# Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Human Figures

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

***Abstract*—This paper discusses the dynamics computation of structure-varying kinematic chains which imply mechanical link systems whose structure may change from open kinematic chain to closed one and vice versa. The proposed algorithm can handle and compute the dynamics and motions of any rigid link systems in a seamless manner without switching among algorithms. The computation is developed on the foundation of the dynamics computation algorithms established in robotics, which is superior in efficiency due to explicit use of the generalized coordinates to those used in the general-purpose motion analysis softwares. Although the structure-varying kinematic chains are commonly found in computing human and animal motions, the computation of their dynamics has not been discussed in literature. The developed computation will provide a general algorithm for the computation of motion and control of humanoid robots and computer graphics human figures.**

***Index Terms*—Closed kinematic chains, dynamics computation, human figures, humanoid, structure-varying systems.**

## I. Introduction

THE ADVANCE of humanoid robot research necessitates efficient computational schemes for simulating, controlling, and generating its motion. In computer graphics (CG), strongly demanded are the tools that can automatically create CG animations with dynamic motions of human and/or animal characters. The key issue concerning these two cases is how to generate various motions considering *physical consistency*, the condition that the motion is physically feasible. In fact, controlling a humanoid robot on the basis of physically consistent motion would contribute to the realization of fast, natural, and stable motions. In CG, physical consistency will lead to cost and time efficiency for generating natural human motions.

The objective of this paper is to establish the foundation for dynamics computation of human figures, namely, the computer models for real human motions. The developed methods will provide the basis for generating physically consistent motions. Compared to conventional robot manipulators for which most dynamics computation algorithms are designed, the motions of human figures have the following two major properties.

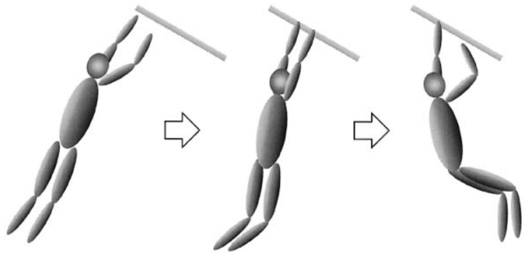
Fig. 1. Structure-varying kinematic chain.

1) The link connectivity may change during the motion from an open kinematic chain to a closed one and vice versa, by catching or releasing an object with the hands as illustrated in Fig. 1. Such systems are said to be *structure-varying* in this paper.
2) The kinematic chain usually has many degrees of freedom, sometimes including complicated closed kinematic chains. In fact, simply holding a bar with the both hands can generate a closed kinematic chain.

Thus, the dynamics computation algorithm for human figures is desired to: 1) readily handle structural changes and 2) efficiently compute the dynamics of closed kinematic chains with many degrees of freedom.

Dynamics computation of closed kinematic chains [1]–[4] and legged mechanisms[5], [6] has been discussed for many years. The dynamics computation algorithms currently used in general-purpose motion analysis softwares [2] can handle any mechanisms and simulate their motions. However, they tend to require enormous amount of computation because of the large number of coordinates they use. In robotics, the dynamics computation algorithms have been developed taking account of their efficiency and adopting the minimal number of coordinates. The algorithms were extended from open kinematic chains to closed ones, where the closed chain is transformed into equivalent tree structure by virtually cutting some joints in closed loops. Most of them use the Lagrange multipliers to compute the constraint force and moment at the cut joints [3]. An alternative approach was proposed by Nakamura and Ghodoussi [4], where the Jacobian matrix of unactuated joints with respect to actuated ones is used instead of the Lagrange multipliers. This approach uses the minimal number of coordinates and is computationally efficient. However, the computation of the Jacobian matrix was

Manuscript received October 13, 1998; revised July 7, 1999. This paper was recommended for publication by Associate Editor H. Zhuang and Editor S. Salcudean upon evaluation of the reviewers' comments. This work was supported by the Advanced Software Enrichment Project of the Information-Technology Promotion Agency (IPA), Japan, the Humanoid Robotics Project, MITI, Japan, and the CREST Program of the Japan Science and Technology Corporation under the "Robot Brain Project."

The authors are with the Department of Mechano-Informatics, University of Tokyo, Tokyo 113-8656, Japan (e-mail: nakamura@ynl.t.u-tokyo.ac.jp; katz@ynl.t.u-tokyo.ac.jp).

Publisher Item Identifier S 1042-296X(00)03575-8.

shown only for simple closed kinematic chains such as parallel five-bar-link structures. Therefore, the systematic computation of the Jacobian matrix of unactuated joints with respect to actuated ones for the general closed kinematic chains remains an open research issue.

The difficulty of handling structure-varying systems may depend on how the link connectivity of the mechanism is described. However, the method of describing link connectivity has seldom been discussed from that point of view. Previous researches represented the relation of links via linear graphs [7], matrices [8], or vectors [9]. From a practical or programming point of view, however, they are not always effective because the program has to search among elements to find out whether there exists a closed loop or even which link is connected to one.

In this paper, we first introduce the generalized coordinates of a closed kinematic chain, which are defined as the independent variables that represent the mobility of the kinematic chain. As for a designed manipulator, we know in advance its degrees of freedom and the variables that represent the motion. For closed kinematic chains found in motions of human figures, on the other hand, since we cannot predict their structures, the generalized coordinates or the degrees of freedom are not defined or computed in advance. We develop a general algorithm that systematically selects the generalized coordinates and computes the degrees of freedom of the system. The developed algorithm is used with the previously proposed efficient algorithm of Nakamura *et al.* [4] and applied to compute the inverse and forward dynamics of general closed kinematic chains.

Second, the method of describing link connectivity is presented. *Pointers*, a function of C/C++ programming language, are applied to describe open kinematic chains. In order to describe closed chains, *virtual links* are also introduced. The method is suitable for implementing the dynamics computation algorithm proposed in this paper. Khalil *et al.* [10] proposed a notation for closed kinematic chains and used a similar concept, but they focused on the notation of geometry of links and did not explicitly use an additional link for a closed loop.

Handling structural change is then discussed. Our description with pointers and virtual links are shown to be powerful in handling structural changes. We also establish the computation of the velocity boundary condition after structural change with collision due to nonzero relative velocities.

Finally, two examples of dynamics simulation of human figures with structural changes are presented and followed by the conclusions.

## II. Dynamics Computation of General Closed Kinematic Chains

### A. Generalized Coordinates of Closed Kinematic Chains

Consider a closed kinematic chain in Fig. 2. Let $N_J$ be the total number of joints in the chain, $\boldsymbol{\theta}_J \in \boldsymbol{R}^{N_J}$ the whole joint angles, $N_A$ the number of actuated joints, $\boldsymbol{\theta}_A \in \boldsymbol{R}^{N_A}$ the actuated joints, and $\boldsymbol{\tau}_A \in \boldsymbol{R}^{N_A}$ the actuator torques. In this section, we assume that the mechanism has rotational or translational joints of single-degree-of-freedom for simplicity sake. Introducing multi-degree-of-freedom joints requires no essential

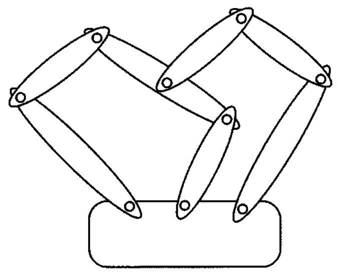
Fig. 2. Closed kinematic chain.

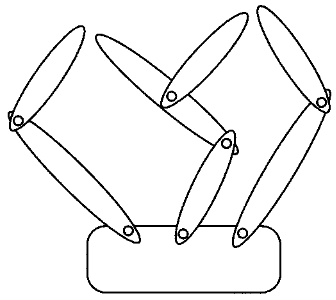
Fig. 3. Tree-structure open kinematic chain.

modification to the algorithm, as to be discussed later in Section VI.

