Natural Motion Animation through Constraining and Deconstraining at Will

Katsu Yamane, *Member*, *IEEE*, and Yoshihiko Nakamura, *Member*, *IEEE*

Abstract—This paper presents a computational technique for creating whole-body motions of human and animal characters without reference motion. Our work enables animators to generate a natural motion by dragging a link to an arbitrary position with any number of links pinned in the global frame, as well as other constraints such as desired joint angles and joint motion ranges. The method leads to an intuitive pin-and-drag interface where the user can generate whole-body motions by simply switching on or off or strengthening or weakening the constraints. This work is based on a new interactive inverse kinematics technique that allows more flexible attachment of pins and various types of constraints. Editing or retargeting captured motion requires only a small modification to the original method, although it can also create natural motions from scratch. We demonstrate the usefullness and advantage of our method with a number of example motion clips.

Index Terms—Animation, online inverse kinematics computation, multiple constraints, motion editing, joint motion range.

1 Introduction

Current technologies of creating realistic motion of human characters rely heavily on either an animator's skill or on motion capture techniques. Moreover, even if a motion clip is created through hard work, it is difficult to modify it for reuse in another scene or for a different character. At the same time, there is an urgent need for digital animation content for films, Internet, and games. Thus, handy and powerful tools for creating and editing whole-body motions without special knowledge are essential.

For this purpose, we have developed an interface for creating natural animation of human characters and implemented it as the computational engine of a CG animation software package. The interface is based on a methodology which we call *pin and drag*, also known as articulated figure positioning [1], [2]. Its basic function is to enable the user to drag a link to an arbitrary position with any number of links pinned in the global frame, as illustrated in Fig. 1. Our results show that this tool is capable of creating natural and human-like motions with only a few pin-and-drag procedures, without any reference motion, even when used by untrained people. The key to this intuitive interface is the reduction of degrees of freedom of a highly complex human character by way of applying constraints on link positions, joint angle errors, and joint motion ranges.

Our approach is also interesting from a biological point of view. In *synergetics* [3], it has been revealed that many natural systems are composed of a combination of a large number of degrees of freedom and constraints. For example, the human body is composed of many bones and muscles. Its apparent degrees of freedom, however, are far fewer than the number of elements included due to an almost equivalent number of constraints. At the human motion level, although the human body has hundreds of degrees of freedom, its motion is constrained by various factors such as internal coupling of joints, joint motion ranges, contacts with the environments, and so forth. Our methodology mathematically imposes constraints by pins and extracts synergetic effects by drags. The reduced degrees of freedom offer easy control and simultaneously give the resulting motions a natural and human-like flavor. This concept has already been applied to the learning and control of robotic systems by forming synergies between actuators [4].

Needless to say, motion capture is a powerful alternative to our approach. Many motion libraries and tools for editing and retargetting captured motions have been developed and are commercially available. Much of the recent research focuses on motion editing with existing motion clips instead of creating new motion from scratch. Published results include retargetting motion to another character [5], [6], blending and connecting multiple motions while preserving the kinematic constraints in the original motions [7], [8], and modifying motion itself using kinematics [9] or dynamics [10], [11], [12]. However, motion capture is not the final solution because of the following two disadvantages: First, users must capture or purchase new motion data every time they need motion not included in their library. Second, motions generated from a single library tend to be relatively uniform. Users may want to change the motion slightly, not only to fit the character or situation, but also to retouch it for aesthetic reasons, which again requires skill and expensive software.

Inverse kinematics is one of the most important techniques for generating motions with kinematic constraints and has been studied in computer animation as well as robotics. There are basically two categories in algorithms for solving inverse kinematics: analytical and numerical.

- *The authors are with the Department of Mechano-Informatics, Graduate School of Information Science and Technology, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan. E-mail: yamane@ynl.t.u-tokyo.ac.jp.*

Manuscript received 22 June 2001; accepted 18 July 2002.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 114398.

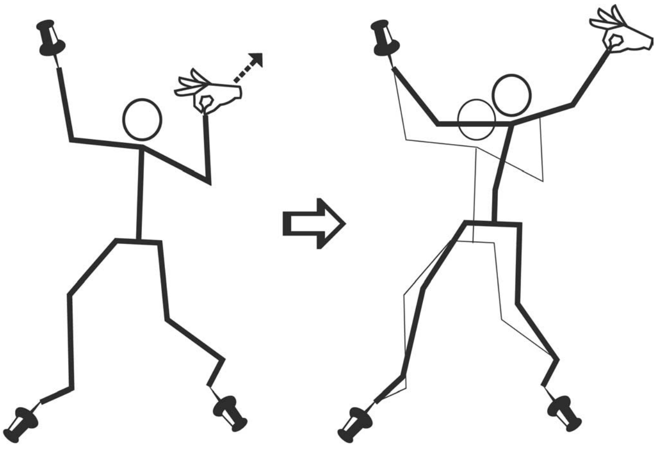


Fig. 1. The concept of pin-and-drag interface.

Analytical inverse kinematics [13], [14] use a closed-form inverse of nonlinear direct kinematics functions. This approach shows good performance in some class of structures such as a single limb of human, but lacks generality. Numerical algorithms vary in the methods they employ to optimize the evaluation functions commonly of the ideal and actual positions of the fixed point. They are usually applicable to general articulated bodies. It is also relatively easy to include other constraints. The techniques in [5], [8] use the whole sequence of motion to perform global optimization, which is not suitable for online motion generation. On the other hand, Choi and Koy [6] employed the closed-loop inverse rate control to enable online modification of captured motion. Lee and Shin [9] proposed a hybrid analytical and numerical algorithm to take the advantage of both approaches.

This paper describes the computational details of our pin-and-drag interface for creating natural motions of human and animal characters without using any captured motion. By applying appropriate fixed pins, the user can create whole-body motion in real time with a single pick-and-drag procedure. Users can also specify other constraints such as desired joint values and joint motion ranges, which are useful for creating cyclic and human-like motions. This method is naturally extended to editing and retargetting existing motions by allowing moving pins and time-dependent desired joint values.

The rest of this paper is organized as follows: We first present an overview of the computation and some preliminaries in Section 2 and then describe the computational details in Section 3. Section 4 discusses how to apply the method to editing motions in real time, followed by some examples of generated motions in Section 5. Finally, we summarize the contributions of this paper in Section 6.

2 Overview

2.1 Pin-and-Drag Interface

The task of the computational engine for the pin-and-drag interface is to generate a motion in which:

1. The link specified by the user (the dragged link) follows the indicated path,
2. Any number of links specified by the user (pinned links) stay at their reference positions,
3. Each joint angle stays in its motion range, and
4. Each joint angle stays as close as possible to the given reference angle.

There are two obvious difficulties in computing the solution that satisfies all of these constraints:

- It is difficult (or virtually impossible) to derive an analytical method that can handle the general cases and
- The constraints often conflict with each other (consider the case where the user drags a link beyond the reachable space determined by the pinned links).

The first problem comes from the fact that the constraints are expressed by a set of complicated nonlinear equations, and the second implies that these equations may not have an exact solution.

