Redundant Inverse Kinematics: Experimental Comparative Review and Two Enhancements

Adrià Colomé and Carme Torras

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 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 to filter the singular values of the Jacobian matrix before calculating its pseudoinverse in order to obtain a more numerically robust result. 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. For 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. Nevertheless, with the rise of redundancies in robots, analytical solutions become harder to obtain [1] [2] and thus again alternatives need to be explored [3] in order to benefit from the additional degrees of freedom.

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 [4] could sometimes fail due to joint limit vulnerations. We tried other open-source IK algorithms [5], 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 [6], distance based methods [7], or even neural networks [8], probably the most popular way is to use closed-loop algorithms. In these Closed Loop Inverse Kinematics (CLIK) algorithms, a first-order Jacobian matrix [9] [10] of the robot is computed, which maps joint velocities into task space velocities, and inverted to map the error into a joint state update which 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 $\Delta\boldsymbol{\theta}^k$:

$$\Delta\boldsymbol{\theta}^k = \alpha J^\star(\mathbf{x}_d - f(\boldsymbol{\theta}^k)) = \alpha J^\star \mathbf{e}, \quad (1)$$

where α is a gain, $J^\star$ is an inverse matrix of the Jacobian, $f(\cdot)$ is the forward kinematics function, $\mathbf{x}_d$ the desired position and $\mathbf{e}$ the positioning error. The first attempts to close the IK loop used the Moore-Penrose pseudoinverse [11] of the Jacobian matrix [12] to invert the differential kinematics equation of the robot. In other works, the Jacobian transpose was used [13], which is faster to compute. These methods 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 [14] [15], reducing this condition number, but not always reducing large joints variations. Some attempts also use second-order derivatives of motion, i.e.: calculating the Hessian matrix of the forward kinematics [16], although this requires much more computation time.

Using first-order derivative methods of the robot's motion also has the drawback that, depending on the goal position, an algorithm can get stuck at an *algorithmic singularity*, a point where the error $\mathbf{e}$ belongs to the kernel of the inverted Jacobian, or in a multiple-task algorithm, a secondary task joint variation may take the contrary value of the primary task, thus the total computed joint variation being $\Delta\boldsymbol{\theta} \simeq 0$.

In a continuous time assumption, the convergence of closed-loop methods can be demonstrated in terms of Lyapunov theory [17] [18]. Nevertheless, these computations have a gain, and the smaller this gain is, the more iterations needed to converge. Thus this gain is lower bounded by the computation capability of a processor, and convergence cannot always be assured by means of Lyapunov theory. Although there exist discrete-time versions of it [19], 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. [20] finds an upper bound of the gain α that guarantees convergence, but restricting the operational space to a subset where the Jacobian is full-rank with bounded singular values, so its application is not general. Nevertheless, this work points out the relevance of the initial error dependency for these methods to converge,

This work is partially funded by the CogSys STREP IntellAct project (FP7-ICT-269959), the CogSys STREP Project GARNICS (FP7-ICT-247947) and the Catalan Research Commission through grant 2009SGR155.

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

showing that they are more robust when used locally. In general, a smaller gain improves convergence rate on one hand, while slows the algorithm on the other.

The main advantadge of redundancy is to be able to perform secondary tasks and/or to choose which solution suits us best. To this purpose, an optimization criterion 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 [21], in which rows are added to the Jacobian, have been used. Among the existing criteria for optimization, the *manipulability* measure [22] [23] is often used. Other criteria such as collision avoidance [24] (by setting a minimum distance to a certain object), minimum effort kinematics [25] or structural stiffness are also used [26]. 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 concerning numerical error propagation, which is sometimes forgotten when analysing these algorithms. Focusing on solving the IK with feasible joint values, two enhancements of the existing literature 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 [27] with a controlled step size [28] to smoothen the motion of the robot. All the analysed algorithms, as well as the proposed enhancements, have been implemented on a Barrett's WAM arm and tested both in simulation and in real experimentation.

