A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks
Nicolas Mansard, Oussama Khatib, Fellow, IEEE, and Abderrahmane Kheddar, Member, IEEE

Abstract—The control approaches based on the task function formalism, and particularly those structured as a prioritized hierarchy of tasks, enable complex behaviors with elegant properties of robustness and portability to be built. However, it is difficult to consider a straightforward integration of tasks described by unilateral constraints in such frameworks. Indeed, unilateral constraints exhibit irregularities that prevent the insertion of unilateral tasks at any priority level, other than the lowest, of a hierarchy. In this paper, we present an original method to generalize the hierarchy-based control schemes to account for unilateral constraints at any priority level. We develop our method first for task sequencing using only the kinematics description; then, we expand it to the task description, using the operational space formulation. The method applies in robotics and computer graphics animation. Its practical implementation is exemplified by realizing a real-manipulator visual servoing task and a humanoid avatar reaching task; both experiments are achieved under the unilateral constraints of joint limits.

Index Terms—Dynamics, humanoid robots, kinematics, redundant robots, visual servoing.

I. INTRODUCTION

The task function approach [1] or the operational space formulation [2] has been introduced to simplify the control problem in robotics. Working directly in a properly chosen task space is more intuitive to simply specify robot objectives. It also enables addressing the control problem directly in the sensor space, which closes the control loop more tightly and improves the robustness and accuracy of the control law [3]–[5]. Finally, since a same task is valid for a large set of robots, task function control schemes are portable and easy to modify and maintain; they can be adapted easily to any robot structure and can be considered in advanced robotic architectures.

Manuscript received January 25, 2008; revised June 13, 2008, November 7, 2008, and March 25, 2009. First published May 15, 2009; current version published June 5, 2009. This paper was recommended for publication by Associate Editor K. Yamane and Editors H. Arai and K. Lynch upon evaluation of the reviewers’ comments. This work was supported in part by the European Community under Contract FP6-034002 ROBOT@CWE.

N. Mansard is with the Laboratoire d’Architecture et d’Analyse des Systèmes, Centre National de la Recherche Scientifique, Université de Toulouse, 31077 Toulouse Cedex 4, France (e-mail: nicolas.mansard@laas.fr).

O. Khatib is with the Artificial Intelligence Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-9010 USA (e-mail: ok@cs.stanford.edu).

A. Kheddar is with the Laboratoire d’Informatique, de Robotique et de Microélectronique de Montpellier, Centre National de la Recherche Scientifique, Université de Montpellier 2, 34092 Montpellier Cedex 5, France, and with the CNRS-National Institute of Advanced Industrial Science and Technology Joint Robotics Laboratory, International Mixed Unit 3218/ Collaborative Research Team, 305-8568 Tsukuba, Japan (e-mail: kheddar@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2009.2020345

In addition, these methods directly produce the kinematics or dynamic model to decouple the motions due to the task from the free remaining motions (i.e., the motions that let the task be invariant) [6], [7]. A secondary task can then be applied in the space of free motions, and recursively, a hierarchic set of tasks (or stack of tasks) can be considered [8], [9]. Methods based on a hierarchy of tasks are gaining soundness in the construction of complex behaviors for redundant robots, especially humanoids [10]–[14]. Tasks are generally defined by a set of equalities of reference such as $e = 0$, where $e = s - s^*(t)$ is an error to be regulated to 0. Each component of a task thus represents a bilateral constraint. On the other hand, there are tasks that would require description through a set of unilateral constraints that are typically represented by inequalities $e_i \leq 0$. Examples of such constraints are joints limits [15], collision avoidance [16], [17], visibility loss [18], [19], or avoidance of singularities [20], [21]. A typical example is given by Fig. 1: The robot has to reach a position with its right end-effector, while ensuring the joint limit constraints. Such tasks present a strong irregularity at the activation point $e_1 = 0$. Due to this irregularity, it is impossible to consider unilateral constraints as classical (bilateral) ones. Specific unilateral constraints have been considered in a large number of works in the past, in particular, via the gradient projection method (GPM) [6], which was revisited recently in humanoid robotics [22]: Unilateral constraint, such as collision avoidance, is embedded in a cost function [23] whose gradient is projected in the space of free motion as a lowest priority task. This can be seen as a regularization of the unilateral function. However, the irregularity persists at the derivative level and prevents the insertion of the unilateral constraint at any level of the task hierarchy but the lowest priority one. Indeed, the unilateral constraints

Fig. 1. Reaching task while maintaining a maximal angle on the elbow.
have mostly been considered as a last objective to be optimized only when enough degrees of freedom (DOFs) are available. A collision-avoidance task is used in [24] at the top-priority level while the irregularities of the task function being smoothed used the damped least square (DLS) inverse [25], [26], which is not always sufficient [27]. The problem has been formulated as a constrained optimization problem in [28]: Inequalities can be directly used to specify the unilateral constraints in the definition of the optimization problem. Other methods use a supervision module that modifies the hierarchy when the unilateral constraint is violated: An equivalent bilateral constraint is introduced as a top-priority task to enforce temporarily the execution of the unilateral constraint [10], [13], [29], [30]. In the particular case of joint-limit constraints (where each constraint is decoupled from the others), this clamping that imposes temporarily the constraint can be damped to ensure a smooth behavior [31].

In this paper, we propose a generic solution to build a smooth control law from any kind of unilateral constraints at any priority level of the hierarchy. To this end, we propose to use a specific inverse operator introduced in [27] to smooth the irregularity of the unilateral constraint while computing the control law. Because the operator is only able to smooth the irregularity of the unilateral activation, the proposed approach still relies on the use of DLS to smooth the control law when crossing a singularity [32]. This operator was first proposed to avoid the loss of visual markers during a visual servoing. However, the method was only applied for the kinematic inverse of a single task. We propose here to extend its formulation to a hierarchically organized set of tasks with unilateral constraints described through inverse kinematics or dynamics.

The paper is organized as follows. The first part presents the proposed method in a formal way: in Section II, we first recall the classical inverse-kinematics solutions and emphasize the causes of discontinuity while considering unilateral constraints. We then extend in Section III the solution of [27] for a hierarchy of tasks and unilateral constraints. The proposed solution is proven to appropriately decouple the tasks of the hierarchy in Section IV. We finally extend the proposed solution for inverse-dynamics control in Section V. Based on this theoretical approach, the second part focuses on the intuitive aspect of the method. Section VI presents with all the details a simple case study inspired from Fig. 1. A set of experiments is discussed in Section VII.

II. INVERSE-KINEMATICS CONTROL

A. Considering One Task

Let \( \mathbf{q} \) be the vector of robot joint positions and \( \mathbf{e} \) be the task function. We consider a controller based on joint velocities \( \dot{\mathbf{q}} \) (we will generalize to torque control in Section V). The Jacobian \( \mathbf{J} \) of the task \( \mathbf{e} \) is defined by

\[
\dot{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J} \dot{\mathbf{q}}. 
\]

Let \( n \) be the number of DOFs \( (n = \text{dim} \mathbf{q}) \) and \( m \) be the size of the task \( (m = \text{dim} \mathbf{e}) \). The controller regulates \( \mathbf{e} \) to \( \mathbf{0} \) according to a reference behavior \( \mathbf{e}^\star \). The joint motion \( \dot{\mathbf{q}} \) realizing \( \mathbf{e}^\star \) is given by the least square inverse of (1)

\[
\dot{\mathbf{q}} = \mathbf{J}^+ \mathbf{e}^\star 
\]

where \( \mathbf{A}^+ \) denotes the Moore–Penrose inverse of \( \mathbf{A} \) [33]. We assume that \( \mathbf{J} \) is perfectly known [34]. The control law is then always stable, and asymptotically stable if \( \mathbf{J} \) is full rank [1].

In the state of the art, an implicit condition of such control schemes is always the constant rankness of the Jacobian \( \mathbf{J} \). Indeed, the pseudoinverse \( \mathbf{J}^+ \) is continuous with respect to \( \mathbf{J} \) only when the rank is constant. When the rank increases or decreases, the continuity is not ensured, which can result in awkward or even dangerous behavior.

B. Evidence of Discontinuity

An unilateral constraint can be written as \( \mathbf{e} < 0 \). In this case, the reference behavior \( \mathbf{e}^\star \) is typically set to

\[
\mathbf{e}^\star = \begin{cases} -\lambda \mathbf{e}, & \text{if } \mathbf{e} > 0 \\ 0, & \text{otherwise} \end{cases} 
\]

Several control laws based on (2) have been proposed [18], [35], [36]. They can be gathered under the common form [27]

\[
\dot{\mathbf{q}} = (\mathbf{HJ})^+ \mathbf{He}^\star 
\]

where \( \mathbf{H} = \text{diag}(h_1, \ldots, h_m) \) is the activation matrix, and

\[
h_i = \begin{cases} 1, & \text{if } e_i > 0 \\ 0, & \text{otherwise} \end{cases} 
\]

This control law is discontinuous. Indeed, consider a task \( \mathbf{e} = (e_1, e_2) \) where the single feature \( e_2 \) gets active. The matrix \( \mathbf{H} \) thus evolves from

\[
\mathbf{H}_t = \begin{bmatrix} \mathbf{I}_{m-1} & 0 \\ 0 & 0 \end{bmatrix}
\]

to \( \mathbf{H}_{t+1} = \mathbf{I}_m \). The control law at \( t \) and \( t+1 \) is thus

\[
\dot{\mathbf{q}}_t = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \end{bmatrix}^+ \begin{bmatrix} e_1^\star \\ e_2^\star \end{bmatrix} 
\]

(5)

These two control laws are obviously not equal in general. Thus, a discontinuity arises between times \( t \) and \( t+1 \).

