Robotics and Autonomous Systems 70 (2015) 191–201

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties

Fabrizio Flacco *, Alessandro De Luca

Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Italy

HIGHLIGHTS

- A method to convert in a discrete-time velocity law any second-order control scheme.
- In redundant robots, preserves the original properties (minimum acceleration/torque).
- Equivalence is obtained in all cases by a proper choice of a vector in the task null space.
- Solutions are simple to implement and are interfaced directly to low-level controllers.
- Validating simulations and experiments on a 7R KUKA LWR IV robot.

ARTICLE INFO

Article history:
Received 9 September 2014
Received in revised form
2 February 2015
Accepted 16 February 2015
Available online 4 March 2015

Keywords:
Inverse differential kinematics
Redundant robots
Discrete-time velocity control

ABSTRACT

With reference to robots that are redundant for a given task, we present a novel and intuitive approach allowing to define a discrete-time joint velocity command that shares the same characteristics of a second-order inverse differential scheme, with specified properties in terms of joint acceleration or torque. Our main goal is to show how commands in the null space of the task can yield different locally optimal solutions, working only at the velocity control level. By following our general method, it is possible to obtain simple implementations of possibly complex robot control laws that (i) can be directly interfaced to the low-level servo loops of a robot, (ii) require less task information and on-line computations, (iii) are still provably good with respect to some target performance. The method is illustrated by considering the conversion into discrete-time velocity commands of control schemes for redundant robots that minimize the (weighted and/or biased) norm of joint acceleration or joint torque. The approach can be extended to auxiliary tasks, possibly organized with priority. Numerical simulations and experimental results are presented for the control of a 7R KUKA LWR IV robot.

1. Introduction

Robot motion interfaces are usually defined at the position/velocity level, and rely on a two-level control architecture (Fig. 1). A low-level controller, or direct layer of joint-level servos, imposes currents to the motors, and thus joint torques to the robot; a high-level, user-defined control program sends desired position/velocity references to the servos, based on proprioceptive sensing (encoders, joint torque sensors), exteroceptive sensing (e.g., visual feedback), and input from the task/trajectory planner. This two-level architecture is used in industrial robots [1,2] but also in most of the robots designed for research [3,4].

In this framework, *kinematic control* schemes are designed without consideration of robot dynamics, assuming that reference position/velocity commands are accurately reproduced by the robot, thanks to the presence of fast and well-tuned low-level loops, to the use of sufficiently small sampling times, and assuming that speed and acceleration involved in the motion task are moderate.

It is well known that *dynamic control* can outperform conventional kinematic controllers, both in terms of speed and precision [5]. For this to happen, we require a relatively complete and accurate dynamic model of the robot and of its actuating devices, together with a so-called *open* architecture that allows to impose joint torques (or motor currents) as user commands.

It has been shown in [4,6] that one can get around closed control architecture and still be able to generate suitable velocity references at the user level, so as to apply desired torque commands to the robot. However, this torque transformer requires a good

* Corresponding author.
E-mail addresses: fflacco@diag.uniroma1.it (F. Flacco), deluca@diag.uniroma1.it (A. De Luca).

http://dx.doi.org/10.1016/j.robot.2015.02.008
0921-8890/© 2015 Elsevier B.V. All rights reserved.

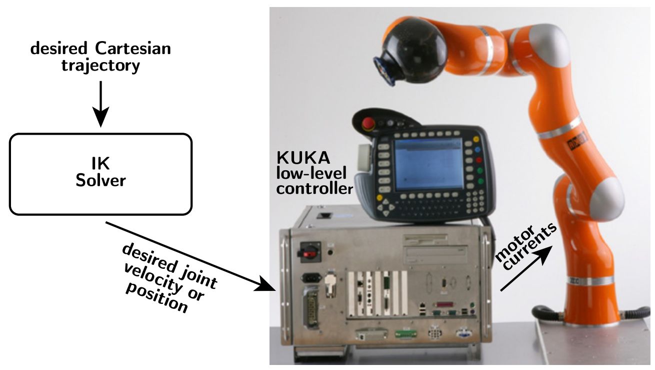

desired Cartesian trajectory
IK Solver
desired joint velocity or position
KUKA low-level controller
motor currents

Fig. 1. Example of a two-level control architecture. The Inverse Kinematics (IK) solver works at the differential level and in discrete time; the KUKA controller KR C2 lr sends to the KUKA LWR IV robot the motor currents necessary to reach the desired joint configuration.

knowledge of the structure and parameters of the low-level control loops, an information that is typically unavailable to a generic end-user.

On the other hand, the latest generation of research-oriented manipulators allows a *torque-controlled* behavior [7–9]. In order to take full advantage of this possibility, a dynamic model of the robot is needed, possibly including also joint/transmission compliance [10]. Moreover, the whole task and control treatment should be moved up to a second-order differential level (acceleration or torques) [11]. This complicates the on-line specification of sensor-based behaviors, since designing an acceleration command to accomplish a desired reactive task is harder than doing the same with a velocity command. Therefore, the use of velocity commands is still an appealing approach.

In fact, velocity control laws are simpler to implement, require the least amount of data (e.g., no need of derivatives involving the Jacobian) and measurements, and can easily accommodate additional constraints, such as the presence of joint range limits and command saturation in redundant robots [12]. Conversely, first-order kinematic laws do not share the smoothness and dynamic optimality properties of acceleration or torque control laws (nor their level of variety).

The objective of this paper is to show how to define a kinematic control scheme at the velocity level[1] that inherits as much as possible the properties and target performance of a given second-order control scheme. We will consider the following working assumptions:

1. Low-level servo loops are present that guarantee ideal execution of any joint velocity (position) command. The reference model of the controlled robot can be considered as a pure integration between joint velocity commands and measured positions.
2. The high-level kinematic controller generates joint velocity references in discrete time, with sufficiently high sampling rate.
3. Task or Cartesian desired commands are available to the robot on line, but only up to the velocity level. Only local optimization schemes are considered.
4. Measurement in the joint space is limited to position, as provided by encoders.
5. The robot is kinematically redundant with respect to the given task.

[1] In the rest of the paper, we assume that velocity reference commands can be directly fed to the low-level controller of the robot. If instead a position reference is required by the robot control interface, a one step discrete-time integration of the velocity command can be used.

While the last assumption is not strictly needed, it enlarges considerably the scope and interest of the present analysis. Some early results on the equivalence of velocity and acceleration redundancy resolution schemes in continuous time can be found in [13], while the emphasis is given here to the discrete-time implementation, as well as to the possible inclusion of robot dynamic model information.

The paper consolidates and expands the results of our recent work [14]. With respect to the conference version, the additional contributions of the present paper include a more detailed presentation of the method and new numerical results, with the addition of dynamic simulations for the torque optimization case. Moreover, we present now also experiments on a 7-DOF robot manipulator.

After summarizing redundancy control schemes and discretization issues in Section 2, we present our main analytical results in Sections 3 and 4. In the former, we show how to design discrete-time velocity commands that are equivalent to a joint acceleration solution of minimum norm or may include the addition of null-space auxiliary tasks, in particular of a damping action for stabilizing joint motion. In the latter, we extend our design to velocity commands that are equivalent to optimal joint torque solutions. Section 5 reports numerical results when the task is to execute a desired trajectory for the end-effector position and our discrete-time velocity control method is used for optimizing acceleration or torque. Experimental results on a KUKA LWR IV robot are presented in Section 6 and in the accompanying video. Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2015.02.008.

2. Preliminaries

2.1. Redundancy resolution

Let $\boldsymbol{q} \in \mathbb{R}^n$ be the generalized (joint) coordinates of a n-DOF robot and $\boldsymbol{x} \in \mathbb{R}^m$ be the variables describing a generic m-dimensional task, with $m \leq n$. The task kinematics is given by the direct map $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q})$. The first-order task kinematics is

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{1}$$

where $\boldsymbol{J} = \partial\boldsymbol{f}/\partial\boldsymbol{q}$ is the $m \times n$ task Jacobian matrix. At a given configuration $\boldsymbol{q}$, all joint velocity solutions to (1) can be expressed as

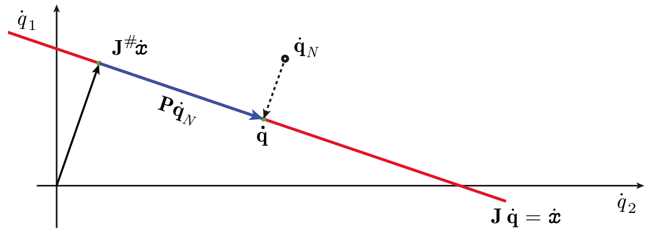
$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + \boldsymbol{P}\,\dot{\boldsymbol{q}}_N, \tag{2}$$

where $\boldsymbol{J}^{\#}$ is the (Moore–Penrose) pseudoinverse of the task Jacobian [15], $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}$ is the $n \times n$ orthogonal projector in the Jacobian null space, and $\dot{\boldsymbol{q}}_N \in \mathbb{R}^n$ is a generic joint velocity that can be used for auxiliary tasks.

