optimization problem requires the task space command vectors $\dot{\mathbf{x}}_a$, $\dot{\mathbf{x}}_r$ and the associated Jacobian matrices $\hat{\mathbf{J}}_a$, $\hat{\mathbf{J}}_r$. Joint velocity bounds and task space limits are gathered in the inequality constraint defined by $\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}$, which restricts the set of valid solutions.

3 Kinematic considerations for cooperative dual-arm robots

This section is dedicated to the *Task Space Adapter* module from Fig. 1 which is in charge of converting the data coming from sensors into information usable in the task space.

3.1 The cooperative task space

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. 4):

– 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 an homogeneous transformation matrix $\mathbf{T}_{ctrl}^{ref}$ expressing the pose of the control frame with respect to the reference frame.

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. Indeed, from a task description point of view, it is not convenient to specify the motion of each end-effector as they are not directly related to the purpose of dual-arm collaborative operations. More importantly, the prioritization strategy presented in Section 5, which allows safe manipulation of objects, cannot be applied with a dissociated management of the arms. In [6], Chiacchio et al. introduced a kinematic representation for dual-arm systems performing cooperative operations. 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. 2:

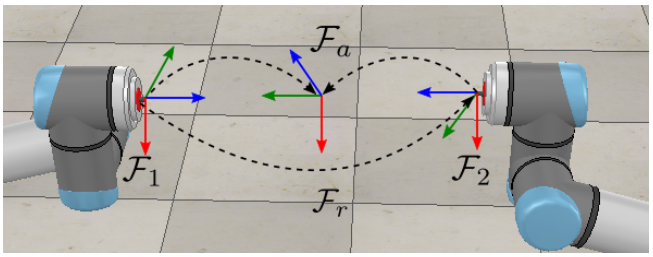


Fig. 2: Representation of the cooperative tasks.

– 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 the two end-effector poses, such as :

$$\begin{aligned} \mathbf{p}_a &= \frac{\mathbf{p}_2 + \mathbf{p}_1}{2} \\ \mathbf{R}_a &= \mathbf{R}_1 \mathbf{R}\left\{n_{12}, \frac{\phi_{12}}{2}\right\}, \end{aligned} \tag{1}$$

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 $\mathbf{n}$. In (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 corresponding transformation matrix is $\mathbf{T}_a$. During dual-arm manipulation, the role of the absolute task is to control the trajectory of the grasped object in the space.

– 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:

$$\begin{aligned} \mathbf{p}_r &= \mathbf{p}_2 - \mathbf{p}_1 \\ \mathbf{R}_r &= \mathbf{R}_2^1, \end{aligned} \tag{2}$$

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.
The corresponding transformation matrix is $\mathbf{T}_r$. The relative task is particularly suited for managing the grasp of an object or for performing assembling operations [1].

Keeping the same idea, we decided to modify the absolute task definition from [6] given in (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. 3, 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

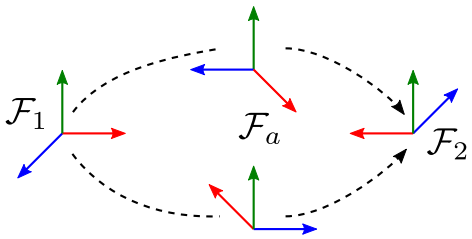
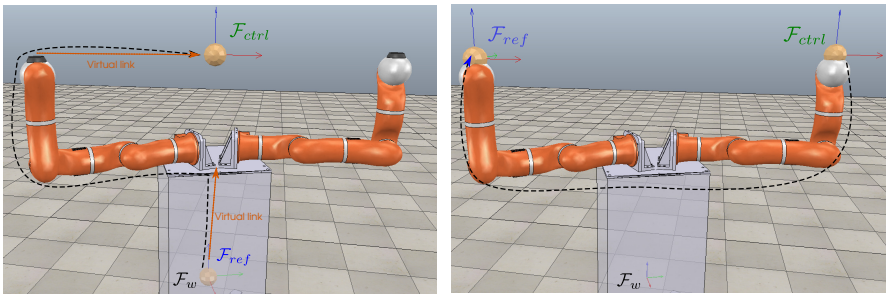


Fig. 3: Discontinuity of the absolute task orientation. In the original version, there exist two solutions for the orientation of $\mathcal{F}_a$ since it is obtained after applying half the rotation needed to align $\mathcal{F}_1$ with $\mathcal{F}_2$.

use, the computation of $\mathbf{R}_a$ may switch from one solution to the other leading to instabilities in the control 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 point, it is more convenient to attach the absolute frame to it.



(a) The absolute task (b) The relative task

Fig. 4: 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 frames 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 ($\mathcal{F}_{ref}$).

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. 4a). 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, (1) becomes:

$$\begin{aligned} \mathbf{p}_a &= \mathbf{p}_1^w + \mathbf{R}_1^w \mathbf{p}_a^1 \\ \mathbf{R}_a &= \mathbf{R}_1^w \mathbf{R}_a^1, \end{aligned} \tag{3}$$

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, the discontinuity problem disappears.

