Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Dynamics Compuation of Structure-Varying Kinematic Chains for Motion Synthesis of Humanoid

Katsu YAMANE and Yoshihiko NAKAMURA
(E-mail: [katz, nakamura]@ynl.t.u-tokyo.ac.jp)

Dept. of Mechano-Informatics, Univ. of Tokyo
7-3-1 Hongo Bunkyo-ku Tokyo, 113-8656 JAPAN

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 structure changes in a seamless manner without switching among algorithms for different kinematic chains. The structure-varying kinematic chains are commonly found in computing human motions. The developed computation will provide the general algorithm for the computation of motion and control of humanoid robots and CG human figures.

Key Words: Structure-varying kinematic chains, Dynamics computation, Humanoid, Closed kinematic chains.

1 Introduction

The advance of humanoid robot research necessitates efficient computational schemes for simulating, controlling and generating its motion. Also in computer graphics (CG), strongly demanded is 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 *dynamical consistency*, the condition that the motion is physically feasible. In fact, controlling a humanoid robot on the basis of dynamically consistent motion would contribute to the realization of fast, natural and stable motions. In CG, dynamical consistency will lead to cost and time efficiency for generating natural human motions.

Compared to conventional robot manipulators for which most dynamics computation algorithms are designed, the major property of human motions is that the link structure 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. In this paper, such systems are said to be *structure-varying*. Therefore, the dynamics computation algorithm for human motion is desired to readily handle structure changes. Dynamics simulation of locomotion has been discussed in several researches [1, 2, 3]. Although they provide efficient algorithms for computing the dynamics of walking human figures, they do not consider more complicated interactions such as catching high bar or other human figures' bodies.

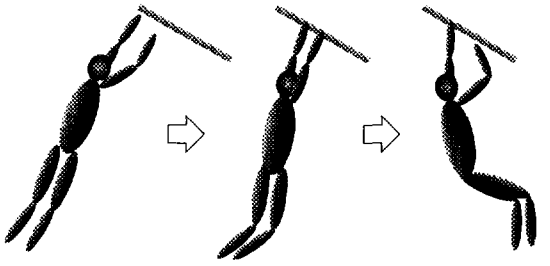

Fig.1: Structure-varying kinematic chain

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 [4], matrices [5] or vectors [6]. 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 briefly summarize our dynamics computation method for closed kinematic chains. Next, we present the method of describing link configuration. *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. Khalil et al.[7] proposed a notation for closed kinematic chains and uses a similar concept, but they focus on the notation of geometry of links and do not explicitly use an additional link for a closed loop.

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

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

2 Dynamics Computation of Closed Kinematic Chains

In this section, a summary of our dynamics computation method is presented.

We first virtually cut the closed loops to make a tree-structure open kinematic chain, and then apply the principle of virtual work to compute the inverse dynamics. This approach was originally proposed by Nakamura et al.[8] and also used in [9]. Finally, forward dynamics is computed by applying unit vector method[10].

Let $\boldsymbol{\theta}_G, \boldsymbol{\theta}_O$ and $\boldsymbol{\theta}_A$ be the generalized coordinate of the closed kinematic chain, joint angles of the virtual tree structure and the joint angles of actuated joints in the original closed kinematic chain, respectively. Using two Jacobian matrices $\boldsymbol{W}$ and $\boldsymbol{S}$ defined as:

$$\boldsymbol{W} \triangleq \frac{\partial \boldsymbol{\theta}_O}{\partial \boldsymbol{\theta}_G} \tag{1}$$

$$\boldsymbol{S} \triangleq \frac{\partial \boldsymbol{\theta}_A}{\partial \boldsymbol{\theta}_G} \tag{2}$$

the following equations are yieleded by the priciple of virtual work:

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

$$\boldsymbol{\tau}_G = \boldsymbol{S}^T \boldsymbol{\tau}_A \tag{4}$$

where $\boldsymbol{\tau}_G$ is the generalized force, $\boldsymbol{\tau}_O$ is the virtual torque of the tree structure and $\boldsymbol{\tau}_A$ is the actuator torque.