There is a close relation between weighted pseudoinversion and equality-constrained least squares programs [16]. The above scheme, as well as most of the other local methods for optimal use of robot redundancy existing in the literature, can be obtained from a general Quadratic Programming (QP) formulation (see, e.g., [13] or [17]), as reported for convenience and completeness in Appendix A. In particular, the joint velocity command (2) is the solution of the QP problem

$$\begin{aligned} &\dot{\boldsymbol{q}} = \arg\min_{\dot{\boldsymbol{q}}\in\mathcal{S}} \frac{1}{2}\|\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_N\|^2 \\ &\text{with } \mathcal{S} = \left\{ \arg\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{J}\dot{\boldsymbol{q}} - \dot{\boldsymbol{x}}\|^2 \right\}. \end{aligned} \tag{3}$$

Fig. 2 shows a pictorial view of the optimal solution (2) to the QP problem (3), where the joint velocity $\dot{\boldsymbol{q}}$ executes the task while



$\dot{q}_1$
$\mathbf{J}^{\#}\dot{\boldsymbol{x}}$
$\dot{\mathbf{q}}_N$
$\boldsymbol{P}\dot{\mathbf{q}}_N$
$\dot{\mathbf{q}}$
$\dot{q}_2$
$\mathbf{J}\,\dot{\mathbf{q}} = \dot{\boldsymbol{x}}$

Fig. 2. Representation of the velocity solution (2) in the joint velocity space of a 2-DOF robot executing a one-dimensional task.

minimizing the distance to the auxiliary (or *biased*) joint velocity $\dot{\boldsymbol{q}}_N$. This property is used in our method.

Differentiating (1), the second-order task kinematics is

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{4}$$

At a given robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, all joint acceleration solutions to (4) can be expressed as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) + \boldsymbol{P}\,\ddot{\boldsymbol{q}}_N \tag{5}$$

being $\ddot{\boldsymbol{q}}_N \in \mathbb{R}^n$ a generic (*preferred*) joint acceleration. Assuming full rank for $\boldsymbol{J}$, the first term on the right side of (5) provides the solution to (4) with minimum joint acceleration norm.

Let the dynamics of a fully actuated robot in free motion be compactly described by[2]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{6}$$

with positive definite inertia matrix $\boldsymbol{M}$, Coriolis, centrifugal, and gravitational terms $\boldsymbol{n}$, and input torque $\boldsymbol{\tau} \in \mathbb{R}^n$. Based on (6), several local optimization-based control schemes have been proposed at the torque level [18–21] for realizing a desired task acceleration $\ddot{\boldsymbol{x}}$ (possibly including also a PD action on the trajectory error). For instance, at a given robot state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, the solution with minimum torque norm of [18] is given by

$$\boldsymbol{\tau} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\right)^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right). \tag{7}$$

When $\boldsymbol{J}$ has full rank, $\left(\boldsymbol{J}\boldsymbol{M}^{-1}\right)^{\#} = \boldsymbol{M}^{-1}\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{M}^{-2}\boldsymbol{J}^T)^{-1}$.

2.2. Discrete-time implementation

Assume that a user-defined control law is implemented in discrete time, with control samples defined at times $t = t_k = kT,\ k = 0, 1, \ldots,$ and kept constant for a sufficiently small sampling time T (say, in the order of few msec). Denote by $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ the sample of a generic vector or matrix variable $\boldsymbol{u}$. Let the robot be driven by joint velocity commands $\dot{\boldsymbol{q}}_k$.

In the common practice, a desired nominal task acceleration $\ddot{\boldsymbol{x}}_k$ is approximated as

$$\ddot{\boldsymbol{x}}_k \simeq \frac{\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{x}}_{k-1}}{T}, \tag{8}$$

so that the task information needed (e.g., about the desired end-effector trajectory) is limited to the velocity level. The time-derivative of the task Jacobian $\boldsymbol{J}$ at $t = t_k$ can be approximated as

$$\dot{\boldsymbol{J}}_k \simeq \frac{\boldsymbol{J}_k - \boldsymbol{J}_{k-1}}{T}, \tag{9}$$

where $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{q}_k)$. Only the current and previous joint position measurements $\boldsymbol{q}_k$ and $\boldsymbol{q}_{k-1}$ are used in (9). Finally, wherever a joint velocity $\dot{\boldsymbol{q}}$ appears in the continuous-time version of a second-order control law, we need to evaluate this quantity as

$$\dot{\boldsymbol{q}} \simeq \dot{\boldsymbol{q}}_{k-1}\left(\simeq \frac{\boldsymbol{q}_k - \boldsymbol{q}_{k-1}}{T}\right), \tag{10}$$

namely using the last velocity command computed at the previous sampling instant, or the current and previous joint position measurements. The latter uses the *forward differences* formula to approximate time differentiation. Indeed, one can simply combine (9) and (10) to directly obtain a discrete-time approximation of $\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$.

To check the significance of the choices (8)–(10), consider the standard controller in the absence of redundancy (and with the robot in a nonsingular configuration) that realizes a desired task acceleration $\ddot{\boldsymbol{x}}$ by imposing the joint acceleration

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}\right). \tag{11}$$

In discrete time, using the above approximations, one obtains

$$\begin{aligned}\ddot{\boldsymbol{q}}_k &= \boldsymbol{J}^{-1}(\boldsymbol{q}_k)\left(\ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}(\boldsymbol{q}_k)\dot{\boldsymbol{q}}_k\right)\\ &\simeq \boldsymbol{J}_k^{-1}\left(\frac{\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{x}}_{k-1}}{T} - \frac{\boldsymbol{J}_k - \boldsymbol{J}_{k-1}}{T}\dot{\boldsymbol{q}}_{k-1}\right)\\ &= \frac{1}{T}\left(\boldsymbol{J}_k^{-1}\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{q}}_{k-1}\right),\end{aligned} \tag{12}$$

where we used the identity $\boldsymbol{J}_{k-1}\dot{\boldsymbol{q}}_{k-1} = \dot{\boldsymbol{x}}_{k-1}$, due to our working assumption of perfect joint velocity execution. Using (12) and *backward differences* to approximate integration over time, i.e.,

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_k T, \tag{13}$$

the joint velocity command at t_k is obtained as

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_k T = \boldsymbol{J}_k^{-1}\dot{\boldsymbol{x}}_k, \tag{14}$$

which is exactly what we would have obtained from the direct discretization of a first-order velocity controller. This shows that the chosen approximation steps provide a consistent evaluation in discrete time, and we will use them next with confidence also for dealing with redundancy.

3. Velocity control for acceleration optimization

As a first illustrative example, we will apply the discretization method of Section 2.2 to define a joint velocity control law that minimizes the joint acceleration norm. In terms of discrete-time velocity, this is intuitively obtained by minimizing the variations in the sequence of velocity commands, which means taking the joint velocity that solves the task (1) and minimizes the distance to the previous velocity command $\dot{\boldsymbol{q}}_{k-1}$. With this in mind, considering the property (3) in Section 2.1 the solution follows as

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_k\dot{\boldsymbol{q}}_{k-1}, \tag{15}$$

where $\boldsymbol{P}_k = \boldsymbol{I} - \boldsymbol{J}_k^{\#}\boldsymbol{J}_k$. It is easily recognized that Eq. (15) is the discrete-time evaluation of the general first-order solution (2), with $\dot{\boldsymbol{q}}_{N,k} = \dot{\boldsymbol{q}}_{k-1}$.

Considering first the case of a full rank Jacobian $\boldsymbol{J}$, we verify that the velocity law (15) corresponds to our discretization method applied to the minimum acceleration norm control law (obtained from Eq. (5), when $\ddot{\boldsymbol{q}}_N = \boldsymbol{0}$), namely

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\left(\ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k\right). \tag{16}$$

In fact, from (13) and (16), using the rules in Section 2.2 we have

$$\begin{aligned}\dot{\boldsymbol{q}}_k &= \dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_k T\\ &\simeq \dot{\boldsymbol{q}}_{k-1} + \boldsymbol{J}_k^{\#}\left(\frac{\dot{\boldsymbol{x}}_k - \dot{\boldsymbol{x}}_{k-1}}{T} - \frac{\boldsymbol{J}_k - \boldsymbol{J}_{k-1}}{T}\dot{\boldsymbol{q}}_{k-1}\right)T\\ &= \dot{\boldsymbol{q}}_{k-1} + \boldsymbol{J}_k^{\#}\left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k\dot{\boldsymbol{q}}_{k-1}\right),\end{aligned} \tag{17}$$

[2] For the sake of simplicity, external forces and friction effects are not considered.

which is identical to (15). Therefore, the velocity control law (17) shares the same optimal properties of the acceleration-level solution (16). In other words, a particular null-space vector in the velocity law (15) guarantees a second-order property to the solution—an interesting new insight in the role of this auxiliary term. Besides, Eq. (17) is very simple to implement, as it does not require differentiating the task Jacobian,[3] nor to compute and then numerically integrate the second-order law. Thus, there is also no need to define a desired task acceleration $\ddot{\boldsymbol{x}}$.

