2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

A Reverse Priority Approach to Multi-Task Control of Redundant Robots

Fabrizio Flacco Alessandro De Luca

Abstract— **A novel method to handle multiple robotic tasks with priorities is presented. The occurrence of singularities, both of the kinematic and algorithmic type, may affect the correct hierarchy in task execution. Existing methods deal with singularities either by using damped least squares solutions or by relaxing the enforcement of secondary tasks. Damped pseudo-inversion mitigates undesired effects near singularities, at the cost of non-negligible task errors and deformation even of the highest priority task. When secondary tasks are not enforced, hierarchy is preserved but these tasks are not executed accurately even when this would be possible. In our approach, joint motion contributions are added following the reverse order of task priorities and working with suitable projection operators. Higher priority tasks are processed at the end, avoiding possible deformations caused by singularities occurring in lower priority tasks. The proposed Reverse Priority (RP) method allows executing at best all tasks while still preserving the desired hierarchy. The effectiveness of the RP method is shown through numerical simulations and with experiments on a 7-dof KUKA LWR.**

I. Introduction

Local inversion of differential kinematics (IK) is the most common approach to control a redundant robot [1]. The main advantages are the simplicity of task design, the possibility of a sensor-based execution, and the easy integration of multiple tasks with priorities [2]. Moreover, the local framework is suitable for controlling robots at the velocity [2], acceleration [3], or torque [4] level.

Prioritized IK solvers are intended to guarantee a hierarchy in the execution of multiple ordered tasks: higher priority tasks will always be executed at best, without being affected by the presence of lower priority tasks [5]. However, when singularities are encountered or when the computed commands will exceed the existing bounds on robot motion, the order of priority will not be preserved in general. In such cases, it may occur that a feasible task, even the highest priority one, is not executed correctly because of the way numerical issues and motion limitations are handled in lower priority tasks.

Recently, a number of methods have been proposed to deal with bounded robot capabilities (limited joint ranges, maximum joint velocities, and so on). In [6], constrained kinematic inversion was modeled as a quadratic programming problem, and then extended so as to cover also a hierarchy of prioritized tasks. Other successful solutions include the Stack of Tasks (SoT) method [7] and the method of Instantaneous Task Specification using Constraints (iTaSC) [8]. Also, we proposed the Saturation in the Null Space (SNS) method [9], [10] that solves IK problems under hard joint constraints with reduced computational costs, by exploiting the particular structure of the optimization problem.

Most of these solutions, however, suffer from the presence of possible singularities. Given a generic task, and assuming unlimited joint motion capabilities of the robot, a *kinematic* singularity occurs when a desired task velocity cannot be realized at the current configuration by any joint velocity. Instead, we have an *algorithmic* singularity when the desired task velocity cannot be realized because it is not compatible with another task of higher priority. Indeed, local IK solvers may not have special problems (except for solution discontinuity) right at singularities. Nonetheless, it is well known that high joint velocities are usually requested in order to obtain a relatively small task velocity already in the vicinity of a singular configuration [11]. Moreover, when the robot is very close to a singularity, the IK problem becomes ill-conditioned and large task errors will also be produced.

The simplest and most widely used solution to this problem is based on a damped pseudo-inversion of the (projected) task Jacobian [12]. The damping factor can be chosen according to different techniques [13], but all share a common drawback: larger damping factors needed to limit joint velocity near a singularity will produce larger task errors and have a stronger influence on higher priority tasks. To milden these effects, a singularity-robust IK method was proposed in [14]. In this approach, the execution of a secondary task is not strictly enforced and thus the primary task will not be affected, whether or not the secondary task is singular. On the other hand, the secondary task will never be realized completely, even when it is feasible and linearly independent from the primary task.

In this paper, we introduce an IK solver for handling multiple robotic tasks with priority that is largely unaffected by algorithmic singularities, and that guarantees task execution in the correct priority order also in case of kinematic singularities. This new method, called Reverse Priority (RP), is based on the idea of computing joint motion contributions starting from the task of lowest priority and moving up to the primary task. A special projection matrix is derived, which ensures preservation of the correct priority order. Our main purpose here is to present the main idea of the RP method, give the tools in order to use it properly, and show its effectiveness through simulations and experiments.

The paper is organized as follows. Section II summarizes

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy ({fflacco,deluca}@diag.uniroma1.it). This work is supported by the European Commission, within the FP7 ICT-287513 SAPHARI project (www.saphari.eu).

the relevant background, with a recall of singularity handling (II-A) and of the singularity-robust method of [14] (II-B), which is used later for comparison. The key idea underlying the RP method is presented in Sec. III, with details about computational issues involved in obtaining the needed projection operator (III-A, and about dealing with so-called configuration space tasks (III-B). A statistical comparison with standard damped pseudo-inverse and singularity-robust methods is proposed in Sec. IV. Finally, the effectiveness of the RP method is illustrated in Sec. V with experimental results on a 7-dof KUKA LWR IV performing three tasks with priority.