II. Preliminaries

Along this work, the notation in Table I will be used.

TABLE I
NOTATION

J	Geometric Jacobian
$J^\dagger$	Jacobian pseudoinverse
$\boldsymbol{\theta} = [\theta_1, ..., \theta_m]$, $\Delta\boldsymbol{\theta}$	Joint state and variation
$\mathbf{x}$	Cartesian robot position (*n-dim)*
$\mathbf{x}_d$	Desired robot position (*n-dim)*
$\mathbf{e} = \mathbf{x}_d - \mathbf{x}$	Position error of the robot
$\kappa(\cdot)$	Condition number of a matrix
$f(\cdot)$	Forward kinematics function
$\sigma_1, ..., \sigma_n$	Jacobian singular values
m	Number of joints
n	Task space dimension

For the positioning error representation as a n-dimentional generalised coordinate vector, as it compares a position error (distance) vs an orientation error (angular), it is often taken the equivalence of $2rad = 1m$ (see [24], pp 137-140). Nevertheless, different metrics can be used to improve the performance of the algorithms [29].

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, and therefore $\delta\mathbf{e}$ on the position error $\mathbf{e}$ (difference between target and current positions). Then, it is fundamental to avoid amplifying this error when computing $\Delta\boldsymbol{\theta}$. To this purpose, the relative error $\delta\boldsymbol{\theta}$ on $\Delta\boldsymbol{\theta}$ coming from the error $\delta\mathbf{e}$ on $\mathbf{e}$ can be computed using the condition number of $J^\star$ [30] [31]:

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

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

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

III. Review of CLIK algorithms

In a redundant manipulator, the Moore-Penrose pseudoinverse is often used in (1), as it is a generalised inverse which is still well-defined when a matrix is rank-deficient. If $J = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ is the Singular Value Decomposition (SVD) of the Jacobian matrix, then its pseudoinverse is:

$$J^\dagger = U\Sigma V^T = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{u}_i\mathbf{v}_i^T, \qquad (2)$$

where $\mathbf{u}_i$, $\mathbf{v}_i$ are the columns of U and V. In (2) we can see there is a discontinuity on the pseudoinverse operator around a singular configuration of a robot (a singular value becomes zero). This discontinuity means the Jacobian Pseudoinverse (JP) algorithm gives large $\Delta\theta$ values with high conditioning. Using the Jacobian Transpose (JT) we can avoid these gains [32], but we do not avoid the conditioning issue. In addition, JT can add chattering around the solution. In fact, for the case of the JP and JT, the condition number is $\kappa(J^\star) = \frac{\sigma_1}{\sigma_n}$ (assuming $\sigma_1 > ... > \sigma_n \geq 0$) and tends to infinity as $\sigma_n \to 0$, thus loosing all the numerical precision in the direction associated with σ_n.

To avoid large gains, reducing the global gain is not a truly effective strategy, as we will be 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 [28] it is proposed a Selective Damping (SD) of the gain on the joints variations derived from each task space error component, which effectively solves the gain issues, but does not solve singularity issues as the loss of rank and algorithmic singularities.

There are some ways of trying to avoid these discontinuities on the singularities, such as Jacobian Damping (JD) which consists of adding a small diagonal term λ when computing the pseudoinverse matrix, or Filtering (JF) the Jacobian matrix [14], in which this λ depends on how close to a singularity the robot is. This modification removes the mentioned discontinuity, but its effect on the condition number or gains may not be strong enough around a singularity. The Error Damping (ED) [33] strategy is to use the norm of the current error to damp the pseudoinverse. This reduces large gains when away from the goal, but if, for instance, the goal is close to a singularity, the error is not a good damping factor. For this reason, the ED can be improved by adding a term $\Omega = diag(\omega_1, ...\omega_n)$ [34]. But in the mentioned case, this would be equivalent to a JD algorithm in the neighbourhood of a singular goal position.

