Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assessment and Two Enhancements

Adrià Colomé, *Student Member, IEEE,* and Carme Torras, *Senior Member, IEEE,*

***Abstract*—Motivated by the need of a robust and practical Inverse Kinematics (IK) algorithm for the WAM robot arm, we reviewed the most used Closed-Loop IK (CLIK) methods for redundant robots, analysing their main points of concern: convergence, numerical error, singularity handling, joint limit avoidance, and the capability of reaching secondary goals. As a result of the experimental comparison, we propose two enhancements. The first is a new filter for the singular values of the Jacobian matrix that guarantees that its conditioning remains stable, while none of the filters found in literature is successful at doing so. The second is to combine a continuous task priority strategy with selective damping to generate smoother trajectories. Experimentation on the WAM robot arm shows that these two enhancements yield an IK algorithm that improves on the reviewed state-of-the-art ones, in terms of the good compromise it achieves between time step length, Jacobian conditioning, multiple task performance, and computational time, thus constituting a very solid option in practice. This proposal is general and applicable to other redundant robots.**

I. INTRODUCTION

Moving robot arms in task space requires efficient and well-behaved Inverse Kinematics (IK) solutions. Along several decades, a lot of effort within the Robotics community has been devoted to obtaining fast and robust IK algorithms. Analytical methods have always been preferred to iterative ones, because their solution is exact and usually faster to compute. However, with the rise of redundancies in robots, analytical solutions become harder to obtain [1] [2] [3] and thus again alternatives need to be explored [4] [5], especially in order to benefit from the additional degrees of freedom [6]. In addition, complex tasks impose more restrictions on IK solutions, such as in the case of medical robots [7] [8].

In tuning the IK of the 7-dof WAM manipulator to the particular requirements of some applications, we noticed that the existing generic KDL algorithm [9] could sometimes fail due to joint limits. We tried other open-source IK algorithms [10], but none performed to entire satisfaction, thus we explored other possibilities for redundant IK.

Although there exist many alternatives for trying to solve the IK problem, such as interval methods [11], distance-based methods [12], or even neural networks [13] [14] [15] and Bézier maps [16] [17], probably the most popular way is to use closed-loop algorithms, where a first-order Jacobian matrix [18] [19] of the robot kinematics is computed, which maps joint velocities into task space velocities, and inverted to map the error into an update of the joint values that is likely to reduce the task error. The updated joint state at step $k+1$ is then $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}^k$, for some computed $\Delta\boldsymbol{\theta}^k$:

$$\Delta\boldsymbol{\theta}^k = \alpha J^\star \mathbf{e}, \tag{1}$$

where α is a gain, $J^\star$ is an inverse of the geometric Jacobian matrix and $\mathbf{e}$ is the positioning error. The first attempts to close the IK loop used the Moore-Penrose pseudoinverse [20] of the Jacobian matrix [21] to invert the differential kinematics equation of the robot. In other works, the Jacobian transpose was used [22], which is faster to compute. As a distinctive advantage over alternative IK methods, CLIK algorithms do not require any previous knowledge or learning process with the robot, other than its Jacobian matrix, this being the main reason for its preferred use over other options. However, CLIK algorithms become unstable when the robot is close to a singularity: the condition number of the Jacobian becomes very large, thus amplifying the numerical error at each iteration, and also requiring large variations in some joints in order to reduce the error in a given direction. To solve these problems, the Jacobian matrix can be damped or filtered [25] [26], reducing this condition number, but not always reducing large joint variations. Some attempts also use second-order derivatives of motion, i.e., calculating the Hessian matrix of the forward kinematics [27], although this requires much more computation time.

In a continuous time assumption, the convergence of closed-loop methods can be demonstrated in terms of Lyapunov theory [28] [29]. Nevertheless, real-time computations have a fixed step, lower bounded by the computation capability of a processor, thus convergence cannot always be ensured by means of Lyapunov theory. Although there exist discrete-time versions of it [30], their application is not immediate, and some additional assumptions must be made.

There is also some literature about the convergence of these methods which takes the discrete-time system as a sequence and proves its convergence. In [31], an upper bound on the gain α that guarantees convergence is found, but restricting the operational space to a subset where the Jacobian is full-rank with bounded singular values, so its application is not

A preliminary version of this research was presented at IROS'12 [60]. This research is partially funded by the CSIC project CINNOVA (201150E088) and the Catalan grant 2009SGR155. A. Colomé is also supported by the Spanish Ministry of Education, Culture and Sport via a FPU doctoral grant (AP2010-1989).

The authors are with Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens Artigas 4-6, 08028 Barcelona, Spain. E-mails: [acolome,torras]@iri.upc.edu

general. Nevertheless, this work points out the relevance of the initial error dependency for these methods to converge, so the closer to the goal, the better these methods perform in the initial steps. In general, a smaller step improves convergence rate on the one hand, but slows the algorithm on the other.

To avoid large gains near a singularity, reducing the global gain is not a truly effective strategy, as we will be also damping the gain in the directions we would like the robot to move. For this reason, and without loss of generality, we will omit the step α from now on. In [32] a Selective Damping (SD) of the gain on the joint variations derived from each task-space error component is proposed, that modifies each gain depending on the corresponding column of the Jacobian and a predefined maximum joint variation γ_{max}. This effectively solves the gain issue, but does not solve singularity issues, such as the loss of rank and algorithmic singularities.

Using first-order derivative algorithms of robot motion has also the drawback that, depending on the goal position, the robot can get stuck at an *algorithmic singularity*, a pose where the error $\mathbf{e}$ belongs to the kernel of the inverted Jacobian, or in a multiple-task algorithm, as we will see later, the joint variation derived from a secondary task takes the opposite value of that for the primary task, thus the total computed joint variation becomes $\Delta\boldsymbol{\theta} \simeq 0$.

The main advantadge of redundancy is to be able to perform secondary tasks and/or to choose which solution suits some criterion best. To this purpose, an optimization function can be set to find, within the set of IK solutions, the one that performs best according to the criterion. The most common procedure is to project a gradient of a secondary task into the kernel of the Jacobian matrix, in order not to affect much the position error. Other algorithms like the Augmented Jacobian or the Extended Jacobian [33], in which rows are added to the Jacobian, have been used. Among the existing criteria for optimization, the *manipulability* measure [34] [35] is often used. Other criteria such as collision avoidance [36] (by setting a minimum distance to a certain object), minimum effort kinematics [37] or structural stiffness are also used [38]. But respecting joint limits is often the main priority when exploiting the redundancies of a robot.

This paper provides an overview of the different CLIK algorithms found in literature, also regarding numerical error propagation, which is sometimes forgotten when analysing these algorithms. Focusing on solving the IK with feasible joint values, two enhancements upon the state-of-the-art are proposed. The first one is a way of filtering the Jacobian matrix that ensures a given numerical conditioning, while the second uses the advantages of the latest works on continuity of inverse operators applied to robotics [39] with a controlled step size [32] to smoothen the motion of the robot. All the analysed algorithms, as well as the proposed enhancements, have been implemented on a Barrett WAM and tested both in simulation and in real experimentation.

II. Preliminaries

A. Notation

Along this work, the notation in Table I will be used and, for the methods presented, we will be using the abbreviations in Table II.

TABLE I
NOTATION

J	Geometric Jacobian
$J^\dagger$	Jacobian pseudoinverse
$\boldsymbol{\theta} = [\theta_1, ..., \theta_m]$, $\Delta\boldsymbol{\theta}$	Joint state, and joint state variation
$\mathbf{e}$	Positioning error of the robot
$\kappa(\cdot)$	Condition number of a matrix
$f(\cdot)$	Forward kinematics function
α	Gain in a CLIK algorithm
$\sigma_1, ..., \sigma_n$	Jacobian singular values
m	Number of joints
n	Task space dimension

TABLE II
METHODS ABBREVIATIONS

Name	Abbreviation	Equation/Section
Jacobian Pseudoinverse	JP	(4)
Jacobian Transpose	JT	(7)
Selective Damping	SD	Section III-A
Jacobian Damping	JD	(9)
Jacobian Filtering	JF	(12)
Error Damping	ED	(13)
Improved Error Damping	IED	(14)
Singular Value Filtering	SVF	Section VII
Jacobian Weighting	JW	(22)
Gradient Projection	GP	(23)
Joint Clamping	JC	(26)
Task Augmentation	TA	Section V-D
Task Priority	TP	(31)
Continuous Task Priority	CTP	(32)

For the position and orientation error representation as an n-dimensional vector, the incommensurability of position and orientation units is a limiting situation, as the Jacobian inverse is not invariant to rescalation and translation. This was a major problem at earlier stages of hybrid control theory [40], which relied on an orthogonal complements structure that was not invariant wrt rescaling of the units taken. One solution to this problem was to use metrics that converted all Jacobian components to energy units [41] [42]. However, when using a Jacobian matrix with disparate units for IK, despite it being dependent on the units taken and the relation between them, the convergence of CLIK algorithms is not affected. Using the orientation error in [36], the equivalence of $2rad = 1m$ is often taken and provides a reasonable error ratio between position and orientation.

B. Condition Number (CN)

Given a system of the type $\Delta\boldsymbol{\theta} = J^\star\mathbf{e}$, where $\star$ denotes an inverse operator, it is very common to have numerical or measurement errors on the robot's task position, or uncertainty on the kinematic parameters of the robot [43] [44]. Therefore, we need to take into account some uncertainty $\delta\mathbf{e}$ on the position error $\mathbf{e}$ (difference between target and current positions). It is fundamental to avoid amplifying this uncertainty when computing $\Delta\boldsymbol{\theta}$. To this purpose, the relative error $\delta\boldsymbol{\theta}$ on $\Delta\boldsymbol{\theta}$ coming from the uncertainty $\delta\mathbf{e}$ on $\mathbf{e}$ can be estimated using the condition number of $J^\star$ [45] [46]:

$$\frac{\|\delta\boldsymbol{\theta}\|}{\|\Delta\boldsymbol{\theta}\|} \leq \kappa(J)\frac{\|\delta\mathbf{e}\|}{\|\mathbf{e}\|}, \tag{2}$$

where $\kappa(J^\star)$ is the condition number of $J^\star$, computed as the ratio of its maximum and minimum singular values:

$$\kappa(J^\star) = \frac{\sigma_{max}(J^\star)}{\sigma_{min}(J^\star)} \tag{3}$$

Note that the dependency of the Jacobian on the units taken and the orientation-translational equivalence chosen may affect the CN of the linear system solved to obtain $\Delta\boldsymbol{\theta}$. Nevertheless, the CN will only grow to infinity when the Jacobian is not full-rank, which happens when approaching a singularity. As singularities are invariant to rescalation, we can conclude that the error propagation when solving the linear system for joint increments will have the same behaviour for any scale chosen for angles and distances.

III. Convergence and Singularities

Regarding Equation (1), to make the algorithm converge to zero error, an appropiate inverse of the Jacobian matrix must be applied. The Jacobian Pseudoinverse (JP) algorithm is widely used, inverting the Jacobian with the Moore-Penrose pseudoinverse matrix [20], a generalized inverse for non-square matrices, that can be defined as $J^\dagger = J^T(JJ^T)^{-1}$ when J is full-rank. With this inverse, the JP update rule is:

$$\Delta\boldsymbol{\theta} = J^\dagger \mathbf{e}, \tag{4}$$

$\mathbf{e}$ being the task error.

When a robot reaches a singularity, the algorithm might get stuck in a point or the pseudoinverse matrix may have too large values. By computing the Singular Value Decomposition (SVD) of J:

$$J = U\Sigma V^T, \tag{5}$$

and taking the singular values of J, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$: $J = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, with $\mathbf{u}_i$ and $\mathbf{v}_i$ being the ith columns of U and V, respectively, and $r = max_i\{i \mid \sigma_i > 0\}$.

As U and V are orthonormal matrices, the pseudoinverse of J is:

$$J^\dagger = V\Sigma^\dagger U^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T. \tag{6}$$

This expression shows that when the robot gets close to a singularity (one σ_i becomes very small), very large gains occur when computing $\Delta\boldsymbol{\theta} = J^\dagger \mathbf{e}$. In addition, the CN of the pseudoinverse is $\kappa(J^\dagger) = \frac{\sigma_1}{\sigma_n}$, which tends to infinity at a singularity, thus losing numerical precision.

To gain computation speed, the Jacobian Transpose (JT) method uses, instead of an inverse of the matrix J, its transpose with the following control rule [47]:

$$\Delta\boldsymbol{\theta} = J^T \mathbf{e}, \tag{7}$$

where J^T is now the transpose of the geometric Jacobian of the manipulator. This method has a computationally very fast step, although it may require more steps than other methods, and not being a least-squares solution can derive in chattering.

In the following subsections other alternatives to Eq.(1) are described, both to reduce the gain magnitude or large conditioning on the matrix inversion.

A. Selectively Damped Pseudoinverse (SD)

As seen in the previous section, a small singular value in the Jacobian yields large gains in all directions. In [32], a new way of controlling the step magnitude is defined, which consists in damping differently the effect of each one of the components of the position error, expressed in the basis of the singular value decomposition of J. Hence small singular values of the Jacobian, which would turn into large gains, are damped more severely.

In [32], a unitary error in the direction of one of the eigenvectors in the task space (columns of U) is taken, $\mathbf{e} = \mathbf{u}_i$, and a bound on the joint variation associated to this error component is obtained, which is then used to damp the gain of the corresponding error component and, finally, the gains over all the error components are added up. This results in a $\Delta\boldsymbol{\theta}$ with limited gains at each component of the task space.

B. Jacobian Damping (JD)

To avoid the discontinuity of the JP operator which results in a large conditioning of the Jacobian, meaning numerical error, the Jacobian Damping or singularity robust pseudoinverse was proposed [23] [24] [25] [26] [38]. If we redefine the Jacobian matrix as:

$$J_D = \sum_{i=1}^{r} \frac{\sigma_i^2 + \lambda^2}{\sigma_i} \mathbf{u}_i \mathbf{v}_i^T, \tag{8}$$

then the Jacobian Damping algorithm will use the following pseudoinverse:

$$J^{\dagger_D} = J_D^\dagger = J^T(JJ^T + \lambda^2 I)^{-1} = \sum_{i=1}^{n} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{u}_i^T, \tag{9}$$

which, for some small λ (usually around 10^{-3}), is almost the same matrix as the ordinary pseudoinverse when $\sigma_i^2 \gg \lambda^2$ $\forall i < n$, and when the smallest singular value σ_n is close to zero, $\lim_{\sigma_n \to 0} \frac{\sigma_n}{\sigma_n^2 + \lambda^2} = 0$, instead of ∞.

The JD algorithm avoids discontinuities in the Jacobian's singular values, and it provides the solution minimizing $\|J\Delta\boldsymbol{\theta} - \mathbf{e}\| + \lambda^2\|\Delta\boldsymbol{\theta}\|$, which will result in a smaller norm solution. However, a small λ value does not guarantee that $\|\Delta\boldsymbol{\theta}\|$ will be small and an analysis of the CN shows it provides no guarantee of keeping the numerical error within an acceptable range.

Let $g(\sigma) = \frac{\sigma}{\sigma^2 + \lambda^2}$ be the function used instead of a trivial inversion $1/\sigma$ when computing the pseudoinverse of the Jacobian as in Eq.(6). This function g, as we see in Fig. 1, has a maximum at $\sigma = \lambda$ with a value of $g(\lambda) = \frac{1}{2\lambda}$.

Now, considering the problem of solving $\mathbf{e} = J_D\Delta\boldsymbol{\theta}$, we can distinguish several cases depending on the least singular value (σ_n) of the geometric Jacobian:

- $\sigma_n > \lambda$, or $\frac{\lambda^2}{\sigma_i} < \sigma_n < \lambda$, $\forall i \neq n$. Then $g(\sigma_n) > g(\sigma_i)$, $\forall i \neq n$ and the condition number is:

$$\kappa(J^{\dagger_D}) = \frac{\sigma_n(\sigma_1^2 + \lambda^2)}{\sigma_1(\sigma_n^2 + \lambda^2)} \overset{\sigma_n \to \lambda}{\longrightarrow} \frac{\lambda^2 + \sigma_1^2}{2\sigma_1\lambda} \in O\left(\frac{1}{\lambda}\right) \tag{10}$$

- $\exists i, j$ so that $\frac{\lambda^2}{\sigma_i} < \sigma_n < \frac{\lambda^2}{\sigma_j}$. Then we have $g(\sigma_i) < g(\sigma_n) < g(\sigma_j)$ and, as the condition number bound will not depend on σ_n, it will be bounded.

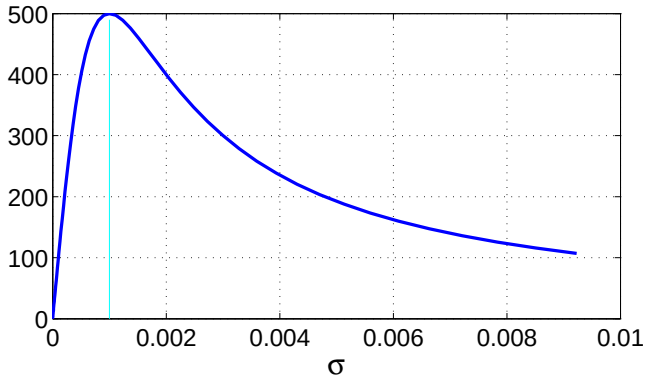

Fig. 1. Inverse of a singular value for the Jacobian Damping algorithm for $\lambda = 10^{-3}$. Note its maximum at $g(\lambda) = \frac{1}{2\lambda}$.

- $\sigma_n < \frac{\lambda^2}{\sigma_i}$, $\forall i \neq n$. Then $g(\sigma_n) < g(\sigma_i)$ and we now have (for some k):

$$\kappa(J^{\dagger_D}) = \frac{\sigma_k(\sigma_n^2 + \lambda^2)}{\sigma_n(\sigma_k^2 + \lambda^2)} \overset{\sigma_n \to 0}{\longrightarrow} \infty \tag{11}$$

This means that, on the one hand, λ should have a high value to avoid the maximum $1/(2\lambda)$ of the condition number at $\sigma_n = \lambda$, but on the other hand, λ must also have a very small value to avoid entering the last case, in which the conditioning tends to infinity as σ_n decreases.

As it is not always necessary to damp the Jacobian, and in many cases the necessary damping may vary, a singular region can be defined so that damping is applied only when entering it. To this purpose, a variable damping factor, leading to Jacobian Filtering (JF) can be used as in [26]:

$$\lambda^2 = \begin{cases} 0 & \text{if } \sigma_n \geq \epsilon \\ (1 - (\frac{\sigma_n}{\epsilon})^2)\lambda_{max}^2 & \text{if } \sigma_n < \epsilon \end{cases} \tag{12}$$

where σ_n is the smallest singular value of the Jacobian matrix, ϵ is the width of a singular region (in terms of singular values) in which the damping factor takes a non-zero value, and λ_{max} is the maximum damping factor allowed.

When using the JF algorithm, the function g becomes $g_F(\sigma) = \frac{\sigma}{\alpha\sigma^2 + \lambda^2}$, with $\alpha = 1 - (1/\epsilon^2)$, so the CN behaviour does not change wrt. the JD algorithm.

C. Error Damping (ED)

Another option for damping the pseudoinverse matrix is to use the current positioning error. In this way, Chan and Lawrence [48] proposed an Error Damped (ED) pseudoinverse matrix defined as:

$$J^{\dagger_{ED}} = J^T \left(JJ^T + EI_m\right)^{-1} = \sum_{i=1}^{n} \frac{\sigma_i}{\sigma_i^2 + E} \mathbf{v}_i \mathbf{u}_i^T, \tag{13}$$

where $E = 1/2\mathbf{e}^T\mathbf{e}$. Using this damping term strongly reduces the gains when far from the goal, but if $\sigma_n < \frac{E}{\sigma_i}$, $\forall i$, then $\kappa(J^{\dagger_{ED}}) \in O\left(\frac{1}{\sigma_n}\right)$ which can still become a large conditioning.

Equation (13) may have small singular values when the error is small. In such case, the condition number would rise again. To avoid this situation, in [49], an improved version of the error-damped pseudoinverse (IED) is proposed by adding a term $\omega I_m = diag(\omega_1, ..., \omega_m)$, being $\omega_i \simeq 10^{-1}l^2 \sim 10^{-3}l^2$, with l the characteristic length of the links [49]:

$$J^{\dagger_{IED}} = J^T \left(JJ^T + EI_m + \omega I_m\right)^{-1} = \sum_{j=1}^{n} \frac{\sigma_i}{\sigma_i^2 + E + \omega_i} \mathbf{v}_i \mathbf{u}_i^T. \tag{14}$$

This last proposal (14) is more robust than the filtering/damping methods, but when the goal position is singular, the inversion behaves in the same way and it may suffer conditioning issues as all the other filtering algorithms.

IV. SINGULAR VALUE FILTERING (SVF)

In order to overcome the conditioning problems of the JD, we propose to modify the Jacobian matrix' singular values so that it never loses rank and its condition number is bounded. To this purpose, if we take the SVD of J:

$$J = U\Sigma V^T = \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{15}$$

then we define

$$\hat{J} = \sum_{i=1}^{m} h_{\nu,\sigma_0}(\sigma_i) \mathbf{u}_i \mathbf{v}_i^T, \tag{16}$$

where

$$h_{\nu,\sigma_0}(\sigma) = \frac{\sigma^3 + \nu\sigma^2 + 2\sigma + 2\sigma_0}{\sigma^2 + \nu\sigma + 2} \tag{17}$$

is our proposed filtering rational function, where σ_0 is the minimum value we want to impose to the singular values of J, and ν is a shape factor, that regulates the curvature (shape) of function $h_{\nu,\sigma_0}(\sigma)$.

With Eq. (17) we can compute (assuming $\sigma_i > \sigma_{i+1}$ $\forall i$)

$$\hat{J}^\dagger = \sum_{i=1}^{n} \frac{1}{h_{\nu,\sigma_0}(\sigma_i)} \mathbf{v}_i \mathbf{u}_i^T, \tag{18}$$

to use it as the pseudoinverse. In this expression, it can be easily seen that $h_{\nu,\sigma_0}(\sigma)$ verifies:

- $h_{\nu,\sigma_0}(\sigma)$ is continuous and differentiable on the positive side of $\mathbb{R}$ which is where the singular values are.
- $\lim_{\sigma\to 0} h_{\nu,\sigma_0}(\sigma) = \sigma_0$, $\forall\nu$, thus σ_0 is the minimum value we will allow for the singular values of the Jacobian matrix.
- $h_{\nu,\sigma_0}(\sigma)$ has an asymptote of equation $y = \sigma$ for $\sigma \to \infty$, as $\lim_{\sigma\to\infty} \frac{h_{\nu,\sigma_0}(\sigma)}{\sigma} = 1$ and $\lim_{\sigma\to\infty} (h_{\nu,\sigma_0}(\sigma) - \sigma) = 0$, $\forall\nu$ and $\forall\sigma_0$.
- $h_{\nu,\sigma_0}(\sigma)$ is monotonic wrt σ if ν and σ_0 are defined verifying $\nu > \sigma_0$ and $2 > \nu\sigma_0$, which are not very restrictive conditions. This monotonicity guarantees that the condition number of the pseudoinverse (18) is always:

$$\kappa(\hat{J}^\dagger) = \frac{(\sigma_1^3 + \nu\sigma_1^2 + 2\sigma_1 + 2\sigma_0)(\sigma_n^2 + \nu\sigma_n + 2)}{(\sigma_1^2 + \nu\sigma_1 + 2)(\sigma_n^3 + \nu\sigma_n^2 + 2\sigma_n + 2\sigma_0)}. \tag{19}$$

Therefore we have:

$$\lim_{\sigma_n \to 0} \kappa(\hat{J}^\dagger) = \frac{(\sigma_1^3 + \nu\sigma_1^2 + 2\sigma_1 + 2\sigma_0)}{\sigma_0(\sigma_1^2 + \nu\sigma_1 + 2)} = \frac{A(\sigma_1)}{\sigma_0}, \tag{20}$$

which is always bounded by the inverse of the minimum value assigned to the singular values.

To sum up, $\hat{J}$ has lower-bounded singular values and tends to J when its singular values move away from 0. Moreover, with this filtering, the Jacobian matrix never loses rank as the singular values are strictly positive.

Figure 2 displays the condition number of different methods in the case of a 4R planar manipulator moving towards a singularity, for a damping factor of $\lambda = 10^{-2}$, and allowing a maximum damping factor on the filtering algorithm (variable damping factor) of $\lambda_{max} = 5\lambda$. The results, plotted in logarithmic scale, show that all previous filtering methods fail, at some point, at keeping the CN stable, while our proposal, with $\sigma_0 = 0.005$, $\nu = 10$, presents a bounded conditioning.

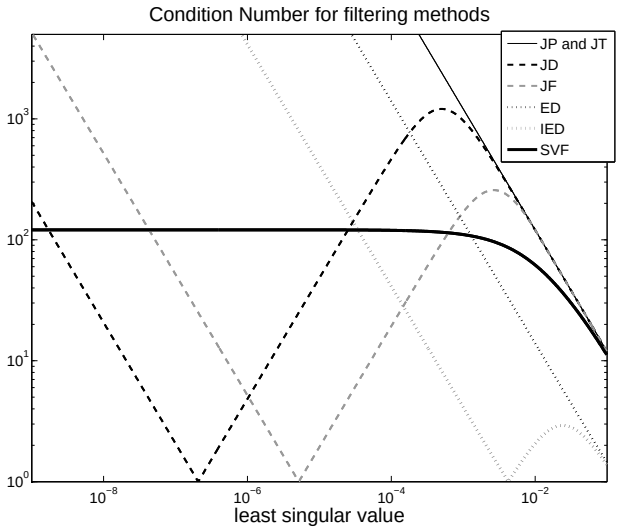

Fig. 2. Condition Number, in logarithmic scale, for different methods on a 4R planar robot approaching a target singular position.

As a first test, these filtering algorithms have been applied to an example trajectory with a 4R manipulator, as recorded in Fig. 3, where it is clear that, while the JP and the JF have very large gains, and the JT and the IED are easily stuck and still unable to solve the gain issue, the SD performs smoothly, and our proposal, combining a method to avoid large condition number (SVF) with a bound on the gains (SD) makes the robot move smoother than with the other methods presented up to this point.

In addition, Table III shows the reprojection error on the task space for each algorithm, i.e., the computed difference on each eigenvector component between JP and the other alternatives. We can see that, the larger the parameter ν is, the smaller the reprojection error, thus a reasonably large number verifying $\sigma_0 < \nu < 2/\sigma_0$ is recommended (a value of 10 has been used throughout this paper).

TABLE III
REPROJECTION ERROR.

Method	$\mathbf{J}^\star$	$\mathbf{J}(\mathbf{J}^\dagger\mathbf{e} - \mathbf{J}^\star\mathbf{e})$
JP	$\frac{1}{\sigma}$	0
JT	σ	$\sigma - 1$
JD	$\frac{\sigma}{\sigma^2+\lambda^2}$	$\frac{\lambda^2}{\sigma^2+\lambda^2}$
JF	$\frac{\sigma}{\sigma^2+\lambda_{max}^2\left(1-(\sigma/\epsilon)^2\right)}$	$\frac{\lambda_{max}^2\left(1-(1/\epsilon)^2\right)}{\sigma^2\left(1-(\sigma/\epsilon)^2\right)+\lambda_{max}^2}$
ED	$\frac{\sigma}{\sigma^2+0.5\cdot\mathbf{e}^T\mathbf{e}}$	$\frac{0.5\cdot\mathbf{e}^T\mathbf{e}}{\sigma^2+0.5\cdot\mathbf{e}^T\mathbf{e}}$
IED	$\frac{\sigma}{\sigma^2+0.5\mathbf{e}^T\mathbf{e}+\omega}$	$\frac{0.5\cdot\mathbf{e}^T\mathbf{e}+\omega}{\sigma^2+0.5\mathbf{e}^T\mathbf{e}+\omega}$
SVF	$\frac{\sigma^2+\nu\sigma+2}{\sigma^3+\nu\sigma^2+2\sigma+2\sigma_0}$	$\frac{\sigma_0}{\sigma^3+\nu\sigma^2+2\sigma+2\sigma_0}$

V. MULTIPLE TASKS

Usually, when computing the IK of a robot, it is a good idea to compute not just a solution of the inverse kinematics, but the solution that behaves best according to a certain criterion. Even more with redundant robots, where the number of solutions may be infinite.

A. Jacobian Weighting (JW)

Apart from considering metrics on the task space, metrics on the joints space can also be used. This metric (which can be variable) can be used to achieve secondary goals, prioritize the motion of certain joints, or even block a joint. In [53], Chan and Dubey defined the $\Delta\boldsymbol{\theta}$ that minimizes

$$\|\Delta\boldsymbol{\theta}\|_W = \sqrt{\Delta\boldsymbol{\theta}^T W \Delta\boldsymbol{\theta}}, \tag{21}$$

where W is an $m \times m$ diagonal matrix applying a weight to each joint depending on its relevance (according to a specified criterion), instead of the common Euclidean norm. Taking $\Delta\boldsymbol{\theta}_W = J_W^\dagger \Delta x$, then

$$\Delta\boldsymbol{\theta} = W^{-1}J^T(JW^{-1}J^T)^{-1}\mathbf{e}, \tag{22}$$

where the influence of $w_i = W_{i,i}$ is that, the greater the w_i, the less θ_i will vary in the given step.

B. Gradient Projection (GP)

Another optimization option is to create a secondary task as a gradient of a function, and project it to the kernel of the primary task.

Given a cost function H, one can calculate its gradient ∇H, and project it onto the kernel of the matrix J. Knowing that for any position of the manipulator, $\delta\mathbf{x} = J\delta\boldsymbol{\theta}$. This means that if $\delta\boldsymbol{\theta} \in ker(J)$, then $\delta\mathbf{e} = 0$, so the added term would not affect the error. In practice, as the step is not infinitely small, the linearization done by projecting the vector to the nullspace would indeed generate some additional error.

This projection onto the kernel is accomplished by multiplying any vector v by the matrix $P = I - J^\dagger J$.

Hence the GP is expressed as:

$$\Delta\boldsymbol{\theta} = J^\dagger\mathbf{e} + \mu P \nabla H, \tag{23}$$

with μ a scalar indicating the magnitude of the projection, and ∇H the vector to project.

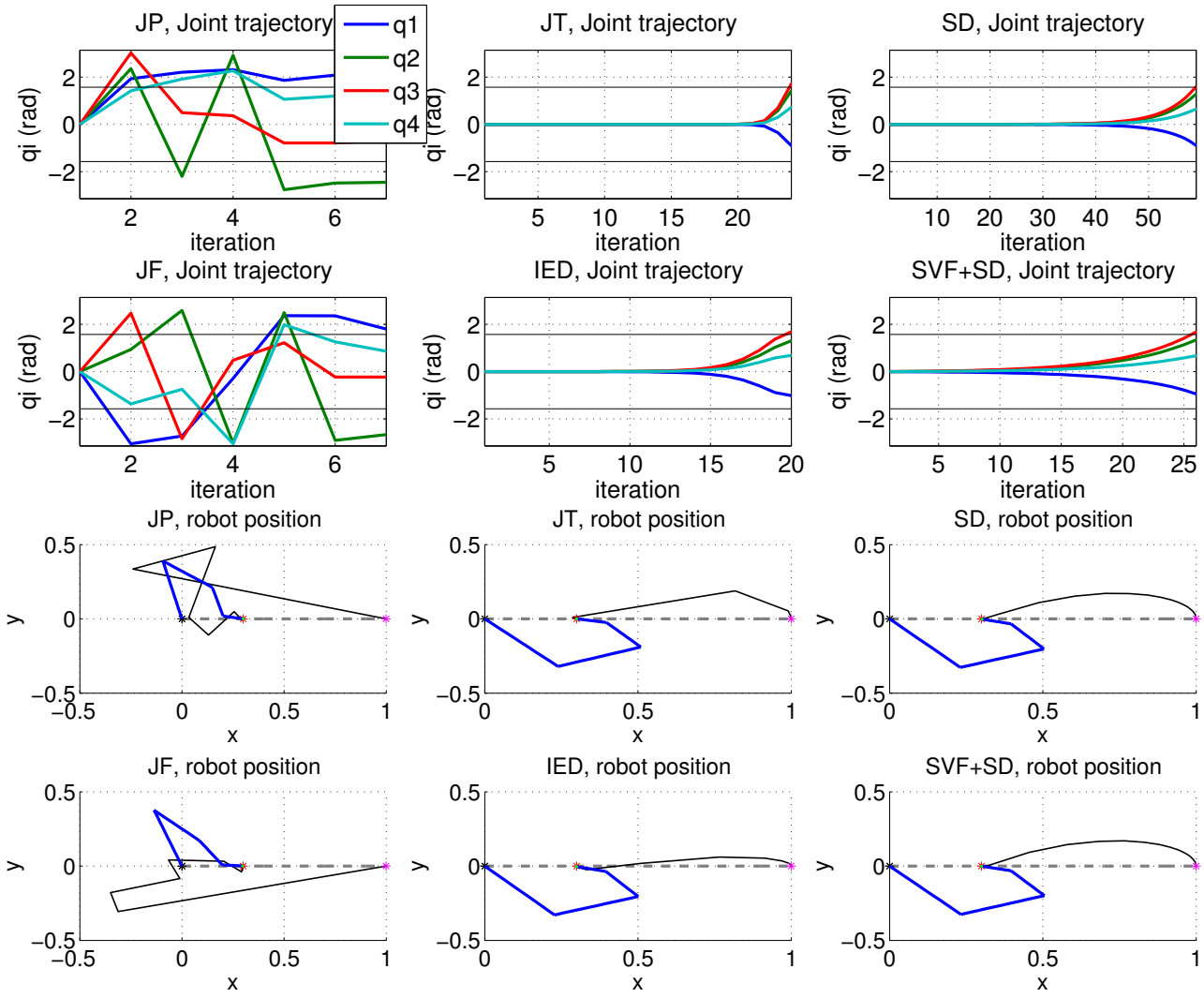

Fig. 3. Behaviour of some CLIK algorithms (see Table II for notation) applied to a 4R planar manipulator. The upper plots show joint trajectories, with joint limits marked with horizontal lines. Note the different scales for the x-axis. The lower plots show the robot frame, evolving from a dashed light color pose at the start position to a darker one, with its end-effector's trajectory marked in a thiner black line.

C. Task Priority (TP)

The GP idea can be generalised to Task Priority (TP) algorithms [50], where an ordered set of tasks $t_1, ..., t_k$ is specified, their errors $\mathbf{e}_1, ...\mathbf{e}_k$ and Jacobians $J_1, ...J_k$ are computed, and next each task error is projected onto the kernel of the Jacobian of the previous tasks. Then $\Delta\boldsymbol{\theta}_1 = J_1^\star \mathbf{e}_1$, where $\star$ is an inverse operator (commonly, the pseudoinverse). For each i from 2 to the number of tasks k, one can compute

$$\Delta\boldsymbol{\theta}_i = \Delta\boldsymbol{\theta}_{i-1} + (J_i \cdot P_{i-1}^A)^\star (\mathbf{e}_i - J_i \Delta\boldsymbol{\theta}_{i-1}), \qquad (24)$$

where $P_{i-1}^A = I - J_i^{A\star} J_i^A$ is the kernel projection operator and J_i^A is an augmented Jacobian of the first i tasks

$J_i^A = \left[\; J_1; \quad \ldots \quad ; J_i \; \right]$.

For online position-tracking problems, a variant of this method can be implemented as explained in [51] to avoid singularity issues.

D. Task Augmentation (TA)

Instead of projecting a secondary task onto the kernel of the robot position tracking task, in the case of a redundant robot a secondary task can be added as a new row of the Jacobian matrix to complete it and obtain a square matrix [52].

Completing the Jacobian matrix to a square matrix has the advantadge of allowing to use its inverse directly when it is full rank. Nevertheless, it is possible that the added rows are linearly dependent on the geometric Jacobian rows. To overcome this problem, a general weighted Jacobian can be used [55] in which a *Gramm-Schmidt* orthonormalisation is performed to ensure that the added task does not originate rank problems.

Moreover, an activation threshold may be used for the added task (e.g., when it entails avoiding joint limits).

VI. JOINT LIMIT AVOIDANCE

Joints usually have limits on their prismatic/rotational ranges, and a solution to the IK with a joint value outside

its limits is not a feasible solution. Hence one of the most important properties of a good IK solution is that it lies inside these limits. The redundant degrees of freedom of a robot are often used to achieve such a goal.

A. Joint Limits as a Secondary Task

A first approach to avoid joint limits may be to use gradient projection. One can use a joint-centering function and project it to the kernel of the main task. For example, taking [56]

$$H = -\frac{1}{2m}\sum_{i=1}^{m}\frac{(\theta_i^{max}-\theta_i^{min})^2}{(\theta_i^{max}-\theta_i)(\theta_i-\theta_i^{min})}, \tag{25}$$

which tends to infinity when approaching a joint limit, and has a minimum value at the joint range's center.

Using the GP equation (23) with the gradient of the H function defined, joints are pushed to their range center values, but joint limits are not always avoided. This is due to the fact that the push-to-center function H is only activated on the kernel of the position reaching task. In [57], the obtained joint variations are rescaled at each iteration to keep joints within the valid range.

B. Avoiding Joint Limits with Activation Matrices

To avoid joint limits, one can also use a weighting matrix that penalizes the motion of the joints approaching a limit [53], or even block them. This is called Joint Clamping (JC).

In [54], a factor is added when updating the joint state at step k: $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + H\Delta\boldsymbol{\theta}^k$ is used instead of $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}^k$, where H is a diagonal activation matrix, with:

$h_i = \begin{cases} 0 & \text{if } q_i > q_i^{max} \text{ or } q_i < q_i^{min} \\ 1 & \text{otherwise} \end{cases}$

This control law clamps any motion that violates joints' limits, but does not push the joints away from them. In this way, when the robot reaches a joint limit, it may loose this degree of freedom, and go on with the others to reach the target. The algorithm follows the expression:

$$\Delta\boldsymbol{\theta} = H(JH)^{\dagger}\mathbf{e}. \tag{26}$$

A problem that may arise when using this algorithm is that, even with an activation matrix continuous wrt. joint activation as in [54], the pseudoinverse operator is not continuous with respect to this activation matrix. Theorem 4.2 in [58] states that the effect of any nonzero h_i in (26) is the same (in the sense that there is no dependency on the value h_i as long as it is not zero). In fact, it can also be seen that damping the pseudoinverse does not solve the problem, outside of a very small interval [59]. To resolve these issues, this previous work proposes a continuous (wrt. activation matrix) pseudoinverse operator, defined as follows.

For task-activation matrices $G = diag(g_1, .., g_n)$:

$$J^{\oplus G} = \sum_{P\in\wp(N)}\left(\prod_{i\in P} g_i\right)\left(\prod_{i\notin P}(1-g_i)\right)J_P^{\dagger}, \tag{27}$$

$\wp(N)$ being the power set of $N = \{1, .., n\}$, and $J_P = G_0 J$, where G_0 is a square diagonal matrix with $G_{0_i} = 1$ if $i \in P$ and 0 otherwise.

And for joint-activation matrices $H = diag(h_1, .., h_m)$:

$$J^{H\oplus} = \sum_{Q\in\wp(M)}\left(\prod_{i\in Q} h_i\right)\left(\prod_{i\notin Q}(1-h_i)\right)J_Q^{\dagger}, \tag{28}$$

$\wp(M)$ being now the power set of $M = \{1, .., m\}$, $J_Q = JH_0$, where $H_{0_i} = 1$ if $i \in P$, and 0 otherwise. The idea behind these pseudoinverse expressions is that the transition between activated and deactivated tasks is smooth wrt. the activation parameters h_i, g_i. For further development of these inverses, see [59].

Therefore, by using:

$$\Delta\boldsymbol{\theta} = J^{H\oplus}\mathbf{e}, \tag{29}$$

the continuity problem is solved. However, when blocking a joint, the robot would still lose one degree of freedom and eventually may not reach the goal.

C. Joint Limits as the Primary Task

Although one can add secondary tasks to avoid joint limits, the only way to guarantee such avoidance is to set it as the primary task and include the positioning goal as a secondary task. To this purpose, it is defined $J_1 = H_m$, m being the number of joints of the manipulator, and matrix H_m as

$$H_m = diag(h_\beta(\theta_i)), \tag{30}$$

where h_β is a continuous function (usually a piecewise function, as in [39]) that progressively deactivates the joint when it reaches a specified distance β from its limits.

Then the main task error in (24) is defined as $\mathbf{e}_1 = -\lambda_{JL}\boldsymbol{\theta}$, λ_{JL} being a scalar to weigh the importance of joint limits. Typically, $\lambda_{JL} \in [0.1, 0.5]$.

With these definitions of $\mathbf{e}_1$ and J_1, $\Delta\boldsymbol{\theta}_1$ is zero when the joints' positions are far from their limits (at a distance greater than β for each joint), and has a *push-to-center* value when in the limits neighbourhood. On its kernel (i.e., the joints which are not forced to move to the center due to their proximity to the limit), a secondary task is applied to reach the goal with $J_2 = J$, the Jacobian matrix of the robot, and $\mathbf{e}_2 = \mathbf{x}_d - \mathbf{x}$, the positioning error.

Then the algorithm following this task-priority hierarchy is:

$$\Delta\boldsymbol{\theta} = J_1\mathbf{e}_1 + J_2(I_m - J_1^{\dagger}J_1)^{\dagger}(\mathbf{e}_2 - J_2J_1\mathbf{e}_1). \tag{31}$$

Moreover, this task-priority scheme can be performed with the continuous pseudoinverse operator [39], to get: $\Delta\boldsymbol{\theta}_1 = J_1^{\oplus H}\mathbf{e}_1$, H being the same matrix defined in (30). And the following tasks would be defined as:

$$\Delta\boldsymbol{\theta}_i = \Delta\boldsymbol{\theta}_{i-1} + J_i^{P_\oplus^{i-1}\oplus}(\mathbf{e}_i - J_i\Delta\boldsymbol{\theta}_{i-1}), \tag{32}$$

where $P_\oplus^0 = I$, and $P_\oplus^i = P_\oplus^{i-1} - J^{P_\oplus^{i-1}\oplus}J_i$ is the *kernel projection* operator. For the case considered, (31) is equivalent to:

$$\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\theta}_1 + J_2^{P_\oplus^1\oplus}(\mathbf{e}_2 - J_2\Delta\boldsymbol{\theta}_1), \tag{33}$$

with $\Delta\boldsymbol{\theta}_1 = J_1^{H\oplus}\mathbf{e}_1 = H(-\lambda_{jl}\boldsymbol{\theta})$, as $J_1 = I_m$ and $\mathbf{e}_1 = -\lambda_{jl}\boldsymbol{\theta}$. Therefore:

$$\Delta\boldsymbol{\theta} = H(-\lambda_{jl}\boldsymbol{\theta}) + J_2^{(I_m-H)\oplus}(\mathbf{e}_2 + \lambda_{jl}J_2H\boldsymbol{\theta}). \tag{34}$$

VII. SMOOTHING ENHANCEMENT

The TP scheme may present large steps and gains, resulting in an almost-chaotic behaviour. To solve these uncontrolled gains, it would be necessary to avoid large steps and condition numbers. Paying attention to (31), we can reorder the terms and separate the position error-dependent terms (**e**) from those that don't depend on it:

$$\Delta\boldsymbol{\theta} = \left(I - J^{(I_m-H)\oplus}J\right) H(-\lambda_{jl}\boldsymbol{\theta}) + J^{(I_m-H)\oplus}\mathbf{e}. \quad (35)$$

We intend to apply the ideas underlying SD [32], so as to damp selectively each one of the task space eigenvectors of the Jacobian matrix J, or its filtered version with SVF, taking care of the dependency of the position variation $J\Delta\boldsymbol{\theta}$ with respect to the position error **e**.

To do so, we have to find a bound for $J\Delta\boldsymbol{\theta}$, i.e., the position variation after each step, which can be written using (35) and separating the position error-depending part (**e**) from the rest as follows:

$$J\Delta\boldsymbol{\theta} = J\left(I - J^{(I-H)\oplus}J\right) H(-\lambda\boldsymbol{\theta}) + JJ^{(I-H)\oplus}\mathbf{e}. \quad (36)$$

Now, after calculating $J^{(I-H)\oplus}$, we can use its SVD, keeping in mind that the result of this decomposition has to be expressed knowing $J^{(I-H)\oplus}$ is an inverse of J, thus

$$J^{(I-H)\oplus} = \hat{V}\hat{\Sigma}^{-1}\hat{U}^T = \sum_{i=1}^{n} \hat{\sigma}^{-1}\mathbf{v}_i\mathbf{u}_i^T. \quad (37)$$

And knowing that $(\mathbf{u}_k^T \cdot \mathbf{e}) = (\mathbf{u}_k^T \cdot \sum_{s=1}^{n}(\mathbf{u}_s^T \cdot e)\mathbf{u}_s) = \sum_{s=1}^{n}(\mathbf{u}_s^T \cdot \mathbf{u}_k)(\mathbf{u}_s^T \cdot e)$ in the expression

$$J^{(I-H)\oplus}\mathbf{e} = \sum_{i=1}^{r} \hat{\sigma}_i^{-1}\mathbf{v}_i\mathbf{u}_i^T\mathbf{e}, \quad (38)$$

we can take, by analogy to the SD algorithm, for $\mathbf{e} = \mathbf{u}_s$, the joints variation $\Delta\boldsymbol{\theta}^s$ used by SD as:

$$J^{(I-H)\oplus}\mathbf{u}_s = \hat{\sigma}_s^{-1}\mathbf{v}_s \Rightarrow \Delta\boldsymbol{\theta}^s = \hat{\sigma}_s^{-1}J\mathbf{v}_s \quad (39)$$

which has an effect on the jth joint of:

$$\Delta\theta_j^s = \hat{\sigma}_s^{-1}J^j v_{j,s}, \quad (40)$$

where $v_{j,s}$ is the jth position on the sth column of matrix V, and J^j is the jth column of matrix J.

Therefore, adding the norms for all joints we get the bound M_s as defined in [32]:

$$\sum_{j=1}^{m}|\Delta\theta_j^s| \leq \hat{\sigma}_s^{-1}\sum_{j=1}^{m}|v_{j,s}|\|J^j\| = M_s. \quad (41)$$

This M_s is a bound on the position change gain in the task space generated by the error-dependent part of the algorithm, for each component of the error, and thus with it we can set, for each $s = 1..n$, the maximum joints change γ_{max}:

$$\gamma_s = min(1, 1/M_s)\gamma_{max} \quad (42)$$

To then proceed exactly as in SD:

We will first compute the joints change for each error component (m-dimensional vector):

$$w_s = \hat{\sigma}_s^{-1}\mathbf{v}_s\left(\mathbf{u}_s^T \cdot \mathbf{e}\right), \quad (43)$$

and we will bound this variation with the γ_s obtained in (42):

$$\Delta q_s = \begin{cases} 1 & \text{if } \|w_s\| < \gamma_s \\ \frac{ws}{\|ws\|}\gamma_s & \text{if } \|w_s\| \geq \gamma_s \end{cases} \quad (44)$$

Now, differing from SD algorithm, we have to add the non error-dependent part of the algorithm to the sum of each component :

$$\Delta\hat{\boldsymbol{\theta}} = (I - J^{(I-H)\oplus}J)H(-\lambda\boldsymbol{\theta}) + \sum_s \Delta q_s, \quad (45)$$

to finally bound the total joint variation by γ_{max}:

$$\Delta\boldsymbol{\theta} = \begin{cases} 1 & \text{if } \|\Delta\hat{\boldsymbol{\theta}}\| < \gamma_{max} \\ \frac{\Delta\hat{\boldsymbol{\theta}}}{\|\Delta\hat{\boldsymbol{\theta}}\|}\gamma_{max} & \text{if } \|\Delta\hat{\boldsymbol{\theta}}\| \geq \gamma_{max} \end{cases} \quad (46)$$

In this way, we ensure that $\Delta\boldsymbol{\theta}$ is bounded, respects joint limits, and it is sufficiently well-conditioned.

The described joint limit-concerned methods have also been compared on an example trajectory in the 4R manipulator in order to check which ones respect joint limits and which do not, as displayed in Fig. 4. In all cases the joint limits are sometimes surpassed, as the push-to-center value action is done *a posteriori*, but the following iteration pushes this joint value back to its feasible interval. As expected, neither JW, nor GP or TA are capable of succesfully avoiding joint limits. On the other hand, JC is uneffective because it tends to block joints, losing degrees of freedom.

Furthermore, CTP, which continuously activates/deactivates joints, is not capable of converging nor of mantaining a smooth trajectory. Our proposal solves these problems and significantly improves performance.

VIII. EXPERIMENTATION

As a benchmark to test both the reviewed and proposed algorithms, all of them have been implemented in *Matlab* and C++ (using a ROS library) in a 7-dof redundant WAM robot arm (with Denavit-Hartenberg parameters as shown in Table IV) and their performance has been tested as global IK solvers. To do so, 1000 random feasible initial and target positions have been generated, using a uniform probability distribution for all joints within their limits, and mapped into a cartesian position with the forward kinematics function.

TABLE IV
DENAVITT-HARTENBERG STANDARD PARAMETERS FOR THE WAM ROBOT ARM, WHERE $d_3 = 0.55$, $d_5 = 0.3$ AND $d_7 = 0.06$.

link	a_i	α_i	d_i	θ_i	θ_i^{min}	θ_i^{max}
1	0	$-\pi/2$	0	θ_1	-2.6	2.6
2	0	$\pi/2$	0	θ_2	-2.0	2.0
3	a	$-\pi/2$	d_3	θ_3	-2.8	2.8
4	-a	$\pi/2$	0	θ_4	-0.9	3.1
5	0	$-\pi/2$	d_5	θ_5	-4.8	1.3
6	0	$\pi/2$	0	θ_6	-1.6	1.6
7	0	0	d_7	θ_7	-2.2	2.2

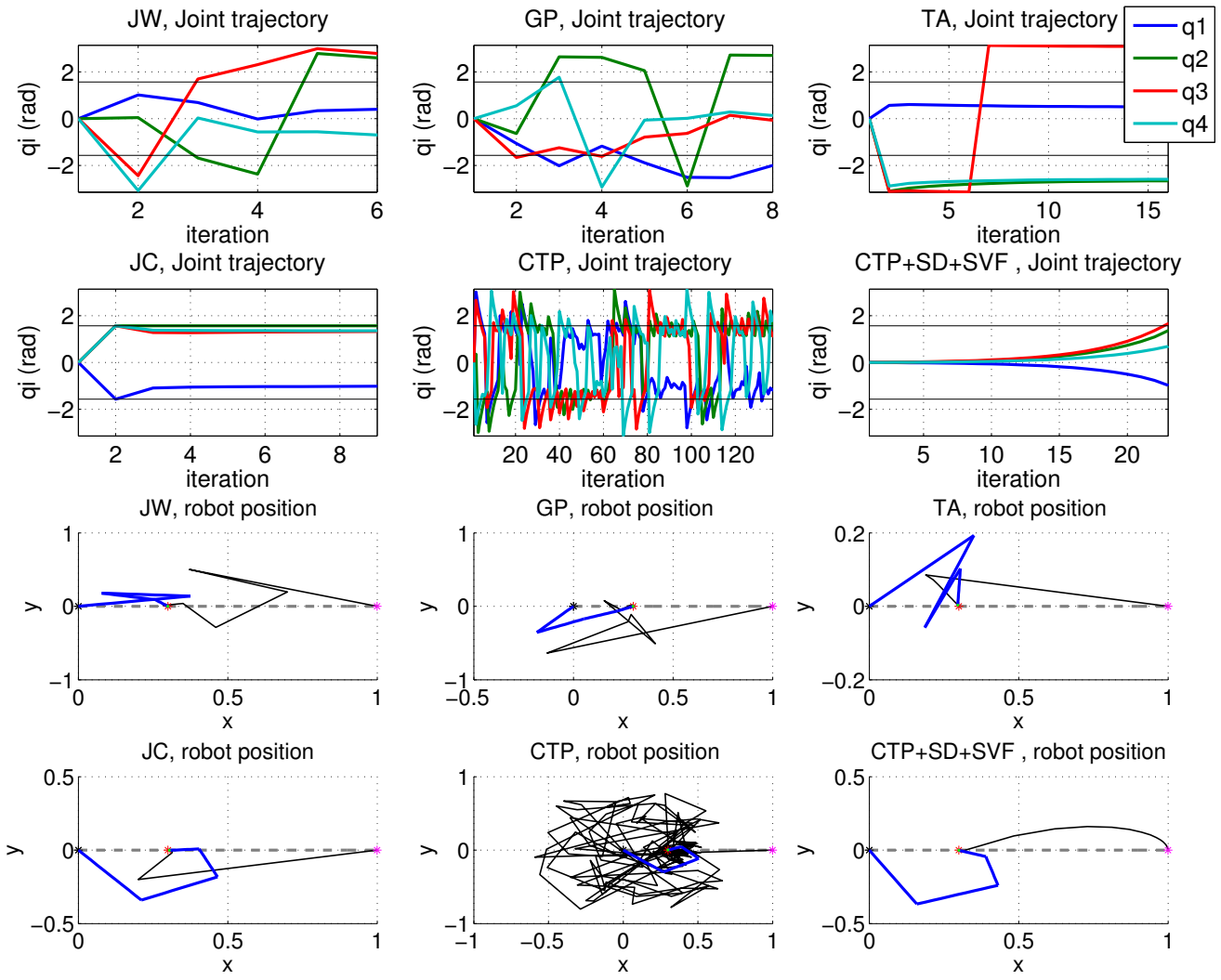

Fig. 4. Behaviour of some CLIK algorithms (see Table II for notation) applied to a 4R planar manipulator. The robot frame evolves from a dashed light color start position (horizontal) to a darker ending one, with its end-effector's trajectory plotted in black (thin line). Note the different scales for the x-axis. In the CTP algorithms, the joints have been allowed to cross joint limits to show how they are capable of moving back to a centered joint value.

Note that, for the algorithms taking into account joint limits, this sampling as such was not adequate to assess their performance, as the arm has several assembly modes (different arm configurations for the same end-effector position), which added to the joint limits restriction, could make the desired configuration sometimes impossible to reach with the same assembly mode as the initial configuration. This fact has an impact on CLIK algorithms by sometimes requiring very different initial and final joint configurations, leading to moving the arm in unintiutive ways so as to have feasible solutions. In these cases, a path planner is needed, which is out of the scope of this study.

To avoid the mentioned situations, in the experiments involving algorithms taking into account joint limits (those in Table VI), we generated the samples by obtaining an initial joint position θ_0 and a final one θ_F. If $|\theta_0^i - \theta_F^i| < A, \forall i = 1..7$, for a given constant value A, the sample was accepted. This ensures that there exists a solution of the desired position in the same assembly mode or another assembly mode close to it. This A parameter will be used later to check the sensitivity of the different algorithms to the initial/ending position distance, as shown in Fig. 5.

TABLE V
BEHAVIOUR OF THE STUDIED METHODS NOT CONCERNED WITH JOINT LIMITS FOR A SAMPLE OF 1000 RANDOM INITIAL AND END POSITIONS FOR THE WAM ROBOT ARM. NOTATION AS IN TABLE II.

Method	% sol.	$\overline{t_{sol}}$ (ms)	$\overline{e_{nosol}}$	$\overline{it_{sol}}$
JP	100.0	42.6	-	12.2
JT	40.70	504.8	0.302	148.7
SD - $\gamma_{max} = 0.5$	98.4	154.3	0.042	43.5
JD - $\lambda = 0.005$	100.0	39.4	-	11.6
JF - $\lambda_{max} = 4\lambda$	100.0	38.6	-	11.2
ED	100.0	36.9	-	10.6
IED - $\Omega = 0.01 I_m$	100.0	38.7	-	11.1
SVF - $nu = 10$, $\sigma_0 = 0.01$	**100.0**	**37.1**	**-**	**10.7**
SVF+ED	**100.0**	**35.8**	**-**	**10.3**
SVF+SD	**99.7**	**145.2**	**0.041**	**41.1**

The results of a *Matlab* simulation can be seen in Tables V and VI, where the columns in Table V show the percentage of solutions found, the average computation time ($\bar{t}_{sol}$), the

TABLE VI
BEHAVIOUR OF THE STUDIED METHODS TAKING INTO ACCOUNT JOINT LIMITS FOR A SAMPLE OF 1000 RANDOM INITIAL AND END POSITIONS FOR THE WAM ROBOT ARM. NOTATION AS IN TABLE II AND PARAMETER A=1.0 RAD.

Method	% sol.	% sol(JL)	$\overline{t}_{sol}$ (ms)	$\overline{e}_{nosol}$	$\overline{it}_{sol}$
JW - as in [53]	100.0	45.3	37.5	-	9.6
GP - $\mu = 0.2$	100.0	34.7	43.0	-	11.0
TA - as in [55]	100.0	84.6	147.3	-	28.3
JC - H as in [39]	73.6	53.5	73.1	0.826	18.9
TP - H as in [39]	33.6	33.6	48.9	1.005	12.0
CTP - H as in [39]	83.7	83.7	366.8	0.381	21.8
CTP+SVF	**86.6**	**86.6**	**300.4**	**0.401**	**48.2**
CTP+SD	**97.1**	**97.1**	**660.0**	**0.867**	**26.2**
CTP+SD+SVF	**98.3**	**98.3**	**568.7**	**0.719**	**22.6**

remaining error when convergence was not achieved ($\overline{e}_{nosol}$) with the position-orientation metric in Section II-A), and the average number of iterations needed to find a solution ($\overline{it}_{sol}$). In Table VI an additional column (second) shows the percentages of solutions where joint limits were respected. As the algorithms in Table V do not consider joint limits, we only show this information in the second Table. The performance of the reviewed state-of-the-art methods is compared with our proposals, which are highlighted in bold face in the Tables. Besides the filtering enhancement SVF in different combinations, we have used the CTP algorithm in (34) together with the SD, as proposed in Section VII, and we have also combined them with SVF to compare results. With these data, we can draw the following conclusions:

- Low convergence ratio of JT. This is due to chattering when activating/deactivating joints, as commented before. The remaining algorithms not considering joint limits always converge, except for SD, due to the limited number of iterations.
- Using SVF improves the speed of JP, requiring less iterations on average and, combined with ED, performs much faster than the rest of methods. Nevertheless, we also recommend using SD+SVF, because this guarantees the steps will always be smooth, even in the case of a singular goal position.
- JW, TA and GP do not respect joint limits. This is due to the fact that avoiding limits is not treated as a priority, thus zero-error positioning prevails.
- All the algorithms not fully respecting joint limits have higher convergence ratios, since they can cross regions with unfeasible joint values to reach the goal.
- The TP algorithm does not converge most of the times. This is due to the discontinuity commented before, causing large gains which then block the joints.
- CTP algorithms do not always converge, but when they do, the solution respects joint limits. This shows that using these limits as a primary task is a successful strategy. Adding SD improves the convergence ratio, and it also reduces computation time. Overall, CTP computation times are large, due in part to the non optimality of *Matlab* for its calculation. This could be reduced by finding an approximate value of the continuous pseudoinverse.

