Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

Inverse Kinematics for Humanoid Robots

Gaurav Tevatia

tevatia@usc.edu
http://www-slab.usc.edu/tevatia

Stefan Schaal

sschaal@usc.edu
http://www-slab.usc.edu/sschaal

Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089-2520, USA
Kawato Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan

Abstract: *Real-time control of the endeffector of a humanoid robot in external coordinates requires computationally efficient solutions of the inverse kinematics problem. In this context, this paper investigates methods of resolved motion rate control (RMRC) that employ optimization criteria to resolve kinematic redundancies. In particular we focus on two established techniques, the pseudo inverse with explicit optimization and the extended Jacobian method. We prove that the extended Jacobian method includes pseudo-inverse methods as a special solution. In terms of computational complexity, however, pseudo-inverse and extended Jacobian differ significantly in favor of pseudo-inverse methods. Employing numerical estimation techniques, we introduce a computationally efficient version of the extended Jacobian with performance comparable to the original version. Our results are illustrated in simulation studies with a multiple degree-of-freedom robot, and were tested on a 30 degree-of-freedom humanoid robot.*

1 Introduction

Most movement tasks are defined in coordinate systems that are different from the actuator space in which motor commands must be issued. Hence, movement planning and learning in task space ([1], [2], [3]) require appropriate coordinate transformations from task to actuator space before motor commands can be computed. We will focus on the case where movement plans are given as external kinematic trajectories—as opposed to complete task-level control laws— on systems with many redundant degrees-of-freedom (DOFs), as typical in humanoid robotics (Figure 1). The transformation from kinematic plans in external coordinates to internal coordinates is the classic inverse kinematics problem, a problem that arises from the fact that inverse transformations are often ill-posed. If we define the intrinsic coordinates of a manipulator as the n-dimensional vector of joint angles $\boldsymbol{\theta} \in \Re^n$, and the position and orientation of the manipulator's endeffector as the m-dimensional vector $\mathbf{x} \in \Re^m$, the forward kinematic function can be written as:

$$\mathbf{x} = \mathrm{f}(\boldsymbol{\theta}) \tag{1}$$

while what we need is the inverse relationship:

$$\boldsymbol{\theta} = \mathrm{f}^{-1}(\mathbf{x}) \tag{2}$$

Figure 1: Humanoid robot in our laboratory

For redundant manipulators, i.e., $n > m$, solutions to Equation (2) are usually non-unique (excluding the degenerate case where no solutions exist at all), and even for $n = m$ multiple solutions can exist (e.g., [4]). Therefore, inverse kinematics algorithms need to address how to determine a particular solution to (2) in face of multiple solutions. Heuristic methods have been suggested, such as freezing DOFs to eliminate redundancy. However, redundant DOFs are not necessarily disadvantageous as they can be used to optimize additional constraints, e.g., manipulability, force constraints, etc. Thus, it is useful to solve the inverse problem (2) by imposing an optimization criterion:

$$g = g(\boldsymbol{\theta}) \tag{3}$$

where g is usually a convex function that has a unique global optimum.

Local methods which are feasible in real time, only compute an optimal change in $\boldsymbol{\theta}$, $\Delta\boldsymbol{\theta}$, for a small change in $\mathbf{x}, \Delta\mathbf{x}$ and then integrate $\Delta\boldsymbol{\theta}$ to generate the entire joint space path. Resolved Motion Rate Control (RMRC) ([6]) is one such local methods. It uses the Jacobian $\mathbf{J}$ of the forward kinematics to describe a change of the endeffector's position as

$$\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{4}$$

This equation can be solved for $\dot{\boldsymbol{\theta}}$ by taking the inverse of $\mathbf{J}$ if it is square i.e. m=n, and non-singular. For a redundant manipulator n is greater than m, e.g., n=30 and m=3 for our

humanoid reaching for an object, which necessitates the use of additional constraints to obtain a unique inverse.

1.1 Pseudo-Inverse Methods

Pseudo-inverse methods are a common choice to invert (4) in face of redundancies, e.g. by using the Moore-Penrose inverse

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\#}\dot{\mathbf{x}} = \mathbf{J}^{T}\left(\mathbf{J}\mathbf{J}^{T}\right)\dot{\mathbf{x}} \tag{5}$$

which minimizes the norm of $\dot{\boldsymbol{\theta}}$. Liegeois ([7]) suggested a more general form of optimization with pseudo-inverses by minimizing an explicit objective function g in the null space of $\mathbf{J}$:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\#}\dot{\mathbf{x}} - \alpha(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})\frac{\partial g}{\partial \boldsymbol{\theta}} \tag{6}$$

Variations of this method, based on the dual projection property of the row space and the null space of the Jacobian([8, 9]), may increase the computational efficiency when there are fewer degrees of redundancy than unconstrained DOFs, i.e. $(n\text{-}m)<m$, but these are expected to be too expensive for highly redundant systems.