A classical solution is then to smooth \( \mathbf{H} \) by introducing an activation buffer before the point of activation of the constraint. In the activation buffer, the activation parameter \( h_i \) evolves progressively from 0 to 1. However, it is also possible to show that, in some cases, the control law (4) is equivalent whether \( \mathbf{H} \) is smooth or not [27]. This means that in such a case, the activation buffer used to smooth the irregularities is not considered due to the inner mathematical simplification of the control law. Using simply the pseudoinverse, the activation buffer is then simply useless. See [27] for detailed proofs and explanations on the equivalence of the control law when the changes in \( \mathbf{H} \) are abrupt and when they are smooth.

C. Extension to k Tasks

The solution (2) computed before is only one particular solution of (1); it is the solution of least norm that realizes the
reference behavior $\dot{e}^*$. If the rank of $\mathbf{J}$ is smaller than $n$, a second criterion can be taken into account using the redundancy formalism [1]. The robot motion is given by

$$\dot{\mathbf{q}} = \mathbf{J}^+ \dot{e}^* + \mathbf{P} \mathbf{z}_{q_2}$$

where $\mathbf{P}$ is the projection operator onto the null space of the matrix $\mathbf{J}$ (i.e., $\mathbf{P} = \mathbf{I}_n - \mathbf{J}^+ \mathbf{J}$), and $\mathbf{z}_{q_2}$ is an arbitrary vector used to apply a secondary control law. Due to $\mathbf{P}$, this secondary motion $\mathbf{z}_{q_2}$ is performed without disturbing the main task $\mathbf{e}$ having higher priority.

Let us now consider two tasks $\mathbf{e}_1$ and $\mathbf{e}_2$. The control law performing exactly $\mathbf{e}_1^*$ and if possible $\mathbf{e}_2^*$ is [9]

$$\dot{\mathbf{q}} = \mathbf{J}_1^+ \dot{e}_1^* + (\mathbf{J}_2 \mathbf{P}_1)^+ (\dot{e}_2^* - \mathbf{J}_2 \mathbf{J}_1^+ \dot{e}_1^*).$$

This equation can be generalized to perform $k$ tasks, while ensuring a proper hierarchy between them [9]

$$\dot{\mathbf{q}}_0 = 0 \quad \text{and} \quad \dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{i-1} + (\mathbf{J}_i \mathbf{P}_{i-1})^+ (\dot{e}_i^* - \mathbf{J}_i \dot{\mathbf{q}}_{i-1})$$

where $i = 1, \ldots, k$, and $\mathbf{P}_i$ is the projector onto the null space of the augmented Jacobian $\mathbf{J}_i^A = (\mathbf{J}_1, \ldots, \mathbf{J}_i)$. The robot joint velocity realizing all the tasks is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_k$. A similar recursion can be established to compute $\mathbf{P}_i$ at low cost [10].

The limited Jacobian $\mathbf{J}_i \mathbf{P}_{i-1}$ is orthogonal to the Jacobian $\mathbf{J}_i$ of any task $\mathbf{e}_j$ having priority over $\mathbf{e}_i$ [32]. All the tasks are thus decoupled from each others. If one task is not feasible, it will be realized only at best under the constraint of not disturbing the previous tasks of higher priority.

D. Discontinuities of the Projection Operator and Introduction of the DLSs

Here also, discontinuities can occur if the rank of the Jacobian matrix $\mathbf{J}_i$ is not constant. In particular, while considering an unilateral constraint task (the inverse $(\mathbf{J}_i \mathbf{P}_{i-1})^+$ is then replaced by $(\mathbf{H} \mathbf{J}_i \mathbf{P}_{i-1})^+ \mathbf{H}$), a change of rank results in a discontinuity in $\dot{\mathbf{q}}_i$, as well as in a change of rank of the associate projection operator $\mathbf{P}_i$. This discontinuity propagates to the next level $i + 1$ and, subsequently, to all the following ones amplifying the discontinuity up to the final control law $\dot{\mathbf{q}}_k$.

A solution is to use the DLSs instead of the pseudoinverse [8]. The DLS will smooth the discontinuities coming from the singularities of the system (a singularity occurs when one task that was feasible becomes nonfeasible [32]). On the other hand, it has been proposed to use the DLS to also smooth the discontinuities arising when a feature gets active [24]. Yet, smoothing such a discontinuity calls for high damping terms, which, in turn, compromise the quality of the execution (in particular, the tasks are not well accomplished and the priority order is not ensured). DLSs are not sufficient in practice to solve this problem [27].

A good explanation of such a discontinuity is given in [31]. This discontinuity is also exemplified in Section VI through the simple case study introduced in Fig. 1.

III. KEEPING THE CONTINUITY AT THE KINEMATICS LEVEL

In this section, we build a new control law whose form is similar to (8), yet ensures the continuity even when considering unilateral constraints. This original solution is based on the inversion operator proposed in [27] to solve the discontinuity of control laws similar to (4). Let us first explain the intuition behind the solution.

A. Intuition

Basically, the control law (4) is based on the inverse matrix $(\mathbf{HJ})^+ \mathbf{H}$, which is very similar in shape to the inverse of $\mathbf{J}$ weighted by $\mathbf{H}$, and is denoted as $\mathbf{J}^* \mathbf{H}$. However, since $\mathbf{H}$ may not be strictly positive, this inverse is not properly defined. Moreover, as shown in [37], some coefficient of the weight $\mathbf{H}$ may not be taken into account depending on the rank of $\mathbf{J}$. The solution proposed in [27] is thus to define a new inverse that is close to $(\mathbf{HJ})^+ \mathbf{H}$ but is always properly defined, whatever the positivity of $\mathbf{H}$ and the rank of $\mathbf{J}$.

The activation matrix $\mathbf{H}$ selects which part of a given task is to be taken into account and which part is not. When the activation matrix is composed only of 0 and 1, we want the robot behavior to be exactly equivalent to the one obtained while considering only the active subpart of the task. This is the case when considering $(\mathbf{HJ})^+ \mathbf{H}$ of (4): There is no problem when the activation parameters are binary. However, discontinuities may appear when one or several activation parameters evolve from 0 to 1. The new inverse operator is built from this observation: as (4), it is equal to $(\mathbf{HJ})^+ \mathbf{H}$ when $\mathbf{H}$ is only composed of 0 and 1. Contrary to (4), it ensures the continuity when any parameters are in between.

In the case where only one parameter $h_i$ evolves from 0 to 1, the continuous inverse is simply an homotopy evolving from $h_i = 0$ to $h_i = 1$. When several parameters evolve from 0 to 1, the continuous inverse is generalized as a multidimensional homotopy driven by $h_i$.

Using this inverse directly, a continuous control law that accomplishes one set of unilateral constraints written under the form $(\mathbf{e}, \mathbf{H})$ is straightforward.

B. Considering One Task

To formalize the proposed idea, we briefly recall the control law proposed in [27] for one task. Consider a task $\mathbf{e}$ (vector, dimension $m$), its Jacobian $\mathbf{J}$ (matrix, dimension $n \times m$ and constant rank $r$), and its activation matrix $\mathbf{H}$ (diagonal matrix of dimension $m \times m$ whose diagonal components $(h_i)_{i=1,\ldots,m}$ are in the interval $[0,1]$). The continuous inverse of $\mathbf{J}$ activated by $\mathbf{H}$ is defined by [27]

$$\mathbf{J}^\mathbf{H} = \sum_{\mathcal{P} \in \mathcal{P}(m)} \left( \prod_{i \in \mathcal{P}} h_i \right) \mathbf{X}_{\mathcal{P}}$$

where $\mathcal{P}(m) = \mathcal{P}([1 \cdots m]) = \{ \mathcal{P} | \mathcal{P} \subset [1 \cdots m] \}$ are all the subsets composed of the $m$ first integers (i.e., the set of all combinations without repetitions), and $\mathbf{X}_{\mathcal{P}}$ are the coupling matrices of $\mathbf{J}$ defined by

$$\text{if } \mathcal{P} = \emptyset, \quad \mathbf{X}_{\emptyset} = \mathbf{0}_{s \times m}$$

otherwise $\forall \mathcal{P} \in \mathcal{P}(m), \quad \mathbf{X}_{\mathcal{P}} = \mathbf{J}^\mathbf{P} - \sum_{\mathcal{Q} \subset \mathcal{P}} \mathbf{X}_{\mathcal{Q}}$
where \( J_P = H_0 J \), with \( H_0 \) being a diagonal matrix whose diagonal components \( h_i \) are equal to 1 if \( i \in \mathcal{P} \), and to 0 otherwise, and \( Q \subseteq \mathcal{P} \) designates all the subsets of \( \mathcal{P} \) that are not \( \mathcal{P} \). This inverse is proven to have two noteworthy properties [27].

1) It is continuous with respect to the variation of the activation matrix \( H \).
2) It is equal to the classical inverse \((HJ)^+H\) when the components of \( H \) are binary (\( h_i = 0 \) or \( h_i = 1 \)).

This means that the resulting control law is continuous and keeps the same behavior and all the properties of local convergence of the corresponding classical control law. The continuous inverse (9) can also be explicitly written as

\[
J^+H = \sum_{P \in \mathbb{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \notin P} \left(1 - h_i\right) J_P. \tag{11}
\]

This assertion is proven in Lemma 2 in Appendix C.

