Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Synergetic CG Choreography through Constraining and Deconstraining at Will

Katsu Yamane and Yoshihiko Nakamura
e-mail: {katz,nakamura}@ynl.t.u-tokyo.ac.jp
Department of Mechano-Informatics, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033 Japan

Abstract

This paper presents an interface for creating whole-body motions of human and animal characters without reference motion. Its basic function is to enable 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. Each constraint can be switched on or off, strengthened or weakened for each joint at a user's will. The interface is based on a new online 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 the method can create natural motions from scratch. We also demonstrate the power and usability of the proposed method by a number of example motion clips.

Key Words — Animation, Online inverse kinematics computation, Multiple constraints, Motion editing, Joint motion range.

1 Introduction

Creating realistic motion of human characters still relies heavily on an animator's skill or motion capture techniques. Moreover, even if a motion clip was created through hard work, it is difficult to modify it to reuse in another scene or for a different character. There is urgent need of ever increasing quantities of digital animation content for films, internet, and games. 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 choreographing 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. Many results show that the tool is capable of creating natural and human-like motions through only a few pin-and-drag procedures without any reference motion, even by untrained people. The key for 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.

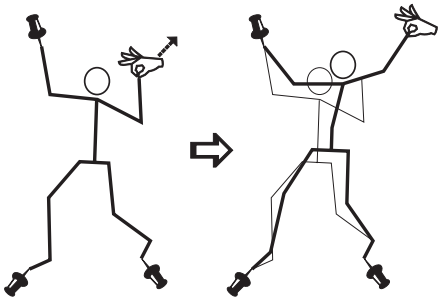

Fig.1: The concept of pin-and-drag interface

This methodology is also interesting from a biological point of view. In *synergetics*, it has been revealed that many natural systems are composed of 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 almost equivalent number of constraints. At the human motion level, although the human body itself 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. Namely, the reduced degrees of freedom offer an ease of control and, simultaneously, provide the resultant motions with a natural and human-like flavor.

Motion capture is a powerful alternative to our approach. In fact, many motion libraries and tools for

editing and retargeting captured motions have been developed and even commercially available. Much of the recent research focuses on motion editing under the existence of motion clips instead of creating a new motion from scratch. Published results include retargeting a motion to another character[3, 4], blending and connecting multiple motions while preserving the kinematic constraints in the original motions[5, 6], and modifying motion itself using kinematics[7] or dynamics[8, 9, 10]. However, motion capture is not the final solution because of the following two disadvantages. First, we have to capture or purchase new motion data every time we need a motion not included in our library. Secondly, 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 just for their taste, which again requires high skill and expensive software.

This paper describes the computational details of the 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 a whole-body motion in real time by a few pick-and-drag procedures. One can also specify other constraints such as desired joint values and joint motion ranges, which are useful for creating cyclic and human-like motions. The method is naturally extended to editing and retargeting existing motions by allowing moving pins and time-dependent desired joint values.

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 (dragged link) follows the indicated path,
2. any number of links specified by the user (pinned links) stay at their fixed positions,
3. each joint angle stays in its motion range, and
4. each joint angle stays as close as possible to the given desired 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 others (simply consider a case where one drags a link beyond its 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 gives 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[11]. The first constraint (the dragged link) is given the higher priority with which the solution never loses exactness. The other constraints are given the lower priority. To satisfy the constraints with the lower priority, the solution is looked for in the null-space of the first constraint. If there is a conflict among the constraints, a least-square optimization is done in the null-space to find the best approximation for the lower-priority constraints.

Although the null-space decomposition and the least-square solution are commonly done using the pseudoinverse, it yields in the neighborhood of singularity extremely large and, therefore, physically infeasible solutions. The singularity-robust (SR) inverse [12] is adopted to avoid the problem, since multiple constraints and conflicts among them are the issue to be dealt with in this paper and 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 once the singularities or conflicts disappear. 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 an "elastic" property, natural response, and reliability.