II. Background

Let $\boldsymbol{q} \in \mathbb{R}^n$ be the generalized (joint) coordinates of a robot with n degrees of freedom, $\boldsymbol{x} \in \mathbb{R}^m$ the variables describing a generic m-dimensional task, with $m \leq n$, and $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{q})$ the associated $m \times n$ task Jacobian matrix. At a given robot configuration $\boldsymbol{q}$, the differential kinematics of the task is

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}. \tag{1}$$

Assuming that $\boldsymbol{J}$ has full rank, $\rho = m$, inversion of the differential map (1) provides infinitely many solutions, all of which can be generated as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + \boldsymbol{P}\dot{\boldsymbol{q}}_{\mathcal{N}} = \dot{\boldsymbol{q}}_{\mathcal{N}} + \boldsymbol{J}^{\#}\left(\dot{\boldsymbol{x}} - \boldsymbol{J}\dot{\boldsymbol{q}}_{\mathcal{N}}\right), \tag{2}$$

where $\boldsymbol{J}^{\#}$ is the Moore-Penrose pseudo-inverse 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 $\dot{\boldsymbol{q}}_{\mathcal{N}} \in \mathbb{R}^n$ is a generic joint velocity. Equation (2) gives the joint velocity $\dot{\boldsymbol{q}}$ that satisfies (1) and is closest to $\dot{\boldsymbol{q}}_{\mathcal{N}}$. The last expression on the right of (2) shows how the solution is actually evaluated. Also, setting $\dot{\boldsymbol{q}}_{\mathcal{N}} = \boldsymbol{0}$ in (2) yields the joint velocity solution of minimum norm.

In general, in order to compute the pseudo-inverse in (2), one needs the Singular Value Decomposition (SVD) of the Jacobian, $\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{U}$ is a $m \times m$ unitary matrix composed by column vectors $\boldsymbol{u}_i$, $\boldsymbol{V}$ is a $n \times n$ unitary matrix composed by column vectors $\boldsymbol{v}_i$, and $\boldsymbol{\Sigma}$ is a $m \times n$ block matrix, with a leading, $m \times m$ diagonal matrix containing the singular values $\sigma_i \geq 0$ ($i = 1, \ldots, m$) of $\boldsymbol{J}$ in decreasing order ($\sigma_i \geq \sigma_j$ for $i < j$), followed by $n - m$ zero columns. It is

$$\boldsymbol{J}^{\#} = \boldsymbol{V}\boldsymbol{\Sigma}^{\#}\boldsymbol{U}^T = \sum_{i=1}^{\rho} \frac{1}{\sigma_i}\boldsymbol{v}_i\boldsymbol{u}_i^T, \tag{3}$$

where $\rho \leq m$ is the rank of $\boldsymbol{J}$.

The use of redundancy can be extended to the satisfaction of multiple simultaneous tasks in the form (1), $\dot{\boldsymbol{x}}_k = \boldsymbol{J}_k\dot{\boldsymbol{q}}$, each of dimension m_k, for $k = 1, \ldots, l$ (typically, with $\sum_{k=1}^{l} m_k \leq n$), which are ordered by their priority, i.e., when $i < j$, task i has a higher priority than task j. The execution of a task of lower priority should not interfere with the execution of tasks having higher priorities. This hierarchy is obtained by projecting the task k in the null space of all higher priority tasks (i.e., all the tasks with index $i \in \{1, \ldots, k-1\}$). An efficient solution is obtained by using the recursive formula [2]

$$\dot{\boldsymbol{q}}_k = \dot{\boldsymbol{q}}_{k-1} + \left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)^{\#}\left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k\dot{\boldsymbol{q}}_{k-1}\right), \tag{4}$$

for $k = 1, \ldots, l$, initialized with $\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$ and $\boldsymbol{P}_{A,0} = \boldsymbol{I}$. Matrix $\boldsymbol{P}_{A,k}$ is the projector in the null space of the augmented Jacobian of the first k tasks

$$\boldsymbol{J}_{A,k} = \begin{pmatrix} \boldsymbol{J}_1^T & \boldsymbol{J}_2^T & \ldots & \boldsymbol{J}_k^T \end{pmatrix}^T. \tag{5}$$

Also the projector in the null space of the augmented task can be computed efficiently, using the recursive expression [16]

$$\boldsymbol{P}_{A,k} = \boldsymbol{P}_{A,k-1} - \left(\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}\right)^{\#}\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}. \tag{6}$$

The multi-task redundancy resolution approach given by eqs. (4) and (6) is the most common in the literature, and therefore we will refer to it as the *standard* method.

A. Singularities

A generic task is labeled as *regular* when the associated Jacobian matrix has full row rank. Conversely, a *kinematic* singularity occurs when the task Jacobian loses rank. In such robot configurations, there is a reduced capability of satisfying a generic task velocity, since motion along at least one direction of the task space is no longer feasible. Moreover, near the singularity the task Jacobian becomes ill conditioned, its pseudo-inverse will contain very high values, and motion along almost unfeasible directions will require large joint velocities. This numerical instability is a typical source of discontinuities in the solution.

In case of multiple tasks being executed with priorities, matrix $\boldsymbol{J}_k\boldsymbol{P}_{A,k-1}$ may become ill conditioned as well, even if the single tasks are regular and each of the associated task Jacobians is well conditioned. Since in this case large joint velocities are generated by eq. (4), the situation is called by analogy an *algorithmic* singularity and occurs when the k-th task is in conflict with (namely, linearly dependent on) a task with higher priority.

In both cases, referring to the SVD used for computing the needed pseudo-inverse, the large generated joint velocities are due to the smallest singular value(s) rapidly approaching zero (and thus $\frac{1}{\sigma_i} \to \infty$, e.g., in eq. (3)). This drawback is mitigated by the so-called damped pseudo-inversion [1], based on the approximation

$$\frac{1}{\sigma_i} \approx \frac{\sigma_i}{\sigma_i^2 + \lambda^2}. \tag{7}$$

The higher is the factor λ, the more damped is the joint velocity near singularities, but the larger will be the error in the execution of the task, so that a modulation of the damping factor near singularities becomes a must. There are different methods for defining a variable damping factor [13]. Here, we shall use

$$\lambda^2 = \begin{cases} 0 & \text{when } \sigma_m \geq \epsilon \\ \left(1 - (\sigma_m/\epsilon)^2\right)\lambda_{max}^2 & \text{otherwise.} \end{cases} \tag{8}$$

The parameter $\epsilon > 0$ monitors the smallest singular value σ_m defining the range of the damping action, while $\lambda_{max} > 0$ is the largest damping factor used near singularities.