As a general problem, pseudo-inverse methods are not conservative, i.e., a closed path in endeffector space does not guarantee a closed path in joint space ([10]). Additionally, it is not always easy to determine the constant α in (6) which controls the influence of the optimization function g. These problems motivated the extended Jacobian method.

1.2 The Extended Jacobian Method (EJM)

The Extended Jacobian Method was suggested by Baillieul ([11]) as an optimization method that obtains a conservative RMRC solution. The goal of the EJM is to zero the gradient of $g(\boldsymbol{\theta})$ in the null space of the Jacobian and track this optimal solution. For this purpose, the gradient of the optimization criterion, $\partial g/\partial \boldsymbol{\theta}$, is projected onto the null space basis vectors of $\mathbf{J}, \boldsymbol{\eta}_i$, given by the columns of $\mathbf{V}_N = [\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{n-m}]$ of the singular value decomposition of $\mathbf{J}$

$$\mathbf{J} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} = \mathbf{U}\begin{pmatrix}\mathbf{S}_R & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{V}_R^{T} \\ \mathbf{V}_N^{T}\end{pmatrix} \tag{7}$$

$$\mathbf{G} = \frac{\partial g}{\partial \boldsymbol{\theta}}^{T}\mathbf{V}_N \tag{8}$$

Note that in (7) and (8), we introduce a partitioned notation of SVD in terms of the null-space and range components, indexed by "N" and "R", respectively. The goal of $\mathbf{G} = \mathbf{0}$ serves to augment the forward kinematics function with n–m additional constraints:

$$\begin{pmatrix}\mathbf{f}(\boldsymbol{\theta}) \\ G_1 \\ \vdots \\ G_{n-m}\end{pmatrix} = \begin{pmatrix}\mathbf{x} \\ 0 \\ \vdots \\ 0\end{pmatrix} \tag{9}$$

where G_i denotes the i-th coefficient of $\mathbf{G}$. For an incremental step $\Delta\mathbf{x}$, the system can be linearized,

$$\begin{pmatrix}\mathbf{J} \\ \dfrac{\partial G_1}{\partial \boldsymbol{\theta}} \\ \vdots \\ \dfrac{\partial G_{n-m}}{\partial \boldsymbol{\theta}}\end{pmatrix}\Delta\boldsymbol{\theta} = \begin{pmatrix}\Delta\mathbf{x} \\ 0 \\ \vdots \\ 0\end{pmatrix} \quad \text{or} \quad \mathbf{J}_{ext}\Delta\boldsymbol{\theta} = \begin{pmatrix}\Delta\mathbf{x} \\ \mathbf{0}\end{pmatrix}$$

where (10)

$$\frac{\partial G_i}{\partial \boldsymbol{\theta}} = \boldsymbol{\eta}_i^{T}\frac{\partial^2 g}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^{T}} + \left(\frac{\partial g}{\partial \boldsymbol{\theta}}\right)^{T}\frac{\partial \boldsymbol{\eta}_i}{\partial \boldsymbol{\theta}} \equiv \mathbf{j}_{i,1}^{E} + \mathbf{j}_{i,2}^{E}$$

In (10), the rank deficient Jacobian matrix was augmented to yield a square matrix $\mathbf{J}_{ext}$ that can be inverted to give

$$\Delta\boldsymbol{\theta} = \begin{pmatrix}\mathbf{J} \\ \mathbf{J}_1^{E} + \mathbf{J}_2^{E}\end{pmatrix}^{-1}\begin{pmatrix}\Delta\mathbf{x} \\ \mathbf{0}\end{pmatrix} = \mathbf{J}_{ext}^{-1}\begin{pmatrix}\Delta\mathbf{x} \\ \mathbf{0}\end{pmatrix} \tag{11}$$

Notice that in (10) the gradient of $\mathbf{G}$ remains zero, i.e. if we start at the optimal posture the EJM will track the optimal solution throughout the movement. $\mathbf{J}_{ext}$ is singular only when $\mathbf{J}$ is singular or when the $\partial\mathbf{G}/\partial\boldsymbol{\theta}$ block has rank< n-m.

1.3 Research Objectives

While for off-line applications the computational complexity of inverse kinematics may not be a major problem, and the quality of the performance can be checked before running the robot, in on-line applications both of these issues become crucially important. For instance, in highly redundant systems as our humanoid in Figure 1, the pseudo-inverse may have unsatisfying performance, while the extended Jacobian may be too expensive. In the following, we will investigate how to make the EJM computationally more efficient in order to achieve the low cost of pseudo-inverse methods. In addition, we will demonstrate that the EJM can be considered as a superset of all pseudo-inverse methods ([9]) with optimization terms. By means of on-line learning techniques with recursive least squares, we will derive the currently most efficient implementation of the EJM.

2 Efficient Ext. Jacobian Algorithms

The computational burden of the EJM stems from two elements: i) the need for a singular value decomposition (SVD)([12]), and ii) the need for computing the change of null space with repect to $\boldsymbol{\theta}$ (cf. term $\mathbf{J}_2^{E}$ in (6)). Since, usually, the latter computation has to be performed numerically, it is necessary to compute n+1 SVDs for every local inverse kinematics step. For real-time applications, this costly operation needs to be avoided.

2.1 Gradient Descent to Optimality

All formulations of the EJM assume that the initial posture is the optimal posture ($\mathbf{G} = 0$) which is then tracked

throughout the movement. If for some reason this optimal posture is lost, these methods continue to track a non-optimal posture ($\mathbf{G} \neq 0$). We propose to modify the EJM to include gradient descent in $\mathbf{G}$ during the course of movement. Adding gradient descent to the EJM requires including a term $-\alpha\mathbf{G}$ in (6), where α is a positive scalar adjusting the strength of gradient descent:

$$\mathbf{J}_{ext}\Delta\boldsymbol{\theta} = \begin{pmatrix} \Delta\mathbf{x} \\ -\alpha\mathbf{G} \end{pmatrix} \tag{12}$$

With the above formulation, whatever the starting posture of the manipulator, the optimal posture will be reached after some transient time, and if for some reason the manipulator deviates from $\mathbf{G} = 0$, it will return to optimal automatically.

2.2 Omitting $\mathbf{J}_2^E$

Given the "stabilized" EJM in (12), it is now possible to simplify the EJM and leave it to the gradient descent to compensate for errors resulting from the simplification. For instance, it is possible to just omit the costly term $\mathbf{J}_2^E$ in (12) , i.e., to neglect the contribution of the change of the null space in the EJM. That this simplification is reasonable can be best understood in the context of the commonly chosen optimization criterion

$$g(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(\theta_{0,i} - \theta_i\right)^2 \tag{13}$$

For this criterion, the Appendix proves that leaving out $\mathbf{J}_2^E$ is analytically identical with Liegeois ([7]) optimized pseudo-inverse in (6), i.e., the pseudo-inverse can be interpreted as a special case of the EJM ([9]). If other optimization criteria are chosen, the simplified EJM and the pseudo-inverse method start to differ, due to the fact the $\mathbf{J}_1^E \neq \mathbf{V}_N^T$ as it was for the special case (13). In later sections, we will empirically show that the simplified EJM has actually better performance than the pseudo inverse for such criteria.

2.3 Estimating $\mathbf{J}_2^E$

The simplified EJM from the previous section is computationally much cheaper than the original EJM, but this speed-up is bought at the cost of decreased tracking performance of the optimal posture—for special cases, the simplified EJM is equivalent to the pseudo-inverse and inherits its characteristics. If maintaining the optimal posture is a primary objective, it is possible to estimate $\mathbf{J}_2^E$ by supervised learning rather than to compute it numerically. It is possible to correctly estimate $\hat{\mathbf{G}}^{n+1}$, i.e., the value of $\mathbf{G}$ at the next RMRC iteration:

$$\begin{aligned} \hat{\mathbf{G}}^{n+1} &= \mathbf{G}^n - \alpha\mathbf{G}^n dt \quad \text{or} \\ \hat{\mathbf{G}}^{n+1} &= \mathbf{G}^n - (\mathbf{J}_1^E + \hat{\mathbf{J}}_2^E + \mathbf{J}_{2,error}^E)\Delta\boldsymbol{\theta} \end{aligned} \tag{14}$$

Due to an incorrect estimate of $\mathbf{J}_2^E$, $\hat{\mathbf{J}}_2^E$, we will actually observe $\mathbf{G}^{n+1}$ to be different from $\hat{\mathbf{G}}^{n+1}$:

$$\mathbf{G}^{n+1} = \mathbf{G}^n - (\mathbf{J}_1^E + \hat{\mathbf{J}}_2^E)\Delta\boldsymbol{\theta} \tag{15}$$

This difference can be used as an error signal for supervised learning. Subtracting (15) from (14) results in

$$\mathbf{J}_{2,error}^E \Delta\boldsymbol{\theta} = \mathbf{G}^{n+1} + \alpha\mathbf{G}^n dt - \mathbf{G}^n \tag{16}$$