As an example, the development of the continuous inverse for a 2-D task \((e = (\epsilon_A, \epsilon_B), H = \text{diag}(h_A, h_B))\) is given as

\[
J^+H = [h_A J_A^+ h_B J_B^+] + h_A h_B X_{AB} = h_B(1 - h_A) J_A^+ + h_A(1 - h_B) J_B^+ + h_A h_B J^+. \tag{12}
\]

Intuitively, when \( h_A \) (respectively \( h_B \)) tends toward 0, the corresponding feature smoothly tends toward not being considered in the control law any more. Reciprocally, when both \( h_A \) and \( h_B \) tend toward 1, the control law tends to be the classical control law (2). Finally, using this inverse, the control law that applies to the task \( e \) upon activation \( H \) is [27]

\[
\dot{\epsilon} = J^+H \dot{\epsilon}. \tag{13}
\]

Using (13), we can assess that each component \( \epsilon_i \) of \( \dot{\epsilon}^* \) is

1) perfectly realized if the corresponding \( h_i \) is equal to 1;
2) not taken into account if \( h_i \) is zero,
3) partially realized otherwise.

Control law (13) can only be used for a single task. We now propose an original solution to extend this generic solution to any set of tasks and constraints.

C. Extension to Two Tasks

As regards the classical redundancy, (13) is not the unique solution. The general solution can be written as

\[
\dot{\epsilon} = \dot{\epsilon}_1 + P^1_{\oplus} z_{\dot{q}_2} \tag{14}
\]

where \( \dot{\epsilon}_1 = J_1 \hat{\epsilon}_1^* \), and \( P^1_{\oplus} = I - J_1 \hat{\epsilon}_1^* J_1 \). Let us now consider two tasks \((e_1, H)\) and \(e_2\). Introducing (14) in (1), we obtain

\[
\dot{\epsilon}_2 = J_2 \dot{\epsilon}_1 + J_2 P^1_{\oplus} z_{\dot{q}_2}. \tag{15}
\]

We now search the optimal \( z_{\dot{q}_2} \) that performs \( e_2 \). Inverting directly this last equation, by analogy to (7), would give us the following control law:

\[
\dot{q}_1 = \dot{q}_1 + P^1_{\oplus} (J_2 P^1_{\oplus} + (\dot{\epsilon}^*_2 - J_2 \dot{\epsilon}_1)). \tag{16}
\]

However, the matrix \( P^1_{\oplus} \) does not have a constant rank. Indeed, the rank of \( P^1_{\oplus} \) evolves from \( n \) (the robot number of DOF) when all the features are inactive to \( n - m \) when all the features are active. Therefore, this control law will lead to the same discontinuities than the classical control laws, each time the rank of \( P^1_{\oplus} \) changes. Instead, we recognize a form similar to (4), with \( P^1_{\oplus} \) at the place of the activation matrix. The same discontinuities as in (4) would thus occur and can be solved once more by using the continuous inverse operator. However, it is first necessary to generalize the continuous inverse for nondiagonal matrix such as \( P^1_{\oplus} \).

D. Continuous Inverse Activated by a Nondiagonal Matrix

Let \( Q \) be any matrix of size \( m \times n, n \leq m \), and \( W \) be a positive symmetric matrix of size \( m \times m \) whose singular values are all between 0 and 1. We denote by \( U, \sigma \) the eigenvalue decomposition of \( W \), with \( \sigma \) being the vector of eigenvalues, and \( S = \text{diag}(\sigma) \) is the corresponding diagonal matrix \((W = USU^T)\). Using this decomposition, we can write

\[
(QW)^+ W = (SQ_a)^+ SU^T \tag{17}
\]

where \( Q_a = U^T Q \).

We now recognize the previous form (4), with \( Q_a \) being the matrix to be inverted and \( S \) being the activation matrix. This time, the matrix \( S \) is a diagonal matrix with components in [0, 1]. The continuity to the change of rank of \( S \) is obtained by applying the continuous inverse of \( Q_a \) activated by \( S \). We thus define the continuous inverse of a matrix \( Q \) activated by any (nondiagonal) positive symmetric matrix whose eigenvalues are in [0, 1] by the following equation:

\[
Q^\ominus (W) = Q_a^\ominus S U^T = \sum_{P \in \mathbb{P}(m)} \left( \prod_{i \in P} \sigma_i \right) X_P Q_a U^T \tag{18}
\]

where \( X_P Q_a \) are the coupling matrices of \( Q_a \).

Until now, we have only defined the continuous inverse activated on the left (i.e., the activation matrix acts on the rows of the matrix by multiplying on the left \( W Q \)). It is easy to generalize to the inverse activated on the right (i.e., where the activation matrix acts on the columns)

\[
Q^\oplus (W) = ((Q^\top)^\ominus W)^\top = \sum_{P \in \mathbb{P}(m)} \left( \prod_{i \in P} \sigma_i \right) X_P^\top \tag{19}
\]

with \( X_P \) being the coupling matrices of \( Q^\top \). The continuous inverse of \( Q \) activated on the right by a no-diagonal symmetric matrix \( W \) is similarly defined by \( Q^\oplus (W) = ((Q^\top)^\ominus W)^\top \).

We have defined two new inverse operators: \( Q^\oplus \) is the continuous inverse where the activation matrix \( W \) acts on the lines of \( Q \) (i.e., in the case of the Jacobian \( J \), the activation acts on the input variables by activating or deactivating some features of \( e \)). In \( Q^\ominus W \), the activation matrix \( W \) acts on the columns (i.e., on the output \( \dot{q} \) by activating some DOFs of the robot).

Both inverses are very similar in shape to the weighted inverse \( Q^\# W \) and \( Q^\# W \). However, contrary to the weighted inverse, the continuous inverse is always properly defined, even when \( W \) is not strictly positive. Moreover, \( W \) is always properly taken into account, whatever the rank of \( J \). We are now going to use this new inverse instead of the weighted inverses inside the control law (16).
E. Extension to k Tasks

Considering the tasks \( (e_1, H) \) and \( e_2 \) of Section III-C, control law (16) can now be replaced. The continuous inverse is valid since the eigenvalues of \( P^1_\oplus \) are in \([0,1]\) [this is obtained by proving that \( \forall x, x^T P^1_\oplus x \leq 1 \), by developing the sum (11)]. With the notations of (14), the control law that performs \( e_2 \) under constraint \( e_1 \) is

\[
\dot{q} = J_1 e_1^* + J_2 P^1_\oplus (\hat{e}_2^* - J_2 \dot{q}_1) \tag{20}
\]

where \( J_2 P^1_\oplus \) is the continuous inverse of \( J_2 \) activated by \( P^1_\oplus \). The corresponding projector \( P^2_\oplus \) is obtained by

\[
P^2_\oplus = P^1_\oplus - J_2 P^1_\oplus J_2. \tag{21}
\]

The control law (20) can then be extended to \( k \) tasks \( (e_1, H) \), \( e_2, \ldots, e_k \) by using the following recursion:

\[
\dot{q}_i = \dot{q}_{i-1} + (J_i)^{P^1_\oplus}(\hat{e}_i^* - J_i \dot{q}_{i-1}), \quad i = 1, \ldots, k.
\tag{22}
\]

A similar recursion is used to compute the operators \( P^i_\oplus \)

\[
P^0_\oplus = I \quad \text{and} \quad P^i_\oplus = P^{i-1}_\oplus - J_1 P^{i-2}_\oplus J_1, \quad i = 1, \ldots, k. \tag{23}
\]

The control law is very similar in shape to the classical control law (8). However, it ensures the continuity whatever the evolution of the activation of the features. Moreover, we show in the next section that it also ensures the priority order of the active features, i.e., when a feature is fully active, it is not disturbed by the tasks of lower priority.

F. Damped Least Square

The continuous inverse is built from a sum of partial pseudoinverses of the Jacobian matrix. It has been proven to be continuous with respect to the evolution of the activation matrix. However, like the classical pseudoinverse, it is sensitive to the singularities of the Jacobian. The set of singular points of the pseudoinverse is the same as the singular points of the continuous inverse. The only points that are regularized by the continuous inverse are the activation points of the unilateral constraints, which were non-regular using the classical approach.

To smooth these singularities of the Jacobian, it is possible to use the DLS instead of the classical pseudoinverse in the definition of the continuous inverse (9). The side effect is that the priority order will no longer be ensured perfectly. The larger the DLS parameter, the more disturbed the priority order will be. However, as the irregularities of the unilateral constraints are smoothed by the continuous inverse construction, the DLS parameter can be set very low. Another solution is to avoid the singular points of \( J \) by setting an explicit constraint to avoid their neighborhood, as proposed in [20] and [21].

The proposed control law is continuous whatever the activation of the unilateral constraints. It is illustrated by a case study in Section VI-C, where we prove in the following section that it also complies with the task hierarchy, i.e., a given task is not disturbed by the secondary tasks.

IV. ORTHOGONALITY OF THE CONTINUOUS INVERSE

In the classical redundancy, the projection operator is orthogonal and guarantees the decoupling (or invariance) of the main task from secondary tasks. This is not the case when using the continuous inverse. Indeed, the projection operator \( P^1_\oplus \) induced in (14) is not orthogonal (i.e., some eigenvalues are not null). In fact, the orthogonality is not required for inverse: Indeed, the motion on the DOF corresponding to none fully active features should not be forbidden for the secondary tasks. However, we want to ensure that the behavior of fully active features is the same as in the classical case. We first prove that the projection defines an orthogonal relation with respect to the Jacobian (i.e., \( J_1 P_1 = 0 \)) on the fully active lines. We then prove that this orthogonality is valid for (22) (i.e., \( J_1 J_2 P^1_\oplus = 0 \)).

