An inverse kinematics method for 3D figures with motion data

Taku Komura, City Univercity of Hong Kong, itaku@cityu.edu.hk
Atsushi Kuroda, GSport Inc, kuroda@gsport.co.jp
Shunsuke Kudoh, University of Tokyo, kudoh@cvl.iis.u-tokyo.ac.jp
Tai Chiew Lan, Hong Kong University of Science & Technology,taicl@cs.ust.hk
Yoshihisa Shinagawa, University of Illinois at Urbanna Champaign, sinagawa@uiuc.edu

Abstract

This paper presents a new inverse kinematics method that utilizes the motion data for realtime control and editing. The key idea is to extract parameters necessary for inverse kinematics from the motion data. These parameters are the weight matrix, which determines the motion of the redundant joints, and the transformation functions that define the motion of the end effectors. User can control the motion by dragging a body segment using a mouse, and the method calculates the new motion using the pre-computed parameters. The method enables interactive editing, warping, and retargeting character motions.

Keywords: *inverse kinematics, motion capture, human animation*

1 Introduction

Interactive graphics tools are attractive means for design and communication. However, despite their attractiveness, interactive technologies for 3D graphics such as VRML and Java3D, have been slow in gaining popularity. The following are two main reasons. First, broadband internet is not widely available for users to download large data sets. Second, interactive technologies for handling complex 3D characters in cyberspace have not been well developed.

Inverse kinematics is one of the most popular method for interactive control of hierarchical characters in 3D space. It addresses the problem of calculating the joint angles of the body segments given the motion of some body segments in 3D space. As the degree of freedom of the character is usually larger than the degree of freedom of the motion specified by the user in cyberspace (usually 2 or 3), this is an under-constrained problem, which has infinite number of solutions.

Although many methods have been proposed to solve inverse kinematics problems, very little research tried to extract features out of motion data for application in inverse kinematics control. In this paper, we propose a new inverse kineatics method to interactively control 3D multibody characters together with motion data. First, we extract the characteristics of the motion by calculating the weight matrix and the relative velocity of the end effectors "throughout the animation sequence". Then, users are allowed to pick and drag some body segments in a frame using the mouse. The motion of the entire body is then determined by the pre-calculated parameters. Since the response of the character is based on characteristics extracted from the motion data, the character posture is edited while retaining the basic features of the motion. With realtime response, the method is suitable as a user interface for interactive manipulation of 3D contents.

2 Related Work

Inverse kinematics is one of the oldest methods for interactively control of 3D multibody systems. Researchers in the robotics area used inverse kinematics to control robot manipulators. As the demand for creating 3D character animation increased, inverse kinematics method plays an important role in determining the keyframe postures of such characters.

Phillips et al [6] created an interactive inverse kinematics system that can handle multiple constraints. They later extended it to control bipedal articulated figures [5]. Zhao and Badler [11] proposed a numerical inverse kinematics method for determining the human posture, which is also applicable to any tree structured system. Mas et al [4] and Boulic et al [1] proposed an inverse kinematics method that allows the control of the center of mass, Koga et al [2] created a motion planning system for the arms that can move objects from one position to another. They used an inverse kinematics algorithm based on neurophysiology to determine the joint angles of the arms and shoulders from the position of the hands.

These days, it is becoming more popular to use the motion capture device to create human animation. In this pa-

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE

per, we propose a new motion control method based on inverse kinematic. It allows motion to be edited while retaining the essence of the motion data attached to the body model. There have been research efforts that use motion capture data to solve inverse kinematics problem. They come under two categories.

One category uses the motion data to define a function that determines the relationship between the position of the end effectors and the redundant joint angles[10, 7]. Wiley et al[10] synthesized the target posture by interpolating various similar postures in the database. Their method requires $O(2^D)$ examples to create the motion, where D is the number of postures of the human figure. Rose et al[7] improved this performance by reducing the number of examples needed to $O(D)$. A limitation of this method is that a great amount of motion similar to the target posture must be prepared in advance.