Let $\boldsymbol{J}^{\#_d}$ be the damped pseudo-inverse of $\boldsymbol{J}$. It should be noted that, when damping is used, it is $\boldsymbol{P} \neq \boldsymbol{I} - \boldsymbol{J}^{\#_d}\boldsymbol{J}$. In this case, the correct projection matrix is computed as

$$\boldsymbol{P} = \boldsymbol{I} - \sum_{i=1} \rho \boldsymbol{v}_i \boldsymbol{v}_i^T. \tag{9}$$

In case of multiple tasks with priority, this property applies also to the projection matrix (6). In the following, we will still refer to the recursive expressions (4) and (6) as the *standard* solution method, even if in practice the damped pseudo-inverse with (7)–(8) has been used and the null-space projection matrix is evaluated via (9).

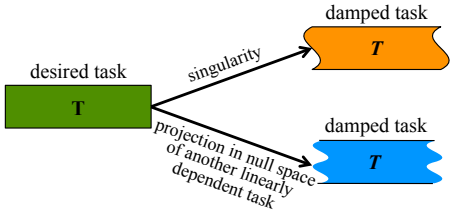

Fig. 1. Pictorial representation of damped solutions for a desired task **T**

To illustrate the effects of using damped pseudo-inversion, consider the representation in Fig. 1 with possible damped solutions associated to a generic task **T** (in green) [1]. If the desired task **T** is in a (kinematic) singularity, the damped solution will deform the execution of the original task, leading to $\boldsymbol{T}$ (in orange). Similarly, if the task **T** is regular but needs to be projected in the null space of other linearly dependent tasks (algorithmic singularity), the damped solution will again deform the feasible part of the original task into $\boldsymbol{T}$ (in light blue). If such contributions were added within a generic step of a recursive task priority solution, the higher priority tasks would be deformed as well. We illustrate this outcome on the three different scenarios in Fig. 2, in which the three tasks **T1**, **T2**, and **T3** have to be executed with the given priority. In the first and second scenarios the tasks are not in conflict, while in the third one they are linearly dependent. The second scenario considers also a singular task, placed at the second priority level (**T2**).

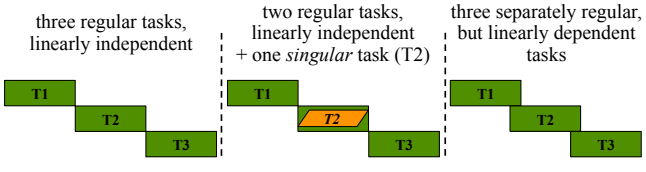

Fig. 2. Three scenarios of three tasks to be executed with priorities

The solutions obtained for the three scenarios using the standard task priority method are illustrated in Fig. 3. The solution for task **T1** is computed first, then the contribution for **T2** is added after projection in the null space of **T1**, and finally the contribution for **T3** is added, as projected in the null space of both **T1** and **T2**. In the first scenario, all three tasks are correctly executed, being separately regular and jointly linear independent. In the second scenario, when the singular **T2** is added, the damped pseudo-inversion of $\boldsymbol{J}_2\boldsymbol{P}_{A,1}$ is needed. A deformed $\boldsymbol{T2}$ is obtained as expected, but since the damping action modifies also $\boldsymbol{P}_{A,1}$, the primary task will be affected too —an undesired behavior: a low priority task modifies the execution of a higher priority task. On the other hand, task **T3** is correctly projected in the null space of higher priority tasks by using eq. (9). Similarly, when the tasks are partially in conflict (third scenario), the damped contribution of **T2** deforms also **T1**, and then the damped contribution of **T3** affects **T2** and again **T1**.

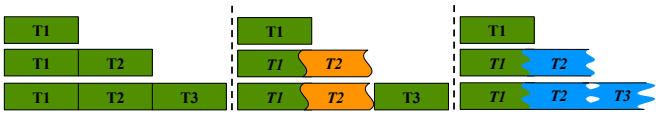

Fig. 3. Solutions for the three scenarios using the standard task priority method in eqs. (4) and (6) —computations proceed from top to bottom

B. Singularity-robust method

A method that overcomes algorithmic singularities was proposed by Chiaverini [14]. The constraints imposed by higher priority tasks, which are recognized as the source of algorithmic singularities, are removed when proceeding to lower priority tasks. In other words, the contribution of a secondary task is not obtained as a projection in the null space of the Jacobian of higher priority tasks, but rather the unconstrained solution to the secondary task is computed separately and then projected in the null space to guarantee the correct priority. The recursive expression of the singularity-robust method is then obtained as

$$\dot{\boldsymbol{q}}_{C,k} = \dot{\boldsymbol{q}}_{C,k-1} + \boldsymbol{P}_{A,k-1}\boldsymbol{J}_k^{\#}\dot{\boldsymbol{x}}_k, \tag{10}$$

for $k = 1, \ldots, l$, using the same null space projector of the standard method and with the same initialization ($\dot{\boldsymbol{q}}_{C,0} = \boldsymbol{0}$, $\boldsymbol{P}_{A,0} = \boldsymbol{I}$).