2.2 Summary of the Algorithm

The algorithm consists of the following five steps:

1. compute the general solutions of joint velocities that moves the dragged link towards 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 values (section 3.3)

4. using the general solution in step 1, find a particular solution that closely satisfies the desired velocities of the constraint variables (section 3.2)

The computational scheme proposed here has a number of advantages over the previous ones with similar objectives, as follows:

- the posture of the whole body is determined by moving a single link; in other words, a single drag affects the whole body
- any link can be dragged
- there is no limitation for the selection and number of pinned links
- constraint variables can be instantly included or removed
- relative importance of the constraint variables can be considered and tuned.

2.3 Relationship with the Previous Works

The main objective of this research 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 is a body of related research in this field, most of these efforts aim to solve the problem of editing or retargetting prerecorded data, deferring the problem of generating new motions as too hard.

Inverse kinematics is the key issue of this research. Many previous works on inverse kinematics problem use global optimization over the spacetime constraint of motion[3, 5, 6, 7]. In [3] the SR-inverse is employed to avoid singularity problems with the Jacobian matrix.

Online computation using local optimization, on the other hand, was investigated by Choi *et al.*[4]. This work is based on the feedback control and the null-space method similar to ours. In [4] the pinned links are assumed 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. Maciejewski[13] discussed a similar idea called the damped pseudoinverse.

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*[14]. It is not surprising that the algorithm for computing the inverse kinematics with multiple goals is similar to ours. However there are several important differences:

- In Badler'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 [15]. We can eliminate the overhead to switch the dragged links, providing a more responsive and comfortable interface to the user.
- Badler's system considered joint motion range of rotational (or 1DOF) joints, and it would be 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.

Representation of motion ranges of 3DOF spherical joints is fundamental to obtain natural behaviors of human characters. Simple inequality representation of the Euler angles is inappropriate due to their nonlinearity[16, 17]. A precise anatomical modeling has been proposed recently[17]. In this paper, we propose a simple representation of spherical joint limit modeling which is efficient for online computation.

3 Computational Details

3.1 The Dragged Link

First we compute $\dot{\boldsymbol{\theta}}$ with which the dragged link exactly follows its reference of 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

$$\dot{\boldsymbol{r}}_P^d = \dot{\boldsymbol{r}}_P^{ref} + \boldsymbol{K}_P(\boldsymbol{r}_P^{ref} - \boldsymbol{r}_P) \tag{1}$$

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

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

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

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_P^\sharp \dot{\boldsymbol{r}}_P^d + (\boldsymbol{E} - \boldsymbol{J}_P^\sharp \boldsymbol{J}_P)\boldsymbol{y}. \tag{3}$$

where $\boldsymbol{J}_P^\sharp$ is the weighted pseudoinverse of $\boldsymbol{J}_P$, $\boldsymbol{E}$ is the identity matrix of the appropriate size, and $\boldsymbol{y}$ is an arbitrary vector. The feedback control is applied only to eliminate the numerical errors. Weighted pseudoinverse[18] may be used in the above equation instead of the normal pseudoinverse to characterize the joint motions.

3.2 Lower-Priority Constraints

The general solutions of Eq.(3) is rewritten by

$$\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\theta}}_0 + \boldsymbol{W}\boldsymbol{y} \tag{4}$$

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

Suppose N_F pinned links whose velocities are denoted by $\dot{\boldsymbol{r}}_{Fi}(i = 1 \ldots N_F)$, N_D joints with their joint velocities $\dot{\boldsymbol{\theta}}_D$, and N_L joints with their joint velocities $\dot{\boldsymbol{\theta}}_L$ 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 $\dot{\boldsymbol{p}}_{aux}$ as follows:

$$\dot{\boldsymbol{p}}_{aux} \triangleq \begin{pmatrix} \dot{\boldsymbol{r}}_{F1}^T & \ldots & \dot{\boldsymbol{r}}_{FN_F}^T & \dot{\boldsymbol{\theta}}_D^T & \dot{\boldsymbol{\theta}}_L^T \end{pmatrix}^T \tag{5}$$

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

$$\dot{\boldsymbol{p}}_{aux} = \boldsymbol{J}_{aux}\dot{\boldsymbol{\theta}}. \tag{6}$$

