Dual-arm relative tasks performance using sparse kinematic control
Sonny Tarbouriech, Benjamin Navarro, Philippe Fraisse, André Crosnier, Andrea Cherubini, Damien Sallé

To cite this version:

HAL Id: hal-01735462
https://hal.archives-ouvertes.fr/hal-01735462v2
Submitted on 14 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Dual-arm relative tasks performance using sparse kinematic control

Sonny Tarbouriech∗† Benjamin Navarro∗ Philippe Fraisse∗ André Crosnier∗ Andrea Cherubini∗ and Damien Sallé†

Abstract—To make production lines more flexible, dual-arm robots are good candidates to be deployed in autonomous assembly units. In this paper, we propose a sparse kinematic control strategy, that minimizes the number of joints actuated for a coordinated task between two arms. The control strategy is based on a hierarchical sparse QP architecture. We present experimental results that highlight the capability of this architecture to produce sparser motions (for an assembly task) than those obtained with standard controllers.

I. INTRODUCTION

The H2020 VERSATILE European Project† aims at contributing to the development of the “Factory of the Future”. In particular, flexibility and versatility are enhanced in production lines, by integrating dual-arm robotic platforms.

Indeed, using multiple cooperative robotic arms significantly increases the potential of manipulators [1]. This type of structures allows the manipulation of large and heavy objects [2] as well as the achievement of difficult tasks that are not feasible with a single arm [3], [4].

However, the resulting system becomes more complex from a control perspective. Arms synchronization has to be perfectly managed throughout a cooperative task, while intrinsic constraints arising from the dual-arm structure reduce the solution space, compromising proper task tracking.

Despite the high number of degrees of freedom (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.

In [5], Chiacchio et al. presented a kinematic representation for dual-arm systems performing coordinated tasks. Their idea was to consider the system as a single entity and to describe the task in terms of cooperative variables, defined by an absolute frame (computed as a function of the two end-effector poses, expressed in the world frame) and a relative frame (expression of one end-effector’s pose with respect to the other).

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 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 [6], sculpting [7] or welding [8].

This is an undesired behavior in industrial environments, as it increases the potential risk of collision with unexpected obstacles. Moreover, this generally results in a motion that is confusing for an operator working near the robot.

Exploiting redundancy for dual-arm systems has been the topic of several studies, in which available DOF were used for obstacle avoidance [9], to increase manipulability [10] or to satisfy joint constraints [11], [12]. These approaches make use of redundancy by defining a hierarchic prioritized task architecture based on the projection on the null space of the Jacobian [13]. This method guarantees that the tasks of lower priority do not affect the performance of the highest priority task. However, there is no explicit management of hard constraints, i.e. constraints that must be satisfied all the time.

A second issue with most approaches is that they impose the motion of all actuators. In fact, inverse kinematics is usually solved using the Jacobian pseudo-inverse; yet, on redundant robots, this solution activates more actuators than needed by the task, leading to non-economic movements. This is an undesired behavior in industrial environments, as it increases the potential risk of collision with unexpected obstacles. Moreover, this generally results in a motion that is confusing for an operator working near the robot.

Instead, an approach originally introduced in the signal processing field [14] aims at generating a sparse, or parsimonious, solution. In the context of robotics, a sparse solution is one that activates as few actuators as possible, reducing the overall motion and, consequently, the risk of unintended contact with the environment. It has also been shown that a parsimonious motion provides more natural human-like movements and is therefore more predictable.
from the operator point of view [15], [16].

In this paper, we propose a parsimonious kinematic control strategy for dual-arm manipulators performing relative tasks under joint limit constraints. Our approach is based on a hierarchy of tasks, implemented as a sequence of quadratic programs (QP) [17]. The main contributions of our framework are:

- The primary task (relative end-effector pose control) is formulated as an optimization problem which returns a sparse solution that minimizes the generated motion, in the actuators’ space.
- A secondary task with lower priority is occasionally activated to move joints away from their physical limits.
- Strict task priority is handled: lower-priority tasks are performed as much as possible without producing any effect on higher-priority tasks. Joint position and velocity limits are integrated as hard constraints to strictly respect the robot’s physical capabilities.