In fact, in the Appendix we show that adding this diagonal term in damping algorithms is not completely robust in terms of conditioning, and there is a tradeoff between the region where the condition number is bounded and the upper bound of the conditioning in this region.

IV. SINGULAR VALUE FILTERING (SVF)

We propose a new way of filtering the Jacobian matrix, which consists in modifying the Jacobian matrix' singular values to obtain an alternative pseudoinverse that is always full-rank and whose condition number is bounded. To this purpose, if we take the SVD of J:

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

then we define

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

where

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

is our proposed filtering rational function with:

σ_0 the minimum value we want to impose to the singular values of J.ν a shape factor.And then we can compute (assuming $\sigma_i > \sigma_{i+1}$, $\forall i$)

$$\hat{J}^{\dagger} = \sum_{i=1}^{n} \frac{1}{h_{\nu,\sigma_0}(\sigma_i)} \mathbf{v}_i \mathbf{u}_i^T, \qquad (4)$$

to use it as the pseudoinverse. Then, 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.
- $\lim_{\sigma\to 0} h_{\nu,\sigma_0}(\sigma) = \sigma_0, \forall\nu$, so 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 with 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 if ν and σ_0 are defined verifying $\nu > \sigma_0$ and $2 > \nu\sigma_0$, which are not very restrictive conditions. On the other hand, the greater ν is, the smaller the value $|h_{\nu,\sigma_0} - \nu|$. This gives us hints on which value to use for ν. In the experimentation, we have taken $\nu = 10$. Monotonicity guarantees that the condition number of the pseudoinverse (4) is always:

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

so we have:

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

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

To sum up, we have that $\hat{J}$ 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 looses rank as the singular values are strictly positive. In Table II we can see the equations defining all the above-mentioned algorithms.

Another advantage of this method can be seen in Fig. 1, where we plot the condition number of different methods in the case of a 4R planar manipulator moving towards a singularity, for a damping factor of $\lambda = 10^{-3}$, and allowing a maximum damping factor on the filtering algorithm (variable damping factor) of $\lambda_{max} = 5\lambda$. As we have already commented, the JP algorithm's condition number tends to infinity, and so does the JT. The JD and JF algorithms perform better, with reduced conditioning, even bounded out of a small interval. Nevertheless, the loss of precision is high. On the other hand, the error-damped methods have very low condition number, but it grows fast as the robot reaches the goal. The proposed method, with $\sigma_0 = 0.005$ and $\nu = 10$, keeps its condition number stable. Our proposal presents the best bounded conditioning, although it can still have considerable gains on their iterations. This can be solved by combining it with the SD.

Note that, as the least singular value approaches very small values compared with the damping factors, the condition number exponentially grows towards infinity, as commented in the Appendix.

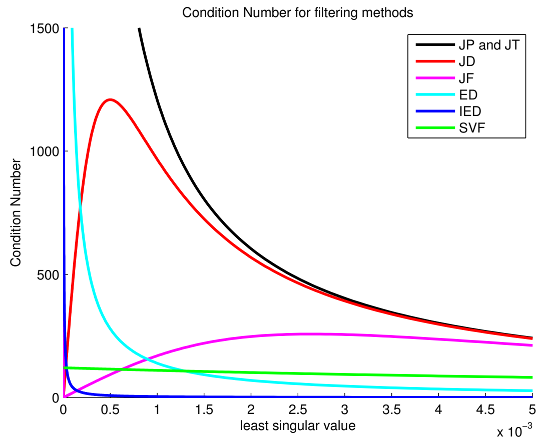

Fig. 1. Condition Number for different methods on a 4R planar robot approaching a desired singular position.