The first problem is solved by introducing differential kinematics that give a linear relationship between the constraints and the joint velocities. In order to deal with the second problem, we divide the four constraints into two priority levels [15]. The first constraint (the dragged link) is given the higher priority and is always satisfied exactly. The other constraints are given the lower priority. To satisfy the constraints with the lower priority, the null-space of the first constraint is used. If there is a conflict among the constraints, the least-square optimization is applied to find the best approximation for the lower-priority constraints.

Although the null-space decomposition and the least-square solution are commonly done with a pseudoinverse, it may result in extremely large and, therefore, physically infeasible solutions in the neighborhood of singularity. The singularity-robust (SR) inverse [16] is adopted to avoid this problem since multiple constraints and conflicts among them are the issue to be dealt with in this paper and, therefore, we necessarily face singularities.

The SR inverse eases the singularity problem by allowing errors near singular points. We introduce the feedback controller as a device for the recovery of errors which the singularities or conflicts introduce. By integrating the SR inverse and the feedback controller into the differential kinematics of constrained kinematic chains, the pin-and-drag interface is equipped with the "elastic" property, the natural response, and increases reliability.

2.2 Differential Kinematics with Redundancy

The Jacobian matrix of the position of a link with respect to the joint angles is defined as:

$$\boldsymbol{J}_i \triangleq \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{\theta}}, \tag{1}$$

where $\boldsymbol{r}_i$ is the position of link i, $\boldsymbol{\theta}$ is the vector composed of all joint angles, and $\boldsymbol{J}_i$ is the Jacobian matrix of $\boldsymbol{r}_i$ with respect to $\boldsymbol{\theta}$. An efficient method for computing the Jacobian matrix can be found in [17]. The velocity of link i and joint angles are related by

$$\dot{\boldsymbol{r}}_i = \boldsymbol{J}_i \dot{\boldsymbol{\theta}}. \tag{2}$$

Note that, if the base link is not fixed to the inertial frame, as is often the case with human figures, its linear and angular velocities are also included in $\dot{\boldsymbol{\theta}}$. If J_i is square and nonsingular, it can be inverted to yield

$$\dot{\boldsymbol{\theta}} = J_i^{-1}\dot{\boldsymbol{r}}_i, \tag{3}$$

by which we can control the joints based on the reference trajectory of $\boldsymbol{r}_i$.

Unfortunately, J_i is not square in our problem since human and animal characters typically have over 30 degrees of freedom (DOF). The general solution of (2) is described using the pseudoinverse $J_i^\sharp$ as

$$\dot{\boldsymbol{\theta}} = J_i^\sharp\dot{\boldsymbol{r}}_i + (\boldsymbol{I} - J_i^\sharp J_i)\boldsymbol{y}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{y}$ is an arbitrary vector. The second term shows the redundancy and reserves the degrees of freedom that we can use for the other constraints [15]. One may use the second term of (4) to find the optimal solution that accomplishes other tasks without breaking (2).

2.3 The Singularity-Robust Inverse

Singularity-robust (SR) inverse [16] is also known as damped pseudoinverse [18]. Consider a linear equation

$$\boldsymbol{Ax} = \boldsymbol{b}. \tag{5}$$

If the coefficient matrix $\boldsymbol{A}$ is not square, we usually use its pseudoinverse $\boldsymbol{A}^\sharp$ to compute the least-square solution with the minimal norm. However, the pseudoinverse solution tends to have extremely large amplitude in the neighborhood of singular points. This is because the pseudoinverse minimizes the norm of the error $|\boldsymbol{b} - \boldsymbol{Ax}|$ first and then minimizes the norm of the solution $|\boldsymbol{x}|$ [16]. The SR inverse, on the other hand, avoids this problem by minimizing the sum of the norms of the error and the solution.

For an m-by-n ($m < n$) matrix $\boldsymbol{A}$, its pseudoinverse is computed by

$$\boldsymbol{A}^\sharp = \boldsymbol{A}^T(\boldsymbol{AA}^T)^{-1}. \tag{6}$$

$\boldsymbol{A}^\sharp$ may have extremely large elements when $\boldsymbol{AA}^T$ is nearly singular. The SR inverse, on the other hand, uses the following equation instead of (6):

$$\boldsymbol{A}^* = \boldsymbol{A}^T(\boldsymbol{AA}^T + k\boldsymbol{I})^{-1}, \tag{7}$$

where $\boldsymbol{A}^*$ is the SR inverse of $\boldsymbol{A}$, $\boldsymbol{I}$ is the identity matrix, and k is the parameter that determines the weighting between the norm of the solution and the error. If we use small k, then the error gets small, but the solution might get large around singular points and vice versa [19].

2.4 The Algorithm

The algorithm to be proposed in this paper consists of the following five steps:

1. Compute the general solutions of joint velocities that move the dragged link toward the indicated position (Section 3.1),
2. Compute the desired velocities of the other constraint variables, taking account of their reference and current values (Section 3.4),
3. Compute the Jacobian matrix of the constraint variables with respect to the joint angles (Section 3.3),
4. Using the general solutions in Step 1, find a particular solution that closely satisfies the desired velocities of the constraint variables (Section 3.2),
5. Numerically integrate the joint velocities to get the joint angles.

The proposed algorithm has a number of advantages over the previous ones with similar objectives:

- Moving a single link determines the posture of the whole body,
- Any link can be dragged or pinned,
- No limit on the number of pinned links,
- Constraint variables can be instantly included or removed,
- Relative importance of the constraint variables can be tuned.

2.5 Comparison with Previous Work

The main objective of this paper is to develop an interface that enables people to generate whole-body motions of articulated figures with little effort and preferably without captured data. Although there are many related works in this field, most of these efforts aim to solve the problem of editing or retargeting prerecorded data. The authors would think it is because the previous researchers viewed generating new motions as too challenging.

Inverse kinematics is the key in our approach. Many previous works in inverse kinematics used global optimization over the spacetime constraint of motion [5], [7], [8], [9]. The SR-inverse was also employed in [5] to avoid singularities in the Jacobian matrix.

Online computation using local optimization, on the other hand, was investigated by Choi and Ko [6] based on the feedback control and the null-space method similar to ours. In [6], the pinned links are placed only at the end-links due to the fact that the increase of constraints makes the Jacobian matrix ill-conditioned and the troubles of singularity cannot be avoided by the use of pseudoinverses. Our approach allows as many pins as we need, even at intermediate links or two neighboring links, thanks to the SR-inverse.

Badler et al. [1] and Phillips et al. [2] also developed a pin-and-drag interface and implemented it as a part of the 3D animation system *Jack* [20]. Our formulation follows the previous works in principle and includes the following improvements:

- In Badler et al.'s system, the link hierarchy was recomputed so that the dragged link becomes the root, while, in ours, the link hierarchy does not change once the structure is given, thanks to the virtual link representation of closed loops proposed in [21]. We can eliminate the overhead to switch the dragged links, providing more responsive and comfortable interface to the user.
- Badler et al.'s system considered joint motion range of rotational (or 1DOF) joints and it would find it difficult to include spherical-joint limits with their projection scheme.