When the Jacobian $\boldsymbol{J}$ is close to a singularity, a damped pseudoinverse is mandatory in order to avoid discontinuities and unfeasible commands [24]. In this case, task execution is slightly deformed in general and the control law (17) is not fully equivalent to (16). Nonetheless, the variation of the velocity will still be minimized.

3.1. Including auxiliary tasks

An important feature of redundancy is the possibility of using motions in the null space of the primary task to accomplish auxiliary tasks, e.g., with the Projected Gradient (PG) method [25], where a preferred joint velocity/acceleration is specified according to the gradient direction of some cost function. Since the null-space vector is already used to suitably modify the property of the solution – see Eq. (15) – it is not immediate to see how this capability can be maintained. We show next how to accommodate auxiliary tasks in the same framework.

At the acceleration command level, consider a preferred joint acceleration $\ddot{\boldsymbol{q}}_{N,k}$ projected in the null space of the task Jacobian

$$\ddot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\left(\ddot{\boldsymbol{x}}_k - \dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k\right) + \boldsymbol{P}_k\ddot{\boldsymbol{q}}_{N,k}. \tag{18}$$

Proceeding as before, the rules of conversion of (18) into a discrete-time velocity control law, with sampling time $T > 0$, provide

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\,\dot{\boldsymbol{x}}_k + \boldsymbol{P}_k\left(\dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_{N,k}\,T\right). \tag{19}$$

The above reasoning for conversion to velocity control laws can be applied similarly also in the case of l multiple tasks, without or with priority [26]. Assume that the multi-task case has been solved at the velocity level by any known methodology, say [27] or [28]. The resulting solution $\dot{\boldsymbol{q}}_{l,k}$ accomplishes (exactly, or as much as possible according to the hierarchy of tasks) all l tasks, while minimizing the joint velocity norm. Then, as for the single-task case, the discrete-time velocity control law that minimizes the joint acceleration norm is the one that minimizes the variation of the joint velocity command, namely

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{l,k} + \boldsymbol{P}_{l,k}\left(\dot{\boldsymbol{q}}_{k-1} + \ddot{\boldsymbol{q}}_{N,k}\,T\right), \tag{20}$$

where $\boldsymbol{P}_{l,k}$ is the projector in the null space of all l tasks.

3.2. Null-space motion damping

As an interesting example of inclusion of auxiliary tasks in the presented framework, we consider the issue of damping the floating robot behavior which may arise in the null space of a given motion task.

When a redundant robot is controlled using minimum norm commands at the second- or higher-order differential level (i.e., from acceleration upwards), one can typically observe the joint variables starting to float over time on the self-motion manifold associated to the task.

This drift phenomenon has a simple explanation. With the robot being controlled at the velocity level, any joint velocity contribution in the null space of the task Jacobian will be set to zero, because of the minimum velocity norm property. A simple case to analyze is when the generic joint i is not useful for performing the task, i.e., the ith column of the task Jacobian is zero. In the minimum joint velocity norm solution, the ith component of $\dot{\boldsymbol{q}}$ is automatically set to zero (joint i will not move). Similarly, when the robot is controlled at the acceleration level, joint accelerations in the null space will be set to zero in the minimum norm solution. However, in the same simple case considered above, while the ith component of $\ddot{\boldsymbol{q}}$ is kept at zero, joint i it will drift at constant velocity if it was not initially at rest, producing a floating motion in the null space of the task Jacobian.

To remove this undesired behavior, occurring especially over long task trajectories, different possibilities have been explored [13,29,30]. The simplest approach is to introduce damping on the null-space motion, as proposed, e.g., in [13]. This can be easily specified at the acceleration level, since the choice

$$\ddot{\boldsymbol{q}} = -k_d\,\dot{\boldsymbol{q}}, \quad k_d \geq 0, \tag{21}$$

in the unconstrained case, i.e., when no task is assigned, will bring the robot to a rest configuration at an exponential rate, tuned by the positive damping factor k_d (which can be in principle arbitrarily large in the continuous time domain). Therefore, the preferred choice in (5) will be $\ddot{\boldsymbol{q}}_N = -k_d\,\dot{\boldsymbol{q}}$, providing

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - k_d\boldsymbol{P}\dot{\boldsymbol{q}}. \tag{22}$$

Applying the discrete-time conversion procedure to (22), from (19) we obtain

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \lambda\boldsymbol{P}_k\dot{\boldsymbol{q}}_{k-1}, \quad \lambda = 1 - k_d T, \tag{23}$$

again with a specific term in the null space of the task Jacobian.

It can be recognized that the scalar $\lambda \leq 1$ in (23) acts as a *forgetting factor* on the previous joint velocity command—a typical tool used in recursive least squares sequences of sampled data [31]. An analysis of the stability behavior of the velocity solution (23) with respect to λ is reported in Appendix B, and provides good insight on the choice of k_d for null-space floating motion suppression at the acceleration level.

4. Extension to torque optimization

Having recognized that all possible discrete-time velocity control solutions differ only for the choice of a specific null-space vector, extending the procedure from an acceleration-level scheme to conversion of a torque-level scheme is rather straightforward. Two analytical examples of interest are presented, which illustrate also the different complexity resulting from slightly different local optimization formulations that deal with robot dynamics. Again, for the sake of simplicity we assume that the Jacobian $\boldsymbol{J}$ has full rank at the current configuration.

4.1. Minimum weighted norm of the torque

Following [19] and later [21], a convenient robot behavior is obtained when minimizing the joint torque in terms of a norm weighted by the *squared inverse of the inertia* matrix:

$$\begin{aligned} &\boldsymbol{\tau} = \arg\min_{\boldsymbol{\tau}\in\mathcal{S}} \frac{1}{2}\boldsymbol{\tau}^T\boldsymbol{M}^{-2}\boldsymbol{\tau} \\ &\text{with } \mathcal{S} = \left\{ \begin{array}{r} \arg\min\limits_{\boldsymbol{\tau}\in\mathbb{R}^n} \dfrac{1}{2}\|\boldsymbol{J}\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{x}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\|^2 \\ \text{s.t. } \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} = \boldsymbol{\tau}. \end{array} \right\}. \end{aligned} \tag{24}$$

[3] The evaluation of $\dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}$ in (11) could be efficiently obtained as part of the forward kinematic pass of a recursive Newton–Euler algorithm (i.e., by setting $\ddot{\boldsymbol{q}} = \boldsymbol{0}$) [22,23]. In comparison, the present method does not need *at all* this computational pass.

In this way, the robot joint motion remains typically bounded even for longer motion tasks, as opposed to the case of the minimum (unweighted) torque norm solution [18].

The optimal solution to (24) is

$$\boldsymbol{\tau} = \boldsymbol{M}\boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right). \tag{25}$$

Associated to this torque, there is a unique joint acceleration

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{P}\boldsymbol{M}^{-1}\boldsymbol{n}. \tag{26}$$

Conversion to the equivalent discrete-time velocity control is easily obtained when starting from (26). Define first the notation $\boldsymbol{n}_{k|k-1} = \boldsymbol{n}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_{k-1})$, where all quadratic velocity terms are evaluated using the previous joint velocity sample.[4] Proceeding as in Section 3, we obtain

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_k\left(\dot{\boldsymbol{q}}_{k-1} - T\boldsymbol{M}_k^{-1}\boldsymbol{n}_{k|k-1}\right). \tag{27}$$

Note that all the needed dynamic information is contained only in the last extra term, which is also scaled by the sampling time T.

4.2. Dynamically consistent redundancy resolution

The other considered torque-level control method for redundant robots is the dynamically consistent approach of [20], which is based on a task-oriented decomposition of the joint torques. Also in this case the problem can be formulated as a QP where the torque difference with respect to a *biased* reference $\boldsymbol{\tau}_N$ is minimized, using a norm weighted by the simple *inverse of the inertia* matrix:

$$\begin{aligned} \boldsymbol{\tau} &= \arg\min_{\boldsymbol{\tau}\in\mathcal{S}} \frac{1}{2}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_N\right)^T \boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_N\right) \\ \text{with } \mathcal{S} &= \left\{ \begin{array}{r} \arg\min_{\boldsymbol{\tau}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{J}\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{x}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\|^2 \\ \text{s.t. } \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} = \boldsymbol{\tau}. \end{array} \right\}. \end{aligned} \tag{28}$$

The optimal solution to (28) is

$$\boldsymbol{\tau} = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{n}\right) + \left(\boldsymbol{I} - \boldsymbol{J}^T(\boldsymbol{J}^T)^{\#}_{\boldsymbol{M}}\right)\boldsymbol{\tau}_N, \tag{29}$$

with the inertia-weighted pseudoinverse of the Jacobian transpose given by

$$(\boldsymbol{J}^T)^{\#}_{\boldsymbol{M}} = \left(\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{J}\boldsymbol{M}^{-1}. \tag{30}$$

The matrix $\boldsymbol{P}^{\perp}_{\boldsymbol{M}} = \boldsymbol{I} - \boldsymbol{J}^T(\boldsymbol{J}^T)^{\#}_{\boldsymbol{M}}$ in (29) is an operator that annihilates all joint torques lying in the range of $\boldsymbol{J}^T$. The unique joint acceleration associated to (29) can be rewritten as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}_{\boldsymbol{M}}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{P}_{\boldsymbol{M}}\boldsymbol{M}^{-1}\left(\boldsymbol{n} - \boldsymbol{\tau}_N\right) \tag{31}$$

where $\boldsymbol{P}_{\boldsymbol{M}}$ is the inertia-weighted projector in the null space of $\boldsymbol{J}$. Eq. (31) serves again as a basis for deriving the discrete-time velocity control, which is