V. MULTIPLE TASKS

Usually, when computing the IK of a robot, it is a good idea not only to compute a solution of the inverse

TABLE II
REVIEWED IK METHODS

Name	Abbreviation	Equation ($\Delta\boldsymbol{\theta} =$)	References
Jacobian Pseudoinverse	JP	$J^{\dagger}\mathbf{e} = \sum_{i=1}^{n} \frac{1}{\sigma_i}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[12]
Jacobian Transpose	JT	$J^{T}\mathbf{e} = \sum_{i=1}^{n} \sigma_i\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[13]
Selective Damping	SD	$J^{\dagger SD}\mathbf{e} = \sum_{i=1}^{n} s(\sigma_i, i, J, \gamma_{max})\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[28]
Damped Jacobian	JD	$J^{\dagger D}\mathbf{e} = J^T(JJ^T + \lambda^2 I)^{-1}\mathbf{e} = \sum_{i=1}^{n} \frac{\sigma_i}{\sigma_i^2+\lambda^2}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[15]
Filtered Jacobian	JF	$J^{\dagger F}\mathbf{e} = J^T(JJ^T + \lambda^2\mathbf{u}_n\mathbf{u}_n^T)^{-1}\mathbf{e} = \sum_{i=1}^{n-1} \frac{1}{\sigma_i}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right) + \frac{\sigma_n}{\sigma_n^2+\lambda^2}\mathbf{v}_n \left(\mathbf{u}_n^T \cdot \mathbf{e}\right)$	[14]
Error Damping	ED	$J^{\dagger ED}\mathbf{e} = J^T\left(JJ^T + EI_n\right)^{-1}\mathbf{e} = \sum_{j=1}^{n} \frac{\sigma_i}{\sigma_i^2+E}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[33]
Improved Error Damping	IED	$J^{\dagger IED}\mathbf{e} = J^T\left(JJ^T + EI_n + \Omega\right)^{-1}\mathbf{e} = \sum_{j=1}^{n} \frac{\sigma_i}{\sigma_i^2+E+\omega_i}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	[34]
Singular Value Filtering	SVF	$\hat{J}^{\dagger}\mathbf{e} = \sum_{i=1}^{n} \frac{\sigma^2+\nu\sigma+2}{\sigma^3+\nu\sigma^2+2\sigma+2\sigma_0}\mathbf{v}_i \left(\mathbf{u}_i^T \cdot \mathbf{e}\right)$	proposed

kinematics, but also the solution which behaves best for a certain criterion. Even more with redundant robots, where the number of solutions may be infinite. Also, joints usually have limits on their prismatic/rotational position, and a solution to the IK with a joint value outside its limits is not a feasible solution, so one of the most important properties of a good IK solution is that it lies inside these limits. To this purpose, the redundancies of a robot are often used to satisfy such constraint goal.

A way to bias the solution given by the pseudoinverse operator is to use a Jacobian Weighting (JW) algorithm, in which a matrix W is used as a metric on the joint space to give more importance to the joints we want to move. A typical use of it is to increase the weight of a joint when it approaches its limit [35], or even to block joints when surpassing their limits, which is called Joint Clamping (JC) [36].

A redundant robot can also have its Jacobian augmented up to a square matrix using Task Augmentation (TA) [37] [38], where gradients of secondary objectives are added as rows to yield a square and invertible matrix, J_V. Note that special care must be taken in order to avoid linear dependency of the Jacobian with its added rows.

Another method to use redundancies for a secondary task is Gradient Projection (GP) [39], which consists on projecting the gradient of a potential function, F, onto the kernel of the main task with the kernel projection operator $P = I - J^{\dagger}J$. This is effective at biasing solutions according to a certain criterion, but does not work, for instance, using a push-to-center value to avoid joint limits. In fact, considering joint limits on the kernel of the main task is not enough to ensure those are avoided. GP can be generalised using what is called Task Priority (TP) [40], in which a list of tasks is performed in a hierarchical order projecting each one onto the kernel of the previous tasks.