- The normal pseudoinverse was used in their computation of inverse kinematics. The SR-inverse we use, in contrast, allows us to apply as many constraints as we need, without worrying about the singularity.

The strict comparison between the proposed and the previous methods is not straightforward. One difference in computational expense comes from the elimination of link-hierarchy recomputation. Another difference is due to the SR-inverse as opposed to the pseudoinverse in the previous works. In [16], the computational comparison was extensively made between the pseudoinverse and the SR-inverse and the difference was negligibly small for full-rank matrices. For non-full-rank matrices, the SR-inverse is known to be significantly more efficient than the pseudoinverse. These algorithmic differences show the computational advantage of the proposed method in this paper.

Representation of the motion range of 3DOF spherical joints is fundamental to obtaining natural behaviors of human characters. A simple inequality representation of the Euler angles is inappropriate due to their nonlinearity [22], [23]. A precise anatomical modeling was recently proposed [23], where three 3DOF spherical joints and a 5DOF joint were used for modeling. It is common in these previous works that the 3DOF of a spherical joint are parameterized by three variables known as the Euler angles and, therefore, suffers from the well-known algorithmic singularity.

In contrast, we propose parameterizing the 3DOF of a spherical joint in a unique definition of three variables. It has a singular point only at a physically insignificant direction. In addition, when the limb configuration goes beyond the motion range, a feedback control is applied to force the range. The feedback control is proposed in a simpler form in this paper than those used in the previous works.

The above-mentioned improvements characterize our algorithm by computationally high efficiency and robustness compared to the previous works.

3 Computational Details

3.1 The Dragged Link

First, we compute $\dot{\boldsymbol{\theta}}$ with which the dragged link exactly follows its reference velocity $\dot{\boldsymbol{r}}_P^{ref}$ and position $\boldsymbol{r}_P^{ref}$. Let $\boldsymbol{r}_P$ denote the current position of the dragged link. Its desired velocity is computed by

$$\dot{\boldsymbol{r}}_P^d = \dot{\boldsymbol{r}}_P^{ref} + \boldsymbol{K}_P(\boldsymbol{r}_P^{ref} - \boldsymbol{r}_P), \tag{8}$$

where $\boldsymbol{K}_P$ is a positive-definite gain matrix. The relationship between $\dot{\boldsymbol{\theta}}$ and $\dot{\boldsymbol{r}}_P$ is given by

$$\dot{\boldsymbol{r}}_P = \boldsymbol{J}_P\dot{\boldsymbol{\theta}}, \tag{9}$$

where $\boldsymbol{J}_P$ is the Jacobian matrix of $\boldsymbol{r}_P$ with respect to the joint angles. The general solution $\dot{\boldsymbol{\theta}}$ for the desired velocity $\dot{\boldsymbol{r}}_P^d$ is computed by

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_P^\sharp \dot{\boldsymbol{r}}_P^d + (\boldsymbol{I} - \boldsymbol{J}_P^\sharp \boldsymbol{J}_P)\boldsymbol{y}. \tag{10}$$

The feedback control is applied only to compensate the numerical errors. A weighted pseudoinverse [19] may be used in the above equation instead of the normal pseudoinverse to characterize the joint motions.

3.2 Lower-Priority Constraints

The general solution of (10) is rewritten by

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_0 + \boldsymbol{W}\boldsymbol{y}, \tag{11}$$

where $\boldsymbol{W} \triangleq \boldsymbol{I} - \boldsymbol{J}_P^\sharp \boldsymbol{J}_P$ and $\dot{\boldsymbol{\theta}}_0 \triangleq \boldsymbol{J}^\sharp \dot{\boldsymbol{r}}_P^d$.

Suppose we have N_F pinned links whose positions are denoted by $\boldsymbol{r}_{Fi}(i = 1 \dots N_F)$, N_D joints with their reference angles $\boldsymbol{\theta}_D$, and N_L joints with their joint values $\boldsymbol{\theta}_L$ out of the motion ranges. Note that N_L may vary anytime during the motion, whereas N_D stays constant until it is changed by the higher level of control. Using the vectors, we define $\boldsymbol{p}_{aux}$ as follows:

$$\boldsymbol{p}_{aux} \triangleq \begin{pmatrix} \boldsymbol{r}_{F1}^T & \dots & \boldsymbol{r}_{FN_F}^T & \boldsymbol{\theta}_D^T & \boldsymbol{\theta}_L^T \end{pmatrix}^T. \tag{12}$$

Velocity $\dot{\boldsymbol{p}}_{aux}$ is related to the joint velocity $\dot{\boldsymbol{\theta}}$ by a relationship similar to (2).

$$\dot{\boldsymbol{p}}_{aux} = \boldsymbol{J}_{aux}\dot{\boldsymbol{\theta}}. \tag{13}$$

Computation of $\boldsymbol{J}_{aux}$ is to be discussed in the following section.

The arbitrary vector $\boldsymbol{y}$ is computed as follows: We first compute the desired velocity $\dot{\boldsymbol{p}}_{aux}^d$ of $\boldsymbol{p}_{aux}$ to take account of the errors between the constraint conditions and their current values as described in Section 3.4. Substituting (11) into (13) yields

$$\dot{\boldsymbol{p}}_{aux} = \dot{\boldsymbol{p}}_{aux}^0 + \boldsymbol{J}_{aux}\boldsymbol{W}\boldsymbol{y}, \tag{14}$$

where $\dot{\boldsymbol{p}}_{aux}^0 \triangleq \boldsymbol{J}_{aux}\dot{\boldsymbol{\theta}}_0$. Using $\boldsymbol{S} \triangleq \boldsymbol{J}_{aux}\boldsymbol{W}$ and

$$\Delta\dot{\boldsymbol{p}}_{aux} \triangleq \dot{\boldsymbol{p}}_{aux}^d - \dot{\boldsymbol{p}}_{aux}^0,$$

we have a simpler form of the equation:

$$\boldsymbol{S}\boldsymbol{y} = \Delta\dot{\boldsymbol{p}}_{aux}. \tag{15}$$

Since $\boldsymbol{S}$ is not always well conditioned, we use the SR inverse to solve this problem. Denoting the SR inverse of $\boldsymbol{S}$ by $\boldsymbol{S}^*$, $\boldsymbol{y}$ is computed by

$$\boldsymbol{y} = \boldsymbol{S}^*\Delta\dot{\boldsymbol{p}}_{aux}. \tag{16}$$

The joint velocity $\dot{\boldsymbol{\theta}}$ is obtained by substituting (16) into (11), which is then integrated to yield the joint angle $\boldsymbol{\theta}$ for animation.

