Inverse Kinematics - Basic Methods

Lukáš Bařinka, l.barinka@sh.cvut.cz
Ing. Roman Berka, berka@sgi.felk.cvut.cz

Dept. of Computer Science & Engineering
Czech Technical University
Prague / Czech Republic

Abstract

Inverse kinematics is used in several fields of applications and computer graphics is one of them. This technique has found a good utilization especially as a tool for animation of articulated structures. The insight of the inverse kinematics is based on basic knowledge in kinematics generally. This paper presents an overview of the methods used to solve the problems concerning inverse kinematics. The difference between forward and inverse kinematics is explained in the beginning of the text. The basic terms and definitions are explained in the background section. Then the methods, their advantages, and disadvantages are reviewed. Finally, some aspects of implementation are discussed.

1 Introduction

Kinematics in computer animation is usually divided into two basic parts – *forward kinematics* and *inverse kinematics*. Forward kinematics is based on the manipulation with the structure, that is done by changes of the joint angles inside the controlled structure (*Figure 1a*). Inverse kinematics is based on the direct manipulation with the end of the structure and the joint angles are derived from changes of the end of the structure (*Figure 1b*).

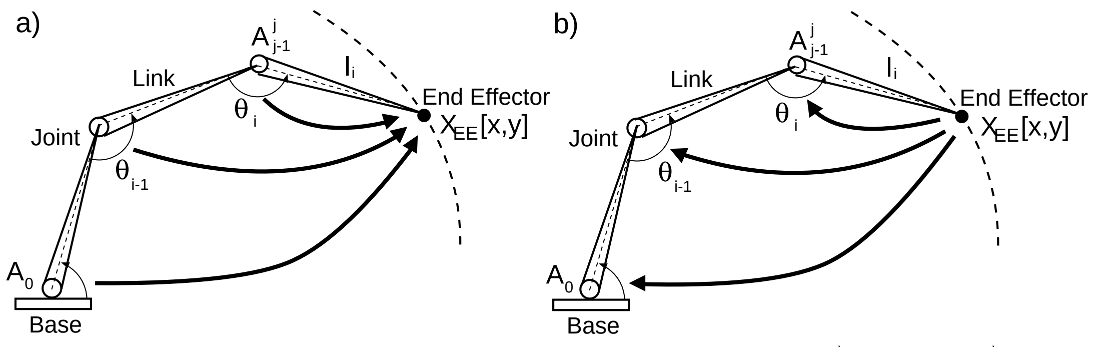

Figure 1: Process of manipulation of the structure through *a) forward b) inverse* kinematics.

Since inverse kinematics makes possible to manipulate the articulated structure by the end effector, it could be used in animation techniques for easy motion control.

The problems relating to the inverse kinematics are known for a long time and are in more detail described in the literature, e.g. [WW92], [Wel93], [Chi96].

2 Background

There are some substitutions in the animation process to reach simplicity in models describing the animated object. Control techniques used in inverse kinematics are often based on a skeleton consisting of links connected with joints (*Figure 1*). That model is usually denoted as the "articulated structure" [WW92]. There are several types of joints (revolute, prismatic, etc.) [WW92], [Mck91]. The computer animation is usually restricted to rotation joints.

The articulated models (*Figure 1*) have hierarchical structure where each link has its own coordinate system (*CS*) and is positioned relatively to the CS of the previous link. The position of the link i in the CS of its predecessor is described by the joint angle. Thus, every joint transformation is local. That fact assists in operations with hierarchical articulated structure (*Figure 1*). The transformation from the CS of any given link i to the world CS is given by concatenation of partial transformations between each two neighbor links from the base to the given link.

The first link in the articulated structure is a *base*. The end of the structure is an *end effector*. The position and the orientation of the base is expressed in the global coordinates. Every articulated structure has one or more end effectors and the motion control of the structure is done through these end effectors. In many cases, the end effector takes place at the end of the structure, like palm, finger, foot, head, etc.