A. Orthogonality of the Projection Operator

The GPM-like formulation of the control law is given by (14). When using the pseudoinverse, the orthogonality is ensured by \( J_1 P_1 = 0 \). In our continuous inverse formulation, \( J_1 P^1_\oplus \neq 0 \). Indeed, let us consider the case of a 1-D task \( e = (e_1) \), whose Jacobian is \( J_1 \) and activation is \( h_1 \). Then, \( J_1 e^H = h_1 J_1^* \) and \( P^1_\oplus = I - h_1 J_1^* J_1 \). The orthogonality is not verified since \( J_1 P^1_\oplus = (1 - h_1) J_1 = 0 \) iff \( h_1 = 1 \). However, as can be seen on this counterexample, the orthogonality of the features whose activation is 1 can be verified.

Theorem 4.1 (Orthogonality): Let \( (e, J) \) be an \( m \)-dimension task whose activation matrix is \( H = \text{diag}(h_1, \ldots, h_m) \). Suppose that \( h_k = 1 \). Then, the \( k \)th line of the matrix \( JP^1_\oplus \) is null.

The proof of this theorem is given in Appendix A3.

Corollary 4.1: In control law (14), the fully active features are not disturbed by the secondary term \( z_{\hat{q}_2} \). Moreover, if feature \( k \) is feasible (i.e., if \( J^* \) is equal to the identity on the line \( k \)) and fully active (\( h_k = 1 \)), then its evolution \( \hat{e}_k \) is equal to the reference behavior \( \hat{e}_k^* \).

Proof: First, let us prove the independence with respect to \( z_{\hat{q}_2} \). The effect of control law is computed from \( \hat{e} = J_0 \hat{q} \). Introducing (14) in this last equality gives \( \hat{e} = JJ^* \hat{q} + JP^1_\oplus z_{\hat{q}_2} \).

We know from Theorem 4.1 that the second term is null on the lines where the activation is 1, whatever the value of \( \hat{q}_2 \), which proves the independence.

Consider \( k \) so that \( h_k = 1 \) and \( JJ^* \) is equal to the identity on line \( k \). If \( k \in P \), then \( JJ^* \) equals the identity on line \( k \). Using the same arguments as in the proof of Theorem 4.1, (74), \( J_0 \hat{q} \) is finally proved to equal \( \hat{e}_k^* \) on the line \( k \).

These results ensure that the behavior of (14) is the same as in the classical GPM case when the feature is fully active. In particular, if \( e_{\hat{q}_2} \) is feasible and its activation is 1, then \( \hat{e}_k^* \) is perfectly realized, whatever the value of the second term acting in the pseudonull space \( P^1_\oplus \). Moreover, since the controller is continuous, the behavior tends to be the same as in the classical case when the activation tends to 1.
B. Orthogonality of the Recurrence Control Law

Consider the controller (20). We aim at proving that whatever the task \((e_2, J_2)\), the behavior of the task having higher priority will remain unchanged.

*Theorem 4.2 (Orthogonality of two tasks):* Consider two matrices \(J_1\) and \(J_2\), and an activation matrix \(H_1\). We denote as \(P_j^0\) the projection operator obtained from the continuous inversion of \(J_1\) activated by \(H_1\). Then, \(J_1 J_2 P_j^0\) is equal to 0 on the lines \(k\), whose activations \(h_k\) are 1.

The proof of the theorem is given in Appendix A4.

*Corollary 4.2:* Consider two tasks \((e_1, J_1)\) and \((e_2, J_2)\). Then, (14) ensures that the behavior of the fully active features of \(e_1\) does not depend on \(e_2\). Moreover, if the feature \(k\) of \(e_1\) is feasible and fully active, then it is perfectly realized, whatever the values of the second task \(e_2\).

*Proof:* The behavior of \(e_1\) is obtained from \(e_1 = J_1 \dot{q}\). Replacing (14) in this last equality gives

\[
e_1 = J_1 J_2 P_j^0 \dot{e}_1^* + J_1 J_2 P_j^0 (\dot{e}_2^* - J_2 \dot{q}_1^*). \tag{24}
\]

From Theorem 4.2, we know that the second term of the equation is null on the lines whose activation is 1, which proves the independence of \(e_1\) with respect to \(e_2\).

If feature \(k\) of \(e_1\) is feasible and fully active, Corollary 4.1 proves that it is perfectly realized if \(J_2 = 0\). Since the feature is independent from the second task, this ensures that the feature is perfectly realized whatever the second task \(e_2\).

The result of course generalizes for the recursive form (22). Due to the continuity of the inverse operator, the behavior of the control law tends to the perfect decoupling when the activation of the features tends to 1.

C. Conclusion

Fully active features are decoupled from secondary tasks. Moreover, if a feature is feasible and fully active, it is perfectly realized. These results assess the choice of our control law, and ensure that the stability and decoupling of the classical redundancy formalism hold when the activation tends to 1.

On the other hand, when a feature is not fully active, it does not entirely forbid the motion of the secondary tasks on its corresponding DOF. Therefore, some DOFs remain available to realize a secondary task, which extends the range of possible motions of the robot.

The proposed control law is written for robots that can be controlled in closed-loop velocity \(\dot{q}\). In the following section, we extend these results to the operational space control.

V. Extension to the Operational Space Control

The operational space control [2] unifies in single formalism forces applied by the robot and its displacement in the free or constraint space. To make the paper self-contained, we recall the generic form of the control law in the operational space for a set of tasks. We then introduce the continuous inverse to build a control law taking into account unilateral constraints in the hierarchy.

A. Classical Control Law

1) One Task: The acceleration of the robot joints \(\ddot{q}\) in free space is defined by the dynamic equation

\[
A \ddot{q} + g + \mu = \tau \tag{25}
\]

with \(A\) is the inertia matrix, \(g\) and \(\mu\) are the gravity and Coriolis forces, and \(\tau\) is the motor torques used as the control input. Given a task \(e\) with Jacobian \(J\), it is possible to write

\[
\ddot{e}_i + b = J A^{-1} \tau \tag{26}
\]

with \(b = J A^{-1} (g + \mu) - J \dot{q}\). We define \(\Omega = J A^{-1} J^\top\) and \(\Lambda = \Omega^{-1}\). By multiplying (26) by \(\Lambda\), we obtain

\[
\Lambda \ddot{e}_i + \Lambda b = J^\top \tau \tag{27}
\]

where \(\bar{J} = A^{-1} J^\top\) is the generalized inverse of \(J\) weighted by \(\Lambda^{-1}\).

2) Two Tasks: This control law is only the least acceleration solution [38]. The general solution is [2]

\[
\tau = J_1^\top \Lambda (\ddot{e}_1^* + b_1) + N_1^\top z_{r_2} \tag{29}
\]

where \(z_{r_2}\) is any vector, and \(N_1 = I - J_1^\top \Lambda_1 J_1 A^{-1}\) is the projection operator that ensures the decoupling of \(z_{r_2}\).

3) \(k\) Tasks: The control scheme can be extended by recurrence to a set of \(k\) tasks. Given a set of \(k\) tasks \(e_1 \cdots e_k\), the control law that performs the tasks while preserving the hierarchy is [39] \((i = 1, \ldots, k)\)

\[
\tau_i = \tau_{i-1} + J_i^\top \Lambda_i j_{i-1} (\ddot{e}_i^* + b - J_i A^{-1} \tau_{i-1}) \tag{30}
\]

with \(\tau_0 = 0, \Omega_{ij-1} = J_i A^{-1} N_{i-1} J_i\), and \(\Lambda_{ij-1} = \Omega_{ij-1}^{-1}\). The control law to be applied on the robot is finally \(\tau = \tau_k\). The projection operator is computed by

\[
N_i = N_{i-1}^\top - J_i^\top \Lambda_{ij-1} J_i A^{-1}
\]

\[
= N_{i-1} - J_i^\top J^\top_{ij-1} \tau \tag{31}
\]

where \(J^\top_{ij-1} = A^{-1} J^\top \Lambda_{ij-1}\).

This control law is a generic and classical form for dynamics inverse control [2], [40]. Discontinuities may appear when considering unilateral constraints. We apply the continuous inverse to generalize this generic controller to unilateral constraints.

B. Unilateral Constraint in the Operational Space

Consider a task and its activation matrix \((e, H)\). We want to determine the torque entry to perform \(e\) under activation \(H\). Applying directly the classical formalism (28) with Jacobian \(H J\) and reference acceleration \(H \ddot{e}\) leads to

\[
\tau = J^\top H (H J A^{-1} J^\top H)^{-1} H (\ddot{e}^* + b). \tag{32}
\]

Similarly to the kinematic case, this control law produces discontinuities and improper behaviors when the rank of \(H J\)
changes, in particular, when $HJ$ is ill-conditioned due to some small activation values in $H$. In this last equation, we recognize the form $J^\top H(JJA^{-1}J^\top H)^{-1} = (HJA^{-1})^\top A$, which is the inverse of $HJA^{-1}$ weighted by the matrix $A$. The matrix $(HJA^{-1})^\top A H$ has a similar shape to (4). To prevent the discontinuity, we also use the continuous inverse.

The continuous inverse of $Q$, activated by $H$ and weighted by $A$, is denoted as $Q^{\oplus H \cdot A}$. This inverse is simply built by replacing in the definitions (9) and (10) all the classical pseudoinverses by the inverses weighted by $A$.