The paper is organized as follows. Section II formulates the kinematic control scheme for dual-arm coordinated motions. Section III introduces the parsimonious task-solving approach as an optimization problem. Our strategy to take into account joint limits is presented in Sect. IV. A description of the developed QP-based hierarchical architecture is given in Sect. V. Experimental results of a screwing scenario performed on a real dual-arm platform are reported in VI. Finally, Sect. VII ends the paper with concluding remarks and future perspectives.

II. DUAL-ARM KINEMATIC CONTROL LOOP

A. Task representation for coordinated motion

The dual-arm robot that we consider here is depicted in Fig. 1. For each manipulator \( i = 1, 2 \), \( p^i_r \) denotes the end-effector position and \( R^i_b \) the rotation matrix, both expressed in a common base frame (superscript "b").

The relative pose corresponds to the coordinates of one end-effector’s frame with respect to the other. If we consider \( F_1 \) as the reference frame, the relative position is given by:

\[
p^1_r = p^1_r.
\]

Similarly, the relative orientation can be defined with reference to frame \( F_1 \) in terms of the rotation matrix

\[
R^1_r = R^2_r.
\]

B. The relative Jacobian definition

The standard inverse kinematic problem consists in finding joint velocities \( \dot{q} \) to obtain a desired task space velocity \( \dot{x} \):

\[
\dot{x} = J \dot{q}.
\]

For a two-manipulator system with relative control, the velocity vectors in the joint and task spaces are defined as:

\[
\dot{x} = \begin{bmatrix} 1 \dot{v}^1_r \\ \dot{\omega}^1_r \end{bmatrix} \quad \text{and} \quad \dot{q} = \begin{bmatrix} q^1 \\ q^2 \end{bmatrix},
\]

with \( \dot{v}^1_r, \dot{\omega}^1_r \) respectively the linear and angular velocity of the task variables.

Solving the inverse kinematic problem using this formalism induces the definition of a relative Jacobian \( J \) that maps the joint velocities of the two robots to their relative motion in task-space. A first expression was proposed in [18]. Following formulations were derived from the Jacobians of each manipulator \( J^1_p \) and \( J^2_p \), leading to simpler and more efficient implementations [19], [20]. However, these expressions omit that the rotational velocity of the reference frame attached to one of the end-effectors can be non-null [21]. Recent work [22] addressed this issue and proposed:

\[
J = [-\Psi^1_r \Omega^1_b - \Omega^1_b \Omega^1_p \Omega^1_b ]^T,
\]

with

\[
\Psi^1_r = \begin{bmatrix} I & -S(p^1_r) \end{bmatrix}, \quad \Omega^1_b = \begin{bmatrix} R^1_b & 0 \\ 0 & R^1_b \end{bmatrix}.
\]

C. Closed-loop inverse kinematic problem

For closed-loop control, the inverse kinematic problem in (3) is reformulated as a tracking problem, whose input is the desired relative task velocity \( \dot{x}^* \) and whose outputs are the corresponding joint velocities \( \dot{q} \).

Let \( e \) denote the six-dimensional error between the desired and current task velocities, such as:

\[
\dot{e} = \dot{x}^* - \dot{x}.
\]

This vector can be partitioned into a position (subscript “p”) and an orientation (subscript “o”) part:

\[
e = \begin{bmatrix} e_p \\ e_o \end{bmatrix}.
\]

The definition of relative position error is straightforward:

\[
e_p = p^1_r - p^1_r.
\]

The definition of an orientation error \( e_o \) was given by Luh et al. [23] and is expressed here as:

\[
e_o = \frac{1}{2} \left( S(n^1_o) n^1_o + S(s^1_o) s^1_o + S(a^1_o) a^1_o \right).
\]

where \( (n, s, a) \) and \( (n^*, s^*, a^*) \) denote respectively the actual and desired unit vector triples of the task frame.