3.3 Computation of $\boldsymbol{J}_{aux}$

Let $\boldsymbol{J}_{Fi}(\mathrm{i} = 1 \dots \mathrm{N_F})$ be the Jacobian matrix of $\boldsymbol{r}_{Fi}$ with respect to the joint angles. Then, for all pinned links, we have

$$\dot{\boldsymbol{r}}_{Fi} = \boldsymbol{J}_{Fi}\dot{\boldsymbol{\theta}}. \tag{17}$$

For the joints with reference angles, the relationship between their velocities $\dot{\boldsymbol{\theta}}_D$ and $\dot{\boldsymbol{\theta}}$ is described by

$$\dot{\boldsymbol{\theta}}_D = \boldsymbol{J}_D\dot{\boldsymbol{\theta}}, \tag{18}$$

where $\boldsymbol{J}_D$ is the matrix whose $(i, j)\mathrm{th}$ element is 1 if the *i*th element of $\boldsymbol{\theta}_D$ corresponds to the *j*th element of $\boldsymbol{\theta}$ and 0 otherwise.

Similarly, we can describe the relationship between $\dot{\boldsymbol{\theta}}$ and the velocity of $\boldsymbol{\theta}_L$ as follows:

$$\dot{\boldsymbol{\theta}}_L = \boldsymbol{J}_L \dot{\boldsymbol{\theta}}, \tag{19}$$

where $\boldsymbol{J}_L$ is the matrix whose (i,j)th element is 1 if the ith element of $\boldsymbol{\theta}_L$ corresponds to the jth element of $\boldsymbol{\theta}$ and 0 otherwise.

Combining the above-defined matrices, $\boldsymbol{J}_{aux}$ is formed as follows:

$$\boldsymbol{J}_{aux} = \begin{pmatrix} \boldsymbol{J}_{F1}^T & \dots & \boldsymbol{J}_{FN_F}^T & \boldsymbol{J}_D^T & \boldsymbol{J}_L^T \end{pmatrix}^T. \tag{20}$$

The computation of columns of $\boldsymbol{J}_{Fi}$, $\boldsymbol{J}_P$, and $\boldsymbol{J}_L$ corresponding to spherical joints is to be discussed in Section 3.5.

3.4 Computation of $\dot{\boldsymbol{p}}_{aux}^d$

The desired velocity of each pinned link $\dot{\boldsymbol{r}}_{Fi}^d$ is computed by the following feedback law:

$$\dot{\boldsymbol{r}}_{Fi}^d = \boldsymbol{K}_{Fi}(\boldsymbol{r}_{Fi}^{ref} - \boldsymbol{r}_{Fi}), \tag{21}$$

where $\boldsymbol{r}_{Fi}^{ref}$ is the reference position and $\boldsymbol{K}_{Fi}$ is a positive-definite gain matrix.

The desired velocity of joints with their reference angles, $\dot{\boldsymbol{\theta}}_D^d$, is computed by

$$\dot{\boldsymbol{\theta}}_D^d = \boldsymbol{K}_D(\boldsymbol{\theta}_D^{ref} - \boldsymbol{\theta}_D), \tag{22}$$

where $\boldsymbol{\theta}_D^{ref}$ represents the reference joint angles and $\boldsymbol{K}_D$ is a positive-definite gain matrix.

The desired velocities of joints that exceed their motion ranges are computed as follows:

$$\dot{\theta}_{Li}^d = \begin{cases} K_{Li}(\theta_{Li}^{max} - \theta_{Li}) & \text{if } (\theta_{Li} > \theta_{Li}^{max}) \\ K_{Li}(\theta_{Li}^{min} - \theta_{Li}) & \text{if } (\theta_{Li} < \theta_{Li}^{min}), \end{cases} \tag{23}$$

where θ_{Li}^{max} and θ_{Li}^{min} are the maximum and minimum joint angles, respectively, and K_{Li} is a positive scalar gain.

Equations (22) and (23) work for 1DOF joints. The following subsection extends the ideas to 3DOF spherical joints.

3.5 Handling Spherical Joints

3.5.1 Reference Joint Displacements

The joint displacement and the joint velocity of a spherical joint are represented by the 3×3 orientation matrix $\boldsymbol{R}_i$ and its associated angular velocity ω_i, respectively, described in its parent link frame [21].

When a spherical joint is given a reference joint displacement $R_{Di} \in R^{3\times 3}$, we compute its desired joint velocity as follows: We first compute the error vector $\boldsymbol{e}_i$ between the current joint displacement $\boldsymbol{R}_i$ and $\boldsymbol{R}_{Di}$ by

$$\boldsymbol{e}_i = \frac{1}{2}\begin{pmatrix} \Delta\boldsymbol{R}_i(1,2) - \Delta\boldsymbol{R}_i(2,3) \\ \Delta\boldsymbol{R}_i(1,3) - \Delta\boldsymbol{R}_i(3,1) \\ \Delta\boldsymbol{R}_i(2,1) - \Delta\boldsymbol{R}_i(3,2) \end{pmatrix} \tag{24}$$

$$\Delta\boldsymbol{R}_i \triangleq \boldsymbol{R}_{Di}\boldsymbol{R}_i^T, \tag{25}$$

where$\Delta\boldsymbol{R}_i(m,n)$ denotes the (m,n)th element of $\Delta\boldsymbol{R}_i$. Then, the desired angular velocity ω_{Di}^d is computed by [24]

$$\boldsymbol{\omega}_{Di}^d = -\boldsymbol{K}_{Di}\boldsymbol{e}_i, \tag{26}$$

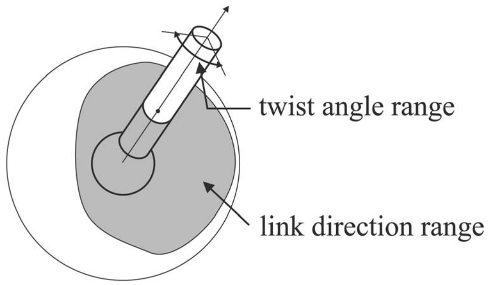


Fig. 2. Joint motion range of a spherical joint.

where $\boldsymbol{K}_{Di}$ is a positive-definite gain matrix. Equations (24)-(26) are used for spherical joints in place of (22).

Also included in $\boldsymbol{J}_{Fi}$, $\boldsymbol{J}_D$, and $\boldsymbol{J}_L$ corresponding to a spherical joint are three columns associated with the angular velocity. Each column is computed just as if there is a rotational joint around the x, y, or z axis.

3.5.2 Joint Motion Range

The motion range of a spherical joint is expressed as a region in a three-dimensional space. Simplicity of the geometric representation of the region is important for real-time computation. The region would show a complex shape if we represent it with common coordinates such as the Euler angle due to their nonlinearity. In this subsection, we propose an intuitive representation of spherical joint motion range. Although one may see a similarity to the *equivalent angle-axis* representation [14], it is different in the sense that our representation consists of two sequential rotations. The representation provides three parameters, two of which describe the link direction and the other denotes the twist angle, as illustrated in Fig. 2.

