Solving Inverse Kinematics – A New Approach to the Extended Jacobian Technique

M. Šoch, R. Lórencz

This paper presents a brief summary of current numerical algorithms for solving the Inverse Kinematics problem. Then a new approach based on the Extended Jacobian technique is compared with the current Jacobian Inversion method. The presented method is intended for use in the field of computer graphics for animation of articulated structures.

Keywords: inverse kinematics, Jacobian inversion, extended Jacobian technique.

1 Introduction

There are two main approaches when animating articulated structures, e.g., human bodies. The first approach, called *Forward Kinematics*, derives the movement of the structure from the movement of all its parts. The second technique, *Inverse Kinematics*, works in the opposite way. The movement of each part is determined by the movement of the complete structure.

Solving *Forward Kinematics* is a straightforward transformation between status vector $\boldsymbol{q}$ – angles between adjacent segments, and position vector $\boldsymbol{x}$ – position of the end point of the structure also called the end-effector or EE. It can be easily written as

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}), \tag{1}$$

while the inverse transformation is required for solving the *Inverse Kinematics* problem. It can be represented by

$$\boldsymbol{q} = \boldsymbol{f}^{-1}(\boldsymbol{x}). \tag{2}$$

The articulated structure is usually redundant, i.e. $n > m$. Thus the solutions to (2) are non-unique (except degenerated cases where no solution exists at all) and even for $n = m$ several solutions can exist. So the *Inverse Kinematics* algorithms have to select a particular solution of (2) in the face of multiple solutions. Heuristic techniques have been proposed, e.g., freezing DOFs to eliminate redundancy. However, the redundant DOFs are not necessarily disadvantageous, as they can be used to optimize additional constraints. Hence it is useful to impose some optimization criterion $g(\boldsymbol{q})$, usually represented by a function with a unique global optimum.

There are two generic approaches to solving the *Inverse Kinematics* problem with optimization criteria. *Global methods* find an optimal path of $\boldsymbol{q}$ with respect to the entire trajectory, usually in computationally expensive off-line calculations. In contrast, *local methods*, which are applicable in real-time, compute only an optimal change of status vector $\boldsymbol{q}$, d$\boldsymbol{q}$ according the small displacement d$\boldsymbol{x}$ and then integrate d$\boldsymbol{q}$ to generate a complete joint space path.

Resolved Motion Rate Control [12] is one such local method. It uses the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}) = \dfrac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial \boldsymbol{q}}$ from *Forward Kinematics* to relate the change of status vector to a change of the end-effector.

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{J}(\boldsymbol{q}) \cdot \mathrm{d}\boldsymbol{q} \tag{3}$$

This equation can be solved for desired d$\boldsymbol{x}$ and unknown d$\boldsymbol{q}$ by taking the inverse of $\boldsymbol{J}(\boldsymbol{q})$ if it is square $(m = n)$ and non-singular. For redundant structures $(n > m)$ the solving of (3) is described in the following sections. There are also analytical methods, optimization-based methods, and others [1, 7].

1.1 Jacobian inversion method

This method comes simply from (3) by inverting the Jacobian $\boldsymbol{J}(\boldsymbol{q})$. Although the standard inversion cannot be applied due to the Jacobian's non-squareness in the general case, Moore-Penrose pseudo-inversion $\boldsymbol{A}^{+} = \boldsymbol{A}^{T} \cdot (\boldsymbol{A} \cdot \boldsymbol{A}^{T})^{-1}$ has to be utilized. Equation (3) results in

$$\mathrm{d}\boldsymbol{q} = \boldsymbol{J}^{+}(\boldsymbol{q}) \cdot \mathrm{d}\boldsymbol{x}. \tag{4}$$

Pseudo-inversion can also be robustly found by Singular Value Decomposition [5]. An advantage of using pseudo-inverse is the minimum norm solution for d$\boldsymbol{q}$ [12]. Liegeois [8] suggested a more general form of optimization with pseudo-inverse by minimizing objective function $g(\boldsymbol{q})$:

$$\mathrm{d}\boldsymbol{q} = \boldsymbol{J}^{+}(\boldsymbol{q}) \cdot \mathrm{d}\boldsymbol{x} - \alpha \left(\boldsymbol{I} - \boldsymbol{J}^{+}(\boldsymbol{q}) \cdot \boldsymbol{J}(\boldsymbol{q}) \right) \frac{\partial g(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{5}$$

where $\boldsymbol{I}$ is identity matrix and α is positive gain constant. Although this formulation requires only that the gradient of $g(\boldsymbol{q})$ be calculated, the gain α is difficult to obtain.