The relative and absolute task are depicted in Fig. 4.

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 $\mathbf{T}_r = \mathbf{T}_2^1$.

3.2 Wrench in the task space

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 to perceive external forces applied to the manipulated object. However, to properly exploit this information, it has to be meaningful in the task space.

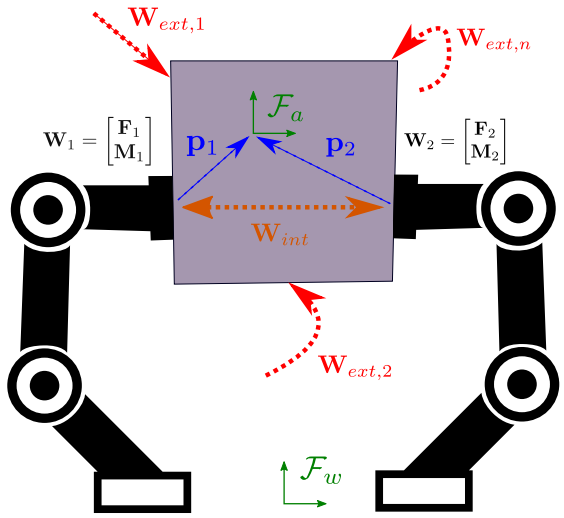


Fig. 5: Dual-arm manipulation of a rigid object. Wrenches $\mathbf{W}_1$ and $\mathbf{W}_2$ are measured at the wrist of each arm. They result from a set of n external wrenches $\mathbf{W}_{ext,k}$ ($k \in [0, 1, 2, \ldots, n]$) exerted on the object and from the internal wrench $\mathbf{W}_{int}$ induced by the tight grasp. An absolute task wrench is computed by projecting the individual wrenches in the absolute task frame $\mathcal{F}_a$ using the "virtual sticks" [26] represented by vector $\mathbf{p}_1$ and $\mathbf{p}_2$.

Let us assume that an object is grasped by the two end-effectors, as depicted in Fig. 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:

$$\mathbf{M}_o(\mathbf{x_o})\dot{\boldsymbol{\upsilon}}_\mathbf{o} + \mathbf{C}_o(\mathbf{x_o}, \boldsymbol{\upsilon_o}) + \mathbf{G}_o(\mathbf{x_o})\boldsymbol{\eta} = \mathbf{W}_{o,ext}, \tag{4}$$

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 the Grasp matrix [22] 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

$$\mathbf{G}_{oi}(\mathbf{x_o}) = \mathbf{H}_{oi}\tilde{\mathbf{G}}_{oi}(\mathbf{x_o}). \tag{5}$$

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, $\mathbf{H}_{oi} = \mathbf{I}_{6\times6}$. Note that we could easily extend our method to other types of contacts (e.g. sliding contacts) but this is beyond the scope of this work.

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

$$\tilde{\mathbf{G}}_{oi}(\mathbf{x_o}) = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{S}(\mathbf{p}_i^w) & \mathbf{I}_{3\times3}\end{bmatrix}, \tag{6}$$

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 (4), $\boldsymbol{\eta} = \begin{bmatrix}\boldsymbol{\eta}_1^T \ \boldsymbol{\eta}_2^T\end{bmatrix}^T$ 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:

$$\boldsymbol{\eta}_i = \mathbf{W}_i = \begin{bmatrix}\mathbf{F}_i \\ \mathbf{M}_i\end{bmatrix}. \tag{7}$$

In the context of physical-human robot interaction where the robot has to operate at low speed, the inertia terms are negligible and the system can be considered as quasistatic. With this assumption, (4) is reduced to:

$$\mathbf{W}_{o,ext} = \mathbf{G}_o(\mathbf{x_o})\boldsymbol{\eta}, \tag{8}$$

Using (5), (6) and (7), the dynamic equation leads to:

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

$$\begin{bmatrix}\mathbf{F}_{o,ext}\\ \mathbf{M}_{o,ext}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3}\\ \mathbf{S}(\mathbf{p}_1^w) & \mathbf{I}_{3\times3} & \mathbf{S}(\mathbf{p}_2^w) & \mathbf{I}_{3\times3}\end{bmatrix}\begin{bmatrix}\mathbf{F}_1\\ \mathbf{M}_1\\ \mathbf{F}_2\\ \mathbf{M}_2\end{bmatrix}, \tag{9}$$

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 $\mathcal{F}_w$):

$$\begin{cases}\mathbf{F}_a = \mathbf{F}_1^w + \mathbf{F}_2^w,\\ \mathbf{M}_a = \mathbf{M}_1^w + \mathbf{p}_1^w \times \mathbf{F}_1^w + \mathbf{M}_2^w + \mathbf{p}_2^w \times \mathbf{F}_2^w,\end{cases} \tag{10}$$

which is equivalent to the formulation introduced in [26], where $\mathbf{p}_1^w$, $\mathbf{p}_2^w$, are vectors representing the *virtual sticks*.