Using (3) yields :

\[
\dot{e} = \dot{x}^* - J \dot{q}.
\]

where \( J \) is defined in (5). To obtain an exponential convergence of the error:

\[
\dot{e} = -Ke, \quad K > 0,
\]

plugging (12) into (11) yields the least square solution for the joint velocity:

\[
\dot{q} = J^+ (\dot{x}^* + Ke).
\]

Here, the inverse kinematic problem has been solved by computing the pseudoinverse \( J^+ \) of the Jacobian matrix [24]. Section III will introduce our alternative – parsimonious and QP-based – method.
III. Parsimonious Task-Solving Approach

When the system is redundant, multiple joint configurations may satisfy the task while creating different behaviors. An explicit resolution of the inverse kinematic problem, when no constraint is specified, relies on the pseudoinverse $J^+$ of the Jacobian, as in (13). It consists in minimizing the Euclidean norm of (3). However, this often requires restricting the admissible range of joint velocities. In particular, it does not account for the physical robot constraints (e.g. joint limits) nor for safety aspects (e.g. collisions). To consider these aspects, an optimization problem has to be formulated and solved by a numerical solver.

A quadratic programming resolution is the usual process to obtain the desired $q$. It minimizes the same 2-norm as the pseudoinverse but allows defining a set of constraints to be satisfied at any time:

$$\min_{\dot{q}} \|J\dot{q} - \dot{x}\|_2^2$$

subject to

$$Ax = b, \quad C\dot{q} \leq d$$

where $A$, $C$ are the linear coefficients matrices and $b$, $d$ the constant vectors in the equality and inequality constraints, respectively. Despite an advanced state of the art in the field, one drawback with (14) 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. Here, we aim at minimizing the number of joints involved in the task, thus reducing the robot motion.

The corresponding optimization problem is:

$$\min_{\dot{q}} \|\dot{q}\|_0$$

subject to

$$\dot{x} = J\dot{q},$$

with $\|\dot{q}\|_0$ the number of nonzero components in $\dot{q}$. 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 [25] that the sparsest solution can also be obtained by solving the following equivalent problem:

$$\min_{\dot{q}} \|\dot{q}\|_1$$

subject to

$$\dot{x} = J\dot{q},$$

which is the formulation of a standard QP of $2p$ variables, $p$ being the dimension of $q$.

IV. Considerations on the Joint Constraints

Taking into account joint limit constraints in dual-arm manipulation has been addressed in [12]. In that paper, joint limits avoidance is integrated in a prioritized hierarchy of tasks, with priority depending on whether the system is redundant or not. However, this architecture is based on the projection on the null space of higher priority tasks, and does not handle strict constraints. Moreover, it does not consider joint velocity limits.

Our approach to address joint limits is twofold:

- setting hard constraints at the joint velocity level for each task, to remain inside joint position and velocity ranges at anytime.
- creating a lower priority task in which a repulsive cost function $\dot{q}_r$ pushes joints away from their limits.

A. Joint Position and Velocity Limits

Let us define the lower and upper bounds on joint positions and velocities as:

$$q \leq \bar{q} \leq \bar{q},$$

$$\dot{q} \leq \dot{q} \leq \dot{q}.$$
Since the control variables are joint velocities, we want to express the position constraint as a velocity constraint. In a discrete control loop of sampling time $T > 0$, the joint velocity remains constant between two iterations. Thus, if $k$ is the current iteration and the position constraint has to be satisfied at $k + 1$, this can be written as:

$$\dot{q} \leq q_k + \dot{q}T \leq q_i,$$  \hspace{1cm} (21)

which, plugged into (20), gives:

$$\frac{q - q_k}{T} \leq \dot{q} \leq \frac{q - q_k}{T},$$  \hspace{1cm} (22)

$$\dot{q} \leq q \leq \overline{q}.$$  \hspace{1cm} (22)

From (22), we keep the most restrictive condition, to define the final lower and upper bounds, respectively $\dot{Q}$ and $\overline{Q}$:

$$\dot{Q} = \max \left\{ \dot{q}, \frac{q - q_k}{T} \right\},$$

$$\overline{Q} = \min \left\{ \dot{q}, \frac{q - q_k}{T} \right\}.$$  \hspace{1cm} (23)

B. Repulsive cost function for joint limits avoidance

Although a dual-arm robot is highly redundant with respect to a relative cooperative task, reaching joint position boundaries limits its operational capabilities. This may lead to suboptimal solutions, due to the reduction of the admissible solution space. To avoid this, we define a repulsive vector representing a target velocity for avoiding joint limits.