The drawback of this simpler approach is that all low priority tasks (except the primary task) are barely taken into account. In practice, a low priority task will be deformed even when it is compatible with higher priority tasks. Consider for instance the case of $l = 2$ tasks, where eq. (10) collapses to (2) with $\dot{\boldsymbol{q}}_{\mathcal{N}} = \boldsymbol{J}_2^{\#}\dot{\boldsymbol{x}}_2$. For the situation shown Fig. 4 ($n = 2$), the singularity-robust solution $\dot{\boldsymbol{q}}_{C,2}$ satisfies the primary task and minimizes the distance to the minimum norm solution $\boldsymbol{J}_2^{\#}\dot{x}_2$ of the secondary task. However, the secondary task is not executed even if there is a simultaneous solution $\dot{\boldsymbol{q}}^*$ to both tasks.

Figure 5 shows the solution for the three illustrative scenarios of Fig. 2 using the method of Chiaverini. The primary task is always executed correctly, but the two other tasks are largely deformed.

[1]Beside adopting color codes, tasks correctly executed are written in roman type, while for tasks that are not completely accomplished we use *italics*.

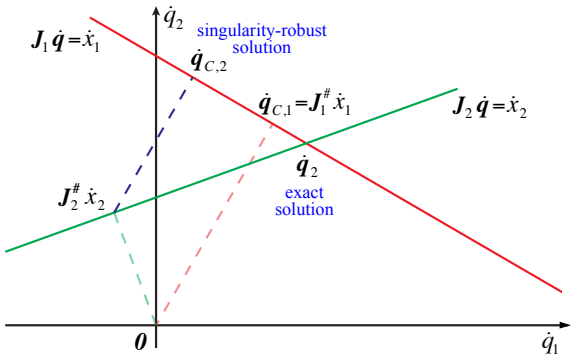

Fig. 4. Construction of the singularity-robust solution for $l = 2$ tasks with priority in a two-dimensional joint velocity space

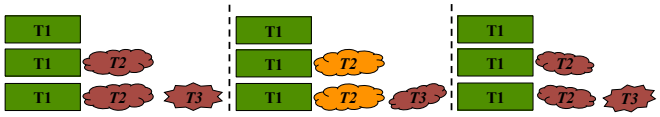

Fig. 5. Solutions for the three scenarios using the singularity-robust method of Chiaverini

III. Reverse Priority Method

Our novel approach is based on the idea of computing first the solution for the lowest priority task, and then recursively adding the contributions of higher priority tasks. Because of this reverse organization of computations, we call the method Reverse Priority (RP). Intuitively speaking, by deferring the evaluation of higher priority tasks to later steps in the recursion, namely after the lower priority tasks have already provided their contributions, will certainly preserve the correct task hierarchy, if things are computed in the proper way, while still allowing the best possible execution of tasks.

Task hierarchy is guaranteed if the higher priority tasks dominate the solution for the common parts of tasks that are in conflict, whereas remaining portions of lower priority tasks that are linearly independent from higher priority tasks will be preserved (not deformed) even after adding the contributions of higher priority tasks.

To obtain such a result, we introduce the reverse priority projection matrix $\widetilde{\boldsymbol{P}}_{k+1}$, which takes into account the null space of those elements of the $l-k-1$ tasks of lowest priority that are linearly independent from the k-th task. Let $\boldsymbol{J}_{i|j}$ be the Jacobian associated to all components of the i-th task that are linearly independent from the j-th task. Introducing the augmented matrix

$$\widetilde{\boldsymbol{J}}_{k+1} = \begin{pmatrix} \boldsymbol{J}_{k+1|k}^T & \boldsymbol{J}_{k+2|k}^T & \dots & \boldsymbol{J}_{l|k}^T \end{pmatrix}^T, \tag{11}$$

we have

$$\widetilde{\boldsymbol{P}}_{k+1} = \boldsymbol{I} - \widetilde{\boldsymbol{J}}_{k+1}^{\#}\widetilde{\boldsymbol{J}}_{k+1}. \tag{12}$$

The RP recursive formula is then

$$\dot{\boldsymbol{q}}_{RP,k} = \dot{\boldsymbol{q}}_{RP,k+1} + \left(\boldsymbol{J}_k\widetilde{\boldsymbol{P}}_{k+1}\right)^{\#}\left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k\dot{\boldsymbol{q}}_{RP,k+1}\right), \tag{13}$$

for $k = l, l-1, \dots, 1$, initialized with $\dot{\boldsymbol{q}}_{RP,l+1} = \boldsymbol{0}$ and $\widetilde{\boldsymbol{P}}_{l+1} = \boldsymbol{I}$. It is easy to verify that the rank of $\boldsymbol{J}_k$ is equal to the rank of $\boldsymbol{J}_k\widetilde{\boldsymbol{P}}_{k+1}$. Therefore, the RP method is *never* affected by algorithmic singularities.

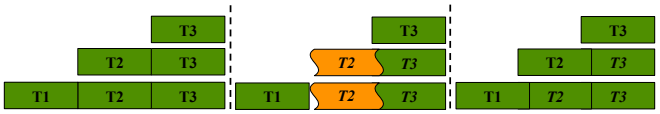

Fig. 6. Solutions for the three scenarios using the proposed Reverse Priority (RP) method

Figure 6 shows the solutions obtained using the RP approach for the three illustrative scenarios of Fig. 2. In the first scenario, all three tasks are correctly executed just as with the standard (forward) method. The singularity of **T2** in the second scenario requests the damped pseudo-inversion of matirx $\boldsymbol{J}_2\widetilde{\boldsymbol{P}}_3$. This will modify the execution of **T3**, but eventually the task **T1** of highest priority will be performed correctly. Therefore, the order of priority is kept by the RP method: a task encountering a kinematic singularity will affect only tasks of lower priority. In the third scenario, the absence of algorithmic singularities in the RP recursion allows avoiding the use of damped solutions. At the end of the procedure, the computed solution will execute the full task **T1**, the portion of **T2** that is linearly independent from **T1**, and the portion of **T3** that is neither in conflict with **T1** nor with **T2**. Again, task hierarchy is completely preserved.