By taking a closer look to (8), 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 [22]. Let $N(\mathbf{G}_o)$ denote the null space projection of the grasping matrix, then from (8) one can write:

$$\boldsymbol{\eta} = \mathbf{G}_o(\mathbf{x_o})^+\mathbf{W}_{o,ext} + N(\mathbf{G}_o)\mathbf{W}_{int} \tag{11}$$

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 [26], 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*):

$$\begin{cases}\mathbf{F}_r = \dfrac{1}{2}(\mathbf{F}_2^1 - \mathbf{F}_1^1),\\ \mathbf{M}_r = \dfrac{1}{2}(\mathbf{M}_2^1 - \mathbf{M}_1^1).\end{cases} \tag{12}$$

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 $\bar{\mathbf{W}}_r = \mathbf{W}_r$.

Note that absolute and relative task wrenches are respectively equivalent to external and internal wrenches. In the rest of this document, we keep the first notation to be homogeneous with respect to the task representation.

4 Closed-loop admittance control for physical interaction

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 processes. In this section, we focus on the *Task Space Command* generation while the *Inverse Kinematics* resolution will be the subject of Section 5.

The task space command aims at achieving a cooperative kinematic task while interacting with the environment. 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. 6.

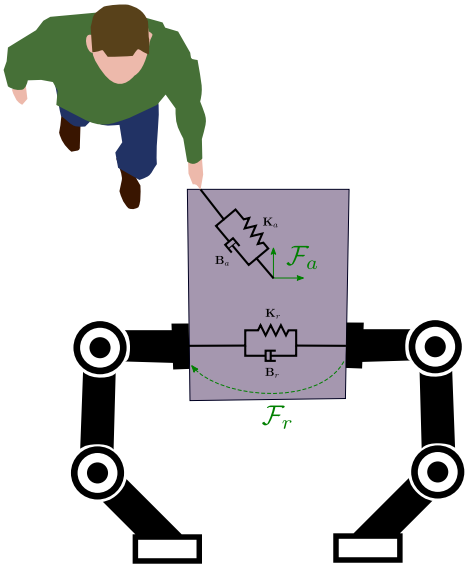


Fig. 6: Representation of the virtual spring-damper systems attached to the relative and absolute task control frames.

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:

$$\mathbf{W} = \mathbf{K}\mathbf{x} + \mathbf{B}\dot{\mathbf{x}}. \tag{13}$$

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 *):

$$\mathbf{W}^* = \mathbf{K}\mathbf{x}^* + \mathbf{B}\dot{\mathbf{x}}^*. \tag{14}$$

By subtracting (14) from (13), we obtain the closed-loop impedance control law:

$$\Delta\mathbf{W} = \mathbf{K}\Delta\mathbf{x} + \mathbf{B}\Delta\dot{\mathbf{x}}, \tag{15}$$

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, (15) can be rewritten in the following form:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}^* + \mathbf{B}^{-1}(\Delta\mathbf{W} - \mathbf{K}\Delta\mathbf{x}). \tag{16}$$

For each task variable, a specific behavior can be obtained. In particular, we can design three control modes from (16) (without loss of generality, every term is now expressed as a scalar value):

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

$$\dot{x} = \dot{x}^* - B^{-1}K\Delta x. \tag{17}$$

  with $B^{-1}K \geq 0$ 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^*.

$$\dot{x} = B^{-1}\Delta W. \tag{18}$$

- **Damping control**: this is a particular case of the force control in which $W^* = 0$. 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.

$$\dot{x} = B^{-1}W. \tag{19}$$

Except for these particular cases, the general admittance control equation (16) can be tuned to give to the robot the desired compliant behavior.

5 Inverse kinematics resolution

In the previous section, we detailed how to compute the task space velocity command $\dot{\mathbf{x}}$ in (20). This section is dedicated to the inverse kinematics resolution, whose aim is to convert the specification of the task in the operational space into the joint space, where actuation takes place.

There exists a relation between joint velocities $\dot{\mathbf{q}}$ and task space velocities $\dot{\mathbf{x}}$ through the Jacobian matrix $\mathbf{J}$:

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}. \tag{20}$$

In the context of a redundant system subject to constraints, a common approach is to solve an optimization problem. It allows to find the best among infinite solutions given: a cost function, and an admissible space defined by equality and inequality constraints.

5.1 Hard constraints consideration

The solution space for the task command described above is restricted by a set of constraints on the robot's motion. These can be distinguished into two types: first, physical limits of the robot, which lead to the specification of admissible joint ranges. This applies at the position, velocity and acceleration level. Second, constraints also appear in the task space perspective. For instance, one might set velocity and/or acceleration limits of the control frame. Finally, obstacles in the environment can be considered under the form of collision avoidance constraints.