As a general problem, pseudo-inverse methods are numerically unstable when the system reaches kinematic singularity [2] – the Jacobian is singular. Furthermore, there is still no control of inner parts of the structure.

1.2 Extended Jacobian method

The Extended Jacobian method was originally developed by Baillieul [2, 3].There are $r = n - m$ rows added to the Jacobian with goal to zero the gradient of $g(\boldsymbol{q})$ in the null space of the Jacobian. Let η_i, $i = 1, \ldots, r$ be the orthonormal set of vectors that span the null space of $\boldsymbol{J}(\boldsymbol{q})$, and $g(\boldsymbol{q})$ is the desired optimization criterion that should be minimized. Equation (3) with the Extended Jacobian follows.

$$\begin{bmatrix} \mathrm{d}\boldsymbol{x} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}(\boldsymbol{q}) \\ \dots \\ \frac{\partial}{\partial \boldsymbol{q}}(\eta_i \nabla g(\boldsymbol{q})) \\ \dots \end{bmatrix}_{i=1,\dots,r} \cdot \mathrm{d}\boldsymbol{q}. \qquad (6)$$

Hence the solution requires only standard inversion instead of pseudo-inversion.

The optimization criterion $g(\boldsymbol{q})$ represents a set of constraints even though its meaning is again more useful for robotics – it takes into account some objective-function (e.g., manipulability), which applies on the whole structure instead of physically-based constraints for each joint.

This method was further extended in [6] from the numerical point of view.

1.3 Jacobian Transposition method

This method comes from the Jacobian Inversion method (section 1.1).The reason for his is the substitution of computationally difficult inversion by simple transposition. This simplification is based on the virtual work principle [9], and it results in:

$$\mathrm{d}\boldsymbol{q} = \boldsymbol{J}^T(\boldsymbol{q}) \cdot \mathrm{d}\boldsymbol{x}. \qquad (7)$$

This method suffers in almost the same way as the Jacobian Inversion method (section 1.1). There is still no control of the internal parts of the structure, while the stability of the solution is much better.

1.4 Constrained Inverse Kinematics problems

The most common constraint on *Inverse Kinematics* systems is the joint limits. Joint constraints are usually represented by inequality constraints. Several different methods can be used to make sure a solution satisfies them.

1.4.1 The Lagrange approach

The Lagrange approach with Lagrange multipliers will find all minima for the objective function, subject to equality constraints, as stated in [7]. Those minima that do not satisfy the joint limit inequalities can be discarded immediately. Any new minima that arise from the inequalities must lie on the boundary of the region defined by those limits. That is when one or more of the joint variables take extreme values [7]. Therefore by setting one q_i to its lower bound or upper bound each time, these minima can be found by solving $2n$ smaller problems, each involving one less variable q_i than the original.

1.4.2 Introducing new variables

The new variables can be introduced to transform each inequality into an equality constraint [7], assuming the i-th joint angle q_i has upper limit u_i and lower limit l_i. The $2n$ new variables are added y_{il} and y_{iu} for the i-th joint to transform each inequality into an equality constraint.

Now the Lagrange approach (section 1.4.1) can be used to solve the original problem plus $2n$ new variables and $2n$ new equality constraints.

1.4.3 Penalty function methods

Another method adds penalty functions to the objective function. The algorithm looks for the minimum value of the objective function so the penalty causes the value of the objective function to increase as the joints approach their limits. The desired result is that the objective function itself effectively prohibits joint limit violations [1]. Unfortunately, penalty function methods are not numerically stable.