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}^{\#}_{\boldsymbol{M},k}\dot{\boldsymbol{x}}_k + \boldsymbol{P}_{\boldsymbol{M},k}\left(\dot{\boldsymbol{q}}_{k-1} - T\boldsymbol{M}_k^{-1}\left(\boldsymbol{n}_{k|k-1} - \boldsymbol{\tau}_{N,k}\right)\right). \tag{32}$$

The complexity of the control law is indeed increased in this case. For instance, the expression of the weighted projector $\boldsymbol{P}_{\boldsymbol{M},k}$ is

$$\boldsymbol{P}_{\boldsymbol{M},k} = \boldsymbol{I} - \boldsymbol{M}_k^{-1}\boldsymbol{J}_k^T\left(\boldsymbol{J}_k\boldsymbol{M}_k^{-1}\boldsymbol{J}_k^T\right)^{-1}\boldsymbol{J}_k. \tag{33}$$

On the other hand, a damping torque may be used for stabilizing robot self-motions. As an additional benefit, this term can be devised so as to simplify the null-space vector in (32). For this, choose the torque $\boldsymbol{\tau}_N$ so as to oppose the *generalized momentum* $\boldsymbol{M}\dot{\boldsymbol{q}}$ with a suitable damping gain $k_d = 1/T$. The continuous time and the realized discrete time versions are

$$\boldsymbol{\tau}_N = -k_d\,\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} \quad\Rightarrow\quad \boldsymbol{\tau}_{N,k} = -\frac{1}{T}\boldsymbol{M}_k\dot{\boldsymbol{q}}_{k-1}. \tag{34}$$

Using this in (32) yields the simpler form

$$\dot{\boldsymbol{q}}_k = \boldsymbol{J}^{\#}_{\boldsymbol{M},k}\dot{\boldsymbol{x}}_k - T\boldsymbol{P}_{\boldsymbol{M},k}\boldsymbol{M}_k^{-1}\boldsymbol{n}_{k|k-1}. \tag{35}$$

5. Numerical results

Taking into account the working assumptions detailed in Section 1, the method has been tested first with Matlab simulations using a model of the KUKA LWR robot with $n = 7$ joints.

A multiple point-to-point motion task is specified for the position $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q})$ of the robot end-effector (the dimension of the task variables is $m = 3$). The robot starts at time $t = 0$ from the initial configuration

$$\boldsymbol{q}(0) = \begin{pmatrix} 28.08 & 104.12 & 114.59 & 94.85 & 14.32 & -28.12 & 0 \end{pmatrix}^T \ \text{[deg]}$$

corresponding to the Cartesian point $\boldsymbol{x}_0 = (-0.4, 0.36, 0.01)$ [m], and moves to three Cartesian points $\boldsymbol{x}_1 = (-0.4, -0.36, 0.01)$, $\boldsymbol{x}_2 = (-0.4, -0.36, 1.16)$, and $\boldsymbol{x}_3 = (-0.4, 0.36, 1.16)$ [m]. These four points are on the same vertical plane. The trajectory between two generic points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ is a linear path, with rest-to-rest timing law given by a doubly-normalized quintic polynomial $s(\xi)$:

$$\begin{aligned} \boldsymbol{x}_d(t) &= \boldsymbol{x}_A + (\boldsymbol{x}_B - \boldsymbol{x}_A)\,s(\xi), \\ s(\xi) &= 6\xi^5 - 15\xi^4 + 10\xi^3, \quad \xi = \frac{t}{T_{AB}} \in [0, 1], \\ \dot{\boldsymbol{x}}_d(t) &= \frac{\boldsymbol{x}_B - \boldsymbol{x}_A}{T_{AB}}\left(30\xi^4 - 60\xi^3 + 30\xi^2\right), \end{aligned} \tag{36}$$

where $T_{AB} = 3$ [s] is the motion time (for each segment). The reference task velocity is given by the desired one complemented by a Cartesian error feedback to recover linearization errors

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_d + \boldsymbol{K}_P\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{q})\right), \tag{37}$$

with $\boldsymbol{K}_P = k_P\boldsymbol{I}$ and the scalar parameter set to $k_P = 10$. When approaching the desired end-point $\boldsymbol{x}_B = \boldsymbol{x}_d(T_{AB})$ of a generic segment AB, the task planner switches to the next desired Cartesian position when the error $\|\boldsymbol{x}_B - \boldsymbol{f}(\boldsymbol{q})\| < \epsilon$, with $\epsilon = 1$ [mm]. The sampling time is $T = 1$ [ms].

5.1. Acceleration optimization

Figs. 3–4 show the results obtained using the discrete-time velocity control (23) with forgetting factor $\lambda = 0.99$, equivalent to a damping of $k_d = 10$ in the acceleration-level control Eq. (22). The Cartesian motion of the robot end-effector and elbow (i.e., the tip of the third link) are shown in Fig. 3. Fig. 4 reports the evolution of the norms of three quantities used as indices of performance: joint acceleration, null-space contribution, and joint velocity (the actual command).

For validation, the same task has been performed with the acceleration-level control law (22). The equivalence between the proposed discrete-time velocity control approach and the velocity resulting from the acceleration-level command is confirmed by the plot of the difference between the joint velocity in the two simulations (Fig. 5), which is zero up to numerical rounding.

A similar consistent result about equivalence was obtained also when using the minimum acceleration norm control without damping (i.e., with $\lambda = 1$). However, the original control law displayed a joint motion drift, letting the robot approach a singular configuration with an increase in acceleration peaks by a factor of 40 (despite a change in λ by just 1%). Even in the presence of such negative event, the sought equivalence was still in place and the converted discrete-time velocity control law displayed exactly the same robot behavior.

[4] This is different from [32], where Coriolis and centrifugal terms were evaluated mixing previous and current velocities.

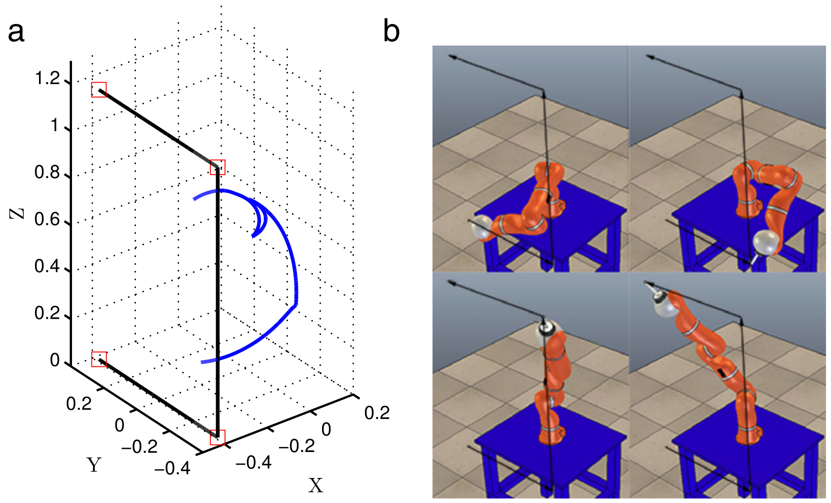


Fig. 3. Execution of the multiple point-to-point motion task with the proposed discrete-time velocity control with null-space damping ($\lambda = 0.99$ in Eq. (23)): (a) end-effector (black) and elbow (blue) traced path; (b) screenshots from the simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

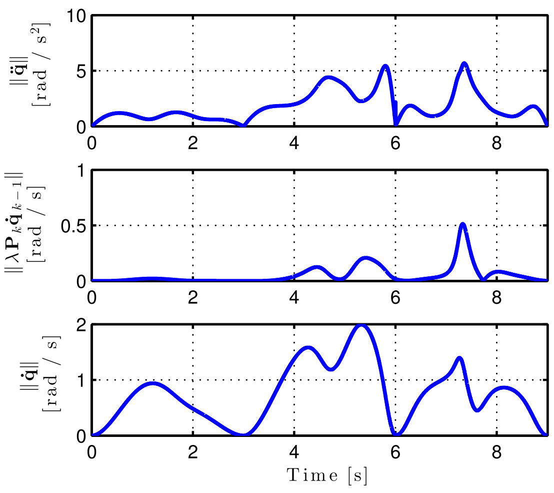


Fig. 4. Numerical results for the proposed discrete-time velocity control with null-space damping ($\lambda = 0.99$ in Eq. (23)): [top] norm of the joint acceleration $\|\ddot{\boldsymbol{q}}_k\|$; [center] norm of the null-space contribution $\|\lambda \boldsymbol{P}_k \dot{\boldsymbol{q}}_{k-1}\|$; [bottom] norm of the joint velocity $\|\dot{\boldsymbol{q}}_k\|$.