Details on how to compute these constraints do not fall within the scope of this work. For the remainder of this study, let us just point out that any constraint can be specified in function of joint velocities under the standard form of linear inequality equations:

$$\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \tag{21}$$

5.2 Task prioritization

A standard method to obtain the desired $\dot{\mathbf{q}}$ is to use a quadratic program. This minimizes the 2-norm of the cost function and allows defining a set of constraints to be satisfied at any time. For the inverse kinematics problem, it results in:

$$\begin{aligned} \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \quad \mathbf{C}\dot{\mathbf{q}} = \mathbf{d} \end{aligned} \tag{22}$$

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

We denote by $\boldsymbol{\epsilon}$ the residual error on the kinematic task after solving (22):

$$\boldsymbol{\epsilon} = \mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}. \tag{23}$$

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

Considering the cooperative dual-arm task space, both the relative and absolute tasks have to be satisfied at the same time (see Section 3.1). However, a relevant choice is to prioritize the relative task over the absolute one. Indeed, it is critical to ensure the relative task fulfillment during bimanual manipulation of an object, to avoid the generation of undesired internal stress and damage the object and/or the robot.

Thus, we propose a 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. Applied to the cooperative dual-arm task space and taking into account constraints, the highest priority task is solved through the following optimization problem:

$$\begin{aligned} \dot{\mathbf{q}}_1 \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_r\dot{\mathbf{q}} - \dot{\mathbf{x}}_r\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \end{aligned} \tag{24}$$

where $\dot{\mathbf{q}}$ refers here to the vector concatenating the joint velocities of the two arms, such as $\dot{\mathbf{q}} = \begin{bmatrix}\dot{\mathbf{q}}_1 & \dot{\mathbf{q}}_2\end{bmatrix}^T$, and $\mathbf{J}_r$, $\dot{\mathbf{x}}_r$, 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:

$$\begin{aligned} \dot{\mathbf{q}}_2 \in \min_{\dot{\mathbf{q}}} \quad & \|\mathbf{J}_a\dot{\mathbf{q}} - \dot{\mathbf{x}}_a\|_2 \\ \text{s.t.} \quad & \mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \\ & \mathbf{J}_r\dot{\mathbf{q}} = \mathbf{J}_r\dot{\mathbf{q}}_1, \end{aligned} \tag{25}$$

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 equality 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 (24). If no 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 (25).

6 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.

6.1 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. 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 to repeat 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.

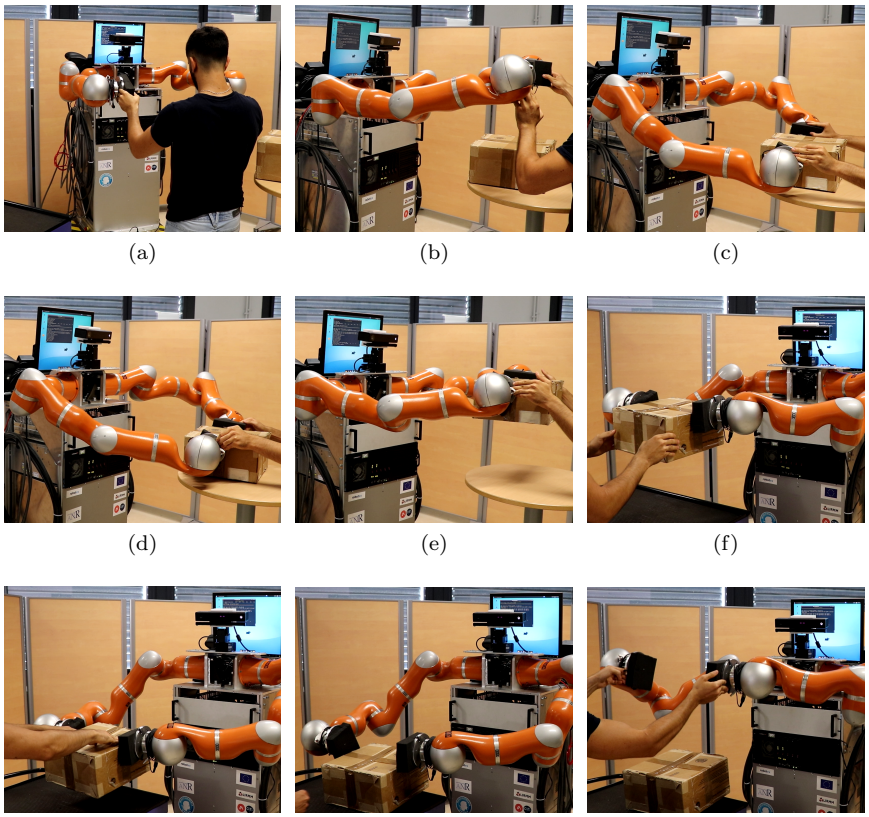


Fig. 7: Some snapshots taken during the teaching phase of the experiment: (a) The operator moves the whole dual-arm platform to a workable location using the mobile base. (b) The operator brings the robot to the first waypoint. (c) The robot is ready to grasp the box. (d) The operators teaches the internal force required to maintain the object. (e) The robot lifts the box to a 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.