Suppose that the closed chain is virtually cut at some joints and forms a tree-structure open kinematic chain in Fig. 3. Let $N_O$ be the number of joints in the tree-structure chain, $\boldsymbol{\theta}_O \in \boldsymbol{R}^{N_O}$ the joint angles, and $\boldsymbol{\tau}_O \in \boldsymbol{R}^{N_O}$ the joint torques. We assume at this moment that all joints in the tree structure, including those unactuated in the original closed chain, are actuated.

Suppose that the tree structure makes the same motion as the original closed chain without force or moment interaction at the virtually cut joints. The joint torques $\boldsymbol{\tau}_O$ required to generate the motion is computed by recursive inverse dynamics algorithms for open kinematic chains [11]–[13]. Note that nonzero values may be obtained for the elements of $\boldsymbol{\theta}_O$ corresponding to the unactuated joints in the original closed kinematic chain.

Let the original closed kinematic chain have $N_F$ degrees of freedom, $\boldsymbol{\theta}_G \in \boldsymbol{R}^{N_F}$ is the generalized coordinates that describe the mobility of the closed kinematic chain, and $\boldsymbol{\tau}_G$ the generalized force. We can form $\boldsymbol{\theta}_G$ by selecting appropriate $N_F$ joints from $\boldsymbol{\theta}_J$, for instance. Since the generalized coordinates determine the motion of the whole mechanism, $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_O$ can be written as follows:

$$\boldsymbol{\theta}_O = \boldsymbol{\theta}_O(\boldsymbol{\theta}_G) \tag{1}$$

$$\boldsymbol{\theta}_A = \boldsymbol{\theta}_A(\boldsymbol{\theta}_G). \tag{2}$$

From (1), the d'Alembert's principle, and the principle of virtual work, the joint torques of the tree structure $\boldsymbol{\tau}_O$ and the generalized forces $\boldsymbol{\tau}_G$ satisfy the following equation [4]:

$$\boldsymbol{\tau}_G^T \delta\boldsymbol{\theta}_G = \boldsymbol{\tau}_O^T \delta\boldsymbol{\theta}_O = \boldsymbol{\tau}_O^T \boldsymbol{W} \delta\boldsymbol{\theta}_G \tag{3}$$

where

$$\boldsymbol{W} \triangleq \frac{\partial\boldsymbol{\theta}_O}{\partial\boldsymbol{\theta}_G} \in \boldsymbol{R}^{N_O \times N_F}. \tag{4}$$

$\delta\boldsymbol{\theta}_O$ and $\delta\boldsymbol{\theta}_G$ are the virtual displacements of $\boldsymbol{\theta}_O$ and $\boldsymbol{\theta}_G$, respectively. Similarly, (2) and the principle of virtual work yield

$$\boldsymbol{\tau}_G^T\delta\boldsymbol{\theta}_G = \boldsymbol{\tau}_A^T\delta\boldsymbol{\theta}_A = \boldsymbol{\tau}_A^T\boldsymbol{S}\delta\boldsymbol{\theta}_G \tag{5}$$

where

$$\boldsymbol{S} \triangleq \frac{\partial\boldsymbol{\theta}_A}{\partial\boldsymbol{\theta}_G} \in \boldsymbol{R}^{N_A\times N_F}. \tag{6}$$

$\delta\boldsymbol{\theta}_A$ is the virtual displacement of $\boldsymbol{\theta}_A$. Since (3) and (5) hold for any $\delta\boldsymbol{\theta}_G$, we have the following equations:

$$\boldsymbol{\tau}_G = \boldsymbol{W}^T\boldsymbol{\tau}_O \tag{7}$$

$$\boldsymbol{\tau}_G = \boldsymbol{S}^T\boldsymbol{\tau}_A. \tag{8}$$

We can compute the actuator torque $\boldsymbol{\tau}_A$ from those of the tree structure $\boldsymbol{\tau}_O$ through the generalized force $\boldsymbol{\tau}_G$ once the sensitivity matrices $\boldsymbol{W}$ and $\boldsymbol{S}$ are computed, which is the subject of Section II-B.

Nakamura *et al.* [4] did not use the generalized coordinates explicitly assuming that $\delta\boldsymbol{\theta}_G$ is taken as a subspace of $\delta\boldsymbol{\theta}_A$. As explained above, introducing the generalized coordinates eliminates unnecessary assumptions and restrictions on virtual cut joints and on the placement of actuated joints.

## B. Computation of $\boldsymbol{W}$ and $\boldsymbol{S}$

For many practical planar closed kinematic chains, $\boldsymbol{W}$ and $\boldsymbol{S}$ become constant and can be formed from visual inspection. It is also known that they are computed relatively easily for some special closed kinematic chains, such as parallel mechanisms. In this section we provide a general method for computing the two matrices.

Consider a loop illustrated in Fig. 4. The linear and angular velocities of the shadowed link $L$ is computed from $\dot{\boldsymbol{\theta}}_A$ as well as $\dot{\boldsymbol{\theta}}_B$ by multiplying the Jacobian matrices $\boldsymbol{J}_A$ and $\boldsymbol{J}_B$ of the position and orientation of link $L$ with respect to $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$, respectively. The closed loop imposes the constraint that the velocity of link $L$ computed from $\boldsymbol{\theta}_A$ should be equal to that from $\boldsymbol{\theta}_B$, namely

$$(\boldsymbol{J}_A \quad -\boldsymbol{J}_B)\begin{pmatrix}\dot{\boldsymbol{\theta}}_A\\ \dot{\boldsymbol{\theta}}_B\end{pmatrix} = \boldsymbol{O}. \tag{9}$$

Extending the discussion to the whole mechanism, the constraint due to the $i$th closed loop is written in the form

$$\boldsymbol{J}_{Li}\dot{\boldsymbol{\theta}}_J = \boldsymbol{O} \tag{10}$$

where $\boldsymbol{J}_{Li}$ is a $6 \times N_J$ matrix. The columns of $\boldsymbol{J}_{Li}$ consist of those of the Jacobian matrices of link $L$ with respect to the joint angles, which can be calculated in the same way as serial kinematic chains [14].

Let $N_L$ be the number of independent closed loop in the structure. Then, we get $N_L$ constraint matrices $\boldsymbol{J}_{Li}$ $(i{=}1, 2, \cdots, N_L)$, which forms the matrix $\boldsymbol{J}_C \in \boldsymbol{R}^{6N_L\times N_J}$ as

$$\boldsymbol{J}_C \triangleq \begin{pmatrix}\boldsymbol{J}_{L1}\\ \boldsymbol{J}_{L2}\\ \vdots\\ \boldsymbol{J}_{LN_L}\end{pmatrix}. \tag{11}$$

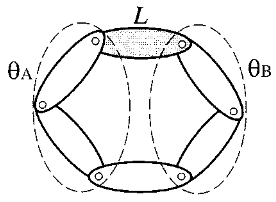

Fig. 4. Closed loop.