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

The arbitrary vector $\boldsymbol{y}$ is computed as follows: We first compute the desired velocity $\dot{\boldsymbol{p}}_{aux}^d$ to take account of the errors between the constraint conditions and their current values as described in section 3.4. Substituting Eq.(4) into Eq.(6) yields

$$\dot{\boldsymbol{p}}_{aux} = \dot{\boldsymbol{p}}_{aux}^0 + \boldsymbol{J}_{aux}\boldsymbol{W}\boldsymbol{y} \tag{7}$$

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:

$$\boldsymbol{S}\boldsymbol{y} = \Delta\dot{\boldsymbol{p}}_{aux}. \tag{8}$$

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

$$\boldsymbol{y} = \boldsymbol{S}^*\Delta\dot{\boldsymbol{p}}_{aux}. \tag{9}$$

The joint velocity $\dot{\boldsymbol{\theta}}$ is obtained by substituting Eq.(9) into Eq.(4), which is then integrated to yield the joint angle $\boldsymbol{\theta}$ for animation.

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

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

$$\dot{\boldsymbol{r}}_{Fi} = \boldsymbol{J}_{Fi}\dot{\boldsymbol{\theta}}. \tag{10}$$

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

$$\dot{\boldsymbol{\theta}}_D = \boldsymbol{J}_D\dot{\boldsymbol{\theta}} \tag{11}$$

where $\boldsymbol{J}_D$ is the matrix whose (i, j)-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:

$$\dot{\boldsymbol{\theta}}_L = \boldsymbol{J}_L\dot{\boldsymbol{\theta}} \tag{12}$$

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

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

$$\boldsymbol{J}_{aux} = \begin{pmatrix} \boldsymbol{J}_{F1}^T & \ldots & \boldsymbol{J}_{FN_F}^T & \boldsymbol{J}_D^T & \boldsymbol{J}_L^T \end{pmatrix}^T. \tag{13}$$

The computation of columns of $\boldsymbol{J}_{Fi}, \boldsymbol{J}_P$ and $\boldsymbol{J}_L$ corresponding to spherical joints is mentioned 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:

$$\dot{\boldsymbol{r}}_{Fi}^d = \boldsymbol{K}_{Fi}(\boldsymbol{r}_{Fi}^{ref} - \boldsymbol{r}_{Fi}) \tag{14}$$

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

$\dot{\boldsymbol{\theta}}_D^d$ and $\dot{\boldsymbol{\theta}}_L^d$ are collections of the desired joint velocities. The number of elements in the vectors corresponding to each joint equals to the degrees of freedom of the joint. A rotational joint, for example, thus has one element, while a spherical joint has three. In this subsection the methods to compute the elements for 1DOF joints are presented. Section 3.5 extends the ideas to 3DOF spherical joints.

If a desired joint angle θ_{Di}^{ref} is given to joint i, its desired joint velocity $\dot{\theta}_{Di}^{d}$ is computed by

$$\dot{\theta}_{Di}^{d} = K_{Di}(\theta_{Di}^{ref} - \theta_i) \tag{15}$$

where $K_{Di} > 0$ is the gain and θ_i is current joint angle.

If joint i is exceeding its joint motion range, its desired velocity $\dot{\theta}_{Li}^{d}$ is computed by

$$\dot{\theta}_{Li}^{d} = \begin{cases} K_{Li}(\theta_{Li}^{max} - \theta_{Li}) & \text{if } (\theta_{Li} > \theta_{Li}^{max}) \\ K_{Li}(\theta_{Li}^{min} - \theta_{Li}) & \text{if } (\theta_{Li} < \theta_{Li}^{min}) \end{cases} \tag{16}$$

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

3.5 Handling Spherical Joints

Desired Joint Value

The joint value $\boldsymbol{R}_i$ and the joint velocity $\boldsymbol{\omega}_i$ of a spherical joint are defined as the 3×3 orientation matrix and its associated angular velocity respectively, described in its parent link frame.