A. Computing the RP projector

Unfortunately, there is no known method for computing the linearly independent Jacobians $\boldsymbol{J}_{i|j}$, except in some trivial or ad hoc cases. Therefore, the projection matrix $\widetilde{\boldsymbol{P}}_{k+1}$, which is the key element of the proposed RP method, cannot be computed in general using directly eqs. (11–12). We propose here an alternative method for computing the RP projector.

To this end, define the reverse augmented Jacobian as

$$\boldsymbol{J}_{RA,k} = \begin{pmatrix} \boldsymbol{J}_k^T & \boldsymbol{J}_{k+1}^T & \dots & \boldsymbol{J}_l^T \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{J}_k \\ \boldsymbol{J}_{RA,k+1} \end{pmatrix}. \tag{14}$$

Denoting by $\bar{\boldsymbol{J}}_{k+1}$ the rows of matrix $\boldsymbol{J}_{RA,k+1}$ that are linearly dependent on $\boldsymbol{J}_k$, this matrix can be rearranged as

$$\boldsymbol{\Pi}_{k+1}\boldsymbol{J}_{RA,k+1} = \begin{pmatrix} \widetilde{\boldsymbol{J}}_{k+1} \\ \bar{\boldsymbol{J}}_{k+1} \end{pmatrix}, \tag{15}$$

for a suitable permutation matrix $\boldsymbol{\Pi}_{k+1}$. The $n \times \sum_{i=k}^{l} m_i$ pseudo-inverse of (14) can be partitioned as

$$\boldsymbol{J}_{RA,k}^{\#} = \begin{pmatrix} \boldsymbol{T}_k & \boldsymbol{H}_k \end{pmatrix}. \tag{16}$$

The $n \times m_k$ matrix $\boldsymbol{T}_k$ is related to the augmentation of $\boldsymbol{J}_{RA,k+1}$ by $\boldsymbol{J}_k$ in (14). From [15, p. 51], we have that

$$\boldsymbol{H}_k = \boldsymbol{J}_{RA,k+1}^{\#} - \boldsymbol{T}_k\boldsymbol{J}_k\boldsymbol{J}_{RA,k+1}^{\#}. \tag{17}$$

Exploiting the structure of the pseudo-inverse of a partitioned matrix [15, p. 19], we obtain also the relation

$$\left(\boldsymbol{I} - \boldsymbol{T}_k\boldsymbol{T}_k^{\#}\right)\boldsymbol{H}_k\boldsymbol{\Pi}_{k+1}^T = \begin{pmatrix} \widetilde{\boldsymbol{J}}_{k+1}^{\#} & \boldsymbol{0} \end{pmatrix}. \tag{18}$$

Finally, the RP projector can be written in the form

$$\begin{aligned}\widetilde{\boldsymbol{P}}_{k+1} &= \boldsymbol{I} - \begin{pmatrix} \widetilde{\boldsymbol{J}}^{\#}_{k+1} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \widetilde{\boldsymbol{J}}_{k+1} \\ \bar{\boldsymbol{J}}_{k+1} \end{pmatrix} \\ &= \boldsymbol{I} - \left(\boldsymbol{I} - \boldsymbol{T}_k \boldsymbol{T}_k^{\#}\right) \boldsymbol{H}_k \boldsymbol{J}_{RA,k+1},\end{aligned} \tag{19}$$

since $\boldsymbol{\Pi}_{k+1}^T \boldsymbol{\Pi}_{k+1} = \boldsymbol{I}$, by definition of permutation matrix. It should be stressed that the actual knowledge about which are the rows in matrix $\boldsymbol{J}_{RA,k+1}$ that are linear dependent and, respectively, independent from $\boldsymbol{J}_k$ is no longer needed.

Using eq. (17) and a bit of math, it is possible to express the reverse projector as

$$\widetilde{\boldsymbol{P}}_{k+1} = \boldsymbol{P}_{AR,k} + \boldsymbol{T}_k \boldsymbol{T}_k^{\#}, \tag{20}$$

where $\boldsymbol{P}_{AR,k}$ is the projection matrix in the null space of $\boldsymbol{J}_{AR,k}$. From (20), it can be proven that

$$\begin{aligned}\left(\boldsymbol{J}_k \widetilde{\boldsymbol{P}}_{k+1}\right)^{\#} &= \left(\boldsymbol{J}_k \boldsymbol{P}_{AR,k} + \boldsymbol{J}_k \boldsymbol{T}_k \boldsymbol{T}_k^{\#}\right)^{\#} \\ &= \left(\boldsymbol{J}_k \boldsymbol{T}_k \boldsymbol{T}_k^{\#}\right)^{\#} = \boldsymbol{T}_k \left(\boldsymbol{J}_k \boldsymbol{T}_k\right)^{\#}.\end{aligned} \tag{21}$$

Finally, substituting (21) in (13) provides the RP recursion

$$\dot{\boldsymbol{q}}_{RP,k} = \dot{\boldsymbol{q}}_{RP,k+1} + \boldsymbol{T}_k \left(\boldsymbol{J}_k \boldsymbol{T}_k\right)^{\#} \left(\dot{\boldsymbol{x}}_k - \boldsymbol{J}_k \dot{\boldsymbol{q}}_{RP,k+1}\right). \tag{22}$$

Note that the only matrix needed is now $\boldsymbol{T}_k$, which can be obtained efficiently as a rank update of $\boldsymbol{J}^{\#}_{RA,k+1}$ (see [15] for more details).