Although $\boldsymbol{J}_C$ represents all the kinematic constraints in the mechanism, not all the rows in $\boldsymbol{J}_C$ are independent, that is, $\boldsymbol{J}_C$ is not always full rank. We extract linearly independent rows from $\boldsymbol{J}_C$ and form $\boldsymbol{J}_{Cm} \in \boldsymbol{R}^{m\times N_J}$ where $m$ is the rank of $\boldsymbol{J}_C$. From (10), $\boldsymbol{J}_{Cm}$ satisfies

$$\boldsymbol{J}_{Cm}\dot{\boldsymbol{\theta}}_J = \boldsymbol{O}. \tag{12}$$

Equation (12) represents the $m$ independent constraints of the closed loops. Since we have $N_J$ joints under $m$ constraints, the remaining degrees of freedom (mobility) of the whole mechanism $N_F$ becomes

$$N_F = N_J - m. \tag{13}$$

Now, we form $\boldsymbol{J}_S$ by extracting $m$ independent columns from $\boldsymbol{J}_{Cm}$ and $\boldsymbol{J}_G$ by gathering the remaining columns. Also divide $\boldsymbol{\theta}_J$ into $\boldsymbol{\theta}_S$ and $\boldsymbol{\theta}_G$ according to the division of $\boldsymbol{J}_{Cm}$. From (12), $\boldsymbol{J}_S$, $\boldsymbol{J}_G$, $\boldsymbol{\theta}_S$, and $\boldsymbol{\theta}_G$ satisfy the equation

$$\boldsymbol{J}_{Cm}\dot{\boldsymbol{\theta}}_J = (\boldsymbol{J}_S \quad \boldsymbol{J}_G)\begin{pmatrix}\dot{\boldsymbol{\theta}}_S\\ \dot{\boldsymbol{\theta}}_G\end{pmatrix} = \boldsymbol{O}. \tag{14}$$

Equivalently

$$\boldsymbol{J}_S\dot{\boldsymbol{\theta}}_S = -\boldsymbol{J}_G\dot{\boldsymbol{\theta}}_G. \tag{15}$$

Since $\boldsymbol{J}_S$ is always invertible, $\dot{\boldsymbol{\theta}}_S$ is uniquely determined by

$$\dot{\boldsymbol{\theta}}_S = \boldsymbol{H}\dot{\boldsymbol{\theta}}_G \tag{16}$$

$$\boldsymbol{H} \triangleq \frac{\partial\boldsymbol{\theta}_S}{\partial\boldsymbol{\theta}_G} = -\boldsymbol{J}_S^{-1}\boldsymbol{J}_G. \tag{17}$$

Equation (16) implies that we can choose $\boldsymbol{\theta}_G$ as the generalized coordinates. It is worth pointing out that the generalized coordinates are automatically selected through the process of forming $\boldsymbol{J}_S$. Note that $\boldsymbol{J}_S$, the independent columns of $\boldsymbol{J}_{Cm}$, corresponds to the *dependent* joint angles $\boldsymbol{\theta}_S$, not to the *independent* ones $\boldsymbol{\theta}_G$.

The Jacobian matrices $\boldsymbol{W}$ and $\boldsymbol{S}$ are formed from $\boldsymbol{H}$ immediately as follows.

- $\boldsymbol{W}$: if the $i$th joint of $\boldsymbol{\theta}_O$ is not a member of the generalized coordinates and corresponds with the $j$th one of $\boldsymbol{\theta}_S$, then include the $j$th row of $\boldsymbol{H}$ as the $i$th row of $\boldsymbol{W}$. If it is a member of the generalized coordinates and corresponds with the $j$th joint of $\boldsymbol{\theta}_G$, then include a unit vector with $j$th element being 1 and others 0 as the $i$th row of $\boldsymbol{W}$. This procedure is shown in Fig. 5.
- $\boldsymbol{S}$: if the $i$th joint of $\boldsymbol{\theta}_A$ is not a member of the generalized coordinates and corresponds with the $j$th one of $\boldsymbol{\theta}_S$, then include the $j$th row of $\boldsymbol{H}$ as the $i$th row of $\boldsymbol{S}$. If it is a member of the generalized coordinates and corresponds

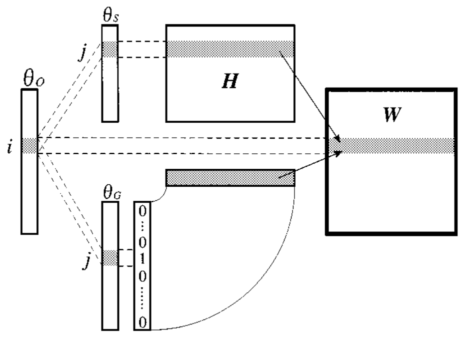

Fig. 5. Forming $\boldsymbol{W}$ from $\boldsymbol{H}$.

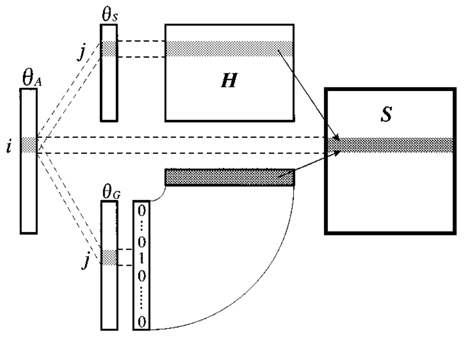

Fig. 6. Forming $\boldsymbol{S}$ from $\boldsymbol{H}$.

with the $j$th joint of $\boldsymbol{\theta}_G$, then include a unit vector with $j$th element being 1 and others 0 as the $i$th row of $\boldsymbol{S}$. This procedure is shown in Fig. 6.

### C. Relationship of Accelerations

Differentiating (16) by time yields

$$\ddot{\boldsymbol{\theta}}_S = \boldsymbol{H}\ddot{\boldsymbol{\theta}}_G + \dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_G \tag{18}$$

which calculates the acceleration of dependent joints $\ddot{\boldsymbol{\theta}}_S$ from the generalized acceleration $\ddot{\boldsymbol{\theta}}_G$. This computation is required in forward dynamics computation. In this section, computation of the second term of the right-hand side of (18) is presented.

From (17), we have

$$\dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_G = -\left\{\frac{d}{dt}(\boldsymbol{J}_S^{-1})\boldsymbol{J}_G + \boldsymbol{J}_S^{-1}\dot{\boldsymbol{J}}_G\right\}\dot{\boldsymbol{\theta}}_G. \tag{19}$$

On the other hand, $\boldsymbol{J}_S^{-1}\boldsymbol{J}_S = \boldsymbol{E}$, where $\boldsymbol{E}$ represents the indentity matrix, yields

$$\frac{d}{dt}(\boldsymbol{J}_S^{-1})\boldsymbol{J}_S + \boldsymbol{J}_S^{-1}\dot{\boldsymbol{J}}_S = \boldsymbol{O}. \tag{20}$$

Using (15) and (20), (19) becomes

$$\dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_G = -\boldsymbol{J}_S^{-1}(\dot{\boldsymbol{J}}_S\dot{\boldsymbol{\theta}}_S + \dot{\boldsymbol{J}}_G\dot{\boldsymbol{\theta}}_G) = -\boldsymbol{J}_S^{-1}\dot{\boldsymbol{J}}_{Cm}\dot{\boldsymbol{\theta}}_J. \tag{21}$$

$\dot{\boldsymbol{J}}_{Cm}\dot{\boldsymbol{\theta}}_J$ is formed by extracting the elements of $\dot{\boldsymbol{J}}_C\dot{\boldsymbol{\theta}}_J$ corresponding to $\boldsymbol{J}_{Cm}$, where $\dot{\boldsymbol{J}}_C\dot{\boldsymbol{\theta}}_J$ is computed in the same algorithm as one for serial manipulators [13].

## III. Inverse and Forward Dynamics of General Closed Kinematic Chains

### A. Inverse Dynamics

The inverse dynamics computation of general closed kinematic chains consists of the following four steps.

1) Compute $\boldsymbol{W}$ and $\boldsymbol{S}$.
2) Compute $\boldsymbol{\tau}_O$ by inverse dynamics computation for the tree structure.
3) Compute $\boldsymbol{\tau}_G$ by (7).
4) Compute $\boldsymbol{\tau}_A$ by solving the linear equation (8).