First, let $\alpha \in [0, 1]$ be a tunable parameter that sets the critical parts of the joint range $[q_i, \overline{q}_i]$ (i.e., the areas in which the repulsive action should be enabled). This leads to the definition of lower and upper activation thresholds (shown for joint $i$ in Fig. 2), respectively:

$$q^*_i = q + \alpha \frac{\overline{q} - q}{2},$$

$$q^*_i = q - \alpha \frac{q_i - q}{2}.$$  \hspace{1cm} (24)

If a joint position is inside $[q^*_i, q^*_i]$, no repulsive action is applied. In this case, the reference velocity for this joint is 0 (to avoid unnecessary motion). Otherwise, it is defined as a ratio of the minimum/maximum velocity that can be applied, depending whether the critical part is near the lower or upper joint limit. The repulsive value $\dot{q}_{r,i}$ associated to joint $i$ is depicted in Fig. 2 and computed as:

$$\dot{q}_{r,i} = \begin{cases} 0, & q_i \in [q^*_i; \overline{q}_i] \\ \left(\frac{q_i - q}{\overline{q}_i - \overline{q}_i}\right) \dot{\overline{q}}_i, & q_i \in [\overline{q}_i; q^*_i] \\ \left(\frac{q - q^*_i}{\overline{q}_i - q^*_i}\right) \dot{q}, & q_i \in [q^*_i; \overline{q}_i]. \end{cases}$$  \hspace{1cm} (25)

V. HIERARCHICAL SPARSE QUADRATIC PROGRAMMING ARCHITECTURE

The parsimonious task solving method and joint limits avoidance strategy are integrated in a hierarchical sparse quadratic programming framework.

The primary task is to ensure proper trajectory tracking of the desired $\dot{x}^*$, while minimizing the generated motion and respecting joint limits. We denote as QP1 the corresponding optimization process and $q_1$ the generated solution:

$$q_1 \in \min_q \frac{1}{2} \|J\dot{q} - (\dot{x}^* + Ke)\|^2_2 + \lambda \|q\|_1, \quad \lambda = 0^+$$

s.t. $\dot{q} \leq q \leq \overline{q}$.  \hspace{1cm} (26)

Here, the $l_2$-norm term solves the closed-loop inverse kinematic problem presented in Section II-C and boundaries for $\dot{q}$ are the ones defined in (23). The $l_1$-norm penalty term $\lambda$ is a real parameter such as $\lambda \in [0^+, \|J^T(\dot{x}^* + Ke)\|_\infty]$ in order to obtain a sparse solution. By taking $\lambda$ as small as possible inside the permissible range provides the sparsest solution amongst those satisfying the task. Indeed, a greater value of $\lambda$ increases the sparsity but also the tracking error.

The secondary task, for joint limits avoidance, is implemented as another QP (referred to as QP2). It provides an additional joint velocity vector $q_2$ solution of:

$$q_2 \in \min_q \|q - q_{r,i}\|_2$$

s.t. $\dot{q} - q_1 \leq q \leq \overline{q} - q_1,$  \hspace{1cm} (27)

$$J\dot{q} = 0.$$

Adding the equality constraint $J\dot{q} = 0$ restricts the solution space to the null space of the primary task, and does not affect its performance. Joint constraints have to be updated to take into account the velocity generated by $q_1$. Finally, the joint velocity sent to the robot is:

$$\dot{q} = q_1 + q_2.$$  \hspace{1cm} (28)

The complete Hierarchical Sparse QP control framework is depicted in Fig. 3.
The efficiency and performance of the presented framework have been tested in simulation and on the real dual-arm cobot BAZAR designed at LIRMM. BAZAR is equipped with two 7-DOF Kuka LWR4 arms and an omni-directional mobile base (fixed in this study).

All experiments are performed on a computer with Intel(R) Core(TM) i7-6600U CPU running Linux. Our approach has been implemented in C++ using the Knowbotics framework developed at LIRMM. Simulation scenarios are performed on the Virtual Robot Experimentation Platform (V-REP)\(^2\). The Fast Research Interface Library (FRI)\(^3\) was used to communicate with the Kuka arms during the real experiment, and the controller sample time was set to \(T = 5\) ms.