At every time step, (16) can be conceived of as the error for a linear neural network

$$\begin{aligned} \mathbf{y} &= \hat{\mathbf{J}}_2^E \Delta\boldsymbol{\theta} \quad \text{where} \\ \mathbf{y}_{error} &= \mathbf{G}^{n+1} + \alpha\mathbf{G}^n dt - \mathbf{G}^n \end{aligned} \tag{17}$$

Assuming that $\mathbf{J}_2^E$ does not change too quickly during the movement, $\hat{\mathbf{J}}_2^E$ can be obtained from recursive least squares (RLS) with forgetting term ([13]):

$$\begin{aligned} \mathbf{P}^{n+1} &= \frac{1}{\lambda}\left(\mathbf{P}^n - \frac{\mathbf{P}^n \Delta\boldsymbol{\theta}^n \Delta\boldsymbol{\theta}^{nT} \mathbf{P}^n}{\lambda - \Delta\boldsymbol{\theta}^{nT} \mathbf{P}^n \Delta\boldsymbol{\theta}^n}\right) \quad \text{where} \quad \lambda \in [0,1] \\ \hat{\mathbf{J}}_2^{E^{n+1}} &= \hat{\mathbf{J}}_2^{E^n} + \mathbf{P}^{n+1} \Delta\boldsymbol{\theta}^n \mathbf{y}_{error}^T \end{aligned} \tag{18}$$

Since $\mathbf{J}_2^E$ is constantly changing, the forgetting rate λ needs to be chosen small, e.g., λ=0.9, such that $\hat{\mathbf{J}}_2^E$ tracks the true $\mathbf{J}_2^E$ as well as possible. RLS with such a small forgetting rate tends be numerically unstable. The problem can be remedied by including ridge regression in RLS ([14]). For this purpose, the matrix $\mathbf{P}$ is initialized as

$$\mathbf{P}^0 = \mathbf{I} / r \tag{19}$$

where $\mathbf{I}$ denotes the identity matrix, and r is a positive scalar, the ridge regression parameter. The larger r the more numerical stabilization is added, however, at the cost that $\hat{\mathbf{J}}_2^E$ becomes biased towards zero. For example, a very large r will cause $\hat{\mathbf{J}}_2^E$ to vanish such that we are left with the simplified EJM from the previous section. Since the forgetting term λ will exponentially eliminate the ridge regression stabilization, it is necessary to add back the fraction that was forgotten. This operation can be accomplished by recursive least squares as explained in ([14]). Fortunately, adding back the ridge parameter, need not be performed every iteration. In our work, we update the ridge regression parameter for one of the RLS input dimensions during every RMRC update such that within n iterations all dimensions get numerically "re-stabilized". In this case, the update equation for the i^{th} dimension becomes

$$\begin{aligned} \mathbf{P}^{n+1} &= \mathbf{P}^n - \frac{\left[\mathbf{P}^n\right]_i \left[\mathbf{P}^n\right]_i^T}{\frac{1}{\Delta r} - P_{ii}}, \quad \Delta r = \sum_{i=1}^{n} (1-\lambda)^i r \\ \hat{\mathbf{J}}_2^{E^{n+1}} &= \hat{\mathbf{J}}_2^{E^n} - \left[\mathbf{P}^n\right]_i \left[\hat{\mathbf{J}}_2^{E^n}\right]_i^T \end{aligned} \tag{20}$$

where $[\cdot]_i$ denotes the i^{th} column of a matrix. The two RLS updates in (18) and (20) that have to performed at every inverse kinematics iteration are only of $O(n^2)$ as opposed to the $O(n^3)$ process that is needed to compute the n+1-fold SVD for obtaining the true $\mathbf{J}_2^E$.

As a technical note, it should be mentioned that when employing SVDs for computing the null space, it is possible that the null space basis $\mathbf{V}_N$ changes discontinuously from one RMRC iteration to the next one. This numerical problem can be avoided by adjusting the basis of the null space computed by SVD, $\mathbf{V}_{N,svd}^{n+1}$, to be maximally close to the previous null space basis $\mathbf{V}_N^n$. This is done by projecting the previous null space in the current null space by the formula

$$\mathbf{V}_N^{n+1} = \mathbf{V}_{N,svd}^{n+1} \mathbf{V}_{N,svd}^{n+1\ T} \mathbf{V}_N^n \qquad (21)$$

Over many iterations the null space basis vectors derived from (21) need to be renormalized even though they decay only very slowly. Theoretically, a Gram-Schmidt orthogonalization should be applied to $\mathbf{V}_N^{n+1}$ occasionally. However, empirically we did not notice any negative effects of omitting the re-orthogonalization.

2.4 Klein's Extensions to the Ext. Jacobian Method

Klein et al. ([15]) suggested a modified EJM that includes the change in null space vectors as unknown parameters. Assuming the system starts at the optimal posture, the null space vectors are re-estimated from a set of linear constraints, derived from the fact that:

- the rows of the Jacobian and null space vectors are perpendicular at all times
- the optimization index G_i remains zero at all times
- the null space basis is orthonormal

Based on these constraints, an update for both joint angles and null space vectors can be computed simultaneously by augmenting $\mathbf{J}_{ext}$ to yield an even larger matrix. Thus, at the cost of inverting a much bigger matrix and computing the derivative of the Jacobian $\partial\mathbf{J}/\partial\theta_k$, SVD calculations can be avoided. Recognizing that, for higher degrees of redundancy, inverting such a large matrix may be even more costly than computing SVDs, Klein et al. ([15]) also derived a compact formulation :

$$\begin{pmatrix} \mathbf{J} \\ \mathbf{V}_N^T \dfrac{\partial^2 g}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T} - \mathbf{W} \end{pmatrix} \Delta\boldsymbol{\theta} = \begin{pmatrix} \Delta\mathbf{x} \\ \mathbf{0} \end{pmatrix} \qquad (22)$$

$$\text{where } w_{ij} = \frac{\partial g}{\partial\boldsymbol{\theta}}^T \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T\right)^{-1} \frac{\partial\mathbf{J}}{\partial\theta_j}\boldsymbol{\eta}_i$$

This version requires that the null-space vectors be computed once at every iteration step of the inverse kinematics, as well as the derivative of the Jacobian $\partial\mathbf{J}/\partial\theta_k$ and the pseudoinverse.

3 Empirical Evaluations

3.1 Computational Complexity

We compared the computational complexity of the six methods that we introduced above: a) the pseudo inverse with optimization, b) the original EJM, c) the simplified EJM, d) the EJM with estimated second term, e) Klein's first method, f) Klein's second method. Assuming a 3-dimensional Cartesian space, Figure 2,shows the number of floating point operations required for every RMSC iteration for the 6 methods as a function of n, the number of joints in the robot. To perform this comparison, we estimated that recursive computation of the (geometric) Jacobian requires $2*3^2*n*(n+1)$ flops and that of the analytical derivative of the Jacobian([16]) requires an additional $4*3*n^2$ flops. All other floating-point operations were determined numerically in Matlab.

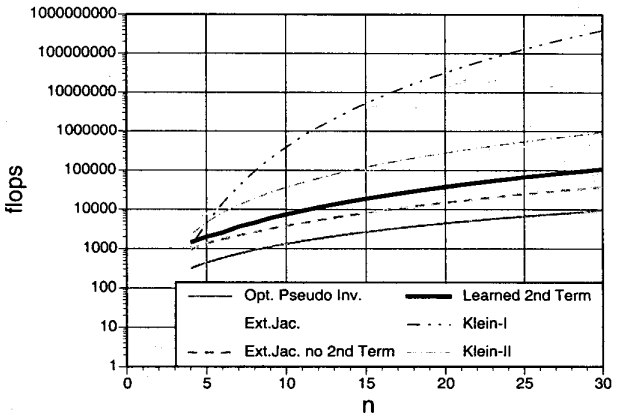

Figure 2: Computational Complexity of RMSC methods

The pseudo-inverse is computationally the most efficient, closely followed by the simplified EJM and the EJM with estimated second term. The original EJM is an order of magnitude more costly than the EJM with estimated second term. While Klein's first method is more efficient than the original EJM for few degrees of freedom, it clearly becomes too expensive for higher degrees of freedom as was expected due the large cost of inverting a much bigger matrix. Klein's second method also becomes as expensive as the original EJM for higher degrees of freedom. It should be noted that there are special cases where the kinematic structure of a robot may allow more efficient computations of some of the methods; Figure 2 can only provide an approximate estimate valid for a general multi-DOF robot.

3.2 Performance in simulation

We compared the different RMRC methods in a simulation of a planar robot arm with 10 revolute DOFs. The task of the robot was to draw circles with 0.2 meters diameter at random locations of the workspace. Each link of the robot was 0.1m long. Kinematic singularities were explicitly avoided in the tests. As each RMRC method is guaranteed to track the circular path in Cartesian space, the criterion for comparisons is how well the optimization criterion is fulfilled in null space, expressed by the value of G_i, which should be zero at all times. For the optimization criterion in (13), Figure 3 demonstrates the time course of the eight-dimensional $\mathbf{G}$ during 5 seconds of circular motion of 1Hz,

starting from a non-optimal posture. Figure 3a illustrates the performance of the original EJM augmented with gradient descent (cf. (12)). Klein's methods can be augmented with gradient descent, too, and would have an similar performance. Figure 3a, constitutes optimal performance and serves as the baseline comparison. Figure 3b demonstrates the performance of the simplified EJM (cf. Section 2.1) which is identical to the pseudo inverse with optimization. The optimization criterion is never fulfilled completely by these methods such that extraneous null space motion remains. Figure 3c shows that the EJM with estimated second term has essentially the same optimal performance as the original EJM. It should be noted, however, that in order to obtain this performance, the forgetting rate λ in (18) and the ridge regression parameter r in (19) need to be adjusted correctly—we used λ=0.95 and r=10^{-7}.