Among these methods, if we want to avoid joint limits, neither JW, TA or GP have success at it. This is due to the secondarity of such a task. And JC may result in permanently blocking degrees of freedom. Then, the best solution is to use a task priority scheme, with its main priority being a push-to-center value of the joints, activated by a (commonly diagonal) matrix $H = diag(h_1, ..., h_m)$, being $h_i > 0$ if θ_i is close to its limits, up to 1 if those are reached, and 0 otherwise, with a gain μ:

$$\Delta\boldsymbol{\theta} = -H(\mu\boldsymbol{\theta}) + \left[J(I_m - H^{\dagger}H)\right]^{\dagger}(\mathbf{e} + JH(\mu\boldsymbol{\theta})) \quad (5)$$

An additional problem that may arise when using this algorithm is that, even with an activation matrix continuous wrt. joint activation as in [36], the pseudoinverse operator is not continuous with respect to this activation matrix. *Theorem 4.2* in [41] states that the effect of any nonzero diagonal element of an activation matrix H is equivalent when using it in a JC or TP algorithm. In fact, it can also be seen that damping the pseudoinverse does not solve the problem, out of a very small interval [42]. Due to these issues, in that work it is presented a continuous (wrt. activation matrix) pseudoinverse operator, defined as:

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

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

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

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

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

$\wp(M)$ being now the power set of $M = \{1, .., m\}$, $J_Q = H_0 J$, where $H_{0_i} = 1$ if $i \in P$ and 0 otherwise. With this pseudoinverse operator, (5) becomes (from Eq. (20) in [27]):

$$\Delta\boldsymbol{\theta} = -H(\mu\boldsymbol{\theta}) + J^{(I_m - H)\oplus}(\mathbf{e} + JH(\mu\boldsymbol{\theta})) \quad (6)$$

Which we will call Continuous Task Priority (CTP). In Table III we can see the commented algorithms for secondary tasks, which have been applied to avoid joint limits.

VI. 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 (5), we can reorder the terms and separate the position error-dependent terms (**e**) from those that don't depend on it):

$$\Delta\boldsymbol{\theta} = \left(I - J^{(I_m-H)\oplus}J\right)H(-\lambda_{jl}\boldsymbol{\theta}) + J^{(I_m-H)\oplus}\mathbf{e}. \quad (7)$$

We intend to apply the ideas underlying the SD [28], 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 (7) and separate the position error-depending part (**e**) from the rest as follows:

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

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

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

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

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

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:

$$J^{(I-H)\oplus}u_s = \hat{\sigma}_s^{-1}v_s \Rightarrow \Delta\boldsymbol{\theta}^s = \hat{\sigma}_s^{-1}Jv_s$$

which has an effect on the jth joint of:

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

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 [28]:

$$\sum_{j=1}^{m}|\Delta\theta_j^s| \le \hat{\sigma}_s^{-1}\sum_{j=1}^{m}|v_{j,s}|\|J^j\| = M_s,$$

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}:

$$\gamma_s = min(1, 1/M_s)\gamma_{max} \quad (8)$$

To then proceed exactly as in the SD:

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

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

and we will bound this variation with the γ_s obtained at (8):

$$\Delta q_s = \begin{cases} 1 & \text{if } \|w_s\| < \gamma_s \\ \frac{ws}{\|ws\|}\gamma_s & \text{if } \|w_s\| \ge \gamma_s \end{cases}$$

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

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

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

$$\Delta\boldsymbol{\theta} = \begin{cases} 1 & \text{if } \|\Delta\hat{\boldsymbol{\theta}}\| < \gamma_{max} \\ \frac{\Delta\hat{\boldsymbol{\theta}}}{\|\Delta\hat{\boldsymbol{\theta}}\|}\gamma_{max} & \text{if } \|\Delta\hat{\boldsymbol{\theta}}\| \ge \gamma_{max} \end{cases}$$

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

VII. EXPERIMENTATION

All the methods described have been implemented in *Matlab* and C++ (using a ROS library) in a 7-dof redundant WAM robot arm (with the *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 between each joint's limits, and mapped into a cartesian position with the forward kinematics function.