The Reflexxes Motion Library [28] is used as trajectory generator for \(x^*\) and \(\dot{x}^*\), given a set of constraints.

We use a gain \(K = 10\) to compensate the tracking error. The \(l_1\)-norm penalization factor is set to \(\lambda = 1 \times 10^{-6}\), and \(\alpha = 0.1\) is taken for the repulsive cost function. Joint limits used are the nominal LWR4 ones.


### A. Assembly task scenario

Our control scheme has been validated on the real robot through an assembly task in which a peg-in-hole operation is simulated. A peg is attached to one arm while a shape with 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. 1. 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. 4.
- then, the screwing operation is performed to insert the peg in the hole.

The scenario is performed with three different task-solving configurations: regular QP from (14), sparse QP (QP1), and hierarchical sparse QP (QP1 + QP2). Hard constraints on joint limits are integrated in any case.

![Standard QP](image1)

![Sparse QP](image2)

![Hierarchical Sparse QP](image3)

Fig. 4. Initial and final configuration reached during the simulated assembly task using the different task-solving configurations.

### B. Results

From a high-level point of view, initial and final configurations obtained for the screwing task reveal the behavioral differences between the methods in terms of generated motions. The screwing operation, that could be intuitively performed by a simple action, generated a high occupancy of

---

\(^2\)http://www.coppeliarobotics.com/

\(^3\)https://cs.stanford.edu/people/tkr/fri/html/
the workspace using the standard QP (Fig. 4(a)). Conversely, the parsimonious solution has resulted in local and economic movements as demonstrated by Fig. 4(b) that shows similar intermediate and final configurations, except for the grippers’ orientations. In the last case (Fig. 4(c)), the secondary task has slightly modified the configurations to avoid limits for the right shoulder and the wrists.

Numerical results from Fig. 5 and I confirm this observation. The overall number of actuated joints as well as the sum of absolute joint velocities have been reduced with the parsimonious approach. Indeed, aligning both end-effectors from an arbitrary configuration required the use of six joints for the most part, corresponding to the number of controlled task variables. Even less joints are needed to execute the screwing movement, because of the geometric properties of the operation. As expected, the standard QP approach minimizes the energy consumption during the task. However, this resolution involves the use of almost every joints, regardless the geometric complexity of the task. When the joint limits repulsive cost function is enabled (Hierarchical Sparse QP), additional joints are temporarily activated to leave critical parts of position ranges: between 5.4 s and 9 s the right shoulder approaches a boundary, and from 34 s to the end of the task, the two wrists reach the repulsive area. However, as seen in Fig. 5(a) and Fig. 5(b), this feature has only local effect on the robot’s behavior and the hierarchical sparse QP still provides much sparser motions than the standard one, as demonstrated in Table I. Note that the hierarchical sparse QP may give a smaller number of actuated joints compared to the sparse QP (as is the case in the middle part of the task). This is due to the algorithms which are local and do not ensure global optimality.

VII. CONCLUSION AND FUTURE WORK

This paper presented a new kinematic control strategy for cooperative dual manipulator robots. Based on a hierarchy of QP, the obtained solution is one that activates the minimal number of joints to perform coordinated motions, while avoiding joint limits. Experiments on a real dual-arm robot revealed the benefits of our task-solving method compared to the standard QP. The resulting economic behavior is an appealing feature for numerous robotic applications.

Future work will focus on addressing parsimony in simultaneous local and global manner by mixing kinematic and impedance control for improving the interaction with human operator and robot in the collaborative manipulation tasks.

ACKNOWLEDGMENT

This research has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement no 731330 (VERSATILE: Innovative robotic applications for highly reconfigurable production lines).

TABLE I

<table>
<thead>
<tr>
<th>Metric</th>
<th>Standard QP</th>
<th>Sparse QP</th>
<th>Hierarchical Sparse QP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int_0^t</td>
<td></td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>$\int_0^t</td>
<td></td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>$\int_0^t</td>
<td></td>
<td>q</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES