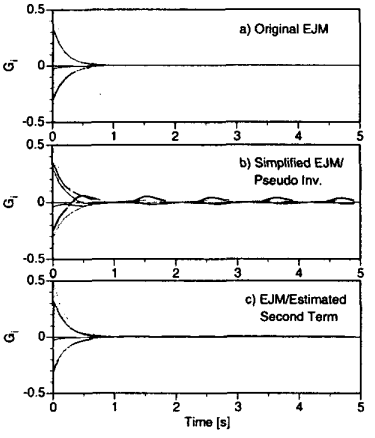

Figure 3: Convergence of optimization for L_2-norm cost criterion (cf. (12)). The desired posture for optimization was chosen to be θ_0=0. There are 8 different lines per chart, corresponding to 8 null space dimensions. All lines converging to zero characterizes optimal performance.

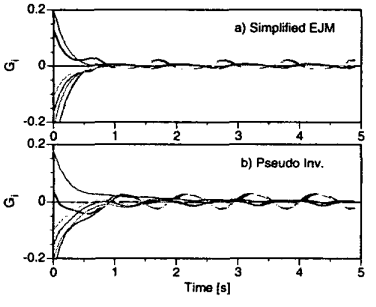

Figure 4: Converge of optimization in null space for L_4-norm cost criterion.

Figure 4 compares the performance of the simplified EJM with the pseudo-inverse under a modified cost criterion that uses the L_4-norm instead of the L_2-norm. In this case, pseudo-inverse and simplified EJM are not identical anymore. As can be seen in Figure 4, the simplified EJM has actually slightly better convergence to and smaller oscillations about the optimal posture. We observed this better performance of the simplified EJM in all empirical tests, which is likely to result from using a more suitable term for $\mathbf{J}_1^E$ in (11)—while the pseudo inverse just employs $\mathbf{J}_1^E = \mathbf{V}_N^T$. As a disadvantage, the simplified EJM is about a factor 3 more computationally expensive than the pseudo-inverse.

3.3 Performance on a 30 DOF Humanoid robot

A humanoid robot with 30 DOF (Figure 1)was used to compare the performance of the two least expensive algorithms, the pseudo-inverse with optimization and the extended Jacobian with estimated second term. The robot started at a non-optimal posture, and with its right hand followed a simulated ball tracing pseudo-random patterns. The patterns, which were generated by superimposing sinusoids of different frequencies in 3 dimensional Cartesian space, enclosed a volume of 0.4m x 0.2m x 0.6m in this XYZ space. To ensure tracking with the relevant degrees of freedom a weighted optimization criterion, e.g. which penalized the hip and body movements greater than the arm movements, was used:

$$g(\boldsymbol{\theta}) = \mathbf{w}\sum_{i=1}^{n}\left(\theta_{0,i} - \theta_i\right)^2 \qquad (23)$$

The desired posture, $\boldsymbol{\theta}_0$, was taken as the mid-range of the mechanical joint limits of the robot. While both the algorithms met the real-time requirements of the system Figure 5 shows that the extended Jacobian with the estimated term had much better convergence than the pseudo-inverse method. The remaining minor oscillations in fig 5b are related to the hip and body degrees of freedom, whose effect as seen, is actually due to an amplification of their cost contribution from having a high weight coefficient in $\mathbf{w}$ in (23) . To achieve this performance the RLS parameters were adjusted to λ=0.95 and r= 10^{-5}. In general, the ridge parameter may need to be increased with increasing DOFs.s

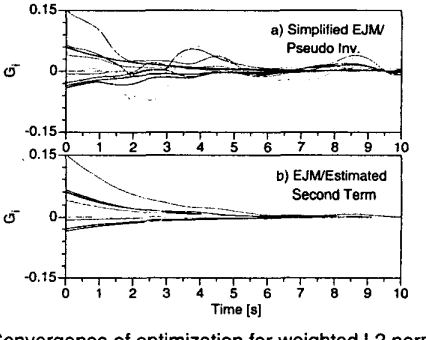

Figure 5: Convergence of optimization for weighted L2 norm cost criterion for 30 DOF robot