Inverse dynamics computation is summarized by the following four steps:

(1) Compute $\boldsymbol{W}$ and $\boldsymbol{S}$

(2) Compute $\boldsymbol{\tau}_O$ by inverse dynamics computation for the tree structure, i.e. Newton-Euler formulation[11]

(3) Compute $\boldsymbol{\tau}_G$ by Eq.(3)

(4) Compute $\boldsymbol{\tau}_A$ by solving the linear equation (4)

To apply this method to general closed kinematic chains, we developed a systematic procedure to automatically select the generalized coordinates $\ddot{\boldsymbol{\theta}}_G$ and compute the Jacobian matrices $\boldsymbol{W}$ and $\boldsymbol{S}$, which was not presented in [8].

First, we derive the independent contraints imposed by the closed loops through the geometric relationships of the joints. Suppose the constraint is expressed as

$$\boldsymbol{J}_{Cm} \dot{\boldsymbol{\theta}}_J = \boldsymbol{O} \tag{5}$$

where $\boldsymbol{\theta}_J$ is the joint angles and $\boldsymbol{J}_{Cm}$ is the constraint matrix. If there are N_J joints and $\boldsymbol{J}_{Cm}$ has m rows, which means that there are m independent constraints, then the degrees of freedom of the whole mechanism N_F is computed by

$$N_F = N_J - m \tag{6}$$

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 Eq.(5), $\boldsymbol{J}_S$, $\boldsymbol{J}_G$, $\boldsymbol{\theta}_S$ and $\boldsymbol{\theta}_G$ satisfy the equation

$$\boldsymbol{J}_{Cm} \dot{\boldsymbol{\theta}}_J = \begin{pmatrix} \boldsymbol{J}_S & \boldsymbol{J}_G \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{\theta}}_S \\ \dot{\boldsymbol{\theta}}_G \end{pmatrix} = \boldsymbol{O} \tag{7}$$

Equivalently,

$$\boldsymbol{J}_S \dot{\boldsymbol{\theta}}_S = -\boldsymbol{J}_G \dot{\boldsymbol{\theta}}_G \tag{8}$$

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

$$\dot{\boldsymbol{\theta}}_S = \boldsymbol{H} \dot{\boldsymbol{\theta}}_G \tag{9}$$

$$\boldsymbol{H} \triangleq \frac{\partial \boldsymbol{\theta}_S}{\partial \boldsymbol{\theta}_G} = -\boldsymbol{J}_S^{-1} \boldsymbol{J}_G \tag{10}$$

Equation (9) implies that we can choose $\boldsymbol{\theta}_G$ as the generalized coordinates. Finally, $\boldsymbol{W}$ and $\boldsymbol{S}$ are formed from $\boldsymbol{H}$ by selecting appropriate rows from $\boldsymbol{H}$.

3 Connectivity Description of Kinematic Chains

3.1 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 a 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.2. The *parent* pointer points the next link connected towards the base link. The *child* pointer, on the other hand, points the next link connected towards an end link. The *brother* pointer points a link with the same *parent*, in case the parent link has several links connected towards end links.

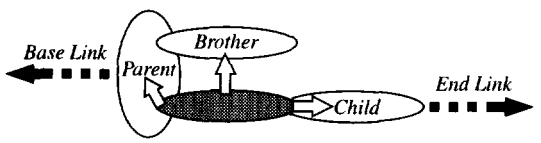

Fig.2: Three pointers to describe open kinematic chains

The recursive dynamics computations of the Newton-Euler formulation [11] 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.

3.2 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.3, 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, 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 chain is not unique and depends on which joint in a closed loop is virtually cut.

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

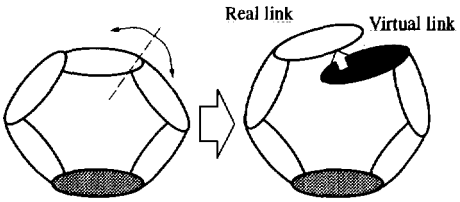

Fig.3: Describing closed loop by virtual link

closed loop. An example of a description of a closed kinematic chain is shown in Fig.4. The advantages of our approach are:

- Suitable for recursive implementation of dynamics computations.
- Easy to find out closed loops, since each closed loop has a corresponding virtual link.
- Simple choice of virtual cut joints for dynamics computation. They coincide with the joints of the virtual links.
- Less data and computation for link connectivities. It is proportional to the number of links.