When a spherical joint is given a desired joint value $\boldsymbol{R}_{Di} \in \boldsymbol{R}^{3\times3}$, we compute its desired joint velocity as follows: We first compute the error vector $\boldsymbol{e}_i$ between the current joint value $\boldsymbol{R}_i$ and $\boldsymbol{R}_{Di}$ by

$$\boldsymbol{e}_i = \frac{1}{2}\begin{pmatrix} \Delta\boldsymbol{R}_i(1,2) - \Delta\boldsymbol{R}_i(2,3) \\ \Delta\boldsymbol{R}_i(1,3) - \Delta\boldsymbol{R}_i(3,1) \\ \Delta\boldsymbol{R}_i(2,1) - \Delta\boldsymbol{R}_i(3,2) \end{pmatrix} \tag{17}$$

$$\Delta\boldsymbol{R}_i \triangleq \boldsymbol{R}_{Di}\boldsymbol{R}_i^T \tag{18}$$

where $\Delta\boldsymbol{R}_i(m,n)$ denotes the (m,n)-th element of $\Delta\boldsymbol{R}_i$. Then, the desired angular velocity $\boldsymbol{\omega}_{Di}^{d}$ is computed by[19]

$$\boldsymbol{\omega}_{Di}^{d} = -\boldsymbol{K}_{Di}\boldsymbol{e}_i \tag{19}$$

where $\boldsymbol{K}_{Di}$ is a positive-definite gain matrix. Equations (17)–(19) are used for spherical joints in place of Eq.(15).

The Jacobian matrix to be included in $\boldsymbol{J}_{Fi}, \boldsymbol{J}_D$ and $\boldsymbol{J}_L$ for a spherical joint has three columns, each of which corresponding to the rotation around x, y or z axis. Each column is computed just as if there is a rotational joint with the axis in the corresponding direction.

Joint Motion Range

The motion range of a spherical joint is expressed as a region in a three dimensional space. The geometric representation of the region is important for real-time computation. The region would show a complex shape if we represent it with the common coordinates such as the Euler angle, for example, 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[20], it is different in the sense that the proposed one consists of two sequential rotations. The representation provides three parameters, two of which describe the link direction and the other denotes the twist angle at each direction, as illustrated in Fig.2.

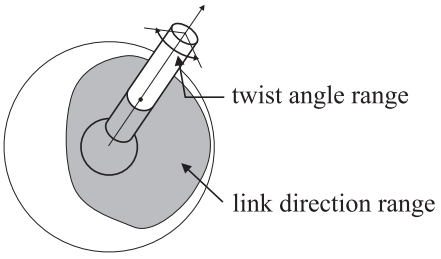

Fig.2: Joint motion range of a spherical joint

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 unit, but $\sin(\gamma_i/2)$ where γ_i represents the angle of rotation. 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.

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

$$\begin{aligned} \boldsymbol{d}_i &= \boldsymbol{R}_i\boldsymbol{d}_i^0 \\ &= (\boldsymbol{R}_i(1,1)\ \boldsymbol{R}_i(2,1)\ \boldsymbol{R}_i(3,1))^T. \end{aligned} \tag{20}$$

Therefore a_y and a_z are computed by

$$a_y = -\frac{\boldsymbol{R}_i(3,1)}{\sqrt{2(1+\boldsymbol{R}_i(1,1))}} \tag{21}$$

$$a_z = \frac{\boldsymbol{R}_i(2,1)}{\sqrt{2(1+\boldsymbol{R}_i(1,1))}}. \tag{22}$$

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 Eqs.(21)(22) show singularity at $\gamma_i = \pm\pi$, it is not a practical problem since it is usually out of the joint limit.

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 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 towards $\boldsymbol{R}_{Si}$. This is achieved by simply substituting $\boldsymbol{R}_{Si}$ into $\boldsymbol{R}_{Di}$ in Eq.(18) and $\boldsymbol{\omega}_{Li}$ into $\boldsymbol{\omega}_{Di}$ in Eq.(19).

In our model, the motion ranges of 10 spherical joints are modeled by 8 to 35 trianglular cylinders depending on their shapes. Thanks to the simplicity of computation, handling spherical joints did not affect the real-time performance of the system.

4 Motion Editing in Motion

The computation in section 3 showed the algorithms for static choreography, where the desired positions of pins, $\boldsymbol{r}^{ref}_{Fi}$, and the desired joint angles, $\boldsymbol{\theta}^{ref}_{Di}$, were assumed constant, although 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 choreography is straightforward. By this extension, we can apply the algorithms to motion editing in motion and retargetting captured data. Changing the pins and the dragged link one after another while editing enables us to add motion flavors step by step and generate rich and complex motions of characters.