4 Conclusion

We investigated resolved motion rate control inverse kinematics algorithms that employ explicit optimization criteria. With the goal of using such algorithms in real-time on highly redundant robots, like humanoid robots, we exam-

ined the computational complexity and performance of six different algorithms. The computationally most efficient method is Liegeois' ([7]) pseudo-inverse. However, this method can cause extraneous motion in null space. By allowing for a 3-fold increase in computational complexity, we introduced a simplified extended Jacobian (EJM) algorithm that can reduce the irrelevant null-space motion for certain optimization criteria. For a specific cost function, this method can be shown to be identical with the pseudo-inverse methods and provides an analytical link between EJM and the various pseudo-inverse algorithms. Allowing for one order of magnitude higher floating-point operations than the pseudo-inverse, we introduced the up-to-now most efficient implementation of the EJM. This new algorithm employs recursive least-squares to estimate the computationally expensive terms of the extended Jacobian and shows performance similar to the original EJM. The original EJM is about two orders of magnitude more expensive than the pseudo-inverse, while other extended Jacobian methods from the literature are even more expensive.

Acknowledgments

This work was made possible by Award #9710312 of the National Science Foundation, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Cooperation, the ATR Human Information Processing Research Labs, and a USC Zumberge grant.

Appendix

Under the cost criterion (13), the simplified EJM is analytically identical to the pseudo-inverse with optimization(6) based on the following argument. First, it is possible to re-write the pseudo-inverse by using the partitioned SVD formula from (7):

$$\mathbf{J}^{\#} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1} = \mathbf{V}\mathbf{S}^T\mathbf{U}^T\left(\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}^T\mathbf{U}^T\right)^{-1}$$

$$= \begin{pmatrix}\mathbf{V}_R & \mathbf{V}_N\end{pmatrix}\begin{pmatrix}\mathbf{S}_R^{-1} \\ \mathbf{0}\end{pmatrix}\mathbf{U}^T = \mathbf{V}_R\mathbf{S}_R^{-1}\mathbf{U}^T$$

By noting that under this cost (13) $\mathbf{J}_1^E = \mathbf{V}_N^T$, the inverted extended Jacobian can be written in terms of the pseudo inverse:

$$\mathbf{J}^{Ext} = \begin{bmatrix}\mathbf{J} \\ \mathbf{V}_N^T\end{bmatrix} = \begin{bmatrix}\mathbf{J} \\ \mathbf{V}_N^T\end{bmatrix}\mathbf{V}\mathbf{V}^T = \begin{bmatrix}\mathbf{J}\mathbf{V}_R & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{V}_R & \mathbf{V}_N\end{bmatrix}^T$$

$$\left(\mathbf{J}^{Ext}\right)^{-1} = \begin{bmatrix}\mathbf{V}_R & \mathbf{V}_N\end{bmatrix}\begin{bmatrix}\left(\mathbf{J}\mathbf{V}_R\right)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}\end{bmatrix} = \begin{bmatrix}\mathbf{V}_R\left(\mathbf{J}\mathbf{V}_R\right)^{-1} & \mathbf{V}_N\end{bmatrix}$$

$$= \begin{bmatrix}\mathbf{V}_R\left(\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}_R\right)^{-1} & \mathbf{V}_N\end{bmatrix} = \begin{bmatrix}\mathbf{V}_R\left(\mathbf{U}\begin{bmatrix}\mathbf{S}_R & 0\end{bmatrix}\begin{bmatrix}\mathbf{V}_R^T \\ \mathbf{V}_N^T\end{bmatrix}\mathbf{V}_R\right)^{-1} & \mathbf{V}_N\end{bmatrix}$$

$$= \begin{bmatrix}\mathbf{V}_R\left(\mathbf{U}\mathbf{S}_R\right)^{-1} & \mathbf{V}_N\end{bmatrix} = \begin{bmatrix}\mathbf{V}_R\mathbf{S}_R^{-1}\mathbf{U}^{-1} & \mathbf{V}_N\end{bmatrix} = \begin{bmatrix}\mathbf{J}^{\#} & \mathbf{V}_N\end{bmatrix}$$

The above result can also be derived from the approach of Chen et al([9]). The addition of the stabilizing term to the right hand side of the EJM leads to a new result Thus, the joint space update of the simplified EJM is

$$\Delta\boldsymbol{\theta}^{Ext} = \left(\mathbf{J}^{Ext}\right)^{-1}\begin{bmatrix}\Delta\mathbf{x} \\ -\alpha\mathbf{G}\end{bmatrix} = \mathbf{J}^{\#}\Delta\mathbf{x} + \alpha\mathbf{V}_N\mathbf{V}_N^T\frac{\partial g}{\partial\boldsymbol{\theta}}$$