The computational complexity of the proposed RP method is similar to that of the standard task priority method. As a drawback left over, there is still the need to compute $\boldsymbol{T}_k$ using damped pseudo-inversion in case of linearly dependent tasks. However, task deformation will be negligible in practice, as shown by the numerical results reported in Sec. IV.

B. Configuration Space Tasks in the RP Framework

We consider the special class of tasks that request a preferred behavior directly in the Configuration Space (CS). A typical example is when the Projected Gradient method [17] is used to specify the joint velocity according to the gradient direction of a cost function. CS tasks are typically placed at lowest level of priority, $\dot{\boldsymbol{x}}_{l+1} = \dot{\boldsymbol{q}}_{CS}$ and so $\boldsymbol{J}_{l+1} = \boldsymbol{I}$, and are thus projected in the null space of all other tasks.

Using the standard method (4), a CS task is added as

$$\dot{\boldsymbol{q}}_{l+1} = \dot{\boldsymbol{q}}_l + \boldsymbol{P}_{A,l}\, \dot{\boldsymbol{q}}_{CS}, \tag{23}$$

In the singularity-robust approach (10), all tasks are separately transformed into a joint velocity and then projected (except for the main one) in the null space of all higher priority tasks. Therefore, a CS task is treated in the same way as with the standard method:

$$\dot{\boldsymbol{q}}_{C,l+1} = \dot{\boldsymbol{q}}_{C,l} + \boldsymbol{P}_{A,l} \dot{\boldsymbol{q}}_{CS}. \tag{24}$$

In the RP framework the CS task is evaluated first, then the second-to-last task is added by considering

$$\boldsymbol{J}_{RA,l} = \begin{pmatrix} \boldsymbol{J}_l \\ \boldsymbol{I} \end{pmatrix}. \tag{25}$$

It is easy to show that $\widetilde{\boldsymbol{P}}_{l+1} = \boldsymbol{I} - \boldsymbol{P}_{A,l}$, and thus

$$\left(\boldsymbol{J}_l \widetilde{\boldsymbol{P}}_{l+1}\right)^{\#} = \boldsymbol{T}_l \left(\boldsymbol{J}_l \boldsymbol{T}_l\right)^{\#} = \boldsymbol{J}_l^{\#}. \tag{26}$$

As a result, a CS task is included in the RP framework simply by initializing $\dot{\boldsymbol{q}}_{RP,l+1} = \dot{\boldsymbol{q}}_{CS}$ in (13) or (22). While a configuration space task does not introduce algorithmic singularities in any of the presented methods, the RP approach turns out to be more robust, because it does not rely on an accurate, exact computation of last projector $\boldsymbol{P}_{A,l}$.

IV. Numerical Results

To show the effectiveness of the proposed method we present some numerical results obtained in Matlab R2012a on a 2.4 GHz Intel Core 2 Duo. We have considered a planar robot with six revolute joints and three two-dimensional tasks, the highest priority task being defined for the end-effector position, the second priority for the tip position of the fourth link, and the lowest priority for the tip of the second link[2]. 'Scenes' are selected by a random choice of link lengths, current configurations, and desired velocities for the three tasks. For each scene, we compute the joint velocity obtained with the standard method (4), the singularity-robust method (10), and the proposed RP method (22), and then evaluate the *normalized* task execution errors

$$\bar{e}_k = \frac{\|\boldsymbol{J}_k \dot{\boldsymbol{q}}_X - \dot{\boldsymbol{x}}_k\|}{\|\dot{\boldsymbol{x}}_k\|}, \qquad k = 1 \ldots 3, \tag{27}$$

where $\dot{\boldsymbol{q}}_X$ equals respectively $\dot{\boldsymbol{q}}_3$, $\dot{\boldsymbol{q}}_{C,3}$, or $\dot{\boldsymbol{q}}_{RP,1}$ for the three methods. In all methods, pseudo-inverses are computed using the damping scheme (7–8) with $\epsilon = 10^{-8}$ and $\lambda^2 = 10^{-12}$.

Table I shows the mean, standard deviation, and maximum values of the errors (27) in a campaign of 100000 random scenes. When comparing worst case executions, with the standard method the two low-priority tasks influence the execution of the primary task leading to a non-negligible error. Using the singularity-robust method, the primary task is perfectly executed but the errors on the two other tasks are very large. On the other hand, the RP method is able to execute at best all three tasks; again, the first task is perfectly executed while errors on the other tasks are very limited and are mainly due to their unfeasible parts (linear dependencies).

[2]We used a realistic kinematics in order to encounter in practice singularities and linear dependencies, which are not easily found when choosing randomly three Jacobians.

TABLE I
STATISTICAL COMPARISON

Method		$\bar{e}_1$	$\bar{e}_2$	$\bar{e}_3$
Standard	mean	5.89 e-6	1.11 e-5	6.36 e-6
	std	1.1 e-3	3.1 e-3	1.9 e-3
	max	0.24	0.97	0.61
Singularity-robust	mean	5.83 e-12	130.6	176.43
	std	7.58 e-10	211.97	2.09 e3
	max	2.34 e-7	3.55 e4	3.00 e5
Reverse Priority	mean	3.85 e-12	1.82 e-5	1.17 e-5
	std	4.05 e-10	4.6 e-3	3.5 e-3
	max	9.62 e-8	1.38	1.09

Based on this statistical comparison, which does not consider ad-hoc scenes but rather an intensive random campaign, we can already state a superiority of the RP method over the two other most commonly used methods. In addition, this is obtained at practically no overhead in computational cost. The mean execution times are in fact similar: 0.354 [ms] for the standard method; 0.325 [ms] for the singularity-robust method, and 0.368 [ms] for the RP method.