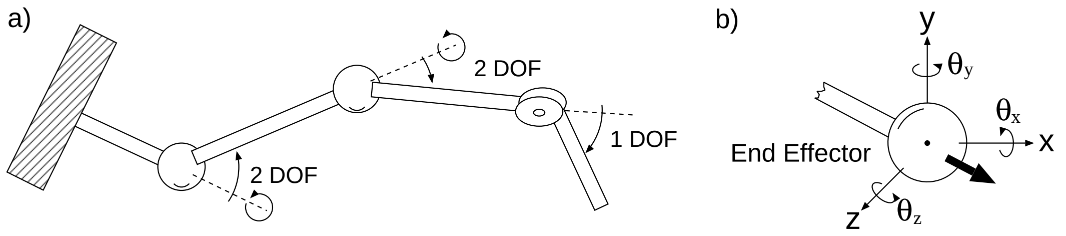

Figure 2: Degrees of freedom of a) the articulated structure (*5 DOF*) b) the end effector.

The state of the articulated structure is represented by a state vector $\theta(\theta_1, \cdots, \theta_n)$ in the joint (angular) space. The position and the orientation of the end effector is represented by the end effector position $X_{EE}(x, y, z, \theta_x, \theta_y, \theta_z)$ in Cartesian space. The relations between the state vector θ and the end effector position X_{EE} are expressed by equations (1) and (2).

Forward kinematics	Inverse kinematics
$X_{EE} = f(\theta) \quad (1)$	$(\theta) = f^{-1}(X_{EE}) \quad (2)$

The complexity of the articulated structure is expressed by the term *degree of freedom (DOF)*. The DOF of the articulated structure is the number of independent variables necessary to specify the state of the structure.

For example, if the joint could revolute in n axes, the degree of freedom of that joint is n. When the structure (*Figure 2a*) consists of three joints and two of them could revolute in two axes and one could revolute in one axe, the DOF of that structure is 5.

The notation for some characteristics differs in various works. The terms in this paper are defined as follows:

State vector (angles)	$\theta(\theta_1, \cdots, \theta_n)$
Transformations (generally) from link $i-1$ to link i	$A_{i-1}^{i}(\theta)$
Initial (base) global transformation of articulated structure	A_0
Total end effector position in the global coordinates	$A_n(\theta) = \prod_{i=1}^{n} A_{i-1}^{i}(\theta)$

As the inversion of the function f is not trivial in all cases, a number of approaches solving the computation of the state vector has been proposed. The next section gives an overview of several such methods.

3 Solutions

A number of methods and their combinations could be used to solve the inverse kinematics. Using each of them separately brings some advantages and also disadvantages. Therefore it is useful to combine them together and often combine them with additional approaches.

3.1 Algebraic methods

The algebraic solution of equation (2) exists only for a restricted class of cases. The joint angles could be expressed using the end effector position. The number of nonlinear equations increases with the DOFs (n DOFs mean n equations). Each joint angle – one by one – could be solved by the system of n equations [Chi96]:

$$\prod_{i=2}^{n} A_{i-1}^{i}(\theta_i) = A_1^{-1}(\theta_1) \cdot A_n \tag{3}$$

The $m + 1^{st}$ joint angle is expressed using the m previous angles. The last joint angle is expressed only by the end effector position and orientation.

The forward kinematics solution for the end effector position $X_{EE}(x, y)$ in 2 DOF structure could be expressed as follows:

$$X_{EE} = (l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2), \quad l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)) \tag{4}$$

Thus, by applying elementary trigonometry the inverse solution is:

$$\theta_2 = \cos^{-1}\frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2} \qquad \theta_1 = \frac{-(l_2\sin\theta_2)x+(l_1+l_2\cos\theta_2)y}{(l_2\sin\theta_2)y+(l_1+l_2\cos\theta_2)x} \tag{5}$$

As the DOFs for most of cases are higher, the state vector is not analytically expressible using such a trivial way. Therefore more sophisticated approaches are necessary.