Since the optimization term of the pseudo-inverse in (6) can be re-expressed([8]) as

$$\mathbf{I} - \mathbf{J}^{\#}\mathbf{J} = \mathbf{I} - \mathbf{V}_R\mathbf{S}_R^{-1}\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{V}\mathbf{V}^T - \mathbf{V}_R\mathbf{S}_R^{-1}\begin{bmatrix}\mathbf{S}_R & \mathbf{0}\end{bmatrix}\mathbf{V}^T$$

$$= \left(\mathbf{V} - \begin{bmatrix}\mathbf{V}_R & \mathbf{0}\end{bmatrix}\right)\mathbf{V}^T = \begin{bmatrix}\mathbf{0} & \mathbf{V}_N\end{bmatrix}\begin{pmatrix}\mathbf{V}_R^T \\ \mathbf{V}_N^T\end{pmatrix} = \mathbf{V}_N\mathbf{V}_N^T$$

we conclude that the simplified EJM and the pseudo-inverse update are identical:

$$\Delta\boldsymbol{\theta}^{\#} = \mathbf{J}^{\#}\Delta\mathbf{x} + \alpha\mathbf{V}_N\mathbf{V}_N^T\frac{\partial g}{\partial\boldsymbol{\theta}} = \Delta\boldsymbol{\theta}^{Ext}$$

References

1. Aboaf, E.W., C.G. Atkeson, and D.J. Reinkensmeyer. *Task-level robot learning*. in *Proceedings of the IEEE International Conference on Robotics and Automation*. 1988. April 24-29, Philadelphia, Pennsylvania.
2. Aboaf, E.W., S.M. Drucker, and C.G. Atkeson. *Task-level robot learing: Juggling a tennis ball more accurately*. in *Proceedings of IEEE Interational Conference on Robotics and Automation*. 1989. May 14-19, Scottsdale, Arizona.
3. Saltzman, E. and S.J.A. Kelso, *Skilled actions: A task-dynamic approach*. Psychological Review, 1987. **94**(1): p. 84-106.
4. Craig, J.J., *Introduction to robotics*. 1986, Reading, MA: Addison-Wesley.
5. Baillieul, J. and D.P. Martin, *Resolution of kinematic redundancy*, in *Proceedings of Symposia in Applied Mathematics*. 1990, American Mathematical Society. p. 49-89.
6. Whitney, D.E., *Resolved motion rate control of manipulators and human prostheses*. IEEE Transactions on Man-Machine Systems, 1969. **10**(2): p. 47-53.
7. Liegeois, A., *Automatic supervisory control of the configuration and behavior of multibody mechanisms*. IEEE Transactions on Systems, Man, and Cybernetics, 1977. **7**(12): p. 868-871.
8. Huang, M.Z. and H. Varma. *Optimal Rate Allocation in Kinematically Redundant Manipulators -- The Dual projection Method*. in *Proc. IEEE International Conference on Robotics and Automation*. 1991. Sacramento, CA.
9. Chen, Y.-C. and I.D. Walker. *A Consistent Null-Space Based approach to Inverse Kinematics of Redundant Robots*. in *IEEE International Conference on Robotics and Automation*. 1993.
10. Klein, C.A. and S. Ahmed, *Repeatable Pseudoinverse Control for Planar Kinematically Redundant Manipulators*. IEEE Transactions on Systems , Man, Cybernetics, 1995. **25**(12): p. 1657-1662.
11. Baillieul, J. *Kinematic programming alternatives for redundant manipulators*. in *IEEE International Conference on Robotics and Automation*. 1985.
12. Maciejewski, A.A. and C.A. Klein, *The Singular Value Decomposition: Computation and application to Robotics*. The International Journal of Robotics Research, 1989. **8**(6): p. 63-79.
13. Ljung, L. and T. Söderström, *Theory and practice of recursive identification*. 1986: Cambridge MIT Press.
14. Schaal, S. and C.G. Atkeson, *Constructive incremental learning from only local information*. Neural Computation, 1998. **10**(8): p. 2047-2084.
15. Klein, C.A., C. Chu-Jenq, and S. Ahmed, *A new formulation of the Extended Jacobian method and its use in mapping algorithmic singularities for kinematically redundant manipulators*. IEEE Transactions on Robotics and Automation, 1995. **11**(1): p. 50-55.
16. Ahmed, S., *Issues in Repeatability of Redundant Manipulator Control*, PhD thesis in *Department of Electrical Engineering*. 1992, Ohio State University.