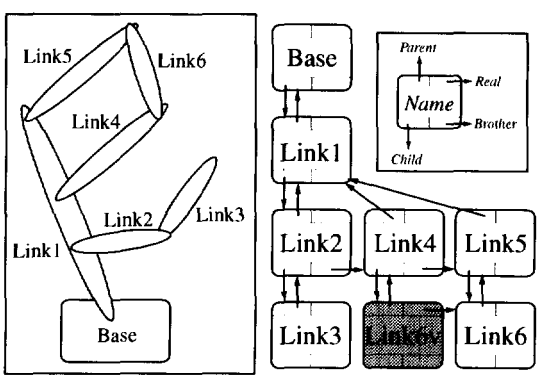

Fig.4: Example of describing kinematic chain

4 Dynamics Computation of Structure-Varying Kinematic Chains

4.1 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 structure 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 the following subsection, we show that the algorithm developed in sections 2 and 3 can attain the goal by taking an advantage of simple maintenance of link connectivity using pointers and virtual links.

4.2 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.5, we add a virtual link at the new joint. The procedure is as simple as:

(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*.

which is easily programmed and computed on line. The descriptions of link configurations before and after the connection are shown in Fig.6.

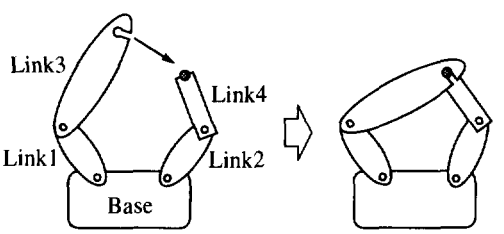

Fig.5: Example of link connection

In the case where a free-floating chain is connected to another chain, the situation becomes complicated. Figure 7 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.8. 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 and the values of some dynamic parameters,

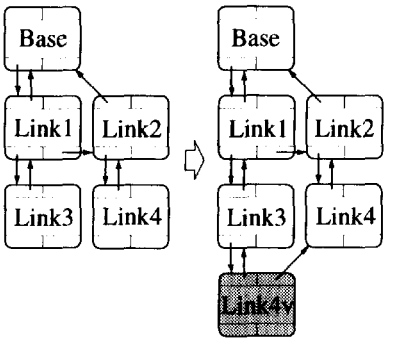

Fig.6: Link structure description before and after connection of Fig.5

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.

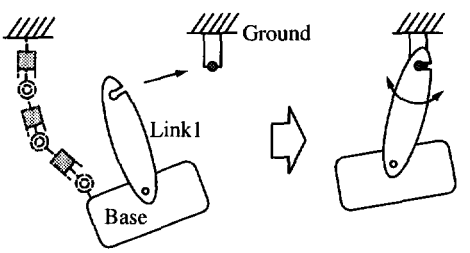

Fig.7: Open kinematic chain generated by link connection

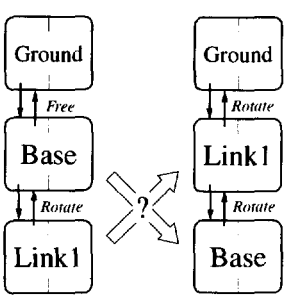

Fig.8: Possible change of link structure description due to connection of Fig.7

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

(1) Create a virtual link of *Link 1* and name it *Link 1v*.

(2) Connect *Link 1v* to *Ground* through the new rotational joint.

Figure 9 shows the description of new structure, which does not require the inversion of the relationship of *Base* and *Link 1*.

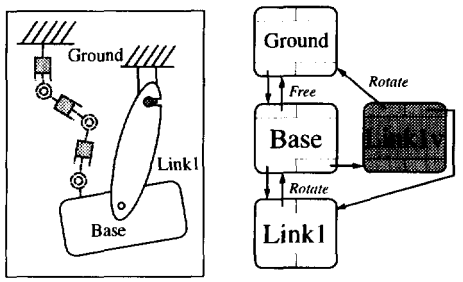

Fig.9: Closed kinematic chain with free joint

Note that the number of links increases only by one as explained in section 5.2 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[12], although the computation for connectivity maintenance cannot take an advantage of parallelism.

In the rest half of this subsection 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.5, 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 foots. Therefore, when human figures are concerned, we 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 5.2. Suppose, in the structure after the connection in Fig.5, that the joint between *Link 1* and *Link 3* is cut. The procedure in this case becomes:

(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.10. The connection between *Link 3* and *Link 4* is maintained by the virtual link *Link 4v*.