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

- The position of the pins are obtained by direct kinematics computation for each frame of the reference motion. Since each pin has reference velocity $\dot{\boldsymbol{r}}^{ref}_{Fi}$, the following equation is used instead of Eq.(14):

$$\dot{\boldsymbol{r}}^{d}_{Fi} = \dot{\boldsymbol{r}}^{ref}_{Fi} + \boldsymbol{K}_{Fi}(\boldsymbol{r}^{ref}_{Fi} - \boldsymbol{r}_{Fi}). \qquad (23)$$

- The desired joint values are set as the joint value data in the reference motion. Using the reference joint velocities $\dot{\boldsymbol{\theta}}^{ref}_{D}$, the following equation is used instead of Eq.(15):

$$\dot{\boldsymbol{\theta}}^{d}_{D} = \dot{\boldsymbol{\theta}}^{ref}_{D} + \boldsymbol{K}_{D}(\boldsymbol{\theta}^{ref}_{D} - \boldsymbol{\theta}_{D}) \qquad (24)$$

5 Examples[1]

The proposed method was implemented as the computational engine of CG animation software *Animanium*^{TM}. 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 computation time is small enough for real time motion generation of a human model with 45 to 54 degrees of freedom on a PC with a PentiumIII 1GHz processor.

5.1 Pin and Drag

Fig.3 shows various postures generated by a single pin-and-drag procedure from the initial posture (a), with both hands and feet pinned.

5.2 Real-Time Motion Generation

Images in Fig.4 are taken from a video clip recorded as 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 a 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.

5.3 Motion Editing in Motion

Figure 5 shows results of editing prerecorded motion. The original motion (above) 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 6 keyframes. Both feet were pinned by moving pins that move along

[1] Video clips of motions generated using the software are available at the following URL: http://www.ynl.t.u-tokyo.ac.jp/~katz/videos/index.html

Fig.4: An example of real-time motion generation

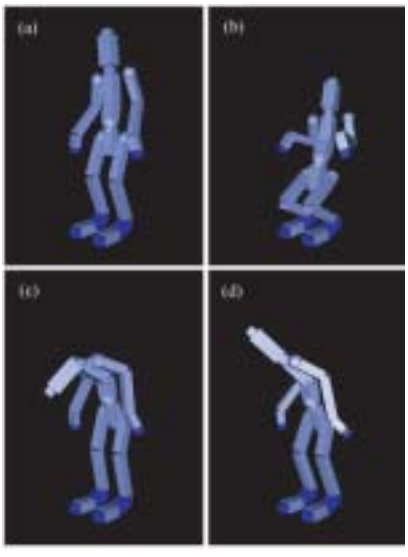

Fig.3: Postures generated by a single pin-and-drag; (a) original posture, (b) pelvis dragged, (c) head dragged, (d) left shoulder dragged

the trajectory determined by the original walking motion. Therefore, 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 towards 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 matrix.
3. Implemented constraints include pins, desired joint angles, and joint motion ranges. All these constraints are handled in a uniform way.
4. Motion editing 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 demonstrated their computational efficiency for real-time choreography without difficulty of use. Examples of created motions demonstrated the usefulness of the developed algorithms and software.

Acknowledgements

This research was supported by the CREST program of the Japan Science and Technology Corporation and the Information-Technology Promotion Agency (IPA) Japan. The first author is supported by the Japan Society for the Promotion of Science. The software is used as the computational engine of *Animanium*^{TM} (SEGA Corporation). The examples in section 5 were generated using this software.

References

[1] N.I. Badler, K.H. Manoochehri, and D. Baraff. Multi-dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, pages 151–169, Chapel Hill, NC, October 1986.