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's skeleton in the 3D Cartesian space. We apply a repulsive action on the absolute task based on the method presented in [13]. We chose to take into account the repulsive action by adding the resulting repulsive translational velocity vector to the translational velocity command

computed for the absolute task frame using (16). This allows to prevent from being too close to unpredictable obstacles without interrupting the task.

In this experiment, a dual-arm robot is mounted on an omnidirectional mobile base, allowing to enlarge the admissible set of motions in the workspace. Taking into account the relatively low reactivity of the mobile part, we establish a prioritized control structure: only the arms are engaged as long as they are sufficient to fulfill the task. Whenever some error remains on the absolute task after solving (25) 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 leaves the arms' reachable space. The mobile base control approach is not part of the core contributions of this work and will not be treated in more details in this paper.

6.2 Setup

The setup consists of dual-arm mobile cobot BAZAR [5]. BAZAR is equipped with two 7-DOF Kuka LWR4 arms attached on a Neobotix MPO-700 omnidirectional mobile base running at $T_{mob} = 20\,\text{ms}$. Force/torque sensors are mounted at each arm wrist. All experiments are performed on a computer with an Intel(R) Xeon(R) E5-2620 v3 CPU running Linux with the PREEMPT-RT patch. The Fast Research Interface Library (FRI)[2] is used to communicate with the Kuka arms at a time rate $T_{arms} = 5\,\text{ms}$.

Our approach has been implemented in C++ in the RKCL (Robot Kinematic Control Library) framework[3]. This library implements the online closed-loop control scheme depicted in Fig. 1. We designed the ecosystem so that distinct hardware components can perform together, although their control rates are different. To make this possible, we separate the execution of communication drivers from the controller by parallelization. The case of BAZAR is illustrated in Fig. 8. The ecosystem is composed of:

- A unique kinematic control loop which executes the different processes sequentially.
- Several drivers in charge of exchanging data between the hardware components and the controller.

each process 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 T_{ctrl} for the kinematic control loop, however, is arbitrary and left free to the user. The best performances are obtained when the control loop time step matches the highest frequency driver so that its bandwith is not artificially limited. In this case, we set $T_{ctrl} = T_{arms}$.

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; the same gains are used for damping controlled variables but the stiffness term is removed

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

[3] `https://gite.lirmm.fr/rkcl`

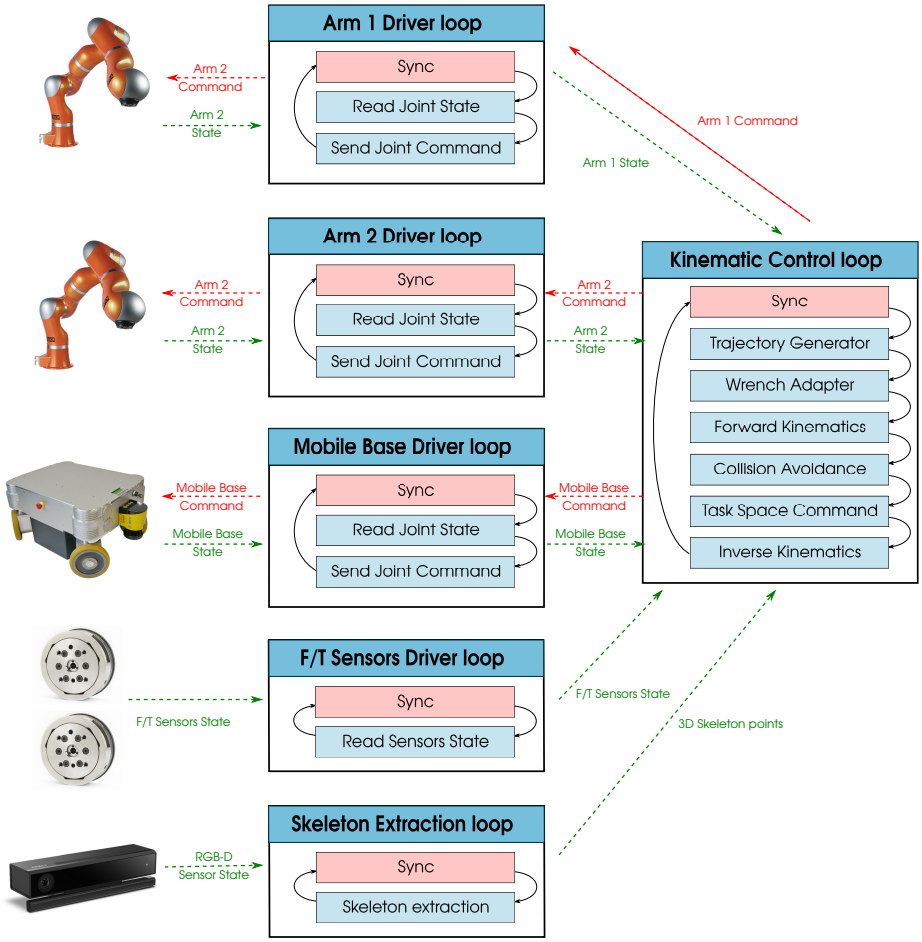


Fig. 8: Parallelization of driver/controller processors in RKCL for dual-arm mobile robot BAZAR. each process is executed independently and at its proper time rate, managed by the *Sync* block. When new data are available (either state for the controller or command for the drivers) they are sent to the concerned processors which will put them into effect at the next iteration.