3.2 Iterative methods

Iterative methods solve the inverse kinematics problem by using a sequence of steps leading to incrementally better solution for the joint angles. The goal is to minimize the difference between the current and desired positions of the end effector. The next sections explain methods leading to the incremental improvement in joint angles space.

3.2.1 Jacobian inversion method

The Jacobian is the multidimensional extension to the differentiation of a single variable [WW92]. As it was mentioned in the introduction, there is a relation between the Cartesian space of the end effector X and the joint space of the joint angles θ. The Jacobian, in fact, transforms the differential angle changes to the differential motions of the end effector [Mck91].

$$\dot{X} = J(\theta)\dot{\theta} \tag{6}$$

The vector $\dot{X}$ represents the linear velocity (dx, dy, dz) and rotational velocity $(\delta_x, \delta_y, \delta_z)$ of the end effector and $\dot{\theta}$ represents time derivative of the state vector (rotational velocity for each joint).

Since the unknown is $\dot{\theta}$, the Jacobian inversion is needed. Hence, the equation (6) is transformed to the form:

$$\dot{\theta} = J^{-1}(\theta)\dot{X} \tag{7}$$

Jacobian construction

If the analytic expression is known for equation (1), then the evaluation of the Jacobian could be done by straightforward differentiation. For example, equation (6) for a structure with 6 DOF in 6D Cartesian space (3D for position, 3D for rotation) – (*Figure 2b*) can be expressed as:

$$[\dot{x}, \dot{y}, \dot{z}, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z] = \left[\frac{\partial f_j}{\partial \theta_i}\right][\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5, \dot{\theta}_6] \tag{8}$$

If the analytic expression is unknown for equation (1), numerical construction for the Jacobian is used. The Jacobian is obtained column by column from the transformation matrices A_i [Chi96].

Iterative model

The Jacobian inversion method works in two phases. The partial transformations based on the joint angles are computed in the first phase. After that, the end effector position and the Jacobian are computed. Then the end effector location is changed.

The second phase contains Jacobian matrix inversion and joint angles changes, using equation (7). The next step lies in the repetition of step one and in the change of the end effector position. The obtained differential of the end effector position dX enters in phase two. The mentioned phases repeat until the error (difference between the current and the desired location of the end effector) comes below a defined value ε or the maximal number of iteration steps is reached: $\| J(d\theta) - dX \| \leq \varepsilon \quad \vee \quad iter \geq maxiter$ (*Figure 3*).

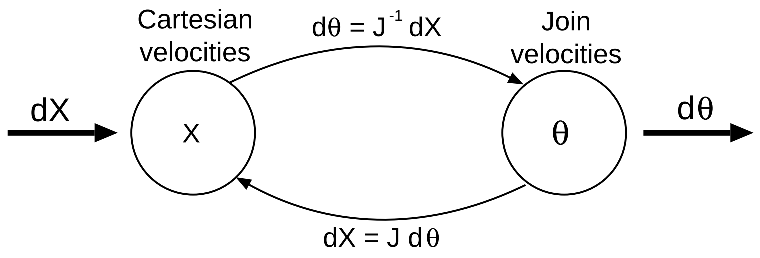

Figure 3: The iterative model for the Jacobian inversion method.

3.2.2 Optimization based method

The basic idea of optimization based method is to take a look at the primary equation $\theta = f^{-1}(X)$ as minimization problem [Wel93]. Thus, the previous equation could be transformed into

$$E(\theta) = (P - X(\theta))^2 \tag{9}$$

After that, standard iterative non-linear optimization techniques could be applied to minimize error function $E(\theta)$, where P is the goal position of end effector and $X(\theta)$ is the current position of the end effector [PFTV90].