Another category uses only a single motion data to solve the redundancy problem of inverse kinematics. Tak et al[8] solved the redundancy problem and then calculated the trajectory of the end effectors so that the posture of the body resembles the posture in the original motion. Our method also comes under this category. However, in contrast to the approach in [8], we extract the feature of the motion at the moment the user controls the model, and apply it to solve the redundancy problem and move the end effectors. Therefore, although the approach by [8] is superior for retargeting motion, our approach has an advantage in creating new postures that retain the feature of the original motion.

3 Inverse Kinematics

Let us specify the posture of the human body model by the vector with degree of freedom (DOF) of the model:

$$\theta = (\theta_0, \theta_1, \ldots, \theta_{n-1})$$

where n is the DOF of the human body. Let us also represent the position and rotation of body segment i_s in Cartesian space by

$$(P_{i_s}, \Theta_{i_s}).$$

where P_{i_s} is a three dimensional vector for segment i_s's position, and Θ_{i_s} is the euler angle vector. Suppose the current state of the body is

$$\theta_0 = (\theta_0^0, \theta_1^0, \ldots, \theta_{n-1}^0),$$

and the state of segment i_s is

$$(P_{i_s}^0, \Theta_{i_s}^0).$$

Obviously, the state of segment i_s is a function of the body's state vector:

$$(P_{i_s}^0, \Theta_{i_s}^0) = g(\theta_0^0, \theta_1^0, \ldots, \theta_{n-1}^0).$$

Inverse kinematics problem can be defined as follows:
Problem: When segment i_s moves from $(P_{i_s}^0, \Theta_{i_s}^0)$ to $(P_{i_s}^0 + \Delta P, \Theta_{i_s}^0 + \Delta\Theta)$, how do the values of the state vector change? This problem is redundant, because n, the DOF of the body, is larger than the dimension of $(\Delta P, \Delta\Theta)$, which is six. The relationship between $(\Delta P, \Delta\Theta)$ and the incrementation of the state vector can be written in the following form:

$$\begin{pmatrix} \Delta P \\ \Delta\Theta \end{pmatrix} = J\Delta\theta. \quad (1)$$

The set of $\Delta\theta$ that satisfies equation (1) can be written in the following form:

$$\Delta\theta = J^+ \begin{pmatrix} \Delta P \\ \Delta\Theta \end{pmatrix} + (I - J^+ J)k$$

where k is an arbitrary vector, and J^+ is called the pseudo inverse matrix of J, which can be calculated by

$$J^+ = (J^T J)^{-1} J^T.$$

Whitney [9] proposed a method to solve the inverse kinematics problems by optimizing a quadratic form which can be written as:

$$Q(\Delta\theta) = \Delta\theta^T W \Delta\theta$$

where W is a $n \times n$ positive definite symmetric matrix, that is called the weighting matrix.

$\Delta\theta$ that optimizes this form can be obtained by

$$\Delta\theta = W^{-1} J^T (J W^{-1} J^T)^{-1} \begin{pmatrix} \Delta P \\ \Delta\Theta \end{pmatrix}. \quad (2)$$

Therefore,

$$\Delta\theta = J^+ \begin{pmatrix} \Delta P \\ \Delta\Theta \end{pmatrix}$$

gives a solution that minimizes the norm of $\Delta\theta$.

For example, the state vector of a 4DOF robot manipulator shown in figure 3 can be written by $\theta = (\theta_0, \theta_1, \theta_2, \theta_3)$ where θ_i is the joint angle of joint i. In this case, the weighting matrix is a 4×4 matrix.

If the following two matrices W_1, W_2:

$$W_1 = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix}, W_2 = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix}$$

are used as the weighting matrix to move the end effector of this manipulator to the left, the joint angles of the system would change as shown in right side of figure 3. This means that the incrementation of joint angle j is determined by the value of element (j, j) of the weighting matrix. When the value of element (j, j) is large compared with the other elements, the incrementation of joint angle j is small. On the other hand, if it is small, the incrementation is large.