Using this definition, the new control law can be written as

$$
\tau = (JA^{-1})^{\oplus H \cdot A} (\dot{e}^* + b).
$$

By using the sum (11) in (33) and factorizing by $J^\top$, the previous control law is finally written as

$$
\tau = J^\top A \oplus (\dot{e}^* + b)
$$

where $A_{\oplus} = \Omega^{\oplus H}$. A secondary term can be taken into account using the projection $N_{\oplus} = I - J^\top A_{\oplus} H A^{-1} = I - J^\top J$, with $J_{\oplus} = A \cdot J^\top A_{\oplus}$. The control law is then given as

$$
\tau = J^\top A_{\oplus} (\dot{e}^* + b) + N_{\oplus}^\top z_{\tau_2}
$$

with $z_{\tau_2}$ being any arbitrary torque.

The final form of the control law is very similar to the bilateral control law (29), where the operational space inertia matrix $A$ has been replaced by its continuous equivalent $A_{\oplus}$. This new inertia matrix is simply used to activate or deactivate the features of $e$ depending on the values of $H$.

C. Extension to Two Tasks

The secondary torque $z_{\tau_2}$ can be used to perform a secondary task. Let $(e_1, H)$ and $e_2$ be two tasks. The control law performing $e_1$ is (35). By introducing this control law in (25), we obtain the equation of motion of the robot constraint by the main task as

$$
Aq + g + \mu = J_1^\top A_{\oplus} (\dot{e}_1^* + b_1) + N_{\oplus}^\top z_{\tau_2}.
$$

By multiplying this last equation by $J_2 A^{-1}$, we obtain the constrained motion expressed in the space of task $e_2$

$$
\dot{e}_2^* + b_2 = J_2 A^{-1} \tau_1 + J_2 A^{-1} N_{\oplus}^\top z_{\tau_2}
$$

where $b_2 = J_2 A^{-1} (g + \mu)$, and $\tau_1$ is the control law defined in (34) applied to $e_1$. It is very tempting to directly inverse (37) to obtain the optimal control $z_{\tau_2}$ performing $e_2$, as has been done in (30), thus obtaining the following control law:

$$
\tau = \tau_1 + N_{\oplus}^\top (J_2 A^{-1} N_{\oplus}^1 A) (\dot{e}_2^* + b_2 - J_2 A^{-1} \tau_1).
$$

However, as in (16), the operator $N_{\parallel}^1$ does not have a constant rank. It is thus necessary to proceed like in Section II-C by using the continuous inverse activated by the projection operator $N_{\parallel}^1$. The generalization of the continuous inverse is only valid for symmetrical matrices whose singular values are between 0 and 1. This is not the case of $N_{\parallel}^1$, since it is not an orthogonal projection operator. However, we can rewrite the inverse weighted by $A$ under the following form:

$$
N_{\parallel}^1 (J_2 A^{-1} N_{\parallel}^1) A = N_{\parallel}^1 \sqrt{A} (J_2 A^{-1} N_{\parallel}^1 \sqrt{A})^\top
$$

$$
= N_{\parallel}^1 \sqrt{A} (J_2 A^{-1/2} (A^{-1/2} N_{\parallel}^1 \sqrt{A}))^\top.
$$

It is therefore possible to normalize $N_{\parallel}^1$ by setting

$$
N_{\parallel}^1 = A^{-1/2} N_{\parallel}^1 \sqrt{A} = I - (J_1 A^{-1/2})^{\oplus H} (J_1 A^{-1/2}).
$$

Using the second part of this equation, it is easy to demonstrate that $N_{\parallel}^1$ is normalized, i.e., it is symmetrical and has proper singular values. The inverse (39) is then $\sqrt{A} N_{\parallel}^1 (J_2 A^{-1/2} N_{\parallel}^1)^\top$. We recognize here the form (4), where $J_2 A^{-1/2}$ is the matrix to be inverted, and $N_{\parallel}^1$ is the activation matrix. Like in (20), we apply the continuous inverse activated on the right by the matrix $N_{\parallel}^1$. The control law that performs the two tasks $(e_1, H)$ and $e_2$ is then given as

$$
\tau = \tau_1 + (J_2 A^{-1})^\top (\dot{e}_2^* + b_2 - J_2 A^{-1} \tau_1)
$$

where $(J_2 A^{-1})^\top (\dot{e}_2^* + b_2 - J_2 A^{-1} \tau_1)$ is the continuous inverse of $J_2 A^{-1}$ weighted by $A$ and activated by $N_{\parallel}^1$, as defined in (33). It is parametrized by two matrices: first, it is weighted by $A$ to be compliant with all the inverses used in the operational space formulation; second, it is activated by the normalized projection operator $N_{\parallel}^1$, which acts on the columns of $J_2 A^{-1}$, i.e., on the output $\tau$, by damping the impulse on the DOF used by the main task $e_1$. The control law finally has a form similar to (30) with $i = 2$ but with the continuous inverse instead of the classical inverse.

D. Extension to $k$ Tasks

This control law can easily be extended to a set of task $(e_1, H), \ldots, e_k$ by analogy to the developments done in [39]

$$
\tau_i = \tau_{i-1} + (J_i A^{-1})^\top (\dot{e}_i^* + b_i - J_i A^{-1} \tau_{i-1})
$$

for any $i = 1, \ldots, k$. The recurrence is initialized with $\tau_0 = 0$. The control law to be applied on the robot is finally $\tau = \tau_k$. A second recurrence is used to compute the projection operator

$$
N_{\parallel}^1 = N_{\parallel}^1 - (J_1 A^{-1}) N_{\parallel}^1 A J_1 A^{-1}.
$$

This final form is a direct generalization of (41) and has a form similar to (30). It is, moreover, possible to prove that the decoupling properties of the classical controller (30) hold for the fully active features. The generalization of the properties proved in Section IV is straightforward when redoing the computations of the proof with the specific matrices introduced by the dynamics.

E. Extension to Free-Floating Robots

Mobile robots, such as humanoids, may have additional underactuated DOFs. In the case of humanoids, when at least one foot is in rigid planar contact with the ground, the robot is controllable, thus providing that the task controller keeps the
contact configuration unchanged. To control such a system, the first solution is to make the assumption that the robot contacts are fixed. The robot can then be controlled as any other classical manipulators. However, this assumption is risky and is not sufficient to denote all the complexities of the robot.

On the other hand, a generic solution is to take the free-floating DOF and the contacts into account in the control model and to deal with the underactuation using reaction forces [38], [39]. As discussed previously, this control scheme is based on (25). The dynamic equation of the system can be written as  

\[
\begin{align*}
\dot{x}_b &= J_c \left[ x_b \right] + \frac{1}{\mu} \left( g + \mu + J_c^\top f_c = S^\top \tau \right) & (44)
\end{align*}
\]

where \( A \) is the mass matrix of the complete free-floating system, \( \ddot{q} \) is the acceleration of the robot joints, \( \dot{x}_b \) is the acceleration of the free-floating robot basis, \( f_c \) are the contact forces with associated Jacobian \( J_c \), and \( S = [0_{6 \times 6} \ I_n] \) is the selection matrix that denotes the underactuation of the free floating (\( \tau \) has the same dimension \( n \) as \( q \)). A rigid contact at the point \( c \) means that the acceleration

\[
\begin{align*}
\ddot{x}_c &= J_c \left[ x_b \right] + \frac{1}{\mu} \left( g + \mu + J_c^\top f_c \right) \dot{q} \quad (45)
\end{align*}
\]

is null. Using this in (44), a relation between contact forces and torques is obtained [38]

\[
\begin{align*}
f_c &= A_c \left( J_c A^{-1} \tr S \tau - J_c J_c^\top (g + \mu) + J_c \left[ x_b \right] \right) \quad (46)
\end{align*}
\]

By introducing this last equation in (44), the dynamics of the free-floating robot constrained by a rigid contact [38] are given as

\[
\begin{align*}
A \left[ \ddot{x}_c \right] + N_c^\top (g + \mu) + J_c^\top A_c J_c \left[ \dot{x}_b \right] \dot{q} &= N_c^\top S^\top \tau 
\end{align*}
\]

where \( N_c = I - J_c J_c^\top A_c A^{-1} \) is the null space of the contact \( J_c \). This last form is similar to (25) and can thus be used similarly to compute a control law equivalent to (42)

\[
\begin{align*}
\tau_i = \tau_{i-1} + (J_i A^{-1} (SN_c)^\top)^{N_i-1 \oplus \#} W 
\end{align*}
\]

for any \( i = 1, \ldots, k \), where \( W = (SN_c A^{-1} (SN_c)^\top)^{-1} \), \( N_i-1 \) \ is the projection operator due to the priority tasks and defined afterward, and \( N_i^1 - W^{-1/2} N_i^1 \oplus / W \). As discussed previously, the recurrence is initialized with \( \tau_0 = 0 \), and the control law to be applied on the robot is \( \tau = \tau_k \). The projection operator is computed through a second recurrence

\[
\begin{align*}
N_i^1 = N_i^1 - (J_i A^{-1} (SN_c)^\top)^{N_i-1 \oplus \#} W 
\end{align*}
\]

for any \( i = 1, \ldots, k \), where \( W = (SN_c A^{-1} (SN_c)^\top)^{-1} \), \( N_i-1 \) \ is the projection operator due to the priority tasks and defined afterward, and \( N_i^1 - W^{-1/2} N_i^1 \oplus / W \). As discussed previously, the recurrence is initialized with \( \tau_0 = 0 \), and the control law to be applied on the robot is \( \tau = \tau_k \). The projection operator is computed through a second recurrence

\[
\begin{align*}
N_i^1 = N_i^1 - (J_i A^{-1} (SN_c)^\top)^{N_i-1 \oplus \#} W 
\end{align*}
\]

The final control law is very similar to the classical controllers [(46) can be compared, for example, with (38, eq. (B.11))]. It is equivalent to the classical one when no constraint is inside the activation buffer and smoothly evolves to a different controller when a constraint comes closer to the activation point.