2 A New approach to the Extended Jacobian technique

We have focused on the Jacobian inversion method due to its mathematical simplicity and purity, as presented in [11]. We have also appreciated the Extended Jacobian technique suggested by Baillieul in [2, 3]. Extension of the Jacobian, to be an every-time matrix with rank at least n, is the only way to avoid using pseudo-inverse and to exploit standard Gaussian inverse. It is also possible to employ constraints on the structure and even on each joint.

However, the constraints have to deal with particular joints, which is more interesting in the targeted field of computer animation. It is inappropriate to incorporate some objective-function which is being optimized as was suggested in [2], which is more useful in robotics for which this method was originally developed.

The other motivation for herein presented method is the absence of similarly-based approaches in computer animation. This is probably caused by the high computational complexity for real-time applications. Thus performance optimization of presented approach is a future goal.

2.1 Notation and numbering

This section will provide an explanation of the indexing of the segments and joints, for better orientation in the further text.

The articulated structure consists of n segments $l_0, \dots, l_{n-1}$ and n joints $0, \dots, n-1$. The end-effector is seen as the n+1-th joint, designated by symbol n. The angles between the adjacent segments are represented by status vector $\boldsymbol{q}$ with $q_0, \dots, q_{n-1}$ items.

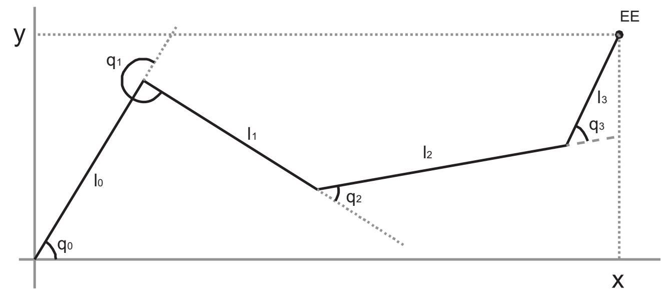

Fig. 1: Articulated structure with notation and numbering

We assume the structure in Fig. 1 with 4 segments l_0, l_1, l_2, l_3, 4 joints 0, 1, 2, 3 and the end-effector labeled with number 4. Status vector $\boldsymbol{q}$ contains 4 elements q_0, q_1, q_2, q_3.

2.2 Treating an articulated structure

It is necessary to extend the Jacobian, as was stated above, in order to avoid problems coming from usage of pseudo-inverse.

It seems natural to control every joint of an articulated structure by some physical characteristic – for instance by its joint-stiffness – rather than by setting its angle value. Thus we decided to treat articulated structures by pairs instead of manipulating them in their entirety. We assumed that a complete articulated structure consists of basic elements – two segments connected by a joint, as is shown in Fig. 2. Using many of these basic elements, it is easy to create an articulated structure with $n > 2$ segments. At each basic element its behavior can be controlled.

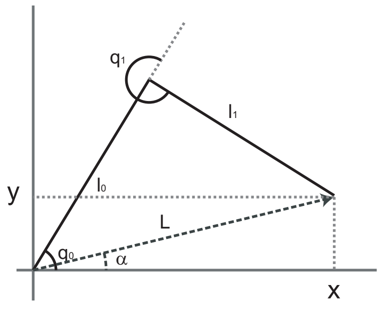

Fig. 2: A basic element as a main part for creating complex articulated structures

The basic element is described by two quantities – either $[x, y]$ in Cartesian coordinates or $[L, \alpha]$ in angle coordinates.

The basic elements constituting a complex structure are not connected at the end-points, but the next basic element is connected into medial joint of its predecessor, so they share one segment of the final complex structure. Each basic element is labeled by a pair of joint numbers within the structure – $i{:}i{+}2$ where i is the joint where the first segment starts and $i{+}2$ is the joint where the second (and last) segment finishes.