($K = 0$); 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.

During the replay phase, the Reflexxes Motion Library [17] is used to generate trajectories between the stored waypoints. To set up the trajectory profile, we specify the following parameters: maximum translational velocity $0.5\mathrm{m\,s^{-1}}$, maximum rotational velocity $0.5\mathrm{rad\,s^{-1}}$, maximum translational acceleration $0.2\mathrm{m\,s^{-2}}$, and maximum rotational acceleration $0.2\mathrm{rad\,s^{-2}}$

We use computer vision to estimate online the location of the human operator. We rely on the OpenPose library [3] which extracts the set of 2D points composing the skeleton of the persons present in a given color image, as shown in Fig. 9. By

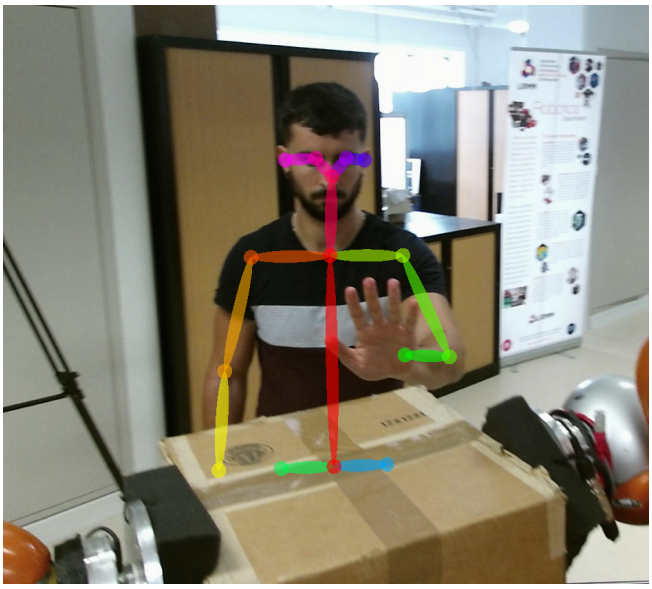


Fig. 9: View from the BAZAR Microsoft Kinect. The human is detected and his skeleton projected in the 3D space using depth. During the replay phase, people entering the robot's workspace are treated as obstacles to avoid.

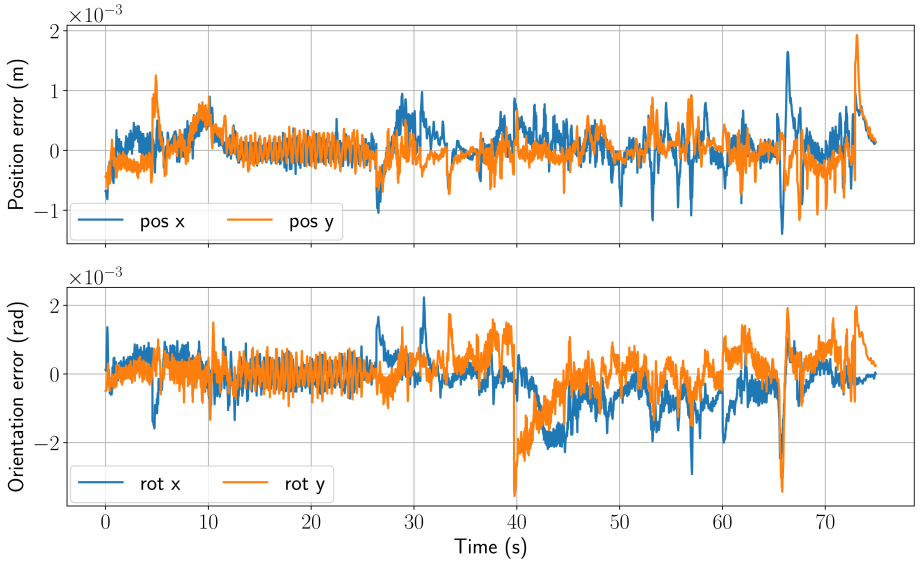
using a Microsoft Kinect V2 RGB-D camera, we can reproject these 2D points in 3D using the depth information provided by the sensor. Each point is treated as an obstacle in the replay phase. By using an NVIDIA GTX 1080 Ti GPU, we achieve around 7 estimations per seconds, which corresponds to a time rate $T_{vis} = 142\,\text{ms}$.