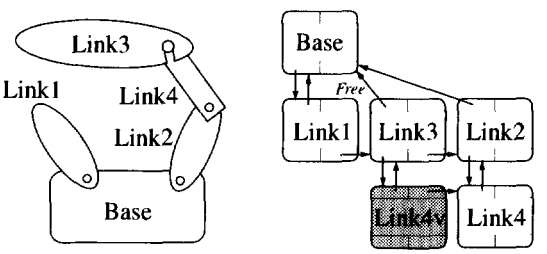

Fig.10: Link structure and its description after cutting

4.3 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.

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 degrees-of-freedom joint $\boldsymbol{\theta}_n \in \boldsymbol{R}^p$ is created. According to the discussion in the previous subsection, 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

$$\boldsymbol{A}_i \Delta\dot{\boldsymbol{\theta}}_i = -\boldsymbol{J}_i^T \boldsymbol{F}_i \quad (i = 1, 2) \tag{11}$$

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

$$\boldsymbol{F}_2 = -\boldsymbol{F}_1 \tag{12}$$

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.

$$\boldsymbol{J}_1(\dot{\boldsymbol{\theta}}_1 + \Delta\dot{\boldsymbol{\theta}}_1) = \boldsymbol{J}_2(\dot{\boldsymbol{\theta}}_2 + \Delta\dot{\boldsymbol{\theta}}_2) + \boldsymbol{J}_n\dot{\boldsymbol{\theta}}_n \tag{13}$$

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

$$\boldsymbol{J}_n^T\boldsymbol{F}_1 = \boldsymbol{O} \tag{14}$$

The change of generalized velocities $\Delta\dot{\boldsymbol{\theta}}_1$ and $\Delta\dot{\boldsymbol{\theta}}_2$ are computed from Eqs.(11) (14) as

$$\Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{A}_1^{-1}\boldsymbol{J}_1^T\boldsymbol{B}^{-1}(\boldsymbol{E}_6 - \boldsymbol{C})\boldsymbol{v} \tag{15}$$

$$\Delta\dot{\boldsymbol{\theta}}_2 = \boldsymbol{A}_2^{-1}\boldsymbol{J}_2^T\boldsymbol{B}^{-1}(\boldsymbol{E}_6 - \boldsymbol{C})\boldsymbol{v} \tag{16}$$

where

$$\boldsymbol{B} = \boldsymbol{J}_1\boldsymbol{A}_1^{-1}\boldsymbol{J}_1^T + \boldsymbol{J}_2\boldsymbol{A}_2^{-1}\boldsymbol{J}_2^T \tag{17}$$

$$\boldsymbol{C} = \boldsymbol{J}_n(\boldsymbol{J}_n^T\boldsymbol{B}^{-1}\boldsymbol{J}_n)^{-1}\boldsymbol{J}_n^T\boldsymbol{B}^{-1} \tag{18}$$

$$\boldsymbol{v} = \boldsymbol{J}_1\dot{\boldsymbol{\theta}}_1 - \boldsymbol{J}_2\dot{\boldsymbol{\theta}}_2 \tag{19}$$

and $\boldsymbol{E}_6$ is a 6×6 identity matrix. $\boldsymbol{B}$ and $\boldsymbol{J}_n^T\boldsymbol{B}^{-1}\boldsymbol{J}_n$ are invertible if either $\boldsymbol{J}_1$ or $\boldsymbol{J}_2$ and $\boldsymbol{J}_n$ are row full rank. Otherwise, in planar mechanisms, for instance, we may choose independent rows from $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$, in which case the sizes of $\boldsymbol{C}$ and $\boldsymbol{v}$ should be changed to appropriate ones.

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 Eqs.(15) and (16).

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 $\boldsymbol{J}_2$ are different. Therefore, the following equation is used in place of Eq.(11):

$$\boldsymbol{A}_1\Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{J}_1^T\boldsymbol{F}_1 - \boldsymbol{J}_2^T\boldsymbol{F}_2 \tag{20}$$

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

$$\Delta\dot{\boldsymbol{\theta}}_1 = -\boldsymbol{A}_1^{-1}\boldsymbol{J}_1^T\tilde{\boldsymbol{B}}^{-1}(\boldsymbol{E}_6 - \tilde{\boldsymbol{C}})\tilde{\boldsymbol{v}} \tag{21}$$

where

$$\tilde{\boldsymbol{B}} = (\boldsymbol{J}_1 - \boldsymbol{J}_2)\boldsymbol{A}_1^{-1}(\boldsymbol{J}_1 - \boldsymbol{J}_2)^T \tag{22}$$

$$\tilde{\boldsymbol{C}} = \boldsymbol{J}_n(\boldsymbol{J}_n^T\tilde{\boldsymbol{B}}^{-1}\boldsymbol{J}_n)^{-1}\boldsymbol{J}_n^T\tilde{\boldsymbol{B}}^{-1} \tag{23}$$

$$\tilde{\boldsymbol{v}} = (\boldsymbol{J}_1 - \boldsymbol{J}_2)\dot{\boldsymbol{\theta}}_1 \tag{24}$$