Each basic element lies in its local coordinate system, i.e., the coordinates $[x_{i,\,i+1}, y_{i,\,i+1}]$ or $[L_{i,\,i+1}, \alpha_{i,\,i+1}]$ are relative. The Cartesian coordinates of the basic element $i{:}i{+}2$ are computed by

$$x_{i,\,i+1} = \sum_{j=i}^{i+1} l_j \cos \underset{i}{q_j}, \quad y_{i,i+1} = \sum_{j=i}^{i+1} l_j \sin \underset{i}{q_j}, \tag{8}$$

where $\underset{i}{q_j} = \sum_{k=i}^{j} q_k$. Its origin is at the beginning of the corresponding basic element and is rotated by an angle equal to the sum of the previous angles from the status vector.

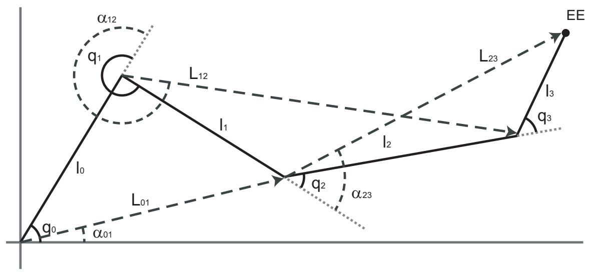

Fig. 3: A complex joint structure consisting of basic elements

For example, in Fig. 3, where three basic elements $0{:}2$, $1{:}3$ and $2{:}4$ form the complete structure, we assume a basic element $2{:}4$ described by $[L_{23}, \alpha_{23}]$, its coordinate system origin lies at the end of the second segment l_1 and is rotated by angle $\sum_{i=0}^{1} q_i$.

2.3 Extending Jacobian

According to the preceding section we can obtain the Jacobian with rank at most $n-1$ – the number of pairs in a complex structure. This is not enough to avoid problems with the Jacobian's singularity. Hence these rows are added into the standard Jacobian used for solving (3) – the rows in the standard Jacobian represent the main link $0-n$ (base to end-effector). Then the Jacobian looks as presented in equation (9).

Using such a Jacobian extension we can control the end-effector and also the behavior of the inner joints.

$$\boldsymbol{J}_{2n,n}(\boldsymbol{q}) = \begin{bmatrix} -l_0 \sin q_0 - l_1 \sin \underset{0}{q_1} & -l_1 \sin \underset{0}{q_1} & 0 & \cdots & 0 \\ 0 & -l_1 \sin q_1 - l_2 \sin \underset{1}{q_2} & -l_2 \sin \underset{1}{q_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -l_{n-1} \sin \underset{n-2}{q_{n-1}} \\ l_0 \cos q_0 + l_1 \cos \underset{0}{q_1} & l_1 \cos \underset{0}{q_1} & 0 & \cdots & 0 \\ 0 & l_1 \cos q_1 + l_2 \cos \underset{1}{q_2} & l_2 \cos \underset{1}{q_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & l_{n-1} \cos \underset{n-2}{q_{n-1}} \\ -\sum_{i=0}^{n-1} l_i \sin \underset{0}{q_i} & -\sum_{i=1}^{n-1} l_i \sin \underset{0}{q_i} & \cdots & \cdots & -l_{n-1} \sin \underset{0}{q_{n-1}} \\ \sum_{i=0}^{n-1} l_i \cos \underset{0}{q_i} & \sum_{i=1}^{n-1} l_i \cos \underset{0}{q_i} & \cdots & \cdots & l_{n-1} \cos \underset{0}{q_{n-1}} \end{bmatrix} \tag{9}$$

The i-th and $(i+n-1)$-th rows of the Jacobian correspond to the i-th basic element of the articulated structure. For instance, the 0-th and $(n-1)$-th rows correspond to the basic element designated as L_{01} in Fig. 3. The last two Jacobian rows describe the main link – from the base to the end-effector.

2.3.1 Extending d$\boldsymbol{x}$

Consequently as the Jacobian is extended, the displacement vector d$\boldsymbol{x}$ also has to be extended to fit the Jacobian's row dimension – $\dim \mathrm{d}\boldsymbol{x} = 2n$. The d$\boldsymbol{x}$ items represent the change of the corresponding basic element or main link $0:n$ (the last two rows). To set up the change of the main link $0:n$ is straightforward – it is the only input of the method. All the remaining items representing a change of the particular basic elements have to be derived from the change of the main link taking into account the desired physical constraints imposed on the joints.