6.3 Results

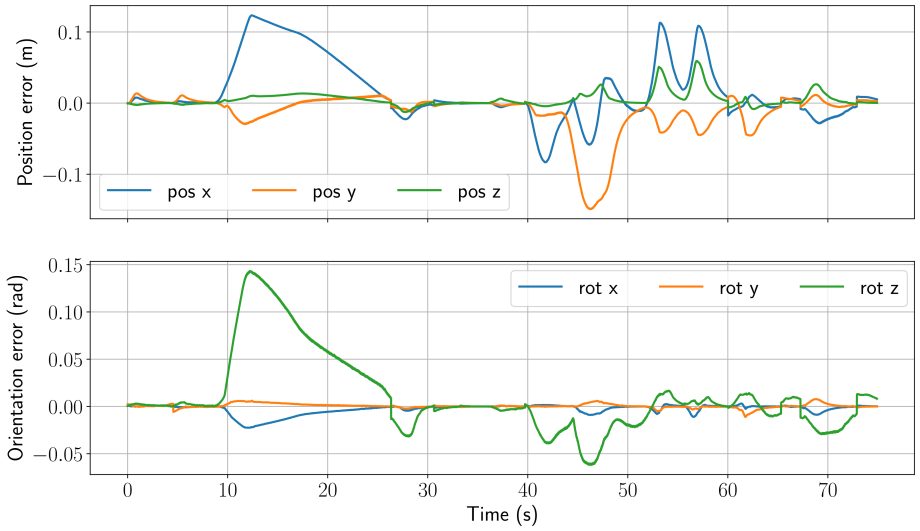
To assess the performance of our dual-arm collaborative framework RKCL and demonstrate that it provides a suitable solution for flexible industrial applications, let us first focus on the reproducibility of the taught operations. Fig. 10 shows the logged error for both the relative and absolute tasks during the whole replay phase. For the 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 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



(a) Relative task pose error



(b) Absolute task pose error

Fig. 10: 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.

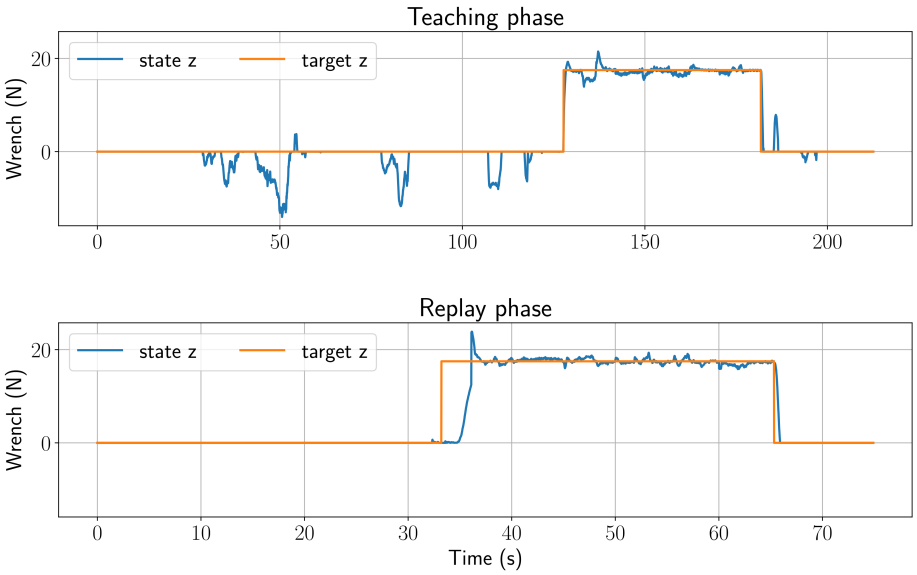


Fig. 11: Evolution of the relative task state and target wrench (z component only). The reference value stored during the teaching phase is properly replicated during the replay phase, to safely manipulate the box.

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. 11, 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. 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.

7 Conclusion

In the prospect of bringing more flexibility to industrial robotic setups, this paper introduces a complete framework for online kinematic control of dual-arm robots. 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. To do so, wrenches measured at the tip of each arm are adequately transformed in the cooperative task space to provide the input of a closed-loop admittance controller. This allows to regulate internal constraints and to interact with the environment. The task prioritization strategy aims at devoting maximum efforts to satisfy the relative task requirements; in particular, this ensures safe manipulation of objects. We spotlight our dual-arm collaborative 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. Future works will focus on exploring predictive strategies for the online redundancy resolution of dual-arm robots. Indeed, switching from a purely local approach to a predictive model would result in a more efficient handling of constraints and would probably provide overall better performances.

ACKNOWLEDGMENT

This work was conducted in the context of the project SHERLOCK, which has received funding from the EU Horizon 2020 research and innovation programme under grant agreement 820689.