If the mechanism does not have actuation redundancy, namely, if the number of actuators equals to the degrees of freedom, $\boldsymbol{S}$ becomes a square matrix. Thus, $\boldsymbol{\tau}_A$ is computed by

$$\boldsymbol{\tau}_A = \boldsymbol{S}^{-T}\boldsymbol{W}^T\boldsymbol{\tau}_O. \tag{22}$$

Otherwise, $\boldsymbol{\tau}_A$ is not determined uniquely, and some optimization method should be applied. Refer to [15] for methods of optimizing actuation redundancy.

### B. Forward Dynamics

Although several forward dynamics algorithms are known for open kinematic chains [2], [8], [9], [11], it is difficult to apply them to closed chains due to the complexity of their structure. The unit vector approach [11], however, can be extended to closed chains easily.

The equation of motion of closed kinematic chains is written in the same form as open chains as

$$\boldsymbol{\tau}_G = \boldsymbol{A}(\boldsymbol{\theta}_G)\ddot{\boldsymbol{\theta}}_G + \boldsymbol{b}(\boldsymbol{\theta}_G, \dot{\boldsymbol{\theta}}_G) \tag{23}$$

where $\boldsymbol{\tau}_G \in \boldsymbol{R}^{N_F}$ is the generalized force, $\boldsymbol{A} \in \boldsymbol{R}^{N_F \times N_F}$ is the inertia matrix, and $\boldsymbol{b} \in \boldsymbol{R}^{N_F}$ includes the sum of centrifugal, Colliolis, and gravitational forces. In open kinematic chains, the joint angles are usually used as the generalized coordinate, and thus the joint torques are the generalized force. Therefore, the accelerations of all joints are computed directly by (23). In closed kinematic chains, on the other hand, the joint torque vector and the generalized force may differ even in their dimensions. Additional computations of transformation of the joint torques into the generalized force and the generalized acceleration into the joint acceleration are required.

The forward dynamics algorithm based on the inverse dynamics algorithm explained in Section II-A and the unit vector approach is summarized as follows.

1) Transform the input joint torques $\boldsymbol{\tau}_A$ into the generalized force $\boldsymbol{\tau}_G$ by (8).
2) Compute the inverse dynamics for the zero generalized acceleration and let the resultant generalized force be $\boldsymbol{b}$. Using (18), the accelerations of the dependent joints $\ddot{\boldsymbol{\theta}}_S$ are given by $\dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_G$, whose computation method is shown in Section II-C.
3) Execute the following computation for $i = 1, 2, \cdots, N_F$.
    a) Compute the inverse dynamics with $\ddot{\boldsymbol{\theta}}_G = \boldsymbol{e}_i$, where $\boldsymbol{e}_i \in \boldsymbol{R}^{N_F}$ is a unit vector whose $i$th element is 1 and others 0. The accelerations of the dependent

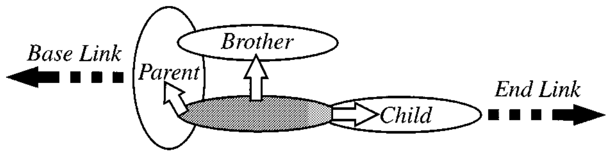

Fig. 7. Three pointers to describe open kinematic chains.

joints are computed by substituting $\boldsymbol{e}_i$ for $\ddot{\boldsymbol{\theta}}_G$ in (18).

b) Let the computed generalized force be $\boldsymbol{f}_i$ and calculate $\boldsymbol{a}_i$ by $\boldsymbol{a}_i = \boldsymbol{f}_i - \boldsymbol{b}$. We can obtain $\boldsymbol{a}_i$ directly by computing the inverse dynamics with $\ddot{\boldsymbol{\theta}}_G = \boldsymbol{e}_i$, $\dot{\boldsymbol{\theta}}_G = 0$, and no gravity, which would save the computational cost a little.

c) Include $\boldsymbol{a}_i$ as the $i$th column of $\boldsymbol{A}$.

4) Using $\boldsymbol{\tau}_G$, $\boldsymbol{b}$, and $\boldsymbol{A}$, compute the generalized acceleration by

$$\ddot{\boldsymbol{\theta}}_G = \boldsymbol{A}^{-1}(\boldsymbol{\tau}_G - \boldsymbol{b}). \tag{24}$$

5) Compute $\ddot{\boldsymbol{\theta}}_S$ by (18), where $\dot{\boldsymbol{H}}\dot{\boldsymbol{\theta}}_G$ is already computed in step 2), to get the accelerations of all joints.

## IV. Connectivity Description of Kinematic Chains

### A. Pointers Describe Open Kinematic Chains

For the efficiency of computation, and for the convenience of implementation, we propose to use *pointers* to describe link connectivity. *Pointer* is an important function of C/C++ programming language and acts as an arrow from one link to another. Since the value of a pointer is the address of a specified datum, we can refer to the data of a link in issue immediately through the pointer to it.

We use three pointers for each link to describe open kinematic chains. The meanings of the pointers are illustrated in Fig. 7. The *parent* pointer points the next link connected toward the base link. The *child* pointer, on the other hand, points the next link connected toward an end link. The *brother* pointer points a link with the same *parent*, in case the parent link has several links connected toward end links.

The recursive dynamics computations of the Newton–Euler formulation [12] are implemented using the three pointers and recursive call of functions. For the forward path computations, the functions are called recursively for the *child* and *brother* links after the computation for itself. For the backward path computations, on the other hand, recursive calls are made before the computation for itself.

### B. Virtual Links Describe Closed Kinematic Chains

The three pointers are applicable only to open kinematic chains, since the parent–child relationship for a closed kinematic chain results in an infinite loop.

First, as illustrated in Fig. 8, we virtually cut one joint in each closed loop to avoid infinite loops, just as we did in the dynamics computation. Since the mechanism is no longer a closed chain,

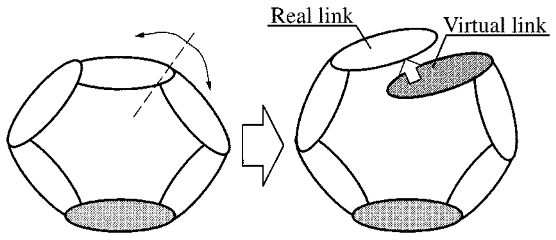

Fig. 8. Describing closed loop by virtual link.

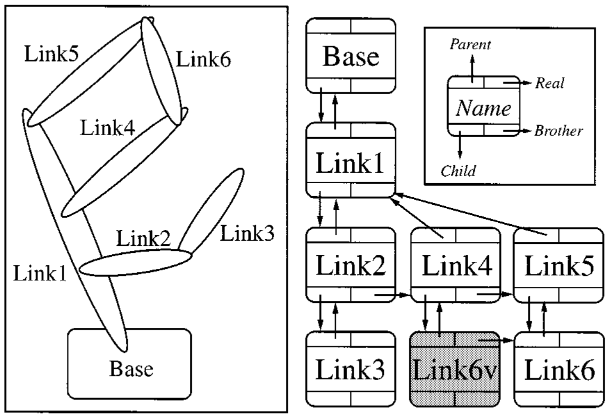

Fig. 9. Example of describing link structure.

we can describe it by the three pointers. To represent the connection at the virtually cut joints, we add a *virtual* link to one of the two links that had been connected by the cut joint. Since *virtual* link is introduced only to describe a closed loop, it has kinematic properties such as joint values and link length, but no dynamic properties such as mass or inertia. In order to indicate the real link of a virtual link, we introduce a new pointer called *real* pointer. The *real* pointer is valid only for virtual links. Note that the description of a closed chains is not unique and depends on which joint in a closed loop is virtually cut.

Virtual links represent the kinematic constraints of closed loops that each virtual link, and its corresponding real link should be in the same position and orientation. If the dynamics computation part finds a virtual link, it recognizes the real link through the *real* pointer and computes the matrix $\boldsymbol{J}_{Li}$ defined in (10).