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE

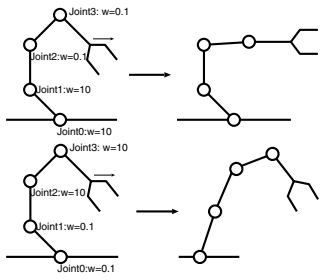

Figure 1. Using two different weighting matrix for inverse kinematics

4 Calculating the inverse Jacobian matrix from the motion

In previous inverse kinematics methodologies, constant values such as inertia or mass of the body segments were used as the weighting matrix elements. However, when a multibody system such as a human body moves in 3D space, it is obvious that such values change all the time, because the distribution of the joint angles changes according to the motion. To accurately reproduce the motion by inverse kinematics, it is necessary to change the weighting matrix according to the posture and motion of the body. In this section, we propose a method to calculate the weighting matrix of the multibody system from its real character motion. Once the weighting matrix is calculated for each posture, the user can change the position or rotation of each segment any time during the motion using inverse kinematics with the weighting matrix obtained in advance.

Suppose the velocity of the system is $\dot{\theta}_i$ at time t_i, and the translation and rotation velocity of end effector j in the Cartesian space is $(\dot{P}_i^j, \dot{\Theta}_i^j)$ $(j = 0, 1, ..., m-1)$ where m is the number of end effectors. All these data can be obtained from a motion data. The relationship between $\dot{\theta}_i$ and all the $(\dot{P}_i^j, \dot{\Theta}_i^j)$ can be written in the following form:

$$\dot{\theta}_i = (J_0^{\#}, ..., J_{m-1}^{\#}) \begin{pmatrix} \dot{P}_i^0 \\ \dot{\Theta}_i^0 \\ \vdots \\ \dot{P}_i^{m-1} \\ \dot{\Theta}_i^{m-1} \end{pmatrix} \quad (3)$$

where $J_0^{\#}, ..., J_{m-1}^{\#}$ are the inverse Jacobian matrices of the end effectors. If an assumption is made here that $J_0^{\#}, ..., J_{m-1}^{\#}$ calculates a velocity vector that minimizes some quadratic form, according to equation 2, equation 3 can be rewritten in the following form:

$$\dot{\theta}_i = W^{\#^{-1}} J_{all}{}^T (J_{all} W^{\#^{-1}} J_{all}{}^T)^{-1} (\dot{r}). \quad (4)$$

where $W^{\#}$ is the weighting matrix, J_{all} is a matrix that vertically lined the Jacobian matrix of all the end effectors: and $\dot{r}$ is a vector that vertically lined down the velocity of the end effectors in the cartesian space:

$$\dot{r} = (\dot{P}_i^0, \dot{\Theta}_i^0, \cdots \dot{P}_i^{m-1}, \dot{\Theta}_i^{m-1})^T$$

Think about solving for the weighting matrix using equation 4. Since we have only n equations as equality constraints, it is possible to solve only for n unknown variables. If we make an assumption here that $W^{\#}$ is a diagonal matrix, the number of unknown variables in this matrix can be decreased down to n:

$$W^{\#} = \begin{pmatrix} w_0 & 0 & ... & 0 \\ 0 & w_1 & ... & 0 \\ \vdots & \vdots & ... & 0 \\ 0 & 0 & ... & w_{n-1} \end{pmatrix},$$

where $w_0, ..., w_{n-1}$ are positive values. Then, equation 4 can be rewritten in the following form

$$\dot{\theta}_i = f(w_0, ..., w_{n-1})\dot{r}.$$

where f is a newly defined nonlinear function. By defining another function g in the following form:

$$g(w_0, ..., w_{n-1}) = \dot{\theta}_i - f(w_0, ..., w_{n-1})\dot{r},$$

solution for the weighting matrix can be obtained by solving for $(w_0, ..., w_{n-1})$ that satisfies $g(w_0, ..., w_{n-1}) = 0$. We have tested various methods to calculate $(w_0, ..., w_{n-1})$ such as the Newton-Rapson method, sequential quadratic programming (SQP) and so on. The best results could be obtained using the SQL method by minimizing $g^2(w_0, ..., w_{n-1})$. The advantage of using the SQL method is that it can handle inequality constraints which is necessary to keep the weighting matrix possitive symetric. We have used the library called *CFSQP* [3] for solving this problem.