5.2. Torque optimization

We have compared also the dynamically consistent torque control law (29), using a null-space damping action as in (34), with its conversion into the discrete-time velocity control law (35) obtained using the proposed method. For this simulation, the dynamics of the KUKA LWR robot was derived using the Robotics Toolbook,[5] modeling each link as a uniform rod of unitary mass.

Figs. 6–8 refer to the same previous point-to-point task with multiple points obtained when using the discrete-time velocity control law (35). As expected the joint velocity norm (Fig. 7) is larger than in the previous simulation (Fig. 4). In fact, the dynamically consistent torque command (29) produces faster motion for those joints that move relatively low inertia.

Finally, the equivalence between the proposed discrete-time velocity control law (35) and the dynamically consistent torque control (29) with damping (34) is confirmed by Fig. 8, where the norm of the difference between the actual joint velocity obtained with the two methods is shown to be totally negligible.

[5] http://petercorke.com/Robotics_Toolbox.html.

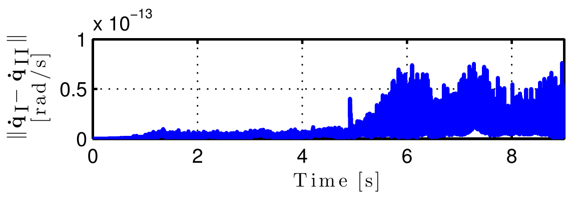


Fig. 5. Norm of the difference between the joint velocity command $\dot{\boldsymbol{q}}_{\text{I}}$ from the discrete-time velocity control law (23) with $\lambda = 0.99$ and the joint velocity $\dot{\boldsymbol{q}}_{\text{II}}$ obtained from the acceleration command (22) with damping gain $k_d = 10$.

6. Experimental results

Experiments have been conducted on a KUKA LWR IV robot to prove the practical effectiveness of the proposed method. The LWR is commanded using the Fast Research Interface (FRI), which allows controlling the desired joint position at high frequency rates, 500 [Hz] in our experiments ($T = 2$ [ms]). The control schemes have been implemented in C++, using the Eigen library [33] for algebraic computations, and working in a ROS environment [34] on a Intel Core i7-2600 CPU 3.4 GHz, with 8 Gb of RAM. The FRI outputs at every sampling instant t_k the current robot configuration $\boldsymbol{q}_k$. The experimental joint velocity and acceleration profiles that we report in this section have been computed *offline* by an accurate numerical differentiation of the joint position measures provided by the robot encoders. The estimations were obtained using a Savitzky–Golay filter [35] with a window of $W = 400$ samples and a polynomial of fifth degree. Due to the filter characteristics, the first and last $W/2$ samples have been discarded (as apparent in the plots). For the experiments, we have considered the same motion task and used the same parameters as in the simulations.

6.1. Acceleration optimization

In a first set of experiments, the standard velocity control law (2) and the acceleration control law (22) have been considered, as well as the proposed conversion of the latter into a discrete-time velocity control law. The accompanying video shows the experiments presented in this section.

Figs. 9–10 show the results obtained with the minimum joint velocity norm control law, i.e., Eq. (2) with $\dot{\boldsymbol{q}}_N = \boldsymbol{0}$. In this experiment, the robot was not able to reach the final position because the commanded motion requested an acceleration beyond the robot

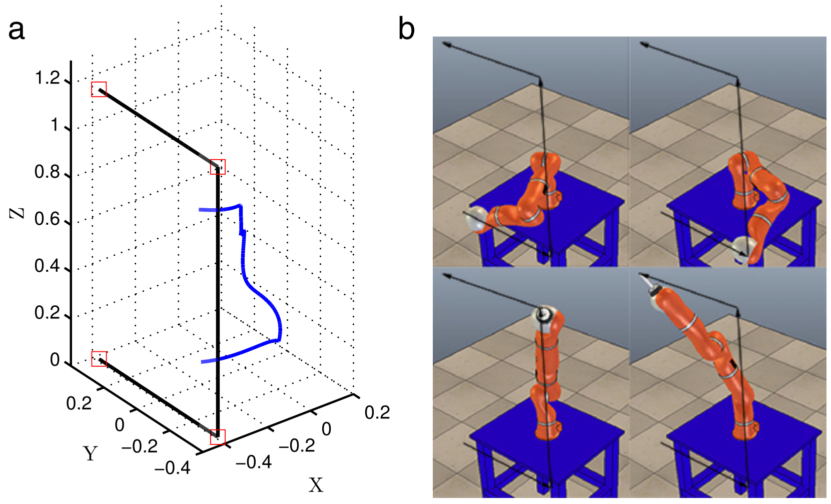


Fig. 6. Execution of the multiple point-to-point motion task with the discrete-time velocity control law optimizing torque as in Eq. (35): (a) end-effector (black) and elbow (blue) traced path; (b) screenshots from the simulation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

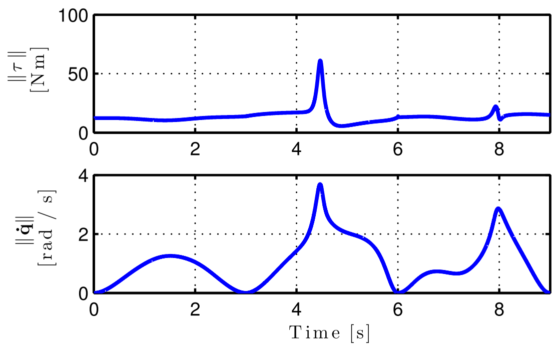


Fig. 7. Numerical results for the discrete-time velocity control law in Eq. (35): [top] norm of the joint torque $\|\boldsymbol{\tau}\|$; [bottom] norm of the joint velocity $\|\dot{\boldsymbol{q}}\|$.

capabilities, and thus the KUKA low-level controller stopped the robot to prevent damages. The critical peaks can be seen after $t = 7$ s in the joint acceleration norm plotted in Fig. 10.

We considered next the discrete-time velocity control law (23) with $\lambda = 0.99$, which is our conversion of the acceleration control law (22) when the joint velocity damping gain is $k_d = 5$. The obtained experimental results are shown in Figs. 11–12. During the first part of task execution, the robot motion is very similar to the one obtained with the standard minimum norm velocity control law. However, in the second part the robot is able to reach the final position, thanks to the strategy of minimizing the variation of the joint velocity. This can also be appreciated when comparing the joint acceleration norm plots of Figs. 10 and 12. It should be emphasized that the obtained behavior depends mainly from the configuration space trajectory exploited by the control law (22) while executing the desired Cartesian trajectory.

Apart from the above desirable property, our objective was also to verify the equivalence of the robot behavior obtained using the proposed discrete-time velocity control design with that resulting from the acceleration-level control (22). This comparison is made in Fig. 13. It is rather evident that the robot controlled in acceleration or with the proposed discrete-time velocity control law performs practically in the same way. The negligible differences reported are mainly due to numerical rounding, non-synchronization of the two experiments, and the limited real time capability of ROS which does not guarantee the acquisition of all joint positions at the same time.

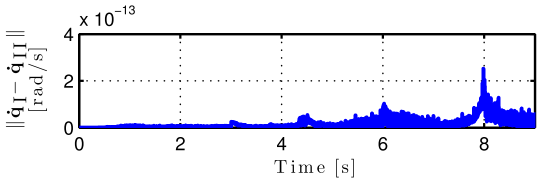


Fig. 8. Norm of the difference between the joint velocity command $\dot{\boldsymbol{q}}_I$ from the discrete-time velocity control law (35) and the joint velocity $\dot{\boldsymbol{q}}_{II}$ obtained from the torque command (29) with damping (34).

6.2. Torque optimization

As in the simulations of Section 5.2, we have compared also in experiments the dynamically consistent torque control law (29) with damping (34) to its implementation as discrete-time velocity control law (35), converted using the proposed method. In these experiments, we have used the accurate dynamic model of the KUKA LWR IV robot identified in [36].

The desired point-to-point motion task between multiple points is identical to the one considered so far, except that the points are now traversed in the reverse order. This has been necessary in order to avoid joint range limits during task execution. Figs. 14–16 refer to the results obtained when using the discrete-time velocity control law (35). Remember that this law has the property of optimizing the norm of the joint torques, weighted by the inverse inertia matrix and biased by the joint torque used for self-motion damping.

Fig. 16 shows the difference between the joint velocity command obtained with the discrete-time velocity control law (35) and the joint velocity obtained with the dynamically consistent torque command (29) with damping (34). The practical equivalence is then confirmed.