Gradient-based optimization could be also used. This may increase the computation cost in each iteration step, but the convergence rate should be better and the number of iteration steps should decrease.

3.2.3 Cyclic coordinate descent (CCD)

Another interesting approach is represented by CCD method [Wel93], [Ebe01]. The CCD is based on minimization applied to each joint separately. The steps in one pass are ordered from the most distant segment to the base segment. The difference between this method and the previous one is that just one joint variable is modified in one step. A number of passes are made over the manipulator to find the global minimum of equation (9).

According to the fact, that the only one joint variable is changing along the minimization process, an analytic solution could be used. That significantly speeds up the minimization problem [Wel93].

3.2.4 Genetic programming

Genetic programming is very interesting approach often used in minimization process. The first way (local solution) of application is to use genetic programming as one of numerical methods to minimize equation (9). In the second way (global solution), this method could be employed to optimize motion of a model as whole.

The goal in the local solution is to achieve the desired position of one end effector, whereas in the global solution, the goal is to produce such a sequence of motions, so that the whole model can reach the desired position and orientation. The first approach is directly connected to solving inverse kinematics.

That means, the genetic programming based motion control could be applied not only to the level of partial movements, but it could be also used to control high level motions (like step, jump, etc.) [Srk99]. The motion simulation is controlled with partial abstraction and by goals for achievement.

There are conditions and limits constructed for each certain task and the evaluation function of success. A number of solutions are combined in each generation and the best solutions according to evaluation function are chosen. The next generation arises by crossing (hybridization) and mutating of best solutions from previous generation. The generations repeat until desired result is achieved.

3.2.5 Jacobian transpose method

Jacobian transpose method removes the problematic Jacobian inversion mentioned above. The annoying inversion is replaced by a simple transposition [Wel93]. The idea is based on the principle of virtual works and generalized forces [Pau81].

The external force $F = [f_x.f_y, f_z, m_x, m_y, m_z]^T$ (consisting of pull f and twist m) is applied to the end effector of the articulated structure and results internal forces and torques in the joints. The relation between the force F and the generalized forces τ is expressed as

$$\tau = J^T F \tag{10}$$

The generalized forces τ could be expressed using either the joint variable accelerations $\ddot{\theta}$ or joint velocities $\dot{\theta}$ [NN90]. The joint accelerations could be used for an accurate dynamic simulation of the manipulator motion. Because the method is not interested in the dynamic behavior, only the joint velocities $\dot{\theta}$ are used for the necessity of this method. Thus equation (10) is supplied by another form:

$$\dot{\theta} = J^T F \tag{11}$$

The force is proportional to the velocity in equation (11). That means the object moves as long as the force takes effect. The inertia and torques are not applied. The scheme of the iterative model is demonstrated by the *Figure 4*.

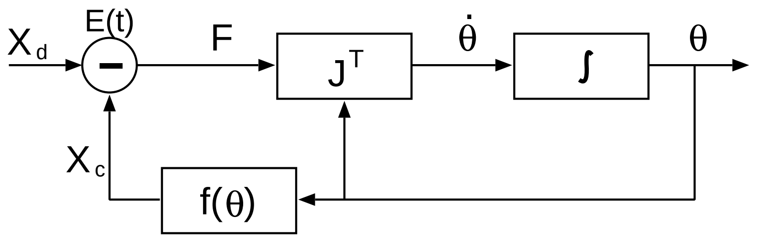

Figure 4: Iterative model adopted for the Jacobian transposition method.

The force F corresponds to error function $E(t)$ expressed by equation $E(t) = X_d(t) - X_c(t)$. E is the error along the time-varying trajectory, X_c is current position and X_d is desired trajectory of the end effector.

3.3 Summary of problems and benefits

Algebraic method

The disadvantage of this method is that it does not guarantee a solution for general structure and even if it exists, it is complicated for increasing DOFs. The second handicap is that the solution is not unique and therefore is not continuous.