When $\boldsymbol{R}_i$ is the identity, the link is at the nominal direction and we represent it by unit vector $\boldsymbol{d}_i^0$. The current link direction $\boldsymbol{d}_i$ is obtained by rotating $\boldsymbol{d}_i^0$ about vector $\boldsymbol{a}_i$ that lies in the two-dimensional plane orthogonal to $\boldsymbol{d}_i^0$. The magnitude of $\boldsymbol{a}_i$ is not 1, but $\sin(\gamma_i/2)$, where γ_i represents the angle of rotation, as seen in Fig. 3. The twist angle α is defined as the angle by which the link frame after the first rotation around $\boldsymbol{a}_i$ is rotated to make the current link frame $\boldsymbol{R}_i$. The entire configuration of a spherical joint is therefore included in a cylinder whose axis is $\boldsymbol{d}_i^0$, radius 1, and height 2π.

In our implementation, $\boldsymbol{d}_i^0$ is set as $(1\ 0\ 0)^T$ for all joints and, therefore, $\boldsymbol{a}_i$ stays in the yz plane, namely, $\boldsymbol{a}_i = (0\ a_y\ a_z)^T$. Thus, the motion range is described by a cylinder with an axis parallel to the α axis as shown in Fig. 4.

a_y, a_z, and α are computed as follows: Since $\boldsymbol{d}_i^0 = (1\ 0\ 0)^T$, we have

$$\begin{aligned} \boldsymbol{d}_i &= \boldsymbol{R}_i\boldsymbol{d}_i^0 \\ &= (\boldsymbol{R}_i(1,1)\ \boldsymbol{R}_i(2,1)\ \boldsymbol{R}_i(3,1))^T. \end{aligned} \tag{27}$$

Therefore, a_y and a_z are computed by

$$a_y = -\frac{\boldsymbol{R}_i(3,1)}{\sqrt{2(1+\boldsymbol{R}_i(1,1))}} \tag{28}$$

$$a_z = \frac{\boldsymbol{R}_i(2,1)}{\sqrt{2(1+\boldsymbol{R}_i(1,1))}}. \tag{29}$$

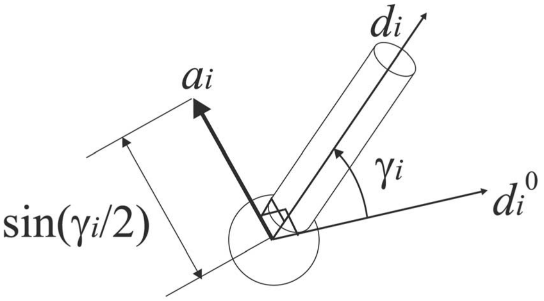


Fig. 3. Graphical representation of $\boldsymbol{a}_i$ and γ_i.

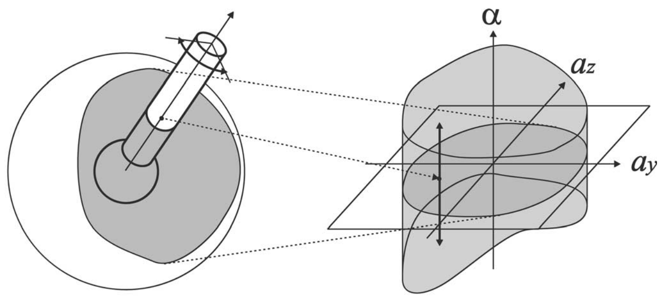


Fig. 4. Motion range of a spherical joint projected onto (a_y, a_z, α) space.

Then, α is computed by comparing the y and z axes of the frame after the rotation around $\boldsymbol{a}_i$ and the actual current frame since their x axes coincide with each other. Although (28) and (29) show singularity at $\gamma_i = \pm\pi$, it is not a practical problem since these values are usually beyond joint limits.

Our next step is to determine whether the computed parameters are inside or outside of the motion range. For ease of computation, we describe the link direction range by a collection of triangle patches in the a_y-a_z plane. The whole motion range is represented by a collection of triangular cylinders with the triangle patches as their footprints and α axis as their axes.

We first look for the triangle in which $(a_y, a_z, 0)$ is included. If no triangle is found, the joint is out of the joint motion range. Otherwise, we then proceed to check if (a_y, a_z, α) is between the upper and lower limits of the triangle.

If the parameters (a_y, a_z, α) are out of the range, we compute the desired velocity to bring the joint back into the range and include it in $\dot{\boldsymbol{p}}^d_{aux}$. For this purpose, we define the standard orientation $\boldsymbol{R}_{Si}$ for each joint, and compute the desired joint velocity $\boldsymbol{\omega}_{Li}$ to move the joint toward $\boldsymbol{R}_{Si}$. This is achieved by simply substituting $\boldsymbol{R}_{Si}$ into $\boldsymbol{R}_{Di}$ in (25) and $\boldsymbol{\omega}_{Li}$ into $\boldsymbol{\omega}_{Di}$ in (26). The more theoretical alternative would be to control the joint toward the configuration on the boundary of the motion range, as in [23]. We did not take this for computational simplicity's sake.

Fig. 5 shows the motion range of a right shoulder. The light colored (or red for readers with colored figures) shows the upper surfaces of triangular cylinders, while the dark colored (or blue) shows their lower surfaces. The vertical axis denotes the twist angle in degree, while the other two denote the rotation angle in forward/backward and left/right directions as indicated in the figure. This example includes 35 triangles in the a_y-a_z plane. We modeled the motion ranges of 10 spherical joints with 8 to 35 triangles, depending on their shapes. Due to the simplicity of computation, handling spherical joints did not affect the real-time performance of the system.

4 Editing Motion in Motion

In Section 3, we discussed the algorithm for static pin-and-drag interface, where the reference positions of pins, $\boldsymbol{r}^{ref}_{Fi}$, and the reference joint angles, $\boldsymbol{\theta}^{ref}_{Di}$, were assumed constant, while the dragged link had its velocity $\dot{\boldsymbol{r}}^{ref}_{P}$. Extending the algorithm to include velocities of $\boldsymbol{r}^{ref}_{Fi}$ and $\boldsymbol{\theta}^{ref}_{Di}$ for dynamic pin-and-drag interface is straightforward. By this extension, we can apply the algorithms to editing motion in motion and retargeting captured data. Changing the pins and the dragged link one after another in motion enables us to generate step by step rich and complex motions of characters.

The following two slight modifications are required to extend the above method to editing motion in motion:

- The positions of the pins are obtained by direct kinematics computation for each frame of the

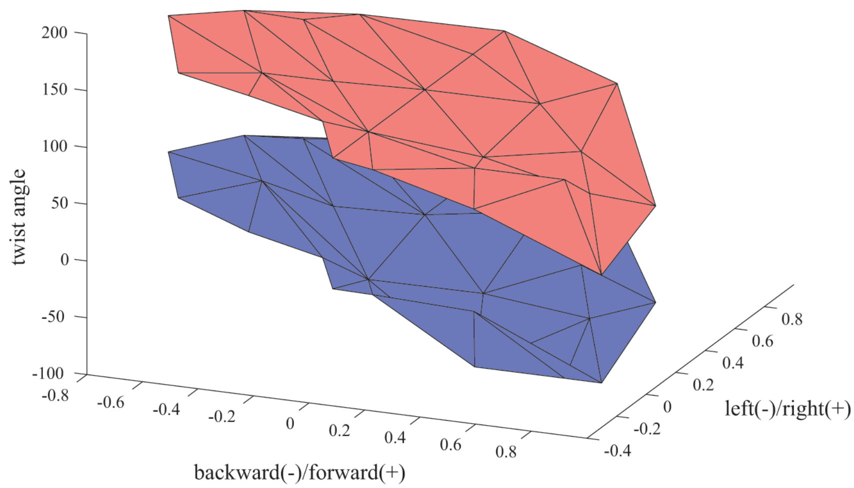


Fig. 5. An example of a motion range of a spherical joint.