5 Multi-Degrees-of-Freedom Joints

5.1 Spherical Joints

Figure 11 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 d.o.f. model of 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 3 d.o.f. 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.11 if spherical joints are considered. In addition, the description of link configuration becomes simpler and requires no discussion of artificial kinematic singularity.

3 d.o.f. 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[13] to avoid singularity. We present below a method of first-order Euler integration of relative

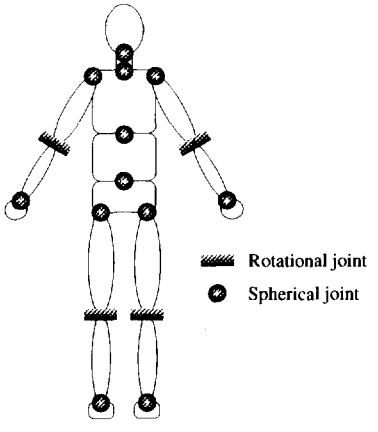

Fig.11: Example of human model

orientation using the Rodrigues' formula[13], 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

$$\boldsymbol{R}_i' = (\boldsymbol{E}_3 + \boldsymbol{\Omega} \sin\theta + \boldsymbol{\Omega}^2(1 - \cos\theta))\boldsymbol{R}_i \qquad (25)$$

where

$$\boldsymbol{\theta} = \boldsymbol{\omega}_i \Delta t \qquad (26)$$

$$\theta = |\boldsymbol{\theta}| \qquad (27)$$

$$\theta(\bar{\omega}_x\ \bar{\omega}_y\ \bar{\omega}_z)^T = \boldsymbol{\theta} \qquad (28)$$

$$\text{if } \theta \neq 0 \quad \boldsymbol{\Omega} = \begin{pmatrix} 0 & -\bar{\omega}_z & \bar{\omega}_y \\ \bar{\omega}_z & 0 & -\bar{\omega}_x \\ -\bar{\omega}_y & \bar{\omega}_x & 0 \end{pmatrix} \qquad (29)$$

$$\text{if } \theta = 0 \quad \boldsymbol{\Omega} = \boldsymbol{O}$$

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

5.2 Free Joints

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

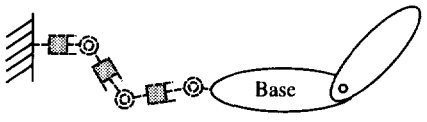

Fig.12: Free-flying kinematic chain

6 Examples

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 200MHz processor and an OpenGL graphic board. The human figure in the simulations has 16 degrees of freedom (4 for each arm and leg) as illustrated in Fig.13. 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 25msec in the both examples.

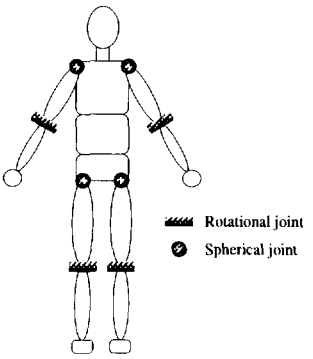

Fig.13: 16 d.o.f. human figure in the simulations

High Bar : Figure 14 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. 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.15. Each hand and the rod of the swing is connected by 3 d.o.f. 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.15.

7 Conclusions

The results obtained in this research are summarized by the following five terms:

(1) Link structure notation via pointers and virtual links is proposed, which is suited for both implementation and execution of dynamics computation algorithms.

(2) A systematic and seamless on-line procedure of connectivity maintenance is developed. Any link connection or joint cutting are handled on line, and there is no need to prepare every possible kinematic chain in advance.

(3) A method of computing the velocity boundary condition after configuration changes is established, which is required when links are connected or cut with non-zero relative velocity.