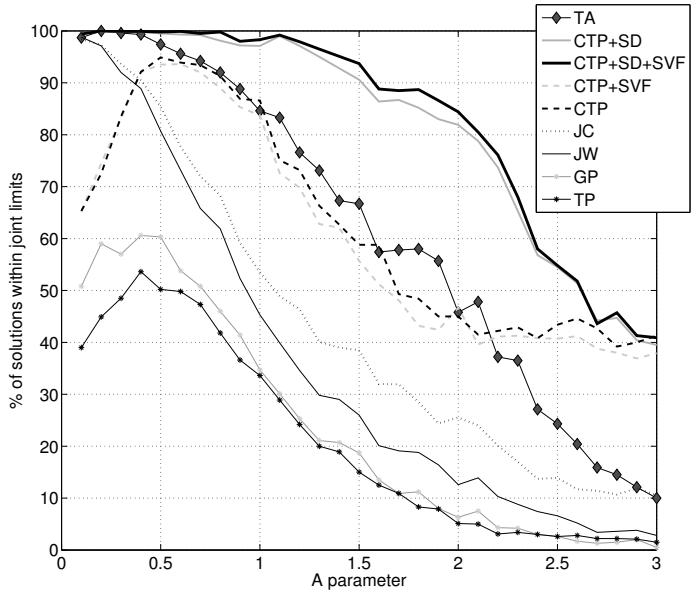

Fig. 5. Percentage of solutions found within joint limits for different sampling thresholds A, an indicator of the distance between initial joint position and goal joint positions.

The non-convergence cases of our CTP proposed algorithms are due to algorithmic singularities. These happen when, close to a joint limit, the push-to-center value of the joint limit avoidance task compensates the position tracking error. This is like saying that the algorithm walks into a dead end in joint space. To avoid this convergence problem, some works in literature try to find a better initial point through a biased random sampling over other possible starting configurations, or it is also possible to use a path planning algorithm in order not to get stuck.

To further test our proposals, we repeated the experiment in Table VI with different values of the parameter A defined before. Figure 5 shows the percentage of solutions found within joint limits for each algorithm and each value of A. There we can see that SVF improves the performance of the CTP algorithms, and that the combination of CTP, SD and SVF outperforms the other algorithms in literature. In addition, we see that some algorithms without the selective damping have worse results for very small values of A; this is because they have large gains, as explained along this work, and they sometimes surpass the goal.

IX. CONCLUSIONS

In this work, the most relevant CLIK algorithms for redundant robots have been compared. Special attention has been paid to three issues: The JP algorithm may have very large gains along certain directions, and reducing the global gain is not the best solution. In fact, having such large gains is similar to a random positioning in joint space, whose topology is equivalent to an *m-torus* mapped into the workspace, and the high convergence rate of JP in Table V is due to the fact that large steps are taken until the end-effector reaches a position from which the goal is achievable. The JT algorithm does not have such problem, but in some cases presents so much

chattering that makes its computational cost grow. Since SD efficiently solves this problem, it is recommended to use such damping in most algorithms.

Looking for good matrix conditioning, we have compared the capability of the different algorithms to avoid amplifying the numerical error in robot positioning. The outcome has been that most of the existing methods do not perform well near a singularity. Filtering or damping the Jacobian matrix improves this conditioning, but with no numerical guarantees. On the other hand, using the current error as a damping factor reduces the condition number, but when close to the goal, the ED algorithm (or its improved version, IED) behaves similarly to filtering or damping.

Therefore, we proposed a new filtering method based on a continuous modification of the singular values of the Jacobian, which we named SVF. We proved theoretically and in practice that our proposal improves the existing methods for numerically filtering or damping the Jacobian pseudoinverse of a matrix. We have also shown that this does not entail a significant growth in the computational cost. With this filtering, the Jacobian matrix can be assumed to be always full rank, without generating much additional error on the algorithms, thus the pseudoinverse operator would not have discontinuities due to a rank change in the Jacobian matrix. This can be used in all control-based methods to improve their performance.

We have also presented a review of first-order approaches to achieve secondary tasks. In particular, we have tried to devise an algorithm that efficiently avoids joint limits. Through experimentation, we have seen that the only way to ensure avoiding such limits is to treat them as the main priority task by adding an activation matrix on this main task. This then results in discontinuities of the pseudoinverse operator when activating or deactivating a joint push-to-center value to avoid a joint limit. However, this shortcoming is solved with the continuous pseudoinverse (CTP) which, when combined with SD and our proposed filtering (SVF), ensures controlled steps and a full-rank behaviour of the Jacobian.

As it is well-known, and it showed up in our testing with a redundant robot such as the Barrett WAM arm, CLIK methods used as global IK solvers do not always reach the goal. This is because of algorithmic singularities, i.e., when the main task and the secondary task compensate one another and the computed joint variation becomes zero. To solve this issue, it is recommended to add a path planner to the algorithm or a randomized initial value to iterate upon, so as to prevent the robot getting stuck in such a situation. Despite the convenience of a path planner to obtain smoother joint changes, our last proposal (CTP+SD+SVF) performs well without such a planner, keeping over a 90 percent success rate with parameter $A \leq 1.5rad$ in Fig. 5, it being the best among all the tested algorithms. Additional experimentation using such algorithms in a trajectory-tracking experiment can be found in http://www.iri.upc.edu/groups/perception/inverseKinematics/ .

As future work we will try to speed up the algorithms, specially the CTP cases where several sparse matrix pseudoinverses must be computed, in order to improve the results as regards to computational time.

REFERENCES

[1] S. Sasaki, "Feasibility Studies of Kinematic Problems in the Case of a Class of Redundant Manipulators." *Robotica*, vol 13, pp 233-241, 1995.

[2] G. K. Singh, J. Claassens, "An analytical Solution for the Inverse Kinematics of a Redundant 7-dof Manipulator with Link Offsets". *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp 2976-2982, 2010.

[3] X. Ding and C. Fang, "A Novel Method of Motion Planning for an Anthropomorphic Arm Based on Movement Primitives".*IEEE/ASME Transactions on Mechatronics*, vol 18, no 2, pp 624-636, 2013.

[4] V. Ruiz de Angulo and C. Torras. "Learning inverse kinematics: Reduced sampling through decomposition into virtual robots". *IEEE Transactions on Systems, Man and Cybernetics*, part B, vol 38 no 6, pp 1571-1577, 2008.

[5] S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras and R. Dillman. "General robot kinematics decomposition without intermediate markers". *IEEE Transactions on Neural Networks and Learning Systems*, vol 23 no 4, pp 620-630, 2012.

[6] S. Chiaverini, G. Oriolo, I.D. Walker, "Kinematically Redundant Manipulators." *Springer Handbook of Robotics* part B, chapter 11, 2008.

[7] K. Ki-Young, "A Novel Surgical Manipulator with Workspace-Conversion Ability for Telesurgery." *IEEE/ASME Trans. on Mechatronics* vol 18, no 1, pp 200-211, 2013.

[8] J. Funda, R.H. Taylor, B. Eldridge, S. Gomory, K.G. Gruben, "Constrained Cartesian Motion Control for Teleoperated Surgical Robots" *IEEE Trans. on Robotics and Automation*, vol 12, no 3, pp. 453-465, 1996.

[9] Inverse Kinematics with KDL. url: http://www.orocos.org/forum/orocos/orocos-users/inverse-kinematics-kdl.

[10] ROS package repository with Barrett WAM/Hand interface. url:http://code.google.com/p/lis-ros-pkg/wiki/README

[11] R. S. Rao, A. Asaithambi, and S. K. Agrawal. "Inverse kinematic solution of robot manipulators using interval analysis." *J. of Mechanical Design*, 120(1): pp 147-150, 1998.

[12] J. M. Porta, Ll. Ros, and F. Thomas. "Inverse kinematic by distance matrix completion." *12th Int. Workshop on Computational Kinematics*, 2005.

[13] V. Ruiz de Angulo and C. Torras. "Self-calibration of a space robot". *IEEE Transactions on Neural Networks*, vol 8, no 4, pp 951-963, 1997.

[14] V. Ruiz de Angulo and C. Torras. "Speeding up the learning of robot kinematics through function decomposition". *IEEE Transactions on Neural Networks*, vol 16, no 4, pp 1504-1512, 2005.

[15] S.F.M. Assal, K. Watanabe, K. Izumi. "Neural Network-Based Kinematic Inversion of Industrial Redundant Robots Using Cooperative Fuzzy Hint for the Joint Limits Avoidance." *IEEE/ASME Trans. on Mechatronics* vol 11, no 5, pp 593-603, 2006.