In summary, the following optimization problem is solved to calculate the weighting vector: $(w_0, ..., w_{n-1})$.

$$\boxed{\min_{w_0,...,w_{n-1}} g^2, where, w_0 > 0, ..., w_{n-1} > 0.}$$

The inequality constraints $w_0 > 0, ..., w_{n-1} > 0$ are added to keep the weighting matrix a positive symmetric.

5 Calculating the relative velocity of the end effectors

If the weighting matrix is known, the velocity of the joint angles between the root of the body and the controlled segment can be calculated using inverse kinematics.

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE

The velocity of the DOFs which do not lie between the controlled segment and the root of the body are not determined unless the motion of the rest of the end effectors are specified. However, such motion must keep the characteristics of the original motion. In this paper, this is done by using the relative velocities of the end effectors in the original motion. Suppose at time t_i, the velocity of segment p is $(\dot{P}_i^p \ \dot{\Theta}_i^p)$, and the translation and rotation velocity of the end effectors $e^0, ..., e^{m-1}$ could be written by $(\dot{P}_i^{e^j} \ \dot{\Theta}_i^{e^j})$. We define here the functions that determine the relative motion of the end effectors and segment p by

$$T_i^{p,e_1}, ..., T_i^{p,e_m}.$$

These functions can be written in the following form:

$$\begin{aligned} T_i^{p,e_1}(\dot{P}_i^p, \dot{\Theta}_i^p) &= (\dot{P}_i^{e_1}, \dot{\Theta}_i^{e_1}) \\ T_i^{p,e_2}(\dot{P}_i^p, \dot{\Theta}_i^p) &= (\dot{P}_i^{e_2}, \dot{\Theta}_i^{e_2}) \\ &\vdots \\ T_i^{p,e_m}(\dot{P}_i^p, \dot{\Theta}_i^p) &= (\dot{P}_i^{e_m}, \dot{\Theta}_i^{e_m}) \end{aligned}$$

There are various solutions for $T_i^{p,e_1}, ..., T_i^{p,e_m}$. In this paper, we define them by a combination of rotation and scaling functions:

$$T_i^{p,e_j} = (R_t^{p,e_j} S_t^{p,e_j}, R_r^{p,e_j} S_r^{p,e_j})(j = 0, ..., m-1)$$

where R_t^{p,e_j} is a rotational transformation, and S_t^{p,e_j} is a linear scaling transformation. The axis of the rotation is defined by

$$\dot{P}_i^p \times \dot{P}_i^{e_j},$$

where $\times$ is an outer product operator. The rotation angle is calculated by

$$acos(\dot{P}_i^p \cdot \dot{P}_i^{e_j}),$$

where $\cdot$ is an inner product operator. The transformation matrix of S_t^{p,e_j} can be written by

$$\begin{pmatrix} \frac{\|\dot{P}_i^{e^j}\|}{\|\dot{P}_i^p\|} & 0 & 0 \\ 0 & \frac{\|\dot{P}_i^{e^j}\|}{\|\dot{P}_i^p\|} & 0 \\ 0 & 0 & \frac{\|\dot{P}_i^{e^j}\|}{\|\dot{P}_i^p\|} \end{pmatrix}$$

where operation $\|x\|$ returns the norm of vector x. To avoid $\frac{\|\dot{P}_i^{e^j}\|}{\|\dot{P}_i^p\|}$ becoming too large, an upper limit is prepared for this scaling.

By defining $T_i^{p,e_1}, ..., T_i^{p,e_m}$ as explained, it is possible to determine the motion of all the end effectors relative to the motion by segment p. As the user move segment p to arbitrary direction, the motion of end effector e_j is calculated by function T_i^{p,e_j}. These motions are based on the original motion, and if p is transfered to the direction parallel to the original motion, the end effectors move as they do in the original motion data.

As the motion of the end effectors relative to the motion of the controlled segment p is known, the last work to be done is to determine the incremental vector for the redundant joints. This is done using the inverse kinematics method explained in the previous section.

As a result, the user can use inverse kinematics to control segment whole body while taking into account the characteristics of the original motion data. The motion created using this method keeps the essense of the original motion.