As the displacement of the main link is easy to define and clear to understand, the definition of the displacements of a particular basic element is difficult to describe according to the required all-structure behavior. Future work and research remains to be done on this.

2.4 Solving Inverse Kinematics

Equation (3) can be comprehended as a *Linear System*

$$\boldsymbol{A}\cdot\boldsymbol{x}=\boldsymbol{b} \qquad (10)$$

and thus can be solved by any appropriate method [4]. Vector $\boldsymbol{x}$ is the vector of unknown variables. In the case of (3), vector $\boldsymbol{x}$ corresponds to vector d$\boldsymbol{q}$, which is being solved. Next, vector $\boldsymbol{b}$ represents the displacement of end-effector d$\boldsymbol{x}$ and matrix $\boldsymbol{A}$ corresponds to Jacobian $\boldsymbol{J}(\boldsymbol{q})$.

Since the Jacobian is normally a matrix with dimensions $(2n, n)$, it is necessary to employ an approximation method for solving (3). We decided to use the *Ordinary Least Square* (*OLS*) method based on normal matrices [10], as it minimizes the norm of residual vector $\|\boldsymbol{r}\|$:

$$\boldsymbol{r}=\boldsymbol{A}\cdot\boldsymbol{x}-\boldsymbol{b}.$$

However, any approximation method can be used [5]. *OLS* produces a square matrix, which can be already solved by the LU decomposition [4] that we decided to use.

3 Experiments and evaluation of results

We performed a comparison test of herein presented Extended Jacobian method with a method based on using pseudo-inverse (section 1.1) as presented in [11]. We decided to make a comparison with the Jacobian inversion method as it is one of the standard methods, together with Jacobian transposition (section 1.3) and CCD – Cyclic Coordinate Descent, for solving the *Inverse Kinematics* problem in the field of computer animation.

Our goal was to acquire the same behavior of both methods, but with good recovery from the singular case.

The testing structure consists of 6 segments with lengths $l_0=90$, $l_1=60$, $l_2=90$, $l_3=150$, $l_4=150$ and $l_5=90$ points. The starting configuration was determined by $q_i=0$. The starting point lies at [–230, 0]. The desired trajectory was formed by an ellipse with $a=400$ and $b=50$ with the mid--point at the origin. Therefore the end-effector touched the ellipse at the point [400, 0].

First we tried to minimize the movement of the inner parts and make a move to the end-effector only. We accomplished this by setting all d$x_i=0$ except those which correspond to a move of the end-effector. This only partly achieved the desired requirements. The structure recovered well from the singular case – see Fig. 4, but its behavior is different from the behavior of the method based on pseudo-inverse.

Due to the these problems with behavior, we tried to minimize the influence of the added rows by multiplying each added row by a positive non-zero constant $w>0$. We expected a minimizing effect of the added rows. We assumed the following: when $w\to 0$ then the Extended Jacobian will become the Non-Extended Jacobian $\boldsymbol{J}_{\mathrm{Ext}}\to\boldsymbol{J}_{\mathrm{NExt}}$. However, this did not achieve the required effect. With $w\to 0$ the behavior remained and the ability to recover from the singular cases disappeared.

The behavior of the tested methods is shown in the Figs. 5 and 6 – Fig. 5 shows the standard Jacobian Inversion method while Fig. 6 displays the herein presented Jacobian Extension method. The pictures show the movement of the structure during the first several steps.