V. Experimental Results

The proposed method has been implemented as a $C{+}{+}$ library in the ROS environment, and tested on a KUKA LWR IV robot. The robot joint velocity $\dot{\boldsymbol{q}}$ is commanded using the Fast Research Interface (FRI) libraries, with a sampling time $T = 2$ [ms].

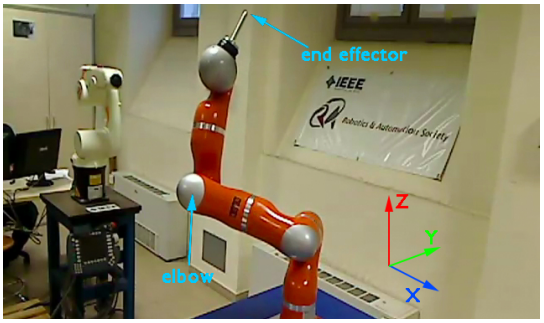

Fig. 7. The KUKA LWR IV in the initial configuration of the experiments

Given a desired end-effector trajectory $\boldsymbol{x}_{ee,d}(t) \in \mathbb{R}^3$, the primary task is specified as

$$\dot{\boldsymbol{x}}_1 = \dot{\boldsymbol{x}}_{ee,d} + k_{P,1}\boldsymbol{e}_1, \tag{28}$$

where $\boldsymbol{e}_1 = \boldsymbol{x}_{ee,d} - \boldsymbol{x}_{ee}$ and $k_{P,1} > 0$ is a scalar gain used to obtain convergence to and/or numerically stable execution of the desired trajectory. The second task requires that the z-coordinate of the elbow (the distal end of the third link) is kept constant at its initial value, while the third (two-dimensional) task requires the same for the x- and the y-coordinate of the elbow. The second and third tasks are thus specified as

$$\dot{x}_2 = k_{P,2}\, e_2, \qquad \dot{\boldsymbol{x}}_3 = k_{P,3}\, \boldsymbol{e}_3, \tag{29}$$

with $e_2 = x_{el,z}(0) - x_{el,z}$, $\boldsymbol{e}_3 = \boldsymbol{x}_{el,xy}(0) - \boldsymbol{x}_{el,xy}$, and scalar gains $k_{P,2} > 0$, $k_{P,3} > 0$.

In all tested methods, matrix pseudo-inversions are computed using damping as in eqs. (7–8), with $\lambda^2_{max} = 0.3$ and $\epsilon = 0.15$. These relatively large values of the damping factor and of its action range are necessary to limit the robot joint velocity near singularities. Starting from the initial configuration $\boldsymbol{q}(0) = \begin{bmatrix} 30 & 80 & 20 & 80 & 0 & -20 & 0 \end{bmatrix}$ [deg], as shown in Fig. 7, the desired end-effector position trajectory (primary task) is the helicoidal-like path shown in Fig. 8, with a feasible constant speed considering the robot motion capabilities. The proportional gains for the feedback action were chosen $k_{P,1} = 10$, $k_{P,2} = 10$ and $k_{P,3} = 100$. When only the primary task is considered, the solution provided by the three considered IK methods are identical.

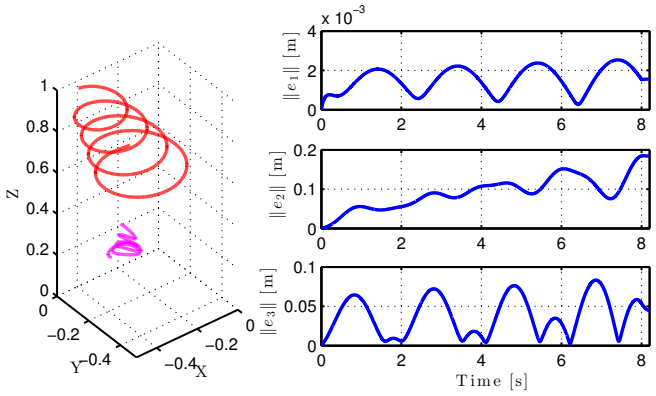

Fig. 8. Experimental results obtained by executing only the primary task. Cartesian trajectories [left]: end effector (red) and elbow (magenta). Task error nomrs [right]: primary task (top) and (uncontrolled) second and third tasks (middle and bottom). Identical results are obtained with all three methods

The prioritized execution of all three tasks using the standard method is presented in Fig. 9. The presence of the two secondary tasks produces an algorithmic singularity that causes an undesired robot behavior. The primary task was not completed by the robot, since the KUKA controller stopped the motion because joint torque limits were exceeded. The associated joint velocities of the robot are plotted in Fig. 10, where it is possible to see the discontinuities due to singularity approaching toward the end of the motion.

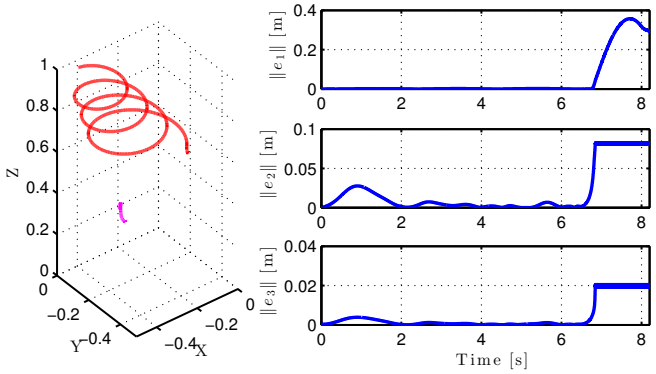

Fig. 9. Experimental results using the standard method. Cartesian trajectories [left]: end effector (red) and elbow (magenta). Task error norms [right]: primary task (top), second task (middle) and third task (bottom)