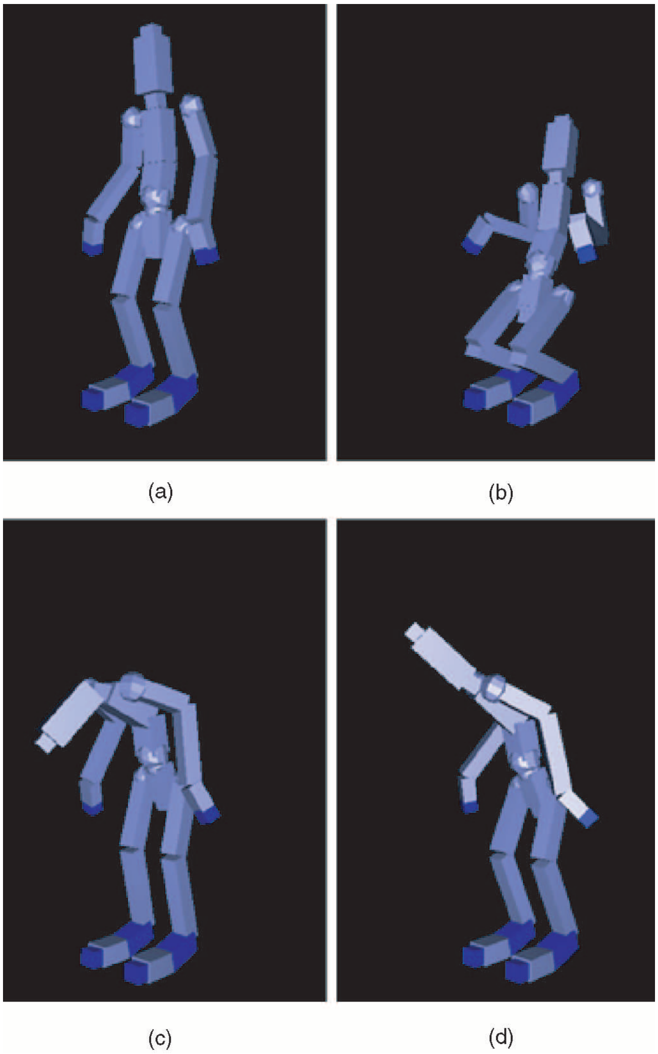


Fig. 6. Postures generated by a single pin-and-drag: (a) original posture, (b) pelvis dragged, (c) head dragged, (d) left shoulder dragged.

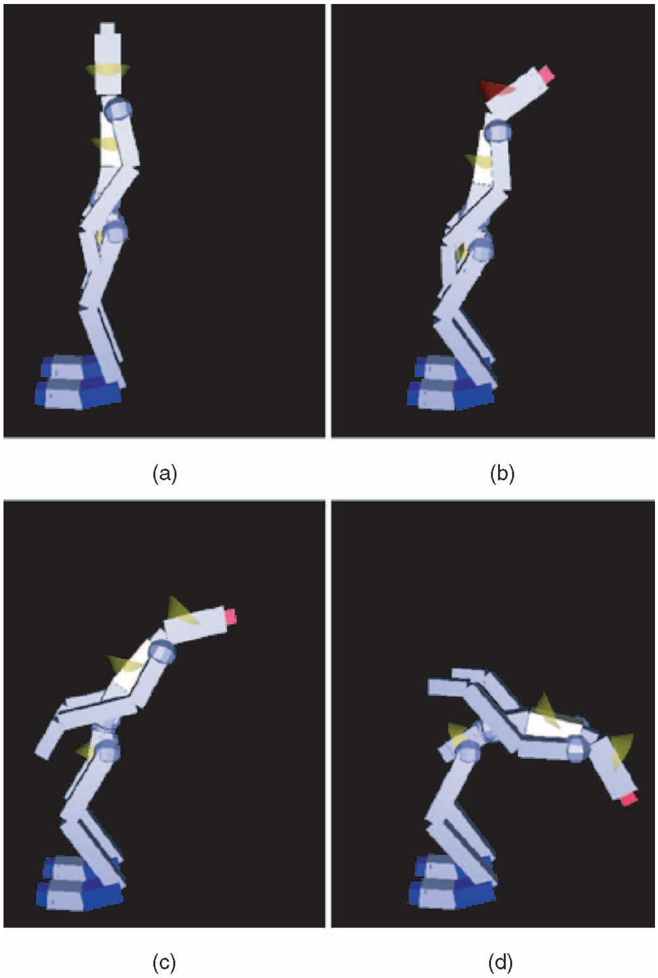


Fig. 7. Effect of applying joint motion range: (a) original posture, where the motion ranges are represented by cones, (b) the neck joint exceeds its range, (c) neck joint returns back into the range, (d) the upper body is bended further without exceeding the ranges.

reference motion. Since each pin has reference velocity $\dot{\boldsymbol{r}}_{Fi}^{ref}$, the following equation is used instead of (21):

$$\dot{\boldsymbol{r}}_{Fi}^{d} = \dot{\boldsymbol{r}}_{Fi}^{ref} + \boldsymbol{K}_{Fi}(\boldsymbol{r}_{Fi}^{ref} - \boldsymbol{r}_{Fi}). \qquad (30)$$

- The reference joint displacements are set as those of the reference motion. Using the reference joint velocities $\dot{\boldsymbol{\theta}}_{D}^{ref}$, the following equation is used instead of (22):

$$\dot{\boldsymbol{\theta}}_{D}^{d} = \dot{\boldsymbol{\theta}}_{D}^{ref} + \boldsymbol{K}_{D}(\boldsymbol{\theta}_{D}^{ref} - \boldsymbol{\theta}_{D}). \qquad (31)$$

5 Examples

The proposed method was implemented as the computational engine of CG animation software *Animanium*™, where the method is used for generating key frames and recording real-time manipulation of human figures. The software is equipped with graphical interfaces to select pinned and dragged links, define weight, feedback gain, and motion range for each joint. The motion of the mouse is mapped into the three-dimensional motion of the dragged link. The users can create appealing animations by simply specifying key frames using the interface, which are then interpolated automatically to generate the animation. The computation time is approximately 33ms on PentiumIII 1GHz processor, for a 48DOF human figure with 1 dragged link, 5 pins, and 20 joints with reference joint displacements and joint motion ranges.

5.1 Pin and Drag

Fig. 6 shows various postures generated by a single pin-and-drag procedure from the initial posture (a), with both hands and feet pinned.