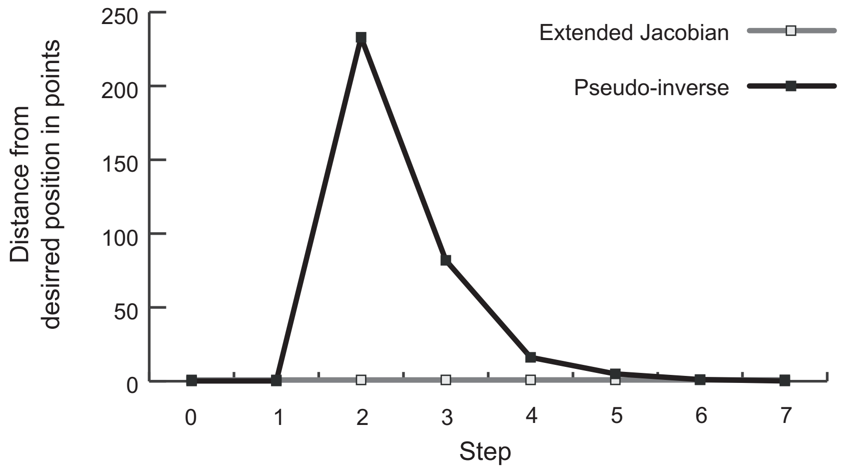

Fig. 4: Graph presenting the distance of the end-effector from the desired position

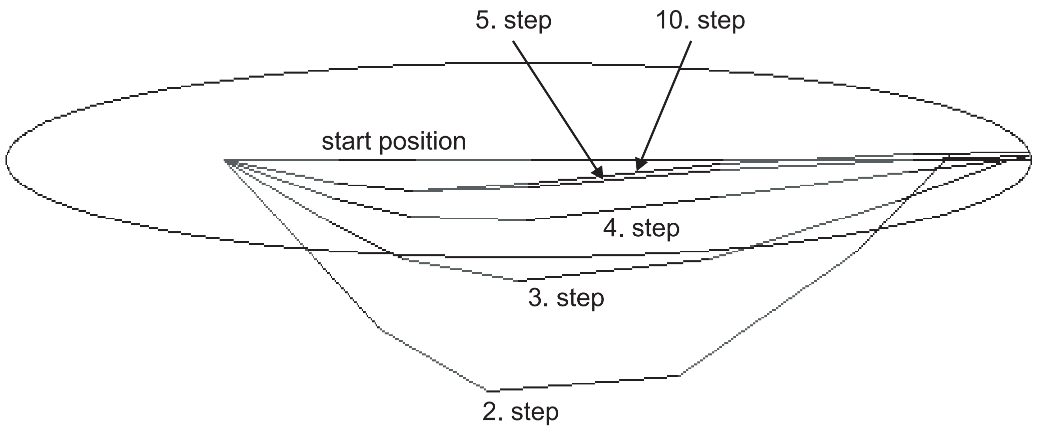

Fig. 5: Behavior of the Jacobian Inversion method during the test

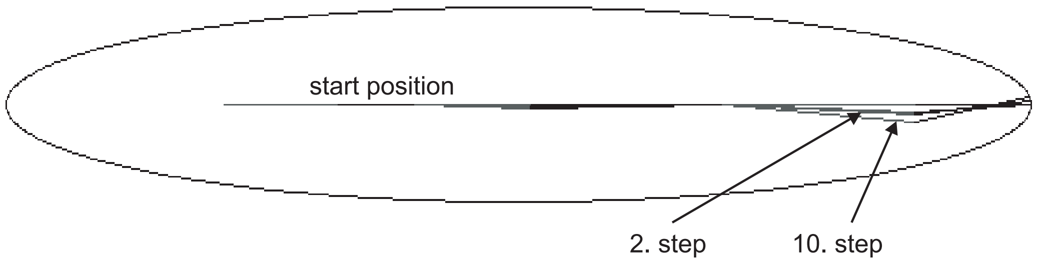

Fig. 6: Behavior of herein presented Jacobian Extension method during the test

The Jacobian Inversion method made a "big jump" when moving from the starting configuration. This was caused by a kinematic singularity and consequent numerical instability of the system. This did not happen with the Extended Jacobian method due to the fact its Jacobian cannot be singular in any case, i.e., its rank is always guaranteed to be n.

4 Conclusions

We have presented a brief overview of methods for solving the *Inverse Kinematics*, together with a new option for solving an identical problem actually in plane mainly for the purposes of computer graphics – for animating articulated structures. The articulated structure is treated by pairs to allow us to control the inner joints based on physical characteristics.

The method is based on an extension of the Jacobian. $2(n-1)$ rows are added into the Jacobian corresponding $n-1$ pairs – basic elements. The system is over-determined (the Jacobian is full-rank), so singular cases coming from usage of pseudo-inverse are avoided. Hence the method is stable.