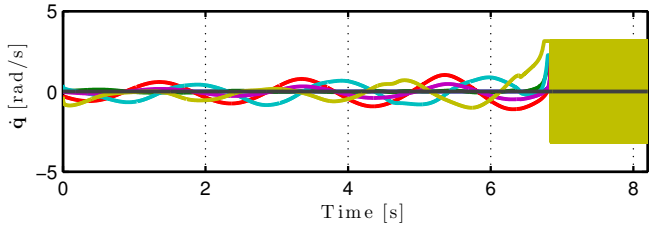

Fig. 10. Velocity profiles of the seven joints obtained when executing the tasks with the standard method (as in Fig. 9)

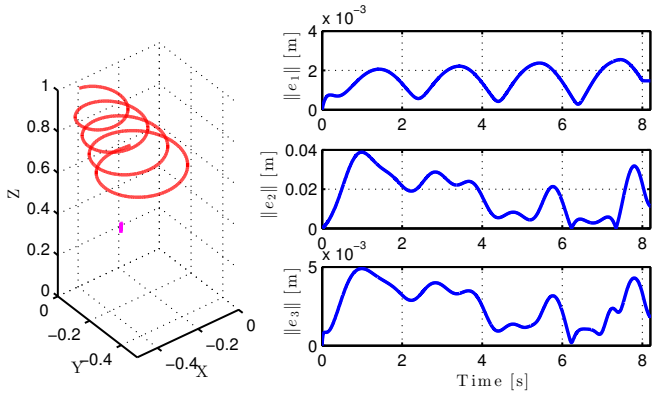

Fig. 11. Experimental results using the singularity-robust method. Cartesian trajectories [left]: end effector (red) and elbow (magenta). Task error norms [right]: primary task (top), second task (middle) and third task (bottom)

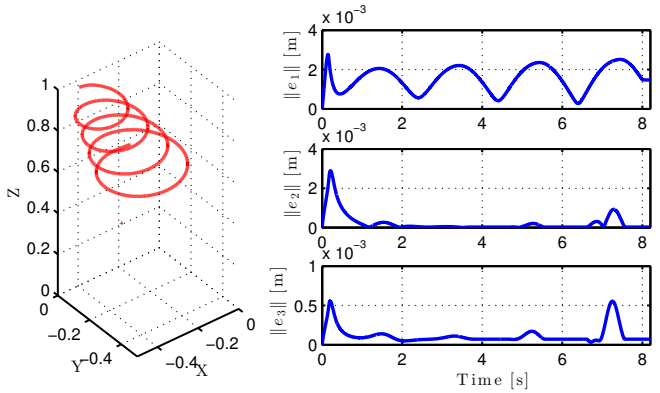

Fig. 12. Experimental results using our Reverse Priority (RP) method. Cartesian trajectories [left]: end effector (red) and elbow (magenta). Task error norms [right]: primary task (top), second task (middle) and third task (bottom)

Figure 11 shows the results obtained when executing the three tasks with the singularity-robust method of Chiaverini. The primary task is correctly executed, but with an undesired motion of the elbow. This behavior was expected, as discussed in Sec. II-B.

The results obtained with the proposed Reverse Priority method are shown in Fig. 12. It is quite evident that our method does not generate algorithmic singularities, which are encountered when using the standard method, while being able to accomplish at best all three tasks, as opposed to the singularity-robust method. The accompanying video shows the execution of all presented experimental results.

VI. Conclusions

We have introduced the Reverse Priority method to handle multiple tasks with strict priority. This new method adds higher priority tasks after having computed the contribution of lower priority tasks, so as to reduce the possible deformation caused by algorithmic singularities. As a result the correct task hierarchy is always preserved, while allowing the best execution of all tasks. The basic underlying idea has been presented, the tools needed to implement the RP method in an efficient way have been reported, and the effectiveness has been shown, both with numerical results on an extensive statistical campaign and with experiments on a KUKA LWR performing three prioritized tasks.

A more complete mathematical analysis and discussion is deferred to future work. One of the most appealing characteristics of the RP method that we plan to explore is the possibility of adding on the fly an hard constraint (as highest priority task) after having already computed the current solution so far. The RP method allows then to evaluate efficiently whether the additional constraint can be enforced or not.

References

[1] S. Chiaverini, G. Oriolo, and I. Walker, "Kinematically redundant manipulators," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 245–268.

[2] B. Siciliano and 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.

[3] A. De Luca, G. Oriolo, and B. Siciliano, "Robot redundancy resolution at the acceleration level," *Laboratory Robotics and Automation*, vol. 4, no. 2, pp. 97–106, 1992.

[4] O. Khatib, "The operational space framework," *JSME Int. J. Ser. C: Dynamics, Control, Robotics, Design and Manufacturing*, vol. 36, no. 3, pp. 277–287, 1993.

[5] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

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

[7] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks," in *Proc. Int. Conf. on Advanced Robotics*, 2009, pp. 1–6.

[8] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *Int. J. of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[9] F. Flacco, A. De Luca, and 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.

[10] F. Flacco, A. De Luca, and O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 3970–3977.

[11] C. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 13, no. 2, pp. 245–250, 1983.

[12] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Trans. on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, 1994.

[13] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *J. of Intelligent and Robotic Systems*, vol. 14, no. 1, pp. 43–68, 1995.

[14] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for realtime kinematic control of robot manipulators," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[15] T. L. Boullion and P. L. Odell, *Generalized Inverse Matrices*. Wiley-Interscience, 1971.

[16] P. Baerlocher and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1998, pp. 323–329.

[17] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.