5.2 Effect of Joint Motion Range

The effect of considering joint motion range is shown in Fig. 7, where the feet are pinned and the head is dragged from the original posture (a) to the final (d). As the user interface for changing joint motion range, the motion ranges of spherical joints are shown by cones representing the link direction ranges in the figure. Although the neck joint exceeds its range in posture (b), which is represented by a

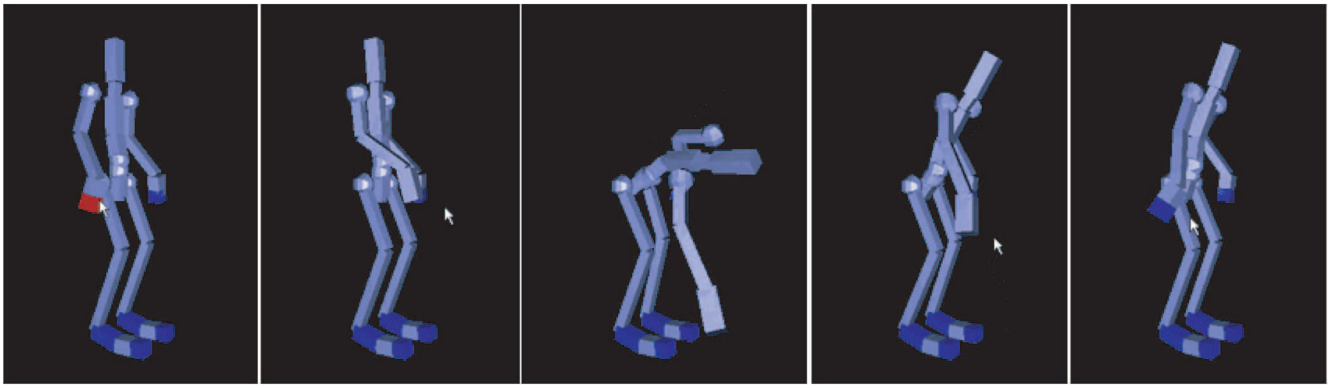


Fig. 8. An example of real-time motion generation.

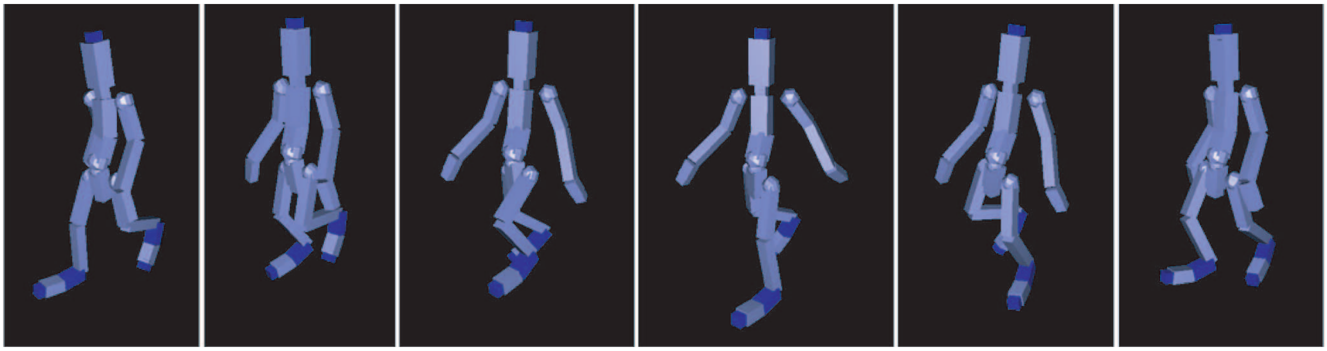


Fig. 9. Original motion used for motion editing in motion.

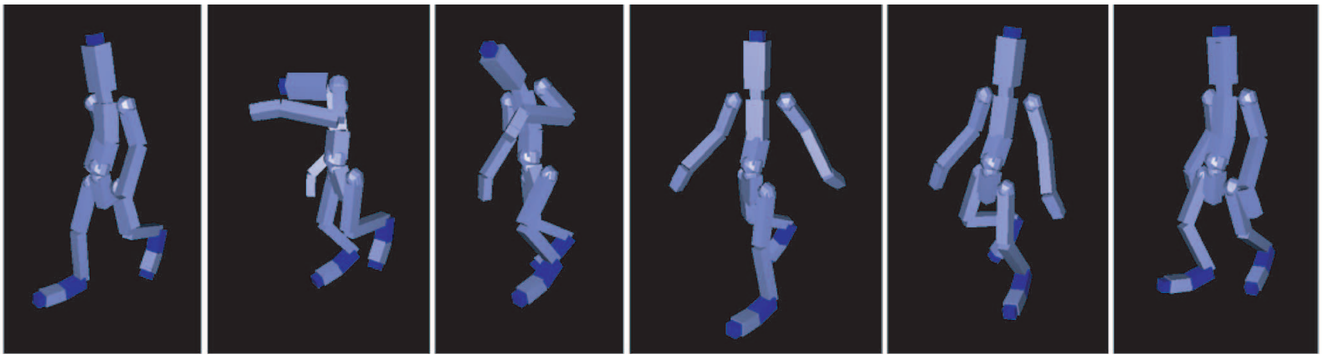


Fig. 10. Modified motion from the motion in Fig. 9.

red cone, it returns back to the range by bending back the chest joint in (c). All joints stay in the range in the final posture (d). Note that this natural-looking behavior is generated in real time by a single pin-and-drag procedure taking account of joint motion ranges.

5.3 Real-Time Motion Generation

The images in Fig. 8 are taken from a video clip recorded also in real time when the user dragged the right hand of the character for 4 seconds. The pins were set at five links—the toes, heels, and the left hand, shown in blue. Note that we can set a pin at link not necessarily at the end of a chain, such as heel links. A single dragging created a realistic motion like picking up an object on the floor.

5.4 Editing Motion in Motion

Figs. 9 and 10 show results of editing prerecorded motion. The motion in Fig. 9 was created by a professional animator using the software and the modification was done by one of the authors. The original motion was a short walk consisting of six keyframes. Since both feet were pinned by moving pins that move along the trajectory determined by the original walking motion, their positions do not change even when we drag other links in the body. We modified the second and third frames by simply dragging the head and left hand so that the motion looks like avoiding an object flying toward the character.

6 Conclusion

The contributions of this paper are summarized as follows:

1. We developed computational algorithms for the singularity-robust pin-and-drag interface to compute natural-looking motions of human character.
2. In contrast to the other numerical inverse kinematics solvers, we can place any number of pins to arbitrary

links without causing troubles due to the singularity of the Jacobian matrices.

3. Implemented constraints include pins, desired joint angles, and joint motion ranges. All these constraints are handled in a uniform way.
4. Editing motion in motion and retargeting captured motions are also realized by applying the proposed method with reference motion data.
5. The computational algorithms were successfully implemented and demonstrated their computational efficiency and ease of use for generating keyframe and real-time animation. Examples of created motions demonstrated the usefulness of the developed algorithms and software.