Several tests were performed to compare presented method with the method based on pseudo-inverse only. The same behavior was required for both methods. Next, the ability to recover from singular cases was also required. In recovering from singular cases, the Extended Jacobian method passed successfully, but the same behavior for both methods was not achieved.

4.1 Future work

The usability of herein presented method in 3D space and exact control of the inner joints have not been evaluated yet.

For the first issue – exact structure control – the inverse transformation $\boldsymbol{f}^{-1}$ of (2) has to be found, either analytically or numerically. Then, a Sensitivity analysis can be performed to realize the dependencies in the system and hence discover the key to exact control.

The next remaining problem – usage in 3D space – seems not to be so difficult now. This is mainly due to the fact that the rotating joints usually operate in a single plane. However, this problem becomes difficult if the joints are capable of unlimited rotation instead of rotation in a specified plane.

The usability of the proposed technique is clear, for enabling animation of articulated structures with user-defined (expressed as functions) constraints imposed on the joints.

After the tasks mentioned above have been dealt with, the focus will turn to the performance and speed of the proposed solution, as speed is dominant requirement for real time computer animations. Also for this reason, the basic methods for solving the *Inverse Kinematics* problem, such as Jacobian inversion, Jacobian transposition, and CCD are only used in computer animation due to their relatively low computational complexity. To achieve the goal of speed we are planning to employ non-standard arithmetic, probably residual arithmetic, together with "special hard-wear" which is available on today's μ-processors, for instance MMX, SSE, 3DNow, etc.

References

[1] Badler, N. I., Phillips, C. B., Webber, B. L.: *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, 1993.

[2] Baillieul, J.: "Kinematic Programming Alternatives for Redundant Manipulators." In IEEE International Conference on Robotics and Automation, 1985, p. 722–728.

[3] Baillieul, J.: "Avoiding Obstacles and Resolving Kinematic Redundancy." In IEEE International Conference on Robotics and Automation, 1986, p. 1698–1704.

[4] Friedberg, S. H., Insel, A. J., Spence, L. E.: *Linear Algebra 3rd Ed*. Upper Saddle River: Prentice-Hall, 1997. ISBN 0-13-233859-9.

[5] Golub, G. H., Van Loan, C. F.: *Matrix Computations 3rd Ed*. Baltimore: The Johns Hopkins University Press, 1996. ISBN 0-8018-5414-8.

[6] Klein, C. A., Chu-Jenq, C., Ahmed, S.: "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, Vol. **11** (February 1995), No. 1, p. 50–55.

[7] Korein, J. U., Badler, N. I.: "Techniques for Generating the Goal-directed Motion of Articulated Structures." IEEE Transactions on Computer Graphics and Applications, November 1982, p. 71–81.

[8] Liegeosis, A.: "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms." IEEE Transactions on Systems, Man, and Cybernetics, Vol. **7** (1977), No. 12, p. 868–871.

[9] Paul, R. P.: *Robot Manipulators: Mathematics, Programming and Control.* Cambridge, Mass.: MIT Press, 1981.

[10] Van Huffel, S., Vandewalle, J.: *The Total Least-Squares Problem*. Philadelphia: SIAM, 1991. ISBN 0-89871-275-0.

[11] Šoch, M., Lórencz, R.: "Solving Inverse Kinematics – Jacobian Inversion Method in a Plane." In 11th International Workshop on Systems, Signals, Image Processing and Ambient Multimedia. Poznaň: PTETiS, 2004, p. 95–97. ISBN 83-906074-8-4.

[12] Whitney, D. E.: "Resolved Motion Rate Control of Manipulators and Human Prosthesis." IEEE Transactions on Man-Machine Systems, MMS-10, June 1969, No. 2, p. 47–53.

Ing. Martin Šoch
e-mail: sochm@fel.cvut.cz

Doc. Ing. Róbert Lórencz, CSc.
e-mail: lorencz@fel.cvut.cz

Department of Computer Science and Engineering

Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo nám. 13
121 35 Praha 2, Czech Republic