However, the generalized solution can be derived for a up to 6 DOF articulated structure (often used in robotics). A solution also exists for structures with more than 6 DOF, when the features of the structure satisfy some conditions [Cra89], [MC94], [MZ94], [Chi96].

Jacobian inversion

One of the problems associated with the Jacobian inversion method is the matrix inversion. The Jacobian matrix dimension grows with the degree of freedom. The computation cost of the matrix inversion is growing quite fast with the dimension. Therefore, this method becomes consuming a lot of time for articulated structures with a large number of DOF (e.g. snake, chain, etc.).

In the case when the degree of freedom is different than 6 in 3D space (3 for location, 3 for rotation) — generally, if the dimension of X is not equal to the dimension of θ, the Jacobian matrix is rectangular and consequently not invertible. In that case, pseudo-inversion could be used for the rectangular matrix inversion by singular value decomposition [GK65]. The disadvantage of the pseudo-inversion is that some numerical errors appears, because that method is approximate and local. If the change of X is too large, according to the facts mentioned above, errors often occur. These errors are called "tracking errors" [WW92]. The solution of that problem is to divide the path into smaller steps.

When the rank of the Jacobian matrix differs from the DOFs (some rows are linearly dependent), the number of solutions is infinite and the matrix is not invertible. Singularities usually occur in the full extended state (*Figure 5a* — the

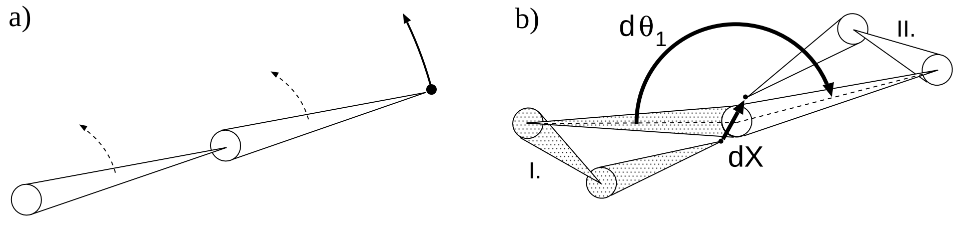

Figure 5: a) Singular state. b) Ill conditioned state.

change of θ_1 and/or θ_2 causes the same differential movements of the end effector). There is not just a single solution in that situation, a choice has to be done, or the prevention of such states (e.g. used in robotics – singular states are not allowed).

If the structure is near the singular state or at the transition between singular and nonsingular state, high velocities and oscillations occur. Example of such a problematic state is shown in *Figure 5b*. In the example $\Delta X \rightarrow 0$, $\Delta\theta \not\rightarrow 0$. It means that the distance in Cartesian space is small, but the distance in joint space is large (compared to the Cartesian space). That state is called "ill conditioned" [WW92].

The good feature of this method is its expected behavior. The articulated structure behaves like rubber. This method is also quite fast for smaller articulated structures therefore it could be used in the real time animations.

Optimization

In the interactive environments, there must be used such an optimization algorithm, running as fast as the refresh rate needs. That could be a problem, because some optimization-based algorithms are quite "dumber". When such algorithm is used, some problems could appear. For example, the algorithm jams in the local minimum and is not able to find the global solution. Some more sophisticated algorithms may be used, but the time cost could be probably higher.

The good idea is the use of any minimization solver which could be applied like plug-in module. It is easy to change the solver and it allows usage of handy solver according to a specific situation. There are not problems with matrix inversion as well.

Cyclic coordinate descent

This method works fine for simple structures, but for more complicated ones could be worse. When the change must be done near the base, the algorithm must pass all the joints in the path from the end effector to the changing joint and therefore the computation slows down [Wel93]. The next feature of the CCD method is that it produces motion more like a void chain than a rubber. That behavior could hamper in the figure animation.