7. Conclusions

We have presented a systematic method for converting any motion control law defined at the acceleration or torque level for redundant robots into a discrete-time joint velocity control law, preserving the same properties of the original second-order scheme. The method has been illustrated with a few kinematic and dynamic examples, including control laws that minimize in norm the joint acceleration, possibly biased towards a preferred value used for damping self-motions, or minimize weighted norms of the

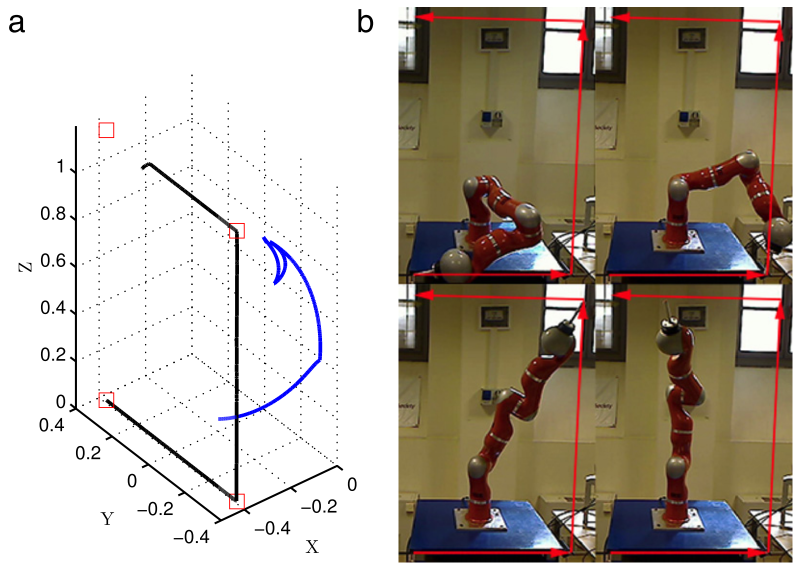


Fig. 9. Minimum joint velocity control: (a) end-effector (black) and elbow (blue) traced path; (b) screenshots from the experiment. The robot was not able to complete the task. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

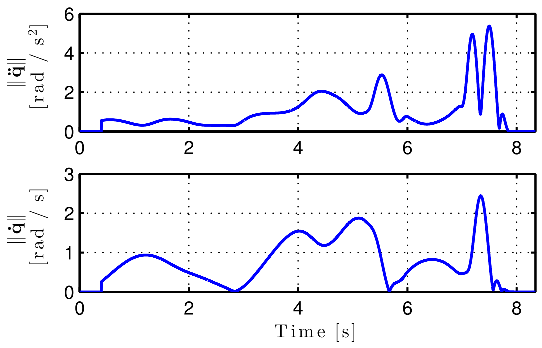


Fig. 10. Experiment with minimum joint velocity control: [top] norm of joint acceleration $\|\ddot{\boldsymbol{q}}\|$; [bottom] norm of joint velocity $\|\dot{\boldsymbol{q}}\|$.

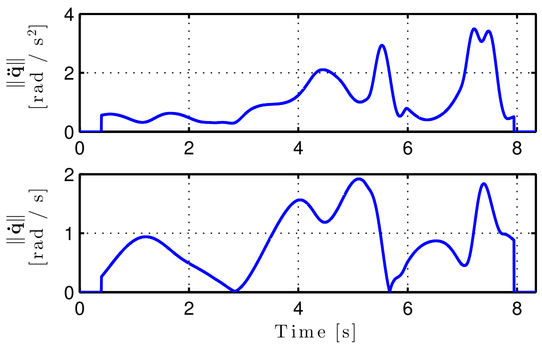


Fig. 12. Experiment with the discrete-time velocity control (23) with null-space damping $\lambda = 0.99$: [top] norm of joint acceleration $\|\ddot{\boldsymbol{q}}\|$; [bottom] norm of joint velocity $\|\dot{\boldsymbol{q}}\|$.

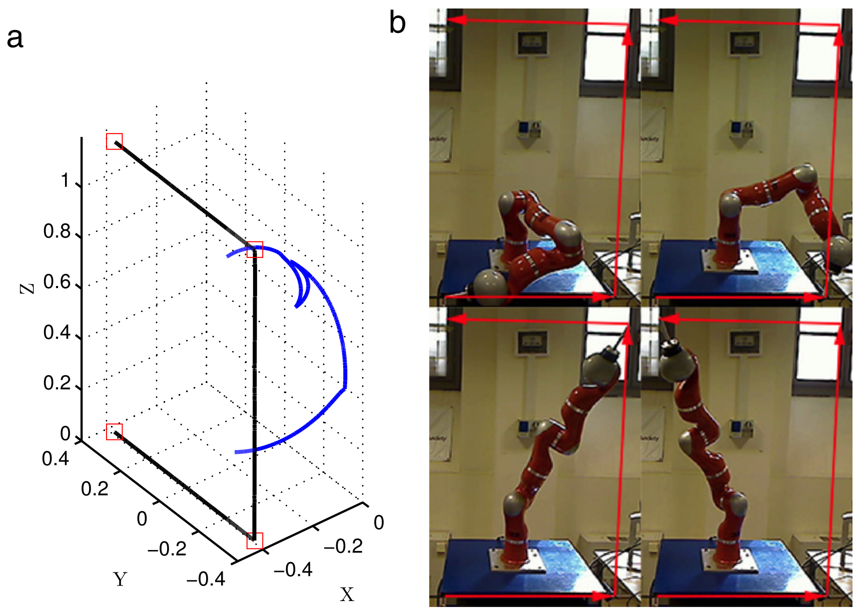


Fig. 11. Discrete-time velocity control (23) with null-space damping $\lambda = 0.99$: (a) end-effector (black) and elbow (blue) traced path; (b) screenshots from the experiment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

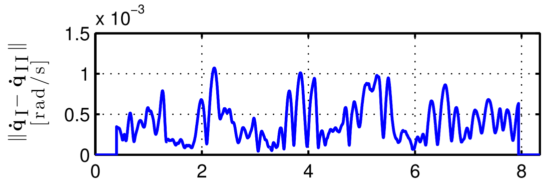


Fig. 13. Norm of the difference between the joint velocity command $\dot{\boldsymbol{q}}_{\rm I}$ from the discrete-time velocity control law (23) with $\lambda = 0.99$ and the experimental joint velocity $\dot{\boldsymbol{q}}_{\rm II}$ associated to the acceleration control (22) with $k_d = 5$.

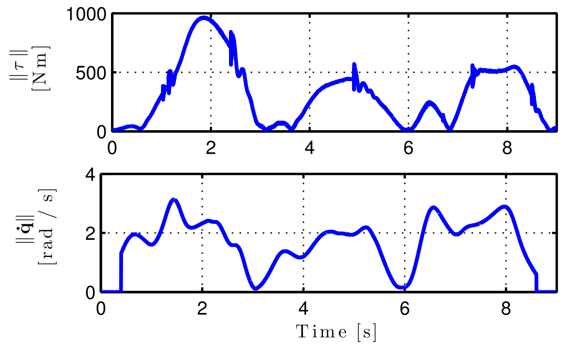


Fig. 15. Experiment with the discrete-time velocity control law (35): [top] norm of the joint torque $\|\boldsymbol{\tau}\|$; [bottom] norm of the joint velocity $\|\dot{\boldsymbol{q}}\|$.

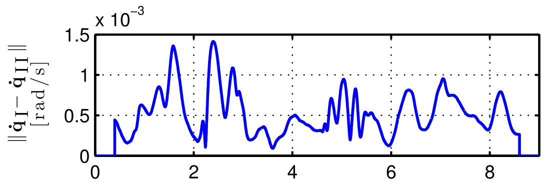


Fig. 16. Norm of the difference between the joint velocity command $\dot{\boldsymbol{q}}_{\rm I}$ from the discrete-time velocity control law (35) and the experimental joint velocity $\dot{\boldsymbol{q}}_{\rm II}$ associated to the torque control (29) with damping (34).

instantaneous joint torque. The equivalent robot behavior obtained with the proposed conversion into a discrete-time velocity control law has been verified through simulations and experiments on a 7R KUKA LWR IV robot performing three-dimensional tasks.

A convenient feature is that the conversion of different control laws leads to velocity laws that differ only by the choice of the joint vector in the null space of the task Jacobian. Moreover, the method allows to handle auxiliary tasks in the original control law, such as when locally optimizing an objective function with the projected gradient method or when introducing a hierarchy of multiple tasks with priority. In doing the conversion, we found also a nice analogy between the addition of null-space damping in acceleration/torque second-order schemes and the introduction of a forgetting factor in discrete-time velocity control. More in general, all the obtained results can be formally derived from a general QP formulation, which provides also insights on the role of kinematic/dynamic terms in the null space of a given task.

The main use of our results is that the obtained velocity control laws can be interfaced directly to the low-level servo loops in any (closed) control architecture for robots. The method preserves the same performance of any target second-order motion control scheme at no additional computational cost, without requiring velocity measurements or numerical differentiation of position measurements. As a further benefit, the implementation avoids the use of a desired task acceleration, which is not trivial to define for sensor-based reactive tasks (e.g., collision avoidance).

We are currently working on the integration with a discrete-time version of our SNS (Saturation in the Null Space) method that deals explicitly with hard bounds in the joint space for redundant robots [37]. This will allow taking into account joint acceleration or torque limits even if the robot is controlled at the velocity level. Another interesting issue to be investigated in our future work is how to use the proposed conversion method to implement velocity-level laws that achieve highly dynamic tasks such as controlling contact forces or realizing desired impedance behaviors in human–robot interaction.

Acknowledgments

This work was supported by the European Commission under grant FP7 ICT-287513 SAPHARI. The authors would like to thank Emanuele Magrini for its contribution on performing the experiments.