F. Conclusion

The control law (42) generalizes the use of a unilateral constraint for an arbitrary set of tasks in the operational space. The final form is very close to the classical control law (30): Very simple computations show that the control law (42) is equal to (30) when the singular values of \( N_i^1 \) are all 0 or 1. This equation is, moreover, easy to extend to (46) for the free-floating robot using the framework introduced in [38] and [39]. The use of the continuous inverse is not very invasive with respect to the classical controllers. It is a close-form algorithm, where only the main inverse of each stage of the hierarchical set of tasks has to be replaced by the new continuous inverse.

Now that all the computations have been realized for any set of tasks, the theory is exemplified with a specific set of tasks through several experiments.

VI. EXAMPLE CASE STUDY: 2-DOF SHOULDER + ELBOW ROBOT WITH ELBOW JOINT LIMIT

We here consider a simple toy example: A 2-DOF planar robot that has to reach a desired position with its end-effector while preserving the joint limit of the second joint. This simple example takes its roots from the case presented in Fig. 1.

In the following, we consider one unilateral constraint (the joint limits) and one task (the end-effector position). The end-effector positioning task is simply

\[
\begin{align*}
e_r^* = -\lambda_r (x - x^*) 
\end{align*}
\]

where \( x \) and \( x^* \) are the current and desired end-effector positions. Theoretically, the gain \( \lambda_r \) has to be positive only. However, if considering a discreet controller, care has to be taken to ensure that the gain is restricted with respect to the sample time interval. The unilateral constraint is classically written as

\[
\begin{align*}
q > -1 \quad \text{and} \quad q < 1
\end{align*}
\]

with \( q \) being the joint position normalized in \([-1, 1]\). We associate with this constraint a vector field that brings the robot in the middle of the joint space

\[
\begin{align*}
e_l^* = -\lambda_l q.
\end{align*}
\]

The Jacobians of \( e_r \) and \( e_l \) are, respectively, denoted as \( J_r \) and \( J_l \). The task \( e_l \) should not be taken into account at any time: We only want a joint limit to be considered when it is going to be violated. A simple solution is to activate the task \( e_l \) only when the robot is reaching the joint limits and to deactivate it when the robot is inside the acceptable joint interval. We define the abrupt activation matrix \( H_l^b \) by

\[
\begin{align*}
H_l^b = \text{diag} \left\{ 0, \quad \text{if} \quad q_l \in [-1, 1] \right\}, \quad \text{otherwise.} \quad (51)
\end{align*}
\]

The exponent makes explicit that the binary diagonal matrix \( H_l^b \) is not continuous. As defined in (3), the equivalent task is \( e_q = H_l^b e_l \).
The transition is smooth everywhere (i.e., has continuous derivatives of all orders), including at the joining points 0 and $\beta$. It is plotted in Fig. 2.

By using this smooth activation function, we would like to have the control law taking progressively into account the task $e_q$ while entering the activation buffer. Simultaneously, we would like to clamp progressively the motion of the robot toward the joint limit. However, it is possible to prove that the control law is equivalent whether $H_{jl}$ or $H_{jl}^b$ is used. At some point, the nonzero values of $H_{jl}$ inside and outside of the inverse $(H_{jl}J_{jl})^+H_{jl}$ get simplified.

If replaying the same experiment of Fig. 3 with the smooth matrix $H_{jl}$, the results are the same: A discontinuity occurs as if there was no damping in the activation matrix. The graph is the same and, hence, is not given here. An exhaustive experimental comparison of the results with or without the smooth activation is provided in [27].

**B. Clamping + Repulsive Potential Field**

This discontinuity can destabilize the control loop by introducing oscillations. Consider the control law (52), with the smooth activation function $H_{jl}$ and a nonzero value of $\lambda_{jl}$. The effect is to clamp the motion toward the joint limit, and also to push the robot away from the limit, following the vector field $e_j$. We replay the simulation of Fig. 3 and with the same initial and desired positions for the new parameters. The results are given in Fig. 4.

The oscillations come from the discontinuities at the activation border: Before the border, the robot is pushed toward the limits by the positioning task. After the border, it is pushed back away from the limit by the repulsive vector field. This oscillation clearly appears in Fig. 4. The robot is thus stuck at the border of the activation buffer by a nondesirable oscillation.

**C. Continuous Control Law**

The constraint is defined by the association $(e_j, H_{jl})$, where $e_{jl}$ is the vector field that pushes the robot inside the joint free space and $H_{jl}$ is our continuous activation matrix evolving continuously from 0 (constraint is inactive) to 1 (constraint is active) in the activation buffer. As given in the previous two paragraphs, the joint limit constraint has priority. The secondary task to be realized under the joint limit constraint is the positioning task $e_r$. For this constraint and this task, the control law (20) can be written as

$$\dot{q} = -\lambda_{jl}J_{jl}^+H_{jl}e_{jl} + J_r^+P_{11}^+(\lambda_r e_r + J_2\lambda_{jl}J_{jl}^+H_{jl}e_{jl}).$$

(57)

In the case where only one joint limit is active ($e_{jl}$ = 0 and $e_{j2} > 0$) and considering first that $\lambda_{jl} = 0$ (that is to say, we only damp the motion of the robot toward the limits without imposing any repulsive force to actively push the robot away from the limit), this last control law becomes

$$\dot{q} = -\lambda_r (1 - h_2)J_{rl}^+e_r - \lambda_r h_2J_{rl}^+e_r$$

where $J_{rl}^+$ is the Jacobian $J_r$ where all but the first column are nullified. In this simple case, the control law is simply a
Fig. 3. Typical example of discontinuity of the control law when the rank of $HJ$ changes. (Top) Position of the arm during the servo. (Bottom) Control law. The goal position is out of reach due to the joint limit. The terminal position is the closest reachable point. When the joint limit is reached, the velocity of the second joint becomes 0, while the first joint velocity changes sign. The control law discontinuity appears as an irregularity in the end-effector trajectory.

Fig. 4. Typical example of oscillations due to the discontinuity when the rank of $HJ$ changes. (Top) Control law. When the activation limit is reached (at time 0.9 s), the control law starts to oscillate between the repulsive vector field $e_{jl}$ and the positioning task $e_r$. (Middle) Joint position. The robot is stuck close to the activation border. (Bottom) Detail of the joint position. The robot is stuck due to oscillations around the activation border.

Fig. 5. Typical use of the continuous inverse. (Top) Position of the arm during the servo. (Bottom) Control law. The parameters of the simulation are the same as in Fig. 3. The control law is given by (58). As in Fig. 3, the final position is the closest reachable point. However, due to the use of the continuous inverse, the trajectory of the robot is smooth, and the control law is continuous.

Fig. 6. Typical example of the continuity of the control law when the continuous inverse is used, with a nonzero gain $\lambda_{jl}$. (Top) Control law. Control law is always continuous. No oscillation can be noted. (Middle) Joint position. The robot converges to a position close to the joint limit. (Bottom) Detail of the joint position. The joint limit is not perfectly reached due to the repulsive vector field $e_{jl}$.

Fig. 7. Typical example of homotopy from the control law that only considers the positioning task $e_r$ to the control law that realizes $e_r$ without using the second joint. The homotopy is driven by the parameter $h_2$. An execution using this control law is given in Fig. 5. The execution parameters and gains are the same as discussed previously. Using the proposed control law, the execution is smooth. No discontinuity appears in the control law. The overall trajectory of the robot is very similar to the discontinuous case, with the main difference being the smoothness of the end-effector trajectory at the point where the activation buffer is reached.

D. Continuous Control Law + Repulsive Potential Field

Finally, we can show that the continuity of the previous control law prevents any oscillation to appear when the gain $\lambda_{jl}$
is not null. The control law (57) is used, with $\lambda_{jl} = 0.1$. The parameters of the simulation are the same as discussed previously. The results of the execution are given in Figs. 6 and 7. In Fig. 6, the control law is continuous, and no oscillation can be noted. The robot finally converges close to the joint limit. Due to the repulsive vector field $e_{jl}$, the final position is a tradeoff between the two tasks $e_{jl}$ and $e_r$. The difference with the previous execution is emphasized in Fig. 7. In the previous execution (see Fig. 5), the final position was the closest reachable point to the goal, since $\lambda_{jl}$ was null. The joint limit was finally reached. In the present case, the robot does not reach the joint limit, since the repulsive vector field has priority over the positioning task.

E. Conclusion

This simple example highlights the problems of the classical pseudoinverse-based control laws. It also detailed the continuous-inverse-based control law in the simple case of one joint limit constraint plus one 2-D positioning task. We show that the control law is equivalent to a simple homotopy between two pseudoinverse-based control laws.

VII. EXPERIMENTS AND RESULTS

Two sets of experiments are presented here: we first quickly demonstrate results with manipulator robots, both in simulation and with a real robot. These experiments have been presented in detail in [41]. We only recall the main results. The second experiment validates the control law for a larger number of tasks on a humanoid avatar.