To summarize, any open or closed kinematic chains are described by four pointers—*parent, child, brother*, and *real*—and a *virtual* link corresponding to each closed loop. An example of a description of a closed kinematic chain is shown in Fig. 9. The advantages of our approach are as follows.

- It is suitable for recursive implementation of dynamics computations.
- It is easy to find out closed loops, since each closed loop has a corresponding virtual link.
- It is a simple choice of virtual cut joints for dynamics computation. They coincide with the joints of the virtual links.
- There are less data and computation for link connectivities. They are proportional to the number of links.

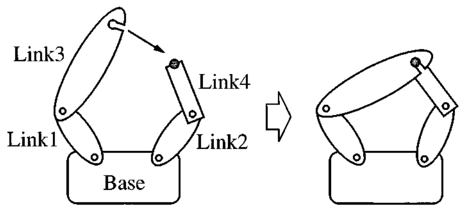

Fig. 10. Example of link connection.

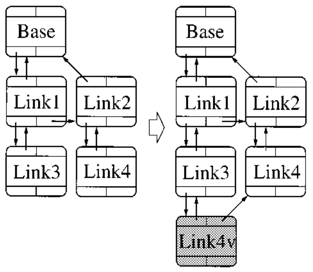

Fig. 11. Description of link connectivity before and after connection of Fig. 10.

## V. Dynamics Computation of Structure-Varying Kinematic Chains

### A. Structure-Varying Systems

Unlike industrial robots, the structure of human figures and animal figures may vary in time as they move. A human figure is a free-floating open kinematic chain of tree structure by himself. When he grabs a high bar with the both hands, he makes a closed kinematic chain. He might form another open kinematic chain by releasing one of the two hands. Even for a simple walk, dynamics computation of human figures might need to switch and use three models of kinematic chain: an open kinematic chain with only the left foot on the ground, a closed kinematic chain with the both feet on the ground, and another open kinematic chain with only the right foot on the ground. With conventional computation algorithms, we would have to prepare several different models and switch between them. We call such systems "structure-varying" ones, whose dynamics computation, to our knowledge, has not been established in literature.

The aim of this section is to develop a general method to handle structure changes seamlessly without switching between different dynamical models and algorithms. In Section V-B, we show that the algorithm developed in Section IV can attain the goal by taking an advantage of simple maintenance of link connectivity using pointers and virtual links.

### B. Link Connectivity Maintenance

First, consider a case in which two links are connected to create a new joint. If a closed loop is generated by the connection, as in a case illustrated in Fig. 10, we add a virtual link at the new joint. The procedure is as simple as the following.

1) Create virtual link *Link 4v* whose real link is *Link 4*.
2) Add *Link 4v* to the data as a child of *Link 3*.

This is easily programmed and computed online. The descriptions of link configurations before and after the connection are shown in Fig. 11.

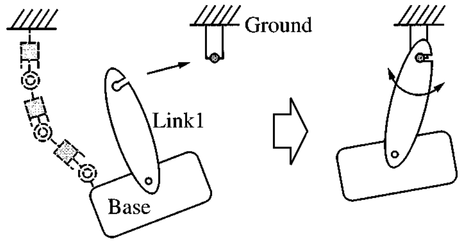

Fig. 12. Open kinematic chain generated by link connection.

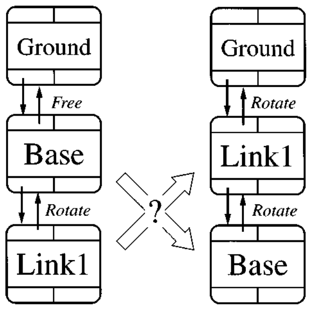

Fig. 13. Possible change of link connectivity description due to connection of Fig. 12.

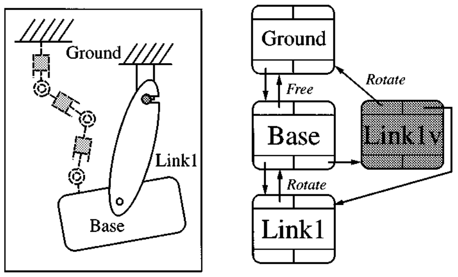

Fig. 14. Closed kinematic chain with free joint.

In the case where a free-floating chain is connected to another chain, the situation becomes complicated. Fig. 12 shows a case where *Link 1* of a free-flying chain is connected to *Ground* and a new joint is created. Since the structure after the connection is apparently an open chain, it seems natural to change the data as shown in Fig. 13. The remarks "*Rotate*" and "*Free*" in the figure indicate the joint types. One must notice, however, that it requires inversion of the parent–child relationship of *Base* and *Link 1*, which results in modification of the Denavit–Hartenberg parameters, the values of some dynamic parameters, and the indexing of joints. The modification is not difficult, but needs additional computation, which is crucial if the structure varies in real–time. When the structure change is known beforehand, the computational burden is reduced by preparing different connectivity models in advance, which would be, however, as tedious and complicated as switching between different dynamical models and algorithms.

We propose to treat this case exactly in the same way as the previous case as follows.

1) Create a virtual link of *Link 1* and name it *Link 1v*.
2) Connect *Link 1v* to *Ground* through the new rotational joint.

Fig. 14 shows the description of new structure, which does not require the inversion of the relationship of *Base* and *Link 1*.

Note that the number of links increases only by one as explained in Section VI-B, although the amount of dynamics computation in this case becomes larger than that when it is treated as an open chain. Therefore, we might need more careful comparison of computation loss due to ease of connectivity maintenance and computation gain due to increase of the number of joints. However, we claim the advantage of the above procedure from the following two viewpoints.

1) Simplicity of algorithm is valuable for programming and, eventually, offers better usability for the end-users.
2) The computation gain due to increase of the number of joints would be reduced in time by employing parallel processing [16], although the computation for connectivity maintenance cannot take an advantage of parallelism.

In the rest half of this section, we discuss the procedure for cutting a connection of two links at the joint between them. Note that this is a physical cutting, while the cutting in dynamics computation was virtual.

If the cut joint is related to a virtual link, the procedure is exactly the opposite of that in link connection. First suppose, in the structure after connection in Fig. 10, the joint between *Link 3* and *Link 4* is cut, which is handled simply by deleting *Link 4v*. For a human figure, connections and cuts commonly occur at the hands or feet. Therefore, when human figures are concerned, we can generally assume that cutting occurs only at the joints related to virtual links.

In general kinematic chains, however, this is not always the case. Even if the cut joint is not related to a virtual link, structure change can be readily handled by introducing a free joint as in Section VI-B. Suppose, in the structure after the connection in Fig. 10, that the joint between *Link 1* and *Link 3* is cut. The procedure in this case becomes the following.

1) Cut the parent–child relation between *Link 1* and *Link 3*.
2) Connect *Link 3* to *Base* by a free joint.

The link structure and its description are shown in Fig. 15. The connection between *Link 3* and *Link 4* is maintained by the virtual link *Link 4v*.

An alternative to deal with this situation would be to cut *Link 3* from *Link 1* and set as a child of *Link 4* in place of the virtual link *Link 4v*, in which case we can eliminate the closed loop. However, this scheme has the same problem as discussed in the examples of Fig. 12, that is, the inversion of parent–child relationship.

## C. Velocity Boundary Condition After Structure Changes

When two links are connected with nonzero relative velocity, discontinuous change of joint velocities occurs due to the collision. When they are cut, we can assume zero relative velocity except for those with explosions. The forward dynamics computation requires the boundary condition of joint velocities after the structure changes. In this section, we present an algorithm to compute the velocity boundary condition.