TABLE IV
DENAVITT-HARTENBERG STANDARD PARAMETERS FOR 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

The results of a *Matlab* simulation can be seen in Table V, where the columns represent (in order) the percentage of solutions found, the percentage of solutions found respecting joint limits, the average computation time for all tests, for those where a solution was found

TABLE III

METHODS USED TO AVOID JOINT LIMITS

Name	Abbreviation	Equation	References
Jacobian Weighting	JW	$\Delta\boldsymbol{\theta} = W^{-1}J^T(JW^{-1}J^T)^{-1}\mathbf{e}$	[35]
Gradient Projection	GP	$\Delta\boldsymbol{\theta} = J^{\dagger}\mathbf{e} + \mu P\nabla F$	[39]
Joint Clamping	JC	$\Delta\boldsymbol{\theta} = H(JH)^{\dagger}\mathbf{e}$	[36]
Task Augmentation	TA	$\Delta\boldsymbol{\theta} = J_v^{\dagger}\mathbf{e}$	[38]
Task Priority	TP	$\Delta\boldsymbol{\theta} = -H(\mu\boldsymbol{\theta}) + \left[J(I_m - H^{\dagger}H)\right]^{\dagger}(\mathbf{e} + JH(\mu\boldsymbol{\theta}))$	[40]
Continuous Task Priority	CTP	$\Delta\boldsymbol{\theta} = H(-\mu\boldsymbol{\theta}) + J^{(I_m-H)\oplus}(\mathbf{e} - JH(-\mu\boldsymbol{\theta}))$	[27]

TABLE V

BEHAVIOUR OF THE STUDIED METHODS FOR A SAMPLE OF 1000 RANDOM INITIAL AND END POSITIONS FOR THE WAM ROBOT ARM. NOTATION AS IN TABLES II AND III.

Method	% sol.	% sol. resp. limits	$\bar{t}$(ms)	$\overline{t_{sol}}$ (ms)	$\overline{t_{JL}}$ (ms)	$\overline{e_{nosol}}$	$\overline{it}$	$\overline{it_{sol}}$
JP	100.0	6.8	42.6	42.6	19.4	-	12.2	12.2
JT	40.70	12.70	710.6	504.8	527	0.302	209.4	148.7
SD - $\gamma_{max} = 0.5$	98.4	33.9	165.2	154.3	140.6	0.042	46.9	43.5
JD - $\lambda = 0.005$	100.0	6.9	39.4	39.4	18.9	-	11.6	11.6
JF - $\lambda_{max} = 4\lambda$	100.0	7.4	38.6	38.6	19.9	-	11.2	11.2
ED	100.0	32.3	36.9	36.9	32.2	-	10.6	10.6
IED - $\Omega = 0.01I_m$	100.0	32.4	38.7	38.7	33.3	-	11.1	11.1
SVF - $nu = 10$, $\sigma_0 = 0.01$	**100.0**	**8.7**	**37.1**	**37.1**	**22.1**	**-**	**10.7**	**10.7**
SVF+ED	**100.0**	**32.3**	**35.8**	**35.8**	**30.9**	**-**	**10.3**	**10.3**
SVF+SD	**99.7**	**34.4**	**147.4**	**145.2**	**133.7**	**0.041**	**41.9**	**41.1**
JW - as in [35]	100.0	7.0	45.7	45.7	20.5	-	13.0	13.0
GP - $\mu = 0.2$	100.0	2.0	55.8	55.8	18.7	-	15.9	15.9
TA - as in [38]	99.2	24.5	21.3	17.5	16.3	0.135	43.6	35.9
JC - H as in [27]	52.6	20.0	476.3	115.7	93.8	0.601	136.4	33.0
TP - H as in [27]	0.5	0.5	934.8	22.3	22.3	1.163	249.8	6.0
CTP - H as in [27]	34.6	34.6	7007.0	2242.0	2242.0	0.440	184.6	59.2
CTP+SVF	**34.6**	**34.6**	**7042.6**	**2267.0**	**2267.0**	**0.395**	**184.8**	**59.7**
CTP+SD	**48.2**	**48.2**	**6876.9**	**2027.3**	**2027.3**	**0.286**	**151.7**	**45.1**
CTP+SD+SVF	**48.5**	**48.5**	**6830.5**	**1867.9**	**1867.9**	**0.276**	**149.2**	**41.1**