ACKNOWLEDGMENTS

This research was conducted while Katsu Yamane was a research fellow of the Japan Society for the Promotion of Science at the University of Tokyo. The theoretical development of this paper was supported by the CREST program of the Japan Science and Technology Corporation through "Development of Machines with Brain-Like Information Processing through Dynamical Connections of Autonomous Behavioral Primitives" (PI: Y. Nakamura). The software development was supported by the Information-Technology Promotion Agency (IPA) Japan through the project "Development of New Application for Motion Generation Based on Control Techniques for Humanoid Robots" (PI: H. Imagawa, SEGA Corporation). The library is used as the computational engine of *Animanium*™ (SEGA Corporation). The examples in Section 5 were generated using the software package.

REFERENCES

[1] N.I. Badler, K.H. Manoochehri, and D. Baraff, "Multi-Dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints," *Proc. 1986 Workshop Interactive 3D Graphics,* pp. 151-169, Oct. 1986.

[2] C.B. Phillips, J. Zhao, and N.I. Badler, "Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints," *Proc. 1990 Workshop Interactive 3D Graphics,* pp. 245-250, Mar. 1990.

[3] H. Haken, *Advanced Synergetics, Instability Hierarchies of Self-Organizing Systems and Devices.* Berlin: Springer Verlag, 1983.

[4] J. Gelfand, M. Flax, R. Endres, S. Lane, and D. Handelman, "Senses, Skills, Reactions and Reflexes: Learning Automatic Behaviors in Multi-Sensory Robotic Systems," *Neural Networks in Robotics,* G. Bekey and K. Goldberg, eds., pp. 319-330, Boston: Kluwer, 1992.

[5] M. Gleicher, "Retargetting Motion to New Characters," *Proc. SIGGRAPH '98,* pp. 33-42, 1998.

[6] K.J. Choi and H.S. Ko, "Online Motion Retargetting," *J. Visualization and Computer Animation,* vol. 11, pp. 223-235, 2000.

[7] C.F. Rose, P.-P. Sloan, and M.F. Choen, "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation," *Eurographics,* vol. 20, no. 3, 2001.

[8] C. Rose, B. Guenter, B. Bodenheimer, and M.F. Cohen, "Efficient Generation of Motion Transitions using Spacetime Constraints," *Proc. SIGGRAPH '96,* pp. 147-154, 1996.

[9] J.H. Lee and S.Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-like Figures," *Proc. SIGGRAPH '99,* pp. 39-48, 1999.

[10] Z. Popovic, "Editing Dynamic Properties of Captured Human Motion," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 670-675, Apr. 2000.

[11] N.S. Pollard and F. Behmaram-Mosavat, "Force-Based Motion Editing for Locomotion Tasks," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 663-669, Apr. 2000.

[12] K. Yamane and Y. Nakamura, "Dynamics Filter—Concept and Implementation of On-Line Motion Generator for Human Figures," *Proc. IEEE Int'l Conf. Robotics and Automation,* vol. 1, pp. 688-695, Apr. 2000.

[13] J.U. Korein and N.I. Badler, "Techniques for Generating the Goal-Directed Motion of Articulated Structures," *IEEE Trans. Computer Graphics and Applications,* pp. 71-81, Nov. 1982.

[14] J.J. Craig, *Introduction to Robotics: Mechanics and Control.* Reading, Mass.: Addison-Wesley, 1986.

[15] Y. Nakamura and H. Hanafusa, "Task Priority Based Redundancy Control of Robot Manipulators," *Int'l J. Robotics Research,* vol. 6, no. 2, pp. 3-15, 1987.

[16] Y. Nakamura and H. Hanafusa, "Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control," *J. Dynamic Systems, Measurement, and Control,* vol. 108, pp. 163-171, 1986.

[17] D.E. Orin and W.W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators," *Int'l J. Robotics Research,* vol. 3, no. 4, pp. 66-75, 1984.

[18] A.A. Maciejewski, "Dealing with the Ill-Conditioned Equations of Motion for Articulated Figures," *IEEE Computer Graphics and Applications,* vol. 10, no. 3, pp. 63-71, May 1990.

[19] Y. Nakamura, *Advanced Robotics—Redundancy and Optimization* Reading, Mass.: Addison-Wesley, 1991.

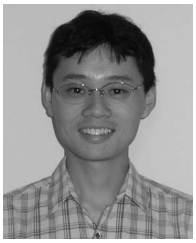
[20] C.B. Phillips and N.I. Badler, "Jack: A Toolkit for Manipulating Articulated Figures," *Proc. ACM/SIGGRAPH Symp. User Interface Software,* 1988.

[21] Y. Nakamura and K. Yamane, "Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Human Figures," *IEEE Trans. Robotics and Automation,* vol. 16, no. 2, pp. 124-134, 2000.

[22] N.I. Badler, C.B. Phillips, and B.L. Webber, *Simulating Humans.* Oxford Univ. Press, 1993.

[23] W. Maurel and D. Thalmann, "Human Shoulder Modeling Including Scapulo-Thoracic Constraint and Joint Sinus Cones," *Computers and Graphics,* vol. 24, pp. 203-218, 2000.

[24] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "Resolved Acceleration Control of Mechanical Manipulators," *IEEE Trans. Automatic Control,* vol. 25, no. 3, pp. 468-474, 1980.



Katsu Yamane received the BS, MS, and PhD degrees from the University of Tokyo, Japan, in mechanical engineering in 1997, 1999, and 2002, respectively. He was a research fellow of the Japan Society for the Promotion of Science from 2000 to 2002 and a postdoctoral fellow at the Robotics Institute, Carnegie Mellon University from 2002 to 2003. He is currently an assistant professor in the Department of Mechano-Informatics at the University of Tokyo. His research interests include dynamics simulation of human figures, control of humanoid robots, entertainment robotics, and physically-based animation in computer graphics. He is a member of the IEEE, IEEE Computer Society, ACM SIGGRAPH, and Robotics Society of Japan.

Yoshihiko Nakamura received the BS, MS, and PhD degrees from Kyoto University, Japan, in precision engineering in 1977, 1978, and 1985, respectively. He was an assistant professor at the Automation Research Laboratory, Kyoto University, from 1982 to 1987. He joined the Department of Mechanical and Environmental Engineering, University of California, Santa Barbara, in 1987 as an assistant professor and became an associate professor in 1990. He was also a codirector of the Center for Robotic Systems and Manufacturing at UCSB. Since 1991, he has been with Department of Mechano-Informatics, University of Tokyo, Japan, and is currently a professor. His fields of research include redundancy in robotic mechanisms, nonholonomy of robotic mechanisms, kinematics and dynamics algorithms for computer graphics, nonlinear dynamics and brain-like information processing, and robotic systems for medical applications. He is a member of the IEEE, the ASME, the SICE, the Robotics Society of Japan, the Japan Society of Mechanical Engineers, the Institute of Systems, Control, and Information Engineers, and the Japan Society of Computer Aided Surgery.