Suppose that the two connecting links belong to chain 1 and chain 2. Let $\boldsymbol{\theta}_i (i = 1, 2)$ be the generalized coordinates of chains 1 and 2, $\boldsymbol{J}_i = \partial \boldsymbol{r}/\partial \boldsymbol{\theta}_i$ the Jacobian matrices of the connection point $\boldsymbol{r}$ with respect to the generalized coordinates, and $\boldsymbol{A}_i$ their inertia matrices.

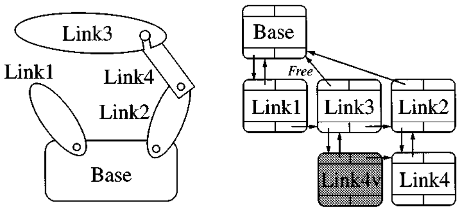

Fig. 15. Link structure and its description after cutting.

Also suppose that the generalized velocities change as much as $\Delta\dot{\boldsymbol{\theta}}_i$ due to the impact forces $\boldsymbol{F}_i$ applied to the two chains at the connection point, and a new $p$-degree-of-freedom joint $\boldsymbol{\theta}_n \in \boldsymbol{R}^p$ is created. According to the discussion in the previous section, a new virtual link is created at the connection point. Let $\boldsymbol{J}_n$ be the Jacobian matrix of the virtual link with respect to $\boldsymbol{\theta}_n$.

The applied force and the change of momentum of each chain are related by

$$\boldsymbol{A}_i \Delta\dot{\boldsymbol{\theta}}_i = -\boldsymbol{J}_i^T \boldsymbol{F}_i (i = 1, 2). \tag{25}$$

Since $\boldsymbol{F}_2$ is the reaction of $\boldsymbol{F}_1$, they satisfy

$$\boldsymbol{F}_2 = -\boldsymbol{F}_1. \tag{26}$$

On the other hand, the following equation is yielded by the condition that the velocity of the virtual link coincides with that of its real link:

$$\boldsymbol{J}_1(\dot{\boldsymbol{\theta}}_1 + \Delta\dot{\boldsymbol{\theta}}_1) = \boldsymbol{J}_2(\dot{\boldsymbol{\theta}}_2 + \Delta\dot{\boldsymbol{\theta}}_2) + \boldsymbol{J}_n\dot{\boldsymbol{\theta}}_n. \tag{27}$$

The impact force due to the collision has zero components along the free unconstrained directions of the new joint $\boldsymbol{\theta}_n$. This condition is expressed as

$$\boldsymbol{J}_n^T \boldsymbol{F}_1 = \boldsymbol{O}. \tag{28}$$

The change of generalized velocities $\Delta\dot{\boldsymbol{\theta}}_1$ and $\Delta\dot{\boldsymbol{\theta}}_2$ are computed from (25) to (28) as

$$\Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{A}_1^{-1}\boldsymbol{J}_1^T\boldsymbol{B}^{-1}(\boldsymbol{E}_6 - \boldsymbol{C})\boldsymbol{v} \tag{29}$$

$$\Delta\dot{\boldsymbol{\theta}}_2 = \boldsymbol{A}_2^{-1}\boldsymbol{J}_2^T\boldsymbol{B}^{-1}(\boldsymbol{E}_6 - \boldsymbol{C})\boldsymbol{v} \tag{30}$$

where

$$\boldsymbol{B} = \boldsymbol{J}_1\boldsymbol{A}_1^{-1}\boldsymbol{J}_1^T + \boldsymbol{J}_2\boldsymbol{A}_2^{-1}\boldsymbol{J}_2^T \tag{31}$$

$$\boldsymbol{C} = \boldsymbol{J}_n(\boldsymbol{J}_n^T\boldsymbol{B}^{-1}\boldsymbol{J}_n)^{-1}\boldsymbol{J}_n^T\boldsymbol{B}^{-1} \tag{32}$$

$$\boldsymbol{v} = \boldsymbol{J}_1\dot{\boldsymbol{\theta}}_1 - \boldsymbol{J}_2\dot{\boldsymbol{\theta}}_2 \tag{33}$$

and $\boldsymbol{E}_6$ is a 6 × 6 identity matrix.

If the connected two links are fixed to each other, namely $p = 0$, $\Delta\dot{\boldsymbol{\theta}}_1$ and $\Delta\dot{\boldsymbol{\theta}}_2$ are computed by substituting $\boldsymbol{O}$ to $\boldsymbol{C}$ in (29) and (30).

When the two links are in the same chain, the generalized coordinates and the mass matrices in the previous discussion coincide with each other, while the Jacobian matrices $\boldsymbol{J}_1$ and

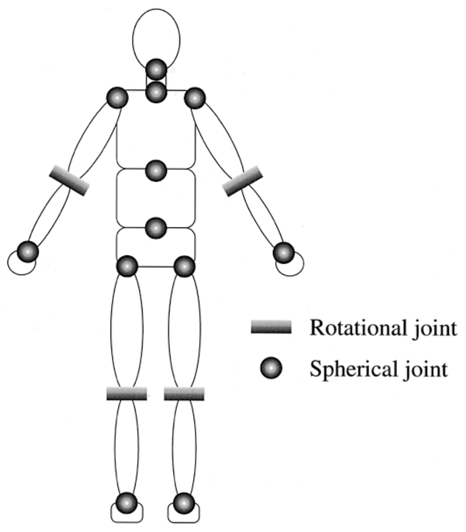

Fig. 16. Example of human model.

$\boldsymbol{J}_2$ are different. Therefore, the following equation is used in place of (25):

$$\boldsymbol{A}_1 \Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{J}_1^T \boldsymbol{F}_1 - \boldsymbol{J}_2^T \boldsymbol{F}_2. \tag{34}$$

The unknown, $\Delta\dot{\boldsymbol{\theta}}_1$, is solved by

$$\Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{A}_1^{-1} \boldsymbol{J}_1^T \tilde{\boldsymbol{B}}^{-1} (\boldsymbol{E}_6 - \tilde{\boldsymbol{C}}) \tilde{\boldsymbol{v}} \tag{35}$$

where

$$\tilde{\boldsymbol{B}} = (\boldsymbol{J}_1 - \boldsymbol{J}_2) \boldsymbol{A}_1^{-1} (\boldsymbol{J}_1 - \boldsymbol{J}_2)^T \tag{36}$$

$$\tilde{\boldsymbol{C}} = \boldsymbol{J}_n (\boldsymbol{J}_n^T \tilde{\boldsymbol{B}}^{-1} \boldsymbol{J}_n)^{-1} \boldsymbol{J}_n^T \tilde{\boldsymbol{B}}^{-1} \tag{37}$$

$$\tilde{\boldsymbol{v}} = (\boldsymbol{J}_1 - \boldsymbol{J}_2) \dot{\boldsymbol{\theta}}_1. \tag{38}$$

## VI. Multi-Degrees-of-Freedom Joints

### A. Spherical Joints

Fig. 16 illustrates the joint configuration of an example of human body model. The 40 degrees of freedom of the model include 4 rotational joints and 12 spherical joints, which shows that many joints in human bodies can be modeled as spherical joints.

In robot manipulators, a spherical joint is mechanically implemented as three successive rotational joints with their axes intersecting at a point. With this mechanical implementation and the modeling, the 40-degree-of-freedom model of the human body would need 41 links to treat in dynamics computation.

Physiological structure or implementation of human body is far more complex and beyond our scope of efficient computation. This fact requires and allows us to adopt a mechanical model that is suitable for computational efficiency and not necessarily constrained by mechanical implementation. As a computational model of human figures, we assume a spherical joint is equipped with a three-degree-of-freedom spherical motor or a similar actuator. With this assumption, we can significantly reduce the number of links. In fact, only 17 links are required for the model in Fig. 16 if spherical joints are considered. In addition, the description of link configuration becomes simpler and requires no discussion of artificial kinematic singularity.