References

1. Almeida, D., Vina, F.E., Karayiannidis, Y.: Bimanual folding assembly: Switched control and contact point estimation. In: IEEE-RAS Int. Conf. on Humanoid Robots (2016)
2. Batinica, A., Nemec, B., Ude, A., Raković, M., Gams, A.: Compliant movement primitives in a bimanual setting. In: IEEE-RAS Int. Conf. on Humanoid Robots, pp. 365–371 (2017)
3. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y.: OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In: arXiv preprint arXiv:1812.08008 (2018)
4. Cherubini, A., Passama, R., Crosnier, A., Lasnier, A., Fraisse, P.: Collaborative manufacturing with physical human–robot interaction. Robotics and Computer-Integrated Manufacturing **40**, 1–13 (2016)
5. Cherubini, A., Passama, R., Navarro, B., Sorour, M., Khelloufi, A., Mazhar, O., Tarbouriech, S., Zhu, J., Tempier, O., Crosnier, A., et al.: A collaborative robot for the factory of the future: Bazar. The Int. Journal of Advanced Manufacturing Technology p. (to appear) (2019)
6. Chiacchio, P., Chiaverini, S., Siciliano, B.: Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system. Journal of dynamic systems, measurement, and control **118**(4), 691–697 (1996)

7. Chiaverini, S.: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Trans. on Robotics and Automation **13**(3), 398–410 (1997)
8. Deniša, M., Gams, A., Ude, A., Petrič, T.: Learning compliant movement primitives through demonstration and statistical generalization. IEEE/ASME transactions on mechatronics **21**(5), 2581–2594 (2015)
9. Faroni, M., Beschi, M., Visioli, A., Tosatti, L.M.: A global approach to manipulability optimisation for a dual-arm manipulator. In: IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA) (2016)
10. Felip, J., Morales, A.: A solution for the cap unscrewing task with a dual arm sensor-based system. In: IEEE-RAS Int. Conf. on Humanoid Robots (2014)
11. Flacco, F., De Luca, A., Khatib, O.: Motion control of redundant robots under joint constraints: Saturation in the null space. In: IEEE Int. Conf. on Robotics and Automation, pp. 285–292 (2012)
12. Flacco, F., De Luca, A., Khatib, O.: Prioritized multi-task motion control of redundant robots under hard joint constraints. In: IEEE/RSJ Int. Conf. on Robots and Intelligent Systems, pp. 3970–3977 (2012)
13. Flacco, F., Kröger, T., De Luca, A., Khatib, O.: A depth space approach to human-robot collision avoidance. In: IEEE Int. Conf. on Robotics and Automation, pp. 338–345 (2012)
14. Hu, Y., Huang, B., Yang, G.Z.: 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)
15. Jamisola, R.S., Roberts, R.G.: A more compact expression of relative jacobian based on individual manipulator jacobians. Robotics and Autonomous Systems **63**(P1), 158–164 (2015)
16. Kanoun, O., Lamiraux, F., Wieber, P.B., Kanehiro, F., Yoshida, E., Laumond, J.P.: Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In: IEEE Int. Conf. on Robotics and Automation, pp. 2939–2944 (2009)
17. Kröger, T.: Opening the door to new sensor-based robot applications — The Reflexxes Motion Libraries. In: IEEE Int. Conf. on Robotics and Automation (2011)
18. Nemec, B., Likar, N., Gams, A., Ude, A.: Bimanual human robot cooperation with adaptive stiffness control. In: IEEE-RAS Int. Conf. on Humanoid Robots, pp. 607–613 (2016)
19. Nemec, B., Likar, N., Gams, A., Ude, A.: Adaptive human robot cooperation scheme for bimanual robots. In: Advances in Robot Kinematics, pp. 371–380. Springer (2018)
20. Ortenzi, D., Muthusamy, R., Freddi, A., Monteriù, A., Kyrki, V.: Dual-arm cooperative manipulation under joint limit constraints. Robotics and Autonomous Systems **99**, 110–120 (2018)
21. Polverini, M.P., Zanchettin, A.M., Incocciati, F., Rocco, P.: Robust constraint-based robot control for bimanual cap rotation. In: IEEE/RSJ Int. Conf. on Robots and Intelligent Systems, pp. 4785–4790 (2017)
22. Prattichizzo, D., Trinkle, J.C.: Grasping. In: Springer handbook of robotics, pp. 955–988. Springer (2016)
23. Saab, L., Mansard, N., Keith, F., Fourquet, J.Y., Souères, P.: Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints. In: IEEE Int. Conf. on Robotics and Automation, pp. 1091–1096 (2011)
24. Saab, L., Ramos, O.E., Keith, F., Mansard, N., Soueres, P., Fourquet, J.Y.: Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Trans. on Robotics **29**(2), 346–362 (2013)
25. Stavridis, S., Doulgeri, Z.: Bimanual assembly of two parts with relative motion generation and task related optimization. In: IEEE/RSJ Int. Conf. on Robots and Intelligent Systems, pp. 7131–7136 (2018)
26. Uchiyama, M., Dauchez, P.: A symmetric hybrid position/force control scheme for the coordination of two robots. In: IEEE/RSJ Int. Conf. on Robots and Intelligent Systems, pp. 350–356 (1988)
27. Xian, Z., Lertkultanon, P., Pham, Q.C.: Closed-chain manipulation of large objects by multi-arm robotic systems. IEEE Robotics and Automation Letters **2**(4), 1832–1839 (2017)