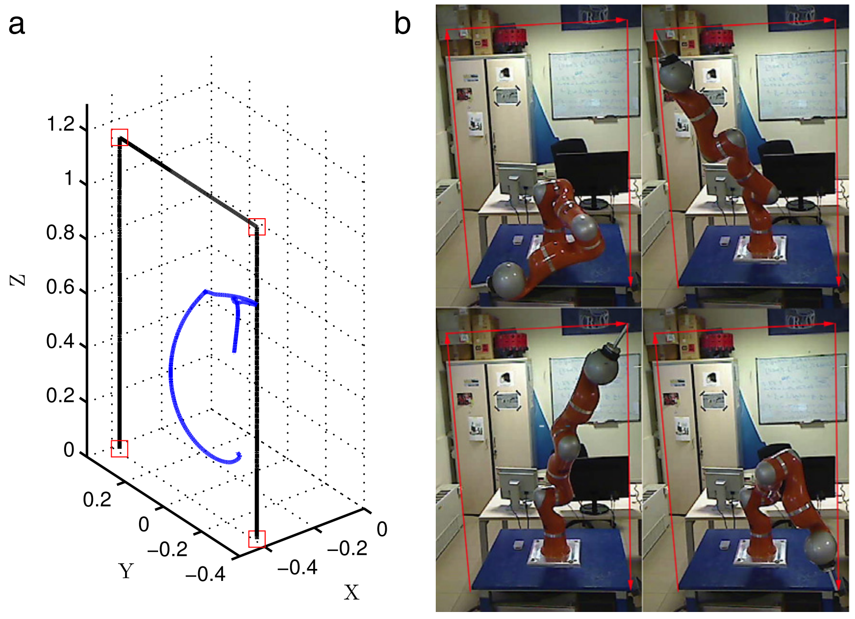


Fig. 14. Discrete-time velocity control law (35) associated to torque optimization: (a) end-effector (black) and elbow (blue) traced path; (b) screenshots from the experiment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Appendix A. QP formulation for optimal use of redundancy

We recall here the relation between equality-constrained least squares programs and weighted pseudoinverse solutions.

Consider the following Quadratic Programming (QP) problem

$$\boldsymbol{y}^* = \arg\min_{\boldsymbol{y}\in\mathcal{S}} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{y}_N\|^2_{\boldsymbol{W}} \tag{A.1}$$

$$\text{with } \mathcal{S} = \left\{\arg\min_{\boldsymbol{y}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{y}-\boldsymbol{b}\|^2\right\} \tag{A.2}$$

where $\boldsymbol{W}$ is a symmetric, positive definite matrix, $\boldsymbol{A}$ is the $(m\times n)$ constraint matrix, with $m \le n$. The manifold $\mathcal{S}$ in (A.2) can be given the explicit structure

$$\mathcal{S} = \left\{\boldsymbol{y}\in\mathbb{R}^n \text{s.t.}\ \boldsymbol{y} = \boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0\right\} \tag{A.3}$$

with $\boldsymbol{y}_0 \in \mathbb{R}^n$ and where we have used the weighted pseudoinverse matrix $\boldsymbol{A}^{\#}_{\boldsymbol{H}}$ [15], with the symmetric matrix $\boldsymbol{H} > 0$ of weights, and being

$$\boldsymbol{P}_{\boldsymbol{H}} = \left(\boldsymbol{I} - \boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{A}\right)\boldsymbol{H}^{-1} \tag{A.4}$$

a symmetric projector in the null space of $\boldsymbol{A}$. In case $\boldsymbol{A}$ has full rank

$$\boldsymbol{A}^{\#}_{\boldsymbol{H}} = \boldsymbol{H}^{-1}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T\right)^{-1}. \tag{A.5}$$

The QP problem (A.1)–(A.2) can be rewritten as

$$\begin{aligned}\boldsymbol{y}^* &= \arg\min_{\boldsymbol{y}_0\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{y}_N\|^2_{\boldsymbol{W}}\\ \text{s.t. } \boldsymbol{y} &= \boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0.\end{aligned} \tag{A.6}$$

Define the Lagrangian function

$$\begin{aligned} L(\boldsymbol{y}_0) &= \frac{1}{2}\ \|\boldsymbol{y}-\boldsymbol{y}_N\|^2_{\boldsymbol{W}}\\ &= \frac{1}{2}\left(\boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0\right)^T\boldsymbol{W}\left(\boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0\right)\\ &\quad -\boldsymbol{y}_N^T\boldsymbol{W}\left(\boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0\right) + c, \end{aligned} \tag{A.7}$$

where c is a constant, thus negligible for the minimization.

The necessary and sufficient conditions for optimality are [38]

$$\nabla_{\boldsymbol{y}_0} L = \left(\frac{\partial L}{\partial \boldsymbol{y}_0}\right)^T = \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{W}\boldsymbol{A}^{\#}_{\boldsymbol{H}}\boldsymbol{b} + \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{W}\boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{y}_0 - \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{W}\boldsymbol{y}_N = \boldsymbol{0}, \tag{A.8}$$

$$\nabla^2_{\boldsymbol{y}_0} L = \boldsymbol{P}_{\boldsymbol{H}}\boldsymbol{W}\boldsymbol{P}_{\boldsymbol{H}} \ge \boldsymbol{0}. \tag{A.9}$$

Choosing $\boldsymbol{H} = \boldsymbol{W}$ we have

$$\boldsymbol{P}_{\boldsymbol{W}}\boldsymbol{W}\boldsymbol{A}^{\#}_{\boldsymbol{W}}\boldsymbol{b} = \left(\boldsymbol{I} - \boldsymbol{A}^{\#}_{\boldsymbol{W}}\boldsymbol{A}\right)\boldsymbol{A}^{\#}_{\boldsymbol{W}}\boldsymbol{b} = \boldsymbol{0} \tag{A.10}$$

because of the defining property of the weighted pseudoinverse. Thus, condition (A.8) becomes

$$\boldsymbol{P}_{\boldsymbol{W}}\boldsymbol{W}\boldsymbol{P}_{\boldsymbol{W}}\boldsymbol{y}_0 - \boldsymbol{P}_{\boldsymbol{W}}\boldsymbol{W}\boldsymbol{y}_N = \boldsymbol{0}, \tag{A.11}$$

which is satisfied by taking

$$\boldsymbol{y}_0 = \boldsymbol{W}\boldsymbol{y}_N, \tag{A.12}$$

and using the idempotency of $\widetilde{\boldsymbol{P}}_{\boldsymbol{W}} = \boldsymbol{P}_{\boldsymbol{W}}\boldsymbol{W}$.

By combining (A.3) with (A.12), we obtain finally

$$\boldsymbol{y}^* = \boldsymbol{A}^{\#}_{\boldsymbol{W}}\,\boldsymbol{b} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}_{\boldsymbol{W}}\boldsymbol{A}\right)\boldsymbol{y}_N. \tag{A.13}$$

By properly defining terms, Eq. (A.13) covers all optimization formulas found in this paper (and in most of the redundancy resolution literature). For example, the minimum torque norm solution (7) is obtained using

$$\begin{aligned} &\boldsymbol{y} = \ddot{\boldsymbol{q}}, \qquad \boldsymbol{W} = \boldsymbol{M}^2, \qquad \boldsymbol{A} = \boldsymbol{J},\\ &\boldsymbol{b} = \ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \qquad \boldsymbol{y}_N = -\boldsymbol{M}^{-1}\boldsymbol{n}, \end{aligned} \tag{A.14}$$

and then inserting the resulting optimal acceleration in (6).

In case matrix $\boldsymbol{A}$ is close to singularity, a damped pseudoinversion is required for stability, and the (weighted) projector $\boldsymbol{P}_{\boldsymbol{W}}$ has to be computed considering only the linearly independent rows of $\boldsymbol{A}$ [39].

Appendix B. Stability analysis of the floating motion damping

For the choice of the scalar forgetting factor λ in (23), a stability analysis can be performed for a particular but significant situation. Suppose that the robot has accomplished its original task and that a specific task is no longer required. Reset the time counter ($k = 0$), and let the initial joint position and velocity at this time instant be, respectively, $\boldsymbol{q}_0$ and $\dot{\boldsymbol{q}}_0 \neq \boldsymbol{0}$ (some joint is still in motion). Since there is no task from now, we set $\boldsymbol{J}_k = \boldsymbol{0}$, $\forall k \ge 0$, and thus $\boldsymbol{P}_k = \boldsymbol{I}$ (all joint motions are in the null space of this vanishing task). Eq. (23) becomes then

$$\dot{\boldsymbol{q}}_k = \lambda\dot{\boldsymbol{q}}_{k-1}. \tag{B.1}$$

Values such that $|\lambda| > 1$ should be discarded, leading to velocity divergence. Moreover, non-negative values of λ will avoid oscillations. For $\lambda = 1$, the joint velocity will not change, and the joint acceleration is clearly zero (minimized in norm). However, the sequence of position samples will diverge, $\|\boldsymbol{q}_k\| \to \infty$ (in practice, some joints will move to their range limits). For $|\lambda| < 1$, Eq. (B.1) is a contraction mapping and the robot configuration will converge to

$$\boldsymbol{q}_\infty = \boldsymbol{q}_0 + \frac{T\dot{\boldsymbol{q}}_0}{1-\lambda}. \tag{B.2}$$

In particular, for $\lambda = 0$, the robot will stop at the next step, at the cost of a very high joint acceleration.