and for those respecting joint limits, the average error using the position-orientation metric in Section II when the solution has not been found, and the average number of iterations and such average when solutions were found. The performance of the reviewed state-of-the-art methods is compared with our proposals, which are highlighted in bold face in the table. Besides the filtering enhancement SVF in different combinations, we have used the CTP algorithm as in (6), together with the SD proposed in Section VI, and we have also combined them with SVF to compare results. Additional experiments with videos can be downloaded at http://www.iri.upc.edu/groups/perception/IK/IKacolome.zip. 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 . The remaining algorithms not considering joint limits always converge, except for SD, due to the limited number of iterations.
- JW, TA and GP methods 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.
- 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.
- Using SVF improves the speed of the JP 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.
- 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 a is successful strategy. Adding SD improves the convergence ratio, and it also reduces their computation time. Overall, the CTP computation times are very large. This may be in part because of *Matlab* not being optimal for such computations, but it should be reduced by finding an approximate value of the continuous pseudoinverse.

The low convergence ratio of CTP algorithms is 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 the algorithm walks into a dead end in the joint space. The algorithms not fully respecting joint limits can cross regions with unfeasible joint values to reach the goal, while CTP algorithms can't. To avoid this convergence problem, some literature works 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. Actually, as mentioned, we have tested these methods as global IK solvers to highlight their differences, but of course they should be used in a more local way, leaving trajectory connectivity issues to a global path planner.

VIII. CONCLUSIONS

Along this work, the most relevant CLIK algorithms for redundant robots have been compared. Special attention has been paid to three issues:

- Large gains in some iterations. JP 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 assimilable to a random positioning in the joint space, whose topology is equivalent to an *m-Torus*, mapped into the workspace, and its high convergence ratio 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, so it is recommended to use such damping in most algorithms.
- Matrix conditioning, We have compared the capability of the different algorithms to avoid amplifying the numerical error on 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 the 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 to numerically filter or damp the Jacobian pseudoinverse of a matrix. We have also seen that this does not mean 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.
- Secondary tasks and joint limits. We have presented some 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 Barrett's 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, to prevent the robot getting stuck in such a situation.

APPENDIX

The JD algorithms and those similar, such as the JF, ED and IED, avoid discontinuities on the Jacobian with respect to its singular values. Nevertheless, if we pay attention to the resulting condition number, we will see that 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 $1/\sigma$ for the singular values when computing the pseudoinverse of the Jacobian.

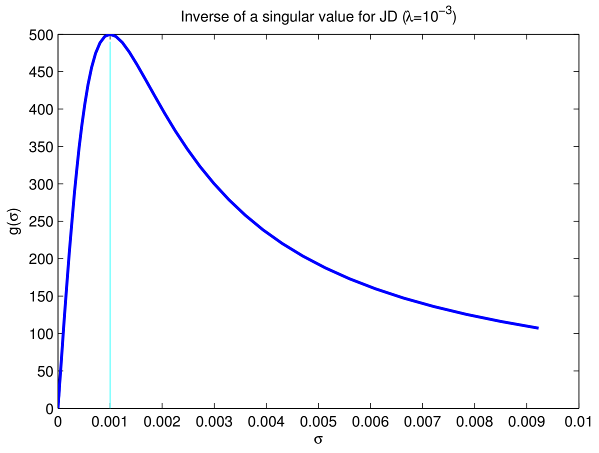

Fig. 2.