6 Experimental results

An experiment to calculate the inverse kinematics parameters such as the weighting matrix and relative velocity of the end effectors from motion data was done. Then, using these parameters, the multibody systems were controlled using inverse kinematics in the Cartesian space and their motions were observed. A 60 DOF human body model (DOF 60) shown in Figure 2 was used for this experiment. The method proposed in this paper was applied to pitching

Figure 2. Human body model

motion (Figure 3), dancing motion (Figure 4), and running motion (Figure 5). First a motion was loaded, then, the

Figure 3. The human body model with the pitching motion

weighting matrix was calculated through the motion, and finally, some part of the body were dragged to arbitrary directions to check the effects of inverse kinematics.

In Figure 6, the results of applying our inverse kinematics method to a pitching motion is shown. In this example,

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE

Figure 4. The human body model with the dancing motion

Figure 5. The running motion

the left hand is dragged to various directions. It is possible to observe that all the body segments are moving at the same time. The motion of the other end effectors (right hand, left and right foot, neck) are defined based on the basic motion data.

The effects of applying the same operation to the dancng motion is shown in Figure 7. The left hand is dragged in this example as well.

Next, an example to edit a posture during a running motion is shown in Figure 8. The posture shown in Figure 8 (a) (side view) and (b) (frontal view) is a snapshot of the original motion. Think about creating a posture jumping over something while running such as one shown in Figure 8 (c) (side view) and (d) (frontal view), from the posture shown in Figure 8 (a)(b). Using previous inverse kinematics method, the user has to go through a number of procedures, such as pulling up the left leg, moving the left hand to the front upward, tilting the chest, and so on, to reach the final posture. However, using our method the user only has to drag the left foot to the upper front to obtain the jumping posture. This is because our inverse kinematics method keeps the feature of the original motion.

7 Discussion

As seeing the results shown in Figure 6, 7, and 8, it is possible to observe the effect of adding parameters of the original motion to the inverse kinematics procedure.

One concern about motion editing is keeping kinematical constraints. For example, the feet must keep ground-contact constraints, and sometimes the hands must keep contact with some other end effectors of the body. Since we use relative velocity of the end effectors calculated from the motion for inverse kinematics control, such kind of constraints are naturally kept. This is one large advantage of our method.

However, there are some limitations of our method. Since the interactive motion by the user depends only on the relative velocity at the time the posture is edited, rhythmical high frequency motion, for example during dance such as Samba, do not appear duing the interactive control by inverse kinematics.

Since the weighting matrix of the motion is calculated only by the motion, joint limits are not taken into account, and thus the motion can look unnatural when a segment is pulled in a great amount.

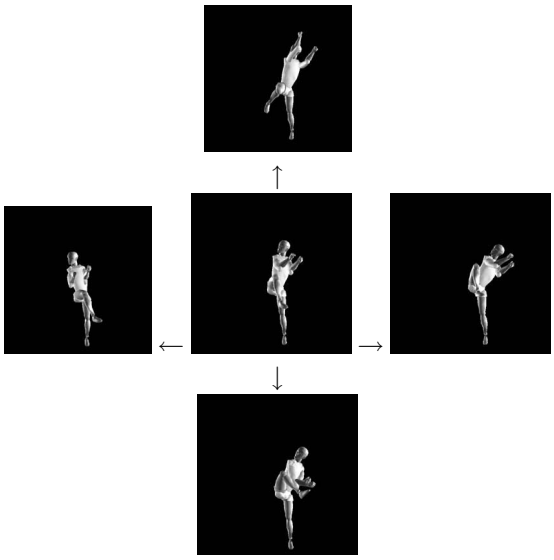

Figure 6. Using inverse kinematics method proposed in this study to control the pitcher's left hand

8 Conclusion and Future Work

In this research, we have proposed a new method that enable users to directly handle motion data of characters using inverse kinematics. Our inverse kinematics method takes into account the characterstics of the original motion, and therefore, users can easily edit the posture of the figure during the motion. The method is based on the concept to extract parameters from the motion: calcluating the weighting matrix and the relative velocities of the end effectors during the motion, and using them for inverse kinematics operation.