We can backtrack this analysis to set actual bounds for $k_d = (1-\lambda)/T \ge 0$ in the continuous-time acceleration solution (22), taking into account the need of its discretization to a velocity control law. In fact, for $k_d \in (0, 1/T)$, a uniformly stable behavior is obtained. For $k_d = 1/T$, the acceleration-level law leads to a first-order control with minimum velocity norm. Convergence is obtained also for $k_d \in (1/T, 2/T)$, but pseudo-oscillating in nature, while for $k_d = 2/T$ the self-motion will be persistently oscillating. Finally, for $k_d > 2/T$ there will be a divergent oscillatory motion in the null space of the task.

References

[1] G. Buizza Avanzini, N. Ceriani, A. Zanchettin, P. Rocco, L. Bascetta, Safety control of industrial robots based on a distributed distance sensor, IEEE Trans. Control Syst. Technol. 22 (6) (2014) 2127–2140.

[2] M. Geravand, F. Flacco, A. De Luca, Human–robot physical interaction and collaboration using an industrial robot with a closed control architecture, in: Proc. IEEE Int. Conf. on Robotics and Automation, 2013, pp. 4000–4007.

[3] G. Hirzinger, J. Bals, M. Otter, J. Stelter, The DLR-KUKA success story: robotics research improves industrial robots, IEEE Robot. Autom. Mag. 12 (3) (2005) 16–23.

[4] O. Khatib, P. Thaulad, T. Yoshikawa, J. Park, A torque-position transformer for task control of position controlled robots, in: Proc. IEEE Int. Conf. on Robotics and Automation, 2008, pp. 1729–1734.

[5] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modeling, Planning and Control, third ed., Springer, London, 2008.

[6] T. Yoshikawa, O. Khatib, Compliant humanoid robot control by the torque transformer, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2009, pp. 3011–3018.

[7] W. Townsend, J. Salisbury, Mechanical design for whole-arm manipulation, in: P. Dario, G. Sandini, P. Aebischer (Eds.), Robots and Biological Systems: Towards a New Bionics? Springer, 1993, pp. 153–164.

[8] G. Hirzinger, A. Albu-Schäffer, M. Hähnle, I. Schaefer, N. Sporer, On a new generation of torque controlled light-weight robots, in: Proc. IEEE Int. Conf. on Robotics and Automation, 2001, pp. 3356–3363.

[9] B. Rooks, The harmonious robot, Ind. Robot. 33 (2) (2006) 125–130.

[10] A. De Luca, W. Book, Robots with flexible elements, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 287–319.

[11] B. Donald, P. Xavier, J. Canny, J. Reif, Kinodynamic motion planning, J. ACM 40 (5) (1993) 1048–1066.

[12] S. Chiaverini, G. Oriolo, I. Walker, Kinematically redundant manipulators, in: B. Siciliano, O. Khatib (Eds.), in: Springer Handbook of Robotics, Springer, 2008, pp. 245–268.
[13] A. De Luca, G. Oriolo, B. Siciliano, Robot redundancy resolution at the acceleration level, Lab. Robot. Autom. 4 (2) (1992) 97–106.
[14] F. Flacco, A. De Luca, Discrete-time velocity control of redundant robots with acceleration/torque optimization properties, in: Proc. IEEE Int. Conf. on Robotics and Automation, 2014, pp. 5139–5144.
[15] T.L. Boullion, P.L. Odell, Generalized Inverse Matrices, Wiley-Interscience, 1971.
[16] L. Eldén, A weighted pseudoinverse, generalized singular values, and constrained least squares problems, BIT 22 (4) (1982) 487–502.
[17] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, S. Schaal, A unifying framework for robot control with redundant DOFs, Auton. Robots 24 (1) (2008) 1–12.
[18] J. Hollerbach, K. Suh, Redundancy resolution of manipulators through torque optimization, IEEE J. Robot. Autom. 3 (4) (1987) 308–316.
[19] K. Kazerounian, Z. Wang, Global versus local optimization in redundancy resolution of robotic manipulators, Int. J. Robot. Res. 7 (5) (1988) 3–12.
[20] O. Khatib, The operational space framework, JSME Int. J. Ser. C: Dyn. Control Robot. Des. Manuf. 36 (3) (1993) 277–287.
[21] S. Ma, Local torque optimization of redundant manipulators in torque-based formulation, in: Proc. 20th Int. Conf. on Industrial Electronics, Control and Instrumentation, Vol. 2, 1994, pp. 697–702.
[22] OROCOS Development Team, Kinematics and Dynamics Library (KDL), 2010. http://www.orocos.org/kdl.
[23] M. Felis, Rigid Body Dynamics Library (RBDL), 2011. http://rbdl.bitbucket.org.
[24] S. Chiaverini, B. Siciliano, O. Egeland, Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator, IEEE Trans. Control Syst. Technol. 2 (2) (1994) 123–134.
[25] A. Liégeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, IEEE Trans. Syst. Man Cybern. 7 (12) (1977) 868–871.
[26] Y. Nakamura, Advanced Robotics: Redundancy and Optimization, Addison-Wesley, 1991.
[27] B. Siciliano, J.J. Slotine, A general framework for managing multiple tasks in highly redundant robotic systems, in: Proc. 5th Int. Conf. on Advanced Robotics, 1991, pp. 1211–1216.
[28] A. Escande, N. Mansard, P.-B. Wieber, Hierarchical quadratic programming, Tech. Rep. LAAS No. 12794, 2012, URL: http://hal.archives-ouvertes.fr/hal-00751924.
[29] K. O'Neil, Divergence of linear acceleration-based redundancy resolution schemes, IEEE J. Robot. Autom. 18 (4) (2002) 625–631.
[30] D. Guo, Y. Zhang, Different-level two-norm and infinity-norm minimization to remedy joint-torque instability/divergence for redundant robot manipulators, Robot. Auton. Syst. 60 (6) (2012) 874–888.
[31] L. Ljung, T. Söderström, Theory and Practice of Recursive Identification, MIT Press, 1983.
[32] K. Kazerounian, A. Nedungadi, An alternative method for minimization of the driving forces in redundant manipulators, in: Proc. IEEE Int. Conf. on Robotics and Automation, 1987, pp. 1701–1706.
[33] G. Guennebaud, B. Jacob, et al. Eigen v3, 2010. http://eigen.tuxfamily.org.
[34] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Robotics, Kobe, JPN, 2009.
[35] A. Savitzky, M.J.E. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (8) (1964) 1627–1639.
[36] C. Gaz, F. Flacco, A. De Luca, Identifying the dynamic model used by the KUKA LWR: a reverse engineering approach, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 1386–1392.
[37] F. Flacco, A. De Luca, O. Khatib, Motion control of redundant robots under joint constraints: saturation in the null space, in: Proc. IEEE Int. Conf. on Robotics and Automation, 2012, pp. 285–292.
[38] D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, 1984.
[39] P. Baerlocher, R. Boulic, An inverse kinematic architecture enforcing an arbitrary number of strict priority levels, Vis. Comput. 6 (20) (2004) 402–417.

Fabrizio Flacco is a PostDoc at the Robotics Laboratory, Sapienza University of Rome since January 2012. He received the Ph.D. in System Engineering at the DIAG (Dipartimento di Ingegneria Informatica, Automatica e Gestionale), Sapienza University of Rome in 2012, and the bachelor degree in Computer Science Engineering and the master degree in Automation Engineering both from the University of Pisa in 2004 and 2007. He joined the Interdepartmental Research Center "E. Piaggio" in 2008. During the Ph.D. he has been a visiting scholar at the Artificial Intelligence Laboratory, Stanford University. His interests are in the general area of human-friendly robotics. In particular, he approached the problematic of safe physical human–robot collaboration from different levels. He worked on the control of variable stiffness actuators (VSA), kinematic control of redundant robots and algorithms for collision avoidance/detection/reaction.

Alessandro De Luca received the Ph.D. in Systems Engineering in 1987 from the Sapienza University of Roma, where he is a professor of Robotics and Automatic Control since 2000. His research interests include motion planning and nonlinear control of flexible, redundant, and underactuated manipulators, and of wheeled nonholonomic mobile robots, as well as fault detection and isolation, visual servoing, and physical human–robot interaction. He published more than 180 journal and conference papers and book chapters. He has been the Editor-in-Chief of the IEEE Transactions on Robotics (2004–2008), the General Chair of IEEE ICRA 2007, and the VP for Publications of the IEEE Robotics and Automation Society (2012–2013). He received the ICRA 1998 and BIOROB 2012 best conference paper awards, the IROS 2008 best application paper award, and the Helmholtz Humboldt research award in 2005. He is the scientific coordinator of the European FP7 project SAPHARI (2011–2015) and the Director of the Master Course in Control Engineering at Sapienza. He is an IEEE Fellow (class of 2007).