The nice feature of the CCD method is that this method is free of singularities and it does not include matrix inversion. In many cases, only few passes are enough to achieve a sufficient precision and therefore this method could be used in real time applications.

Genetic programming

Genetic programming based methods are very interesting, but very slow in general. Also, an appropriate solution is not guaranteed (in reasonable time).

On the other hand, this method is useful for more complex and independent motion control and animation, as e.g. walk, creep, etc. Also multiple re-usability of that approach is possible.

Jacobian transposition

Some troubles like in the Jacobian inversion are still there, e.g. singularities, ill conditioning, but one huge problem – matrix inversion – disappears (different DOF and state space dimension).

4 Implementation

Creation of the articulated structure is basic and important first step in animation. Sufficient time should be dedicated to structure abilities specification. It could spare a lot of time in implementation.

For easiness in the implementation, it is handy to select the proper model representation. The model should be simple on the one hand and general on the other. Than a standard (general) algorithm could be applied without larger customization. Such a representation is also useful in various programs and environments.

The last but not least advise is to strictly follow the transformation chain and the sequence of partial operations, because the operations are not commutative in general, and in opposite case, the structure could have a strange behavior. The same order of operations is also needed in a hardware implementation.

5 Conclusions

Inverse Kinematics experienced a great evolution in the last twenty years. Many methods to solve have been developed and many approaches applied. A special branch was created around these ideas and other thoughts are still coming.

The basic methods were approached — *Jacobian inversion, Jacobian transposition, Optimization, Cyclic coordinate descent, Genetic programming.* The main ideas in each method were described. A comparison was based on access to problem and ideas of solutions. Every method mentioned above could be applied as solver for inversion kinematics in articulated figure animation. Advantages and disadvantage of these methods were mentioned in *Section 3.3.*

The basis for further work should be the implementation of several methods and comparison based on features and behaviors. The interesting objective is also to combine some of the methods together and thus, eliminate some problems. Also, a comparison of possibilities to incorporate constraints is a possible way to explore.

References

[Chi96] *K. W. Chin.* Closed-form and generalized inverse kinematic solutions for animating the human articulated structure. *Curtin University of Technology, 1996.*

[Cra89] *J. J. Craig.* Introduction to robotics: Mechanics and control. *Addison–Wesley, 1994.*

[Ebe01] *D. H. Eberly.* 3D game engine design. *ISBN 1-55860-593-2, 2001.*

[GK65] *G. Golub, W. Kahan.* Calculating the singular values and pseudoinverse of a matrix. *Journal SIAM Numerical Analysis, 1965.*

[Mck91] *P. J. McKerrow.* Introduction to Robotics. *Addison-Wesley, Electronic Systems Engineering Series, Wokingham, 1991.*

[MC94] *D. Manoch, J. F. Canny.* Efficient inverse kinematics for general 6r madipulators. *IEEE Transactions on robotics and automation, 1994.*

[MZ94] *D. Manoch, Y. Zhu.* A fast algorithm and system for the inverse kinematics of general serial manipulators. *IEEE Conference on robotics and automation, 1994.*

[NN90] *Z. R. Novakovic, B. Nemec.* A solution of the inverse kinematics problem using the sliding mode. *IEEE Transactions on robotics and automation, 1990.*

[Pau81] *R. P. Paul.* Robot manipulators: Mathematics, Programming and control. *MIT Press, Cambridge, MA, 1981.*

[PFTV90] *W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling.* Numerical recipes in C - The art of scientific computing. *Cambridge University Press, Cambridge, MA, 1990.*

[Srk99] *P. Srkal.* The animation of evolutionary developing structures (in Czech). *Czech Technical University of Prague, 1999.*

[Wel93] *Ch. Welman.* Inverse kinematics and geometric constraints for articulated figure manipulation. *Simon Fraser University, 1993.*

[WW92] *A. Watt & M. Watt.* Advanced animation and rendering techniques. *Addison–Wesley, 1992.*