For future work, the following extensions can be considered:

- In this research, we only used a single motion to determine the effects of inverse kinemtics. It is also possible

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE

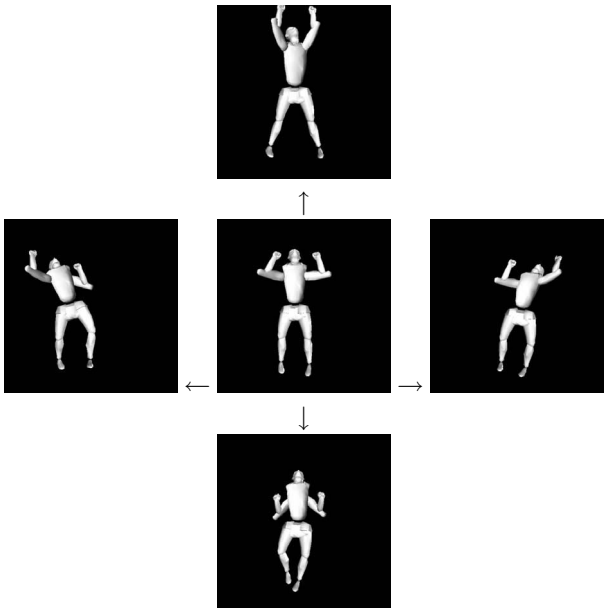

Figure 7. Using inverse kinematics to control the dancer's left han

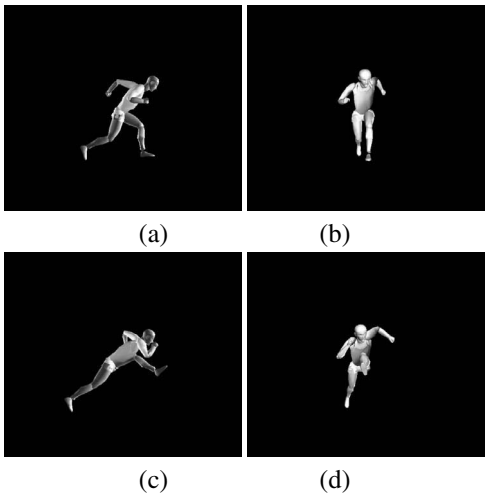

Figure 8. Creating a jumping posture ((c) front view (d) side view) from a running posture ((a) front view (b) side view)

to calculate the inverse dynamics motion by taking into account a series of similar but different motions. Using multiple data to determine inverse kinematics parameters, the inverse kinematics engine can be more robust.

- Our idea can be easily extended to multibody models whose scaling factors change during the motion. Scaling effects are often used in computer animations to exaggerate the feelings of the characters, or to emphasize physical phenomena such as crashes. By adding such kinds of effects to our inverse kinematics engine, users can have more exciting interaction with the character models.

References

[1] R. Boulic, R. Mas-Sanso, and D. Thalmann. Complex character positioning based on a compatible flow model of multiple supports. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):245–261, 1997.

[2] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. *Computer Graphics*, 28:395–408, 1994.

[3] C. Laurence, J. L. Zhou, and A. L. Tits. *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear ((Minimax) Optiization Problems, Generating Iterates Satisfying All Inequality Constraints*. Electrical Engineering Department and Institute for Systems Research, University of Maryland, 1997.

[4] R. Mas, R. Boulic, and D.Thalmann. A robust approach for the control of the center of mass with inverse kinetics. *Computers and Graphics*, 20(5):693–701, 199.

[5] C. B. Phillips and N. I. Badeler. Interactive behaviors for bipedal articulated figures. *Computer Graphics*, 25(4):359–362, 1991.

[6] C. B. Phillips, J. Zhao, and N. I. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics*, 24(2):245–250, 1990.

[7] C. Rose, P.-P. J. Sloan, and M. F. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3), 2001.

[8] S. Tak and H. Ko. Example guided inverse kinematics. *Proceedings of the International Conference on Computer Graphics and Imaging (CGIM) 2000*, 2000.

[9] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10:47–53, 1969.

[10] D. Wiley and J. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17:39–45, 1997.

[11] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

Proceedings of the Computer Graphics International (CGI'03)
1530-1052/03 $17.00 © 2003 IEEE