A. Definition of the Tasks and Constraints

1) Joint Limits Avoidance: As discussed previously, $q$ is the normalized joint position of the robot. The normalized upper and lower joint limits are thus 1 and $-1$. We note that $e_{jl} = q$ is the task to be considered. Given $n$ joints, the joint limits can be expressed by $2 \times n$ unilateral constraints

$$\forall i = 1, \ldots, n, \quad e_{jl} > 1, \quad e_{jl} < 1. \quad (59)$$

The corresponding unilateral task can be written as $(e_{jl}, H_{jl})$, with $H_{jl}$ being the activation matrix defined in (54).

2) Visual Servoing: The two manipulator robots embed a camera mounted on their end-effector. The main task to be realized under the joint limit constraint is thus a visual servoing task [3], [4]. The task is defined directly as a difference between the value of some features measured in the image and their respective values computed at the desired position. For the experiments, we have used visual features derived from the image moments [42]: The robot has to move with respect to a given target in order to center the target in the field of view and to correct the apparent size of the target in the image. The task is thus of dimension 3, which is denoted as $e_v = (X_g, Y_g, Z)$. See [41] for the explicit definition of the task.

3) Other Tasks: On the humanoid avatar, the main task $e_r$ to be realized under joint limit constraint is simply a positioning task (3 DOF) of the end-effector of the robot at a desired position. To maintain balance, we also consider a task $e_{com}$ that regulates the projection of the center of mass (CoM) at the center of the support polygon. Finally, three tasks are used to regulate the robot posture: $e_{chest}$ locks the chest of the robot by imposing a zero velocity on this joint, $e_{hip}$ regulates the rotation of the hip body to a reference posture, and $e_z$ regulates the altitude of the
CoM to its initial position. Finally, an additional friction task has to be added if all the DOFs of the robot are not constrained in order to ensure the stability of the control law. The friction task is simply defined as $e_f = -\lambda f \dot{q}$.

B. Experiments With Manipulator Robots

1) In Simulation: We first consider a 7-DOF PA-10 robot in simulation. We consider the following hierarchy of task $[e_{j1}, e_v, e_f]$, where the joint limits avoidance is ensured by putting the corresponding task at the higher priority position. An overview of the execution is given in Fig. 8. The target is positioned so that the required motion is not feasible inside the joint limits. The robot reaches equilibrium, which is the closest point to the target inside the joint space. The joint limits are not violated while the robot stops as close as possible to the desired position. We then move the target inside the joint limit boundary. The robot moves freely away from its limits to reach the desired position. The details of the execution along with several other simulation with the PA-10 are given in [41].

2) With the Real Puma Robot: As in simulation, a camera is embedded on the effector. The same hierarchy of tasks is used. The experiment is summarized in Fig. 9. The robot regulates the main task while preserving the joint limits. During the execution, an external force is applied on the third joint to disturb the execution. Since only three DOFs of task $e_v$ are regulated, the robot moves freely to follow the external force until it is eventually blocked by the joint limit constraint $e_{j1}$. Additional details about the experiments are given in [41].

C. Experiment on the Humanoid Avatar

The last experiment has been realized in simulation with a 31-DOF humanoid avatar. In this experiment, we assess the control law for a much larger number of tasks due to the large number of available DOFs. Basically, the objective of the control is to reach an object with the hand while maintaining the balance. However, we require the robot to use only the DOF of the arm to reach the object, as long as it is possible. The DOFs of the chest and the legs are to be used only when the object is out of the arm limits. The main objective is to force the robot to use first its lightest joints before using the chest or even the legs [14]. Motion planning may be used to design such an action while considering constraints [43]–[45]. However, planning approaches still suffer from the lack of dynamics, rely on an execution layer for reactivity, and are sensor-based and robust, which is the topic of this paper; basically, our research can be coupled with planning techniques as a steering method for instance.

The set of tasks is thus composed of seven tasks in the following order of priority: 1) CoM ($e_{com}$); 2) joint limits ($e_{j1}$); 3) hip altitude ($e_h$); 4) right-arm hand position ($e_r$); 5) hip orientation ($e_{hip}$); 6) chest orientation ($e_{chest}$); and 7) friction for stability ($e_r$). The control frequency is 1 kHz. The activation buffer width is set to $\beta = 0.1$. The value of the damping parameter has been set to $\eta = 10^{-2}$. The gains of the tasks are $\lambda_{com} = 250$, $\lambda_{j1} = 50$, $\lambda_r = 100$, $\lambda_{hip} = \lambda_{chest} = 120$, and $\lambda_f = 50$.

A typical execution is summed up by Figs. 10–14. The global behavior is shown in Fig. 10. As shown in Fig. 11, the robot arm moves first toward the goal position. The shoulder joint is doing the main part of the job. It soon reaches its limits at time 1. The other joints of the arm are then used to overcome the blocking of the shoulder. However, they are less convenient for completing...
We then changed the goal position arbitrarily to a further (and unreachable) desired positions during the execution. The goal is reached at (4).

Fig. 14. Experiment in Section VII-C. Hand current (continuous line) and desired (slash line) positions during the execution. The hand position converges to the closest reachable point to the goal. We then moved arbitrarily the goal position to an unreachable position. At instant 4, the task \( e_t \) is then completed as much as possible, while maintaining \( e_{com} \) and \( e_Z \) and respecting the condition \( e_H \). The robot finally stops at the reachable position that is closest to the required goal.

The last experiment shows that it is possible to define a behavior by composition of several tasks that can be in conflict at some specific position, depending on the robot configuration. The unilateral constraints acting like a varying-dimension task, the behavior is equivalent to a sequence of different tasks that are selected automatically, depending on the active constraints.

VIII. CONCLUSION

In this paper, we proposed an original scheme to compute a generic control law from a hierarchic set of both unilateral constraints and bilateral tasks. This scheme ensures that the unilateral constraints will not be violated whatever the actions of the tasks. The proposed scheme is generic and could be applied for various sets of tasks, constraints, and robots. We demonstrate its validity theoretically and by applying it on different types of robots with a common visual servoing scheme while satisfying the joint limit constraint. We have shown that it also applies to define the behavior of complex robots, such as a humanoid. Future work will focus on the optimization of the numerical computation algorithm and on the application to the real humanoid robot.

APPENDIX

PROOFS OF THE THEOREMS OF SECTION IV

The proofs are self-content, but for compactness, only the mathematical formulations are given. The intuition behind is given in Section IV. Before proving the two main theorems, we need two intermediate results in the two first sections.

A. Explicit Formulation of the Continuous Inverse

We first here prove that the continuous inverse \( J^{+H} \) can be written as a weighted sum of all the partial pseudo-inverses of \( J \), and we compute the explicit coefficients of the sum.

Lemma 1: For a given integer \( n \), let us define the following matrix index by the subparts of the integers \( n \):

\[
Y^n_P = J^+_P - \sum_{Q \subseteq P} Y^n_Q
\]  

where \( J^+_P \) is a shortcut for \( J^+_P \cup \{n\} \) and \( Y^n_Q = J^n_Q \). Then

\[
Y^n_P = X^n_P + X_P.
\]  

Remark 1: The definition (60) is similar to (10): \( Y^n_P \) can be seen as the coupling between the feature \( n \) and all the other features of \( P \). The lemma simply says that this coupling equals the coupling of \( P \) plus the coupling of \( P \cup \{n\} \).

The introduction of \( Y^n_P \) is necessary for the proof of the following lemma where the continuous inverse is explicitly computed as a sum of partial pseudo-inverses.

Proof: Consider \( P \) as any subset of \( \{1 \cdots n\} \). We consider the following partition of \( P: P = R \cup \{k\} \), with \( k \) being an integer of \( P \). Assuming (61) holds for \( R \) and letting us prove that it...
holds for $\mathcal{P}$, then

$$
Y^n_P = J^n_{Rkn} - \sum_{Q \subseteq R} Y^n_Q - \sum_{Q \subseteq R} Y^n_{Qk} - Y^n_R
$$

$$
= J^n_{Rkn} - \sum_{Q \subseteq R} (X_{Qn} + X_Q) - \sum_{Q \subseteq R} (X_{Qkn} + X_{Qk}) - X_{Rn} - X_R.
$$

By construction of $X$, we have

$$
X_{Rkn} = \left[ J^n_{Rkn} - \sum_{Q \subseteq R} (X_{Qn} + X_Q) - X_{Rn} \right]
$$

which finally proves that (61) is true for $\mathcal{P} = \mathcal{R} \cup \{k\}$. Equation (61) holds for $\mathcal{P} = \emptyset$. By recurrence, (61) holds for any subset $\mathcal{P}$. □

**Lemma 2 (Explicit form of the continuous inverse $J^e_H$):** The continuous inverse (9) is equal to

$$
J^e_H = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \notin P} (1 - h_i) J^n_P.
$$

**Remark 2:** Intuitively, the continuous inverse (9) was only built as a sum of partial pseudoinverses. The lemma gives the explicit formulation of the coefficients corresponding to each partial pseudoinverse in the sum.