[16] S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras and R. Dillman. "Rapid learning of humanoid body schemas with kinematic Bézier maps", *9th IEEE-RAS International Conference on Humanoid Robots*, pp. 431-438, 2009.

[17] S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras and R. Dillman. "Kinematic Bézier maps". *IEEE Transactions on Systems, Man and Cybernetics*, part B, vol 42, no 4, pp 1215-1230, 2012.

[18] O. Khatib. "A unified approach for motion and force control of robot manipulators." *Int. Conf. on Robotics and Automation (ICRA)* vol. RA-3, no 1, pp 43-53, 1987.

[19] D.E. Orin and W.W. Schrader. "Efficient computation of the jacobian for robot manipulators." *Int. J. Robot Research*, vol 3, no 4, pp. 66-75, 1984.

[20] A. Ben-Israel, T. Greville. Generalized Inverses. *Springer-Verlag* 2003. ISBN 0-387-00293-6.

[21] D. E. Whitney. "Resolved motion rate control of manipulators and human prostheses." *IEEE Trans. on Man-Machine Systems*, vol 10, pp 47-53, 1969.

[22] W. A. Wolovich and H. Elliott. "A computational technique for inverse kinematics." *23rd IEEE Conf. In Decision and Control*, vol 23, pp 1359-1363, 1984.

[23] Y. Nakamura and H. Hanafusa. "Inverse Knematic Solutions with Singularity Robustness for Robot Manipulator Control." *Journal of Dynamic Systems, Measurement and Control*, vol 108, pp 163-171, 1986.

[24] C. W. Wampler II, "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods." *IEEE Transactions on Systems, Man and Cybernetics*, vol SMC-16, no 1, pp 93-101, 1986.

[25] S. Chiaverini, O. Egeland, and R.K. Kanestrom. "Achieving user-defined accuracy with damped least-squares inverse kinematics." *Fifth Int. Conf. on Advanced Robotics 'Robots in Unstructured Environments', 91 ICAR.*, vol 1, pp 672-677, 1991.
[26] 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*, pp 123-134, 1994.
[27] B. Siciliano. "A closed-loop inverse kinematic scheme for on-line joint-based robot control." *Robotica*, no 8, pp 231-243, 1990.
[28] H. Das, J.E. Slotine, T.B. Sheridan. "Inverse kinematic algorithms for redundant systems." *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol 1, pp 43-48, 1988.
[29] G. Antonelli. "Stability Analysis for Prioritized Closed-Loop Inverse Kinematic Algorithms for Redundant Robotic Systems." *IEEE Trans. on Robotics* vol 25, no 5, pp 5892-5897, 2009.
[30] Z Jiang. "A converse lyapunov theorem for discrete-time systems with disturbances." *Systems & Control Letters*, vol 45, pp 49-58, 2002.
[31] P. Falco and C. Natale. "On the stability of closed-loop inverse kinematics algorithms for redundant robots." *IEEE Trans. on Robotics*, vol 27, no 4, pp 780-784, 2011.
[32] S. R. Buss, J.-S. Kim. "Selectively Damped Least-Squares for Inverse Kinematics." *J. of Graphics Tools*, vol 10, pp 37-49, 2004.
[33] J. Baillieul. "Kinematic programming alternatives for redundant manipulators." *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol 2, pp 722-728, 1985.
[34] T. Yoshikawa. "Dynamic manipulability of robot manipulators." *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol 2, pp 1033-1038, 1985.
[35] T. Yoshikawa. "Analysis and Control of Robot Manipulators with Redundancy." *First Int. Symposium Robotics Research*, pp 735-748, 1984.
[36] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. "Modelling, Planning and Control." *Advanced Textbooks in Control and Signal Processing. Springer.* 2009.
[37] A. S. Deo and I.A. Walker. "Minimum Effort Inverse Kinematics for Redundant Manipulators." *IEEE Trans. on Robotics and Automation*, vol 13, no 5, pp 767-775, 1997.
[38] Y. Nakamura, "Advanced Robotics: Redundancy and Optimization", *Addison-Wesley Pub. Co.*, 1991.
[39] N. Mansard, O. Khatib, A. Kheddar, "A unified Approach to Integrate Unilateral Constraints in the Stack of Tasks." *IEEE Trans. on Robotics*, vol 25, no 3, pp 670-685, 2009.
[40] J. Duffy, "The Fallacy of Modern Hybrid Control Theory that is Based on 'Orthogonal Complements' of twist and Wrench Spaces." *Journal of Robotic Systems*, vol 7, is 2, pp 139-144, 1999.
[41] K.L. Doty, C. Melchiorri, E. M. Schwartz and C. Bonivento, "Robot Manipulability", *IEEE Trans. on Robotics and Automation*, vol 11, no 3, 1995.
[42] F. Ranjbaran, J. Angeles, A. Kecskemethy. "On the Kinematic Conditioning of Robotic Manipulators" *IEEE Int. Conf. on Robotics and Automation (ICRA)* vol 4, pp 3167-3172, 1996.
[43] F. Aghili, "Adaptive Control of Manipulators Forming Closed Kinematic Chain With Inaccurate Kinematic Model" *IEEE/ASME Transactions on Mechatronics*, vol 18, no 5, pp 1544-1554, 2013.
[44] P. In-Won, L. Bum-Joo, Ch. Se-Hyoung, H. Young-Dae, K. Jong-Hwan, "Laser-Based Kinematic Calibration of Robot Manipulator Using Differential Kinematics" *IEEE/ASME Trans. on Mechatronics* vol 17, no 6, pp 1059-1067, 2012.
[45] C.A. Klein, B.E. Blaho, "Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators." *Int. J. of Robotics Research*, vol 6, no 2, pp 72-83, 1987.
[46] A.K. Cline, C.B. Moler, G.W.Steward and J.H. Wilkinson. "An Estimate for the Condition Number of a Matrix." *SIAM J. on Numerical Analysis*, vol 16, no 2, pp 368-375, 1979.
[47] S. R. Buss. "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods." *Unpublished*, http://math.ucsd.edu/ sbuss/ResearchWeb, 2004.
[48] S. K. Chan, P. D. Lawrence, "General Inverse Kinematics with the Error Damped Pseudoinverse." *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol.2, pp 834-839, 1988.
[49] T. Sugihara, "Solvability-Unconcerned Inverse Kinematics by the Levenberg-Marquardt Method." *IEEE Trans. on Robotics*, vol 27, no 5, pp 984-991, oct 2011.
[50] Y. Nakamura, H. Hanafusa, T. Yoshikawa, "Task-priority based redundancy control of robot manipulators." *Int. J. of Robotics Research*, vol 6, no 2, pp 3-15, 1987.
[51] S. Chiaverini. "Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators." *IEEE Trans. on Robotics and Automation*, vol 13, no 3, pp 398-410, 1997.
[52] L. Sciavicco, B. Siciliano. "A solution to the inverse kinematic problem for redundant manipulators." *IEEE J. of Robotics and Automation* vol 4, pp 403-410, 1988.
[53] T. Fung Chan, R. V. Dubey. "A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators." *IEEE Trans. on Robotics and Automation* vol 11, no2, pp 286 - 292, 1995.
[54] D. Raunhardt, R. Boulic. "Progressive Clamping." *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp 4414-4419, 2007.
[55] J. Xiang, C. Zhong, Wei Wei. "General Weighted Least-Norm Control for Redundant Manipulators." *IEEE Trans. on Robotics*, vol 26, no 4, pp 660-669, 2010.
[56] H. Zghal, R.V. Dubey, J.A. Euler, "Efficient gradient projection optimization for manipulators with multiple degrees of redundancy." *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)* vol 2, pp 1006-1011, 1990.
[57] F. Flacco, A. De Lucca, O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints." *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3970-3977, 2012.
[58] N. Mansard, A. Remazeilles, and F. Chaumette, "Continuity of varying-feature-set control laws." *IRISA Technical report*, 2007. url: *ftp://ftp.irisa.fr/techreports/2007/PI-1864.pdf*
[59] N. Mansard, A. Remazeilles, F. Chaumette, "Continuity of Varying-Feature-Set Control Laws." *IEEE Trans. on Automatic Control*, vol 54, no 11, pp 2493-2505, 2009.
[60] A. Colomé and C. Torras, "Redundant inverse kinematics: Experimental comparative review and two enhancements." *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5333-5340, 2012.

Adrià Colomé (http://www.iri.upc.edu/staff/acolome) is a PhD student at the Institut de Robòtica I Informàtica Industrial (IRI) in Barcelona. He obtained two BSc degrees in Mathematics and Industrial Engineering in 2009, and a MSc degree in Automatic Control, Robotics and Computer Vision in 2011, all from the Technical University of Catalonia (UPC). Since then, he has been working on serial redundant robots, analysing their kinematics, control, and how to represent and learn trajectories with them.

Carme Torras (http://www.iri.upc.edu/people/torras) is Research Professor at the Spanish Scientific Research Council (CSIC). She received M.Sc. degrees in Mathematics and Computer Science from the Universitat de Barcelona and the University of Massachusetts, Amherst, respectively, and a Ph.D. degree in Computer Science from the Technical University of Catalonia (UPC). Prof. Torras has published five books and about two hundred papers in the areas of robot kinematics, neurocomputing, computer vision and geometric reasoning. She has been local project leader of several European projects, among which the FP6 IP project "Perception, Action and COgnition through Learning of Object-Action Complexes" (PACO-PLUS), and the FP7 STREP projects "GARdeNIng with a Cognitive System" (GARNICS) and "Intelligent observation and execution of Actions and manipulations" (IntellAct). She was awarded the Narcís Monturiol Medal of the Generalitat de Catalunya in 2000, and she became ECCAI Fellow in 2007, member of Academia Europaea in 2010, and member of the Royal Academy of Sciences and Arts of Barcelona in 2013. Prof. Torras is Editor of the IEEE Transactions on Robotics.