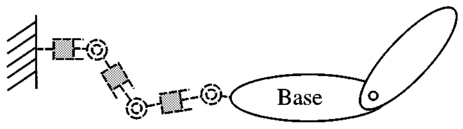

Fig. 17. Free-flying kinematic chain.

Three-degree-of-freedom spherical joints cause a difficulty in numerical integration of relative orientation between the two links connected by them. Although the Euler angles representation can avoid such problem, it has the problem of singularity. Integration problem would arise when we apply other methods such as the Euler parameters [14] to avoid singularity. We present below a method of first-order Euler integration of relative orientation using the Rodrigues' formula [14], which is often used for finite spatial rotation.

Let $\boldsymbol{\omega}_i$ be the relative angular velocity and $\boldsymbol{R}_i$ the relative orientation of link $i$ with respect to its parent link at time $t$. The relative orientation at $t + \Delta t$, $\boldsymbol{R}'_i$, is computed by

$$\boldsymbol{R}'_i = (\boldsymbol{E}_3 + \boldsymbol{\Omega} \sin\theta + \boldsymbol{\Omega}^2 (1 - \cos\theta)) \boldsymbol{R}_i \tag{39}$$

where

$$\boldsymbol{\theta} = \boldsymbol{\omega}_i \Delta t \tag{40}$$

$$\theta = |\boldsymbol{\theta}| \tag{41}$$

$$\theta (\overline{\omega}_x \overline{\omega}_y \overline{\omega}_z)^T = \boldsymbol{\theta} \tag{42}$$

$$\text{if } \theta \neq 0,\ \boldsymbol{\Omega} = \begin{pmatrix} 0 & -\overline{\omega}_z & \overline{\omega}_y \\ \overline{\omega}_z & 0 & -\overline{\omega}_x \\ -\overline{\omega}_y & \overline{\omega}_x & 0 \end{pmatrix} \tag{43}$$

$$\text{if } \theta = 0,\ \boldsymbol{\Omega} = \boldsymbol{O} \tag{44}$$

and $\boldsymbol{E}_3$ is a 3 × 3 identity matrix.

Spherical joints may be regarded as a combination of three rotaional joints whose axes lies on the $x$, $y$, and $z$ axes of the link coordinate. Therefore, relative angular velocities and accelerations of the two links expressed in link coordinates correspond to the joint velocity and acceleration of a single-degree-of-freedom joint.

### B. Free Joints

In order to treat the cases where the base link is not fixed to the inertial frame, we introduce a six-degree-of-freedom "free" joint between the base link and the inertial frame whose actuator torque is always zero, as illustrated in Fig. 17. Thus, forward dynamics is computed in the same way as base-fixed chains. Note that the six-degree-of-freedom joint is not decomposed into six single-degree-of-freedom joints but is treated as one joint. This can reduce the amount of computations especially for the recursive computation of kinematics and dynamics.

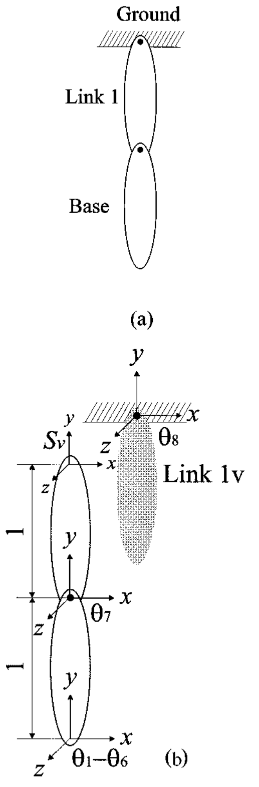

Fig. 18. Configuration in the numerical example.

Free joints may be regarded as a combination of three linear joints that can move in $x$, $y$, and $z$ directions, and a spherical joint. Therefore, the linear and angular velocities and accelerations of the free link expressed in the link coordinate correspond to the joint velocity and acceleration of a single-degree-of-freedom joint. Integrating the angular elements of joint velocities and accelerations is done in just the same way as spherical joints. The three elements of the linear part can be integrated independently as in three separate linear joints.

## VII. Examples

### A. Computation of $\boldsymbol{W}$ and $\boldsymbol{S}$

We first show an example of computing the matrices $\boldsymbol{W}$ and $\boldsymbol{S}$ for a simple structure. Consider the case already illustrated in Fig. 14, where the two-linked free-flying chain was connected to the ground through a new rotational joint attached to its end link. This structure would be treated as an open kinematic chain with two joints in conventional methods where we would have to pay the cost of replacing the kinematic model of the free-flying chain with the new two-joint model. In our scheme, on the other hand, it is described as a closed kinematic chain with eight joints, six of which are not actuated, maintaining the original parent–child relationship for generality and simplicity.

Suppose the chain is in a simple configuration shown in Fig. 18(a). Fig. 18(b) shows the existence of the virtual link *Link 1v* and the link coordinates. Coordinates $S_v$ is fixed on *Link 1* and coincides to *Link 1v* coordinates. Let $\dot{\theta}_1$–$\dot{\theta}_6$ be the linear and angular velocities of *Base*, $\dot{\theta}_7$ the joint angle of *Link 1*, $\dot{\theta}_8$ the joint angle of *Link 1v*, and $\dot{\boldsymbol{\theta}}_J = (\theta_1\theta_2\cdots\theta_8)^T$. From the constraint that linear and angular velocities of *Link 1v* coordinates should be equal to those of $S_v$, we can derive the constraint matrix $\boldsymbol{J}_C$ as

$$\boldsymbol{J}_C = \left(\begin{array}{ccc|ccc|c|c} 1 & 0 & 0 & 0 & 0 & -2 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{array}\right). \quad (45)$$

Since $m$, the row rank of $\boldsymbol{J}_C$ is 6, $\boldsymbol{J}_{Cm}$ is equal to $\boldsymbol{J}_C$, and the degrees of freedom of the structure become $N_F = N_J - m = 8 - 6 = 2$, which is equivalent to that of serial chain with two joints.

Now, we choose two dependent columns from $\boldsymbol{J}_{Cm}$ to select the two generalized coordinates. By simple visual inspection, it turns out that each of columns 2–5 are linearly independent and cannot be replaced by a linear combination of the other, and should be included in $\boldsymbol{J}_S$ of (14). Therefore, the generalized coordinates should be chosen from $\theta_1$, $x$ position of *Base*, $\theta_6$, rotation of *Base* about its $z$ axis, $\theta_7$, joint angle of *Link 1*, or $\theta_8$, joint angle of *Link 1v*. If we choose $\theta_1$ and $\theta_7$ as the generalized coordinates, $\boldsymbol{H}$ is computed as

$$\boldsymbol{H} = -\begin{pmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} 1 & -2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.5 & -0.5 \\ 0.5 & 0.5 \end{pmatrix}. \quad (46)$$

Since we have one closed loop, we have to cut one joint, $\theta_8$, to obtain the virtual open chain, which consists of seven joints $\theta_1$–$\theta_7$. The actuated joints in this case are $\theta_7$ and $\theta_8$. Thus, following the procedures illustrated in Figs. 5 and 6, the two matrices $\boldsymbol{W}$ and $\boldsymbol{S}$ are formed as follows:

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.5 & -0.5 \\ 0 & 1 \end{pmatrix} \quad (47)$$

$$\boldsymbol{S} = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \end{pmatrix}. \quad (48)$$

### B. Simulations

We show two examples of simulations of a simple human figure with structure changes. The algorithm is implemented using Microsoft Visual C++ and runs on a PC with Pentium Pro 200-MHz processor and an OpenGL graphic board. The

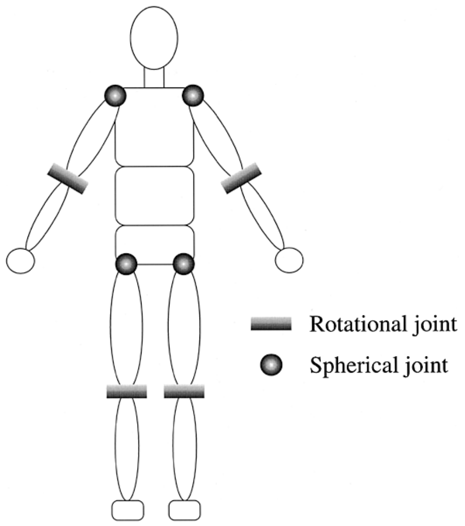

Fig. 19. Sixteen-degree-of-freedom human figure in the simulations.