[2] C.B. Phillips, J. Zhao, and N.I. Badler. Interactive Real-time Articulated Figure Manipulation Using Multiple Kinematic Constraints. In *Proceedings of*

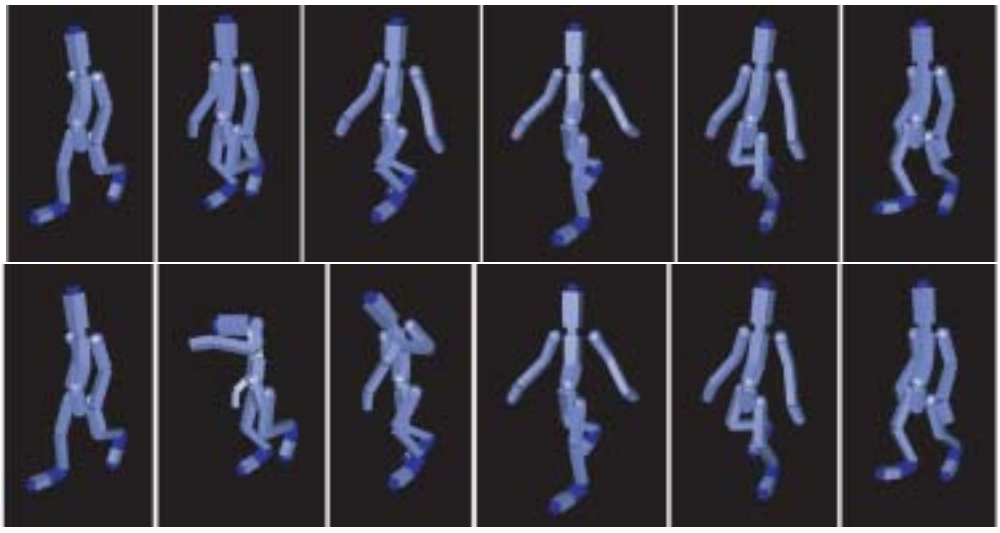

Fig.5: The original (above) and modified motions(below)

the 1990 Workshop on Interactive 3D Graphics, pages 245–250, March 1990.

[3] Michael Gleicher. Retargetting Motion to New Characters. In *Proceedings of SIGGRAPH '98*, pages 33–42, 1998.

[4] K.J. Choi and H.S. Ko. Online Motion Retargetting. *The Journal of Visualization and Computer Animation*, 11:223–235, 2000.

[5] C.F. Rose, P.-P. Sloan, and M.F. Choen. Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation. *Eurographics*, 20(3), 2001.

[6] C. Rose, B. Guenter, B. Bodenheimer, and M.F. Cohen. Efficient Generation of Motion Transitions using Spacetime Constraints. In *In Proceedings of SIGGRAPH'96*, pages 147–154, 1996.

[7] J.H. Lee and S.Y. Shin. A Hierarchical Approach to Interactive Motion Editing for Human-like Figures. In *Proceedings of SIGGRAPH'99*, pages 39–48, 1999.

[8] Z. Popovic. Editing Dynamic Properties of Captured Human Motion. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 670–675, San Francisco, CA, April 2000.

[9] N.S. Pollard and F. Behmaram-Mosavat. Force-Based Motion Editing for Locomotion Tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 663–669, San Francisco, CA, April 2000.

[10] K. Yamane and Y. Nakamura. Dynamics Filter — Concept and Implementation of On-Line Motion Generator for Human Figures. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 688–695, San Francisco, CA, April 2000.

[11] Y. Nakamura and H. Hanafusa. Task Priority Based Redundancy Control of Robot Manipulators. *International Journal of Robotics Research*, 6(2):3–15, 1987.

[12] Y. Nakamura and H. Hanafusa. Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[13] A.A. Maciejewski. Dealing with the Ill-conditioned Equations of Motion for Articulated Figures. *IEEE Computer Graphics and Applications*, 10(3):63–71, May 1990.

[14] C.B. Phillips and N.I. Badler. Jack: A Toolkit for Manipulating Articulated Figures. In *Proceedings of ACM/SIGGRAPH Symposium on User Interface Software*, 1988.

[15] Y. Nakamura and K. Yamane. Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Human Figures. *IEEE Transactions on Robotics and Automation*, 16(2):124–134, 2000.

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

[17] W. Maurel and D. Thalmann. Human Shoulder Modeling Including Scapulo-Thoracic Constraint and Joint Sinus Cones. *Computers and Graphics*, 24:203–218, 2000.

[18] Y Nakamura. *Advanced Robotics—Redundancy and Optimization*. Addison–Wesley , Reading, MA, 1991.

[19] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul. Resolved Acceleration Control of Mechanical Manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980.

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