(4) It is shown that the number of links is reduced by considering 3 d.o.f. spherical joints. We proposed to use 3 d.o.f. spherical joints to model

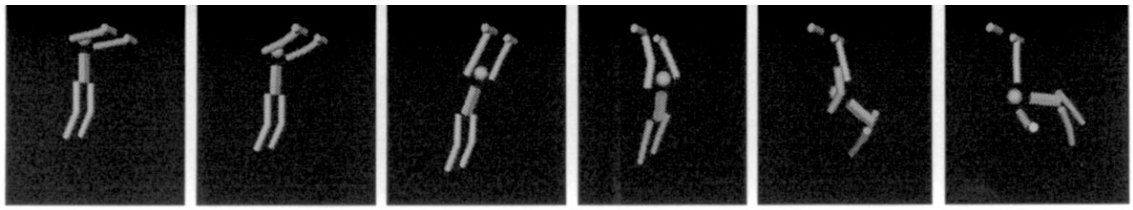

Fig.14: High bar example of simulation of human figure

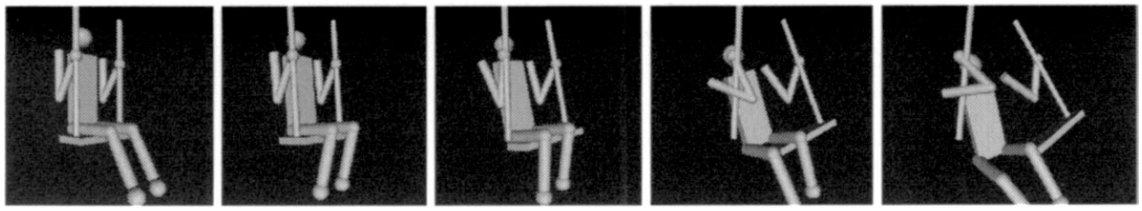

Fig.15: Swing example of simulation of human figure

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.

(5) The algorithms were implemented and examples of simulating motions of human figures verified their feasibility and computational efficiency.

Acknowledgments

We would like to thank Mr. Fumio Nagashima, Fujitsu Laboratories Ltd. for offering the programming environment and discussions on connectivity description. This research was supported by the Advanced Software Enrichment Project of the Information-technology Promotion Agency (IPA), Japan, and Humanoid Robotics Project, MITI, Japan.

References

[1] S.Y. Oh and D. Orin. Dynamic Computer Simulation of Multiple Closed-Chain Robotic Mechanisms. In *Proc. IEEE Int. Conf. on Robotics and Automation*, vol.1, pp.15 20, 1986.

[2] 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. on Robotics and Automation*, vol.2, pp.1088 1093, 1997.

[3] D. Surla and M. Rackovic. Closed-Form Mathematical Model of the Dynamics of Anthropomorphic Locomotion Robotic Mechanism. In *2nd ECPD Int. Conf. on Advanced Robotics, Intelligent Automation and Active Systems*, pp.327 331, 1996.

[4] J.J. McPhee. Automatic Generation of Motion Equations for Planar Mechanical Systems Using the New Set of "Branch Coordinates". *Mech. Mach. Theory*, 33(6):805 823, 1998.

[5] Farid M.L. Amirouche. *Computational Methods in Multibody Dynamics*. Prentice Hall, 1992.

[6] R. Featherstone. *Robot Dynamics Algorithm*. Kluwer, 1987.

[7] W. Khalil and J.F. Kleinfinger. A New Geometric Notation for Open and Closed-Loop Robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, vol.2, pp.1174 1179, 1986.

[8] Y. Nakamura and M. Ghodoussi. Dynamics Computation of Closed-Link Robot Mechanisms with Nonredundant and Redundant Actuators. *IEEE Trans. Robotics Automat.*, 5(3):294 302, 1989.

[9] A. Codourey and E. Burdet. A Body-Oriented Method for Finding a Linear Form of the Dynamic Equation of Fully Parallel Robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.1612 1618, 1997.

[10] M.W. Walker and D.E. Orin. Efficient Dynamic Computer Simulation of Robot Manipulators. *ASME J. of Dynamic Systems, Measurement and Control*, 104:205 211, 1982.

[11] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul. On-line Computational Scheme for Mechanical Manipulators. *ASME J. of Dynamic Systems, Measurement and Control*, 104:69 76, 1980.

[12] K. Yamane, M. Okada, N. Komine, and Y. Nakamura. Parallel Dynamics Computation and H_∞ Acceleration Control of Parallel Manipulators for Acceleration Display. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.2301 2308, 1998.

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