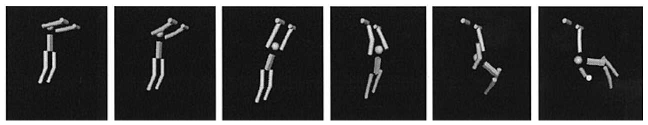

Fig. 20. High bar example of simulation of human figure.

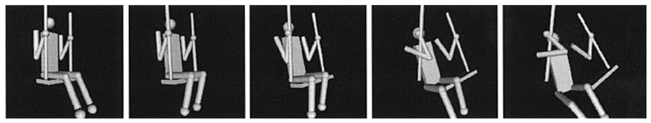

Fig. 21. Swing example of simulation of human figure.

human figure in the simulations has 16 degrees of freedom (four for each arm and leg) as illustrated in Fig. 19. We applied zero torques except for the case when we need to restrict the joint angles within their limits. The sampling time for the forward dynamics is approximately 25 ms in both examples.

**High Bar**: Fig. 20 shows a human figure playing high bar and releasing the right hand during the motion. Initially, there is a rotational joint between each hand and the bar, one of which is cut at an arbitrary given time. The connections are maintained, as discussed before, by two virtual links, whose real links are left and right hands. One of them is deleted when the cut occurs. The figure will be completely free-flying if the other virtual link is also deleted. In this case, including free joints and joints at the hands, we initially had 24 degrees of freedom in total and used 23 degrees of freedom after releasing the right hand.

**Swing**: What happens if a swing breaks down while you are playing on it? The answer is shown in Fig. 21. Each hand and the rod of the swing are connected by a three-degree-of-freedom spherical joint. There is a translational joint between each thigh and the plate of the swing, which is programmed to be cut when the thigh goes out of the plate. In this case, including the swing, we initially had 30 degrees of freedom in total and used 28 degrees of freedom in the final figure of Fig. 21.

## VIII. Conclusion

The results obtained in this research are summarized by the following six terms.

1) A dynamics computation algorithm for general closed kinematic chains is developed by introducing the concept of the generalized coordinates of closed kinematic chains to describe their mobility and by developing an algorithm to identify them automatically.
2) Link connectivity notation via pointers and virtual links is proposed, which is suited for both implementation and execution of dynamics computation algorithms.
3) A systematic and seamless online procedure of connectivity maintenance is developed. Any link connection or joint cutting are handled online, and there is no need to prepare every possible kinematic chain in advance.
4) A method of computing the velocity boundary condition after configuration changes is established, which is required when links are connected or cut with nonzero relative velocity.
5) It is shown that the number of links is reduced by considering 3-DOF spherical joints. We proposed to use 3-DOF spherical joints to model human figures for representational and computational simplicity. A method of integrating the relative orientation of the two links connected by a spherical joint is also presented.
6) The algorithms were implemented and examples of simulating motions of human figures verified their feasibility and computational efficiency.

## Acknowledgment

The authors would like to thank F. Nagashima, Fujitsu Laboratories Ltd. for offering the programming environment and discussions on connectivity description.

## References

[1] S. Y. Oh and D. Orin, "Dynamic computer simulation of multiple closed-chain robotic mechanisms," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. RA-1, Feb. 1986, pp. 15–20.
[2] E. J. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems*, ser. Allyn and Bacon in Engineering. Boston, MA: Allyn and Bacon, 1989.
[3] J. F. Kleinfinger and W. Khalil, "Dynamic modeling of closed-loop robots," in *Proc. 16th Int. Symp. Ind. Robots*, 1986, pp. 401–412.
[4] Y. Nakamura and M. Ghodoussi, "Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 294–302, June 1989.
[5] B. Perrin, C. Chevallereau, and C. Verdier, "Calculation of the direct dynamics model of walking robots: Comparison between two methods," in *Proc. IEEE Int.Conf. Robot. Automat.*, vol. 2, Apr. 1997, pp. 1088–1093.
[6] D. Surla and M. Rackovic, "Closed-form mathematical model of the dynamics of anthropomorphic locomotion robotic mechanism," in *Proc. 2nd ECPD Int. Conf. Adv. Robotics, Intell. Automat. Active Syst.*, 1996, pp. 327–331.
[7] J. J. McPhee, "Automatic generation of motion equations for planar mechanical systems using the new set of 'branch coordinates'," *Mech. Mach. Theory*, vol. 33, no. 6, pp. 805–823, 1998.
[8] F. M. L. Amirouche, *Computational Methods in Multibody Dynamics*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[9] R. Featherstone, *Robot Dynamics Algorithm*. Norwell, MA: Kluwer, 1987.
[10] W. Khalil and J. F. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. RA-2, Apr. 1986, pp. 1174–1179.

[11] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robot manipulators," *ASME Journal on Dynamic Systems, Measurement and Control*, vol. 104, pp. 205–211, 1982.
[12] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *ASME J. Dyn. Syst., Meas. Contr.*, vol. 104, pp. 69–76, 1980.
[13] Y. Nakamura, Y. Yokokohji, H. Hanafusa, and T. Yoshikawa, "Unified recursive formulation of kinematics and dynamics for robot manipulators," in *Proc. Japan–USA Symp. Flexible Automat.*, 1986, pp. 53–60.
[14] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Reading, MA: Addison-Wesley, 1986.
[15] Y. Nakamura and T. Ropponen, "Actuation redundancy of a closed link manipulator," in *Proc. 1990 Amer. Contr. Conf.*, 1990, pp. 2294–2299.
[16] K. Yamane, M. Okada, N. Komine, and Y. Nakamura, "Parallel dynamics computation and $H_\infty$ acceleration control of parallel manipulators for acceleration display," in *Proc. 1998 IEEE Int. Conf. Robot. Automat.*, 1998, pp. 2301–2308.

**Yoshihiko Nakamura** (M'87) was born in Osaka, Japan, in 1954. He received the B.S., M.S., and Ph.D. degrees from Kyoto University, Kyoto, 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 at Santa Barbara (UCSB), in 1987, as an Assistant Professor, and became an Associate Professor in 1990. He was also a Co-Director of the Center for Robotic Systems and Manufacturing at UCSB. Since 1991, he has been with the Department of Mechano-Informatics, University of Tokyo, Tokyo, Japan, and is currently a Professor. His research interests include redundant manipulators, actuation redundancy of closed kinematic chains, multi-robot coordination, multi-fingered robot hands, space robot control, motion control of mechanical systems with nonholonomic constraints, and medical robotics. He is the author of the textbook *Advanced Robotics: Redundancy and Optimization* (Reading, MA: Addison-Wesley).

**Katsu Yamane** was born in Kobe, Japan, in 1974. He received the B.S. and M.S. degrees in mechanical engineering from the University of Tokyo, Tokyo, Japan, in 1997 and 1999, respectively. Currently, he is working toward the Ph.D. degree at the Department of Mechano-Informatics, University of Tokyo.

Since 2000, he has been a Research Fellow of the Japan Society for the Promotion of Science. His research interests include dynamics simulation of human figures, control of humanoid robots, and generation of physically consistent animation in computer graphics.