This function g, as we see in Fig. 2, has a maximum at $\sigma = \lambda$ with a value of $g(\lambda) = \frac{1}{2\lambda}$.

Now, we can distinguish different cases depending on the least singular value of the 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:

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

- $\exists i, j$ 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 will not depend on σ_n, it will be bounded.

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

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

This means that, on the one hand, λ should have a high value to avoid this maximum 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.

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 order of magnitude does not change. And using error damping means having a very large damping factor, thus if the goal is a singular position (for example, reaching the furthest point with an arm), the results are equivalent to the behaviour of the JD algorithm.

References

[1] M. Shimizu, H. Kakuya, W.-K. Yoon, K. Kitagaki, and K. Kosuge. ”Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution.” *IEEE Trans. on Robotics*, pp 1131-1142, 2008.

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

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

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

[6] 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.

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

[8] 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.

[9] 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.

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

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

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

[13] 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.

[14] 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.

[15] 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.

[16] B. Siciliano. ”A Closed-Loop Inverse Kinematic Scheme for On-line Joint-based Robot Control.” *Robotica*, vol 8, pp 231-243, 1990.

[17] 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.

[18] 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.

[19] Z Jiang. ”A converse lyapunov theorem for discrete-time systems with disturbances.” *Systems & Control Letters*, vol 45, pp 49-58, 2002.

[20] P. Falco and C. Natale. ”On the stability of closed-loop inverse kinematics algorithms for redundant robots.” *IEEE Trans. on Robotics*, vol 99, pp 1-5, 2011.

[21] J. Baillieul. ”Kinematic programming alternatives for redundant manipulators.” *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol 2, pp 722-728, 1985.

[22] T. Yoshikawa. ”Dynamic manipulability of robot manipulators.” *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol 2, pp 1033-1038, 1985.

[23] T. Yoshikawa. ”Analysis and Control of Robot Manipulators with Redundancy.” *First Int. Symposium Robotics Research*, pp 735-748, 1984.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. ”Modelling, Planning and Control.” Advanced Textbooks in Control and Signal Processing. Springer, 1st edition, 2009.

[25] 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.

[26] Yoshihiko Nakamura, ”Advanced Robotics: Redundancy and Optimization”, *Addison-Wesley Pub. Co.*, 1991.

[27] 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.

[28] S. R. Buss, J.-S. Kim. ”Selectively Damped Least-Squares for Inverse Kinematics.” *J. of Graphics Tools*, vol 10, pp 37-49, 2004.

[29] 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.

[30] C.A. Klein, B.E. Blaho, ”Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators.” *Int. Journal of Robotics Research*, vol 6, no 2, pp 72-83, 1987.

[31] 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.

[32] S. R. Buss. ”Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods.” *Unpublished*, http://math.ucsd.edu/˜sbuss/ResearchWeb, 2004.

[33] Stephen K. Chan and Peter D. Lawrence. “General Inverse Kinematics with the Error Damped Pseudoinverse.“ *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol.2, pp 834-839, 1988.

[34] T. Sugihara, ”Solvability-Unconcerned Inverse Kinematics by the Levenberg-Marquardt Method.” *IEEE Trans. on Robotics*, vol 27, no 5, pp 984-991, oct 2011.

[35] 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.

[36] D. Raunhardt, R. Boulic. ”Progressive Clamping.” *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp 4414-4419, 2007.

[37] 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.

[38] 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.

[39] 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.

[40] Y. Nakamura, H. Hanafusa, T. Yoshikawa, ”Task-priority based redundancy control of robot manipulators.” *Int. Journal of Robotics Research*, vol 6, no 2, pp 3-15, 1987.

[41] 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*

[42] 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.