**Proof:** We denote by $J^e_H$ the continuous inverse where only the $m$ first components are taken into account, and $G_m = \sum_{P \in \mathcal{P}(m)} (\prod_{i \in P} h_i) Y^p_P$. The proof can be obtained by recurrence to $m$. We make the two following assumptions:

$$
\mathcal{H}_m : J^e_H = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \notin P} (1 - h_i) J^n_P.
$$

$$
\mathcal{H}'_m : G_m = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \notin P} (1 - h_i) J^n_{Pm+1}.
$$

Suppose the two hypotheses are true at rank $m$. The second part of (64) is denoted as $M_m$

$$
M_{m+1} = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \notin P} (1 - h_i) J^n_P + \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) h_{m+1} \prod_{i \notin P} (1 - h_i) J^n_{Pm+1}.
$$

Using (61), we can write for any $\mathcal{P}$ that $(1 - h_{m+1}) X_P + h_{m+1} Y^m_P = X_P - h_{m+1} X_P + h_{m+1} X_P + h_{m+1} X_{Pm+1}$. Simplifying the second term, and reordering the sum, we finally obtain

$$
M_{m+1} = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) X_P
$$

$$
+ \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) h_{m+1} X_{Pm+1}
$$

which finally gives $M_{m+1} = J^e_H$. The dual hypothesis $\mathcal{H}'_m$ is similarly obtained. Both $\mathcal{H}_0$ and $\mathcal{H}'_0$ are also true. By recurrence, this last result is finally extended to any case, which proves the lemma.

### B. Sum of the Coefficients $h_i$

The coefficients of the continuous inverse explicit formulation (64) have a very noteworthy property: When they are summed by themselves (i.e., if removing the term $J_P$ from the sum), they can be reduced to a very simple expression. More interesting, this sum equals 1 as soon as one activation parameter $h_i$ equals 1. This is what we are going to prove here.

The sum of all the scalar coefficients of (64) is denoted as $S_m$ and is defined by

$$
S_m = \sum_{P \in \mathcal{P}_m, P \neq \emptyset} \left( \prod_{i \in P} h_i \right) \left( \prod_{i \notin P} (1 - h_i) \right).
$$

The following three lemmas are necessary to compute the explicit formulation of $S_m$ and prove that $S_m$ simply equals 1 when one of $h_i$ equals 1.

**Lemma 3:** The sum $S$ can be computed by recurrence from

$$
S_{m+1} = S_m + h_{m+1} D_m
$$

where $D_m = \prod_{i=1}^m (1 - h_i) = (1 - h_m) D_{m-1}$, and $S_0 = 0$.

**Proof:** The sum (69) is separated in two parts according to whether $m + 1$ is part of $\mathcal{P}$ or not:

$$
S_{m+1} = \sum_{P \in \mathcal{P}_m, m + 1 \notin \mathcal{P}} \left( \prod_{i \in P} h_i \right) \left( \prod_{i \notin P} (1 - h_i) \right)
$$

$$
+ \sum_{P \in \mathcal{P}_m, m + 1 \in \mathcal{P}} \left( \prod_{i \in P} h_i \right) \left( \prod_{i \notin P} (1 - h_i) \right).
$$

Taking the terms $h_{m+1}$ out of the sums, we obtain

$$
S_{m+1} = h_{m+1} \sum_{P \in \mathcal{P}_m, P \neq \emptyset} \left( \prod_{i \in P} h_i \right) \left( \prod_{i \notin P} (1 - h_i) \right)
$$

$$
+ (1 - h_{m+1}) S_m + h_{m+1} \prod_{i=1}^m (1 - h_i).
$$
We recognize $S_m$ in the second term and $D_m$ in the last term, which finally proves the lemma.

**Lemma 4**: For all $m$, the sum $S_m + D_m$ equals 1.

**Proof**: From Lemma 3, we can write $S_m + D_m = S_{m-1} + h_m D_{m-1} + D_m$. By construction, $D_m = (1 - h_m) D_{m-1}$. Then, $S_m + D_m = S_{m-1} + D_{m-1} = \cdots = S_0 + D_0 = 1$.

**Lemma 5 (Sum of the coefficients $h_k$)**: If one of the coefficient $h_i$ equals 1, then the entire sum $S_m$ equals 1.

**Proof (Theorem A.5)**: Suppose that there is an index $k$ so that $h_k = 1$. Then, Lemma 4 gives $S_k = S_{k-1} + D_{k-1} = 1$. Moreover, since $h_k = 1$, for all $j > k$, $D_j = 0$, and $S_j = S_{j-1} = \cdots = S_k = 1$, which proves the theorem.

From the two main results of these last two sections, the theorems of Section IV can easily be proven.

### C. Proof of Theorem 4.1

**Proof (Theorem 4.1)**: By Lemma 2, the continuous inverse can be written as a sum of partial inverses. The product $J_{P_0}^-$ can thus be written as

$$J_{P_0}^- = J - \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \in P} (1 - h_i) J J_{P_0}^+ J.$$  \hfill (73)

Suppose that $h_k = 1$. From Lemma 5, the sum of $h_i$ in the second term of the sum is equal to 1; the Jacobian $J$ can thus be factorized in the sum

$$J_{P_0}^- = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} h_i \right) \prod_{i \in P} (1 - h_i) (J - J J_{P_0}^+ J).$$  \hfill (74)

Since $h_k = 1$, all the terms of the sums where $k \notin P$ are null. On the other hand, if $k \in P$, then the $k$th lines of $J$ and $J J_{P_0}^+ J$ are equal (by fundamental property of the pseudoinverse $J_{P_0}^+$). The $k$th lines of $J_{P_0}^-$ are thus null.

### D. Proof of Theorem 4.2

**Proof (Theorem 4.2)**: The eigenvalue decomposition of $P_0^+$ is denoted as $(V, S = \text{diag}(\sigma))$. By construction of the continuous inverse, we can write

$$J_{2} P_{1}^+ = V (J_{2} V)^{S_{1}}.$$  \hfill (75)

From Theorem 1.2, this last equality can be developed as

$$J_{2} P_{1}^+ = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} \sigma_i \right) \prod_{i \in P} (1 - \sigma_i) V (J_{2} S_P)^+$$  \hfill (76)

where $S_P$ is the diagonal matrix whose coefficients are equal to 1 if their indexes are in $P$ and 0 otherwise. Multiplying the previous equation by $J_1$ gives

$$J_1 J_2 P_{1}^+ = \sum_{P \in \mathcal{P}(m)} \left( \prod_{i \in P} \sigma_i \right) \prod_{i \in P} (1 - \sigma_i) J_1 V (J_2 V)^{S}.$$  \hfill (77)

From Theorem 4.1, we know that $J_1 P_{1}^+ = J_1 V S V^T$ is null on the lines $k$ whose activation is $h_k = 1$. Since $V$ is full rank, then for any $x$ in $\text{Span}(S)$, $J_1 V x$ is null on line $k$.

Consider a subpart $P$ [i.e., an index of the sum (77)]. Then, the corresponding term of the sum (77) is null if there is a $k$ so that $\sigma_k = 0$ and $k \in P$. Otherwise, since all zero components of $S$ are also zero components of $S_P$, then $\text{Span}(J_1 V (J_2 S_P)^+)$ is included in $\text{Span}(S)$. Therefore, all nonzero terms $J_1 V (J_2 S_P)^+$ of $J_1 J_2 P_{1}^+$ are null on the line $k$, which proves the theorem.

### ACKNOWLEDGMENT

The authors would like to particularly thank J. Park, L. Sentis, and P. Evrard for valuable comments and help for both the theoretical and experimental parts of this paper.

### REFERENCES


Nicolas Mansard graduated from the Ecole Nationale Superieure d’Informatique et de Mathematiques Appliquees, Grenoble, France, in 2003 and received the M.S. degree in robotics and image processing from the University Joseph Fourier, Grenoble, in 2003 and the Ph.D. degree in computer science from the University of Rennes, Rennes, France, in 2006.

He was with the Lagadic Group, Institut National de Recherche en Informatique et Automatique, Bretagne, France, for three years. He was with Stanford University, Stanford, CA, for one year. He was also with the Joint Robotics Laboratory, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan, for one year. He is currently with Gepetto Group, Laboratoire d’Architecture et d’Analyse des Systmes/Centre National de la Recherche Scientifique, Toulouse, France. His current research interests include sensor-based robot animation and, more specifically, the integration of reactive control schemes into real robot applications.

Dr. Mansard received, from the French Research Group Modeling Analysis and Conduct of Systems (MACS) and French Society for Telecommunication and Computer Science, an award for his Ph.D. thesis. He also received the prize for the best thesis of the Région Bretagne.


He is currently a Professor of computer science with Stanford University, Stanford, CA. His current research interests include human-centered robotics, human motion synthesis, humanoid robotics, dynamic simulation, haptic interfaces, and human-friendly robot design. He has authored or coauthored more than 200 publications in the field. He was the Editor of the Robotics Review. He is the Editor of the *Springers Tracts in Advanced Robotics*.

Prof. Khatib is a Distinguished Lecturer of the IEEE. He was the recipient of the Japan Robot Association Award. He is currently the President of the International Foundation of Robotics Research. He was the Program Chair of the 2000 International Conference on Robotics and Automation. He was the Director of the Stanford Computer Forum.

Abderrahmane Kheddar (M’04) received the B.S. degree in computer science from the Institut National d’Informatique Algiers, Algeria, and the D.E.A. (Master by Research) and the Ph.D. degree in robotics from the University of Paris 6, Paris, France.

From September 2003 to August 2008, he was a Professor with the University of Évry, Évry, France. He is currently the Director of Research with the Centre National de la Recherche Scientifique (CNRS), where he has been the Director of the CNRS National Institute of Advanced Industrial Science and Technology Joint Robotics Laboratory, UMI3218/ Collaborative Research Team, Tsukuba, Japan, since December 2008. He is currently a member of the Editorial Board of the *Journal of Intelligent and Robotic Systems*. His current research interests include haptics and humanoids.

Dr. Kheddar is the founding member and a Senior Adviser of the IEEE/Robotics and Automation Society Chapter on haptics. He is currently a member of the Editorial Board and a founding member of the IEEE TRANSACTIONS ON HAPTICS.