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

Efficient Parallel Dynamics Computation of Human Figures

Katsu YAMANE*1 Yoshihiko NAKAMURA*1*2
e-mail: {katz,nakamura}@ynl.t.u-tokyo.ac.jp
*1Department of Mechano-Informatics, University of Tokyo
*2CREST, Japan Science and Technology Corporation

Abstract

An efficient parallel algorithm for forward dynamics computation of human figures is to be proposed. The algorithm is capable of handling any kinematic chains including structure-varying ones. The asymptotic complexity of the algorithm is $O(N)$ in serial computation and $O(\log N)$ in parallel computation on $O(N)$ processors for most practical kinematic chains. The idea is to assemble a kinematic chain by adding the joints one by one and compute the constraint forces at the new joints using the Principle of Virtual Work. The parallelism of the algorithm can be adapted for parallel processing systems with any number of processors by simply changing the assembly order. Simulation examples on an 8-node cluster demonstrate the effectiveness of the algorithm.

Key Words: Forward Dynamics, Parallel Computation, Human Figures, $O(\log N)$ Complexity.

1 Introduction

Forward dynamics of kinematic chains has wide range of applications. The authors[1] have developed a scheme for computing the forward dynamics of structure-varying kinematic chains and applied it to a humanoid simulator[2]. The scheme can seemlessly handle any open and closed kinematic chains and transitions among them. However, the algorithm requires $O(N^3)$ computational cost, where N is the degrees of freedom of the kinematic chain, and would have difficulty in handling larger systems such as crowd, precise anatomical human models, and so on.

In this paper, we propose an efficient algorithm for parallel forward dynamics computation of structure-varying open and closed kinematic chains. The algorithm achieves $O(N)$ asymptotic complexity for serial computation and $O(\log N)$ time complexity for parallel computation on $O(N)$ processors for most practical kinematic chains. The basic procedure of the algorithm is, starting from a collection of free-flying links, to assemble the chain by adding joints one by one and compute the constraint force at the new joint. Then, since the constraint force computed at the assembly phase does not include the effect of joints added after the joint, we need the disassembly phase to compute the constraint forces in the completed chain, where the joints are removed in the reverse order of the assembly. The concept is illustrated in Fig.1.

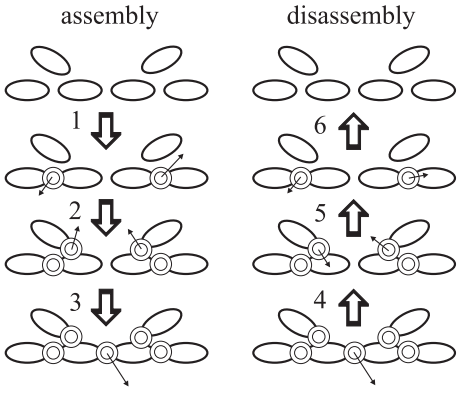

Fig.1: The idea of the algorithm

Parallel computation is realized by utilizing the order of assembly. During the assembly phase, if the intermediate chains do not have connections between each others, we can process the joint connection in parallel. Increasing parallelism, however, simultaneously leads to increase of total computational cost. It is therefore important to select appropriate assembly schedule to optimize the algorithm to the number of processors available.

Recent works on dynamics computation of kinematic chains include $O(N)$ algorithms for serial computation[3]-[8], $O(\log N)$ algorithms for parallel computation [8]-[11], and iterative approximation algorithms[11, 12]. Our method has three major advantages over previous ones:

1. Our method is based on a simple physical intuition rather than complicated mathematical devices[7, 8]. This fact eases the understanding and implementation of the algorithm.

2. The same algorithm can be used for serial and parallel computations on any number of processors. The difference is only in the order of adding and removing the joints in assembly and disassembly phases. Therefore, we can easily customize the algorithm for the available computational resource.

3. The algorithm gives an exact solution.

0-7803-7272-7/02/$17.00 © 2002 IEEE

2 Overview

The idea of the algorithm is illusrated in Fig.1. The free-flying links are assembled by adding a joint one by one to form the target kinematic chain, and then disassembled by removing the joints in the reverse order to return to the initial state. We call the intermediate kinematic chains found in the course of assembling as *subchains*. The target kinematic chain may be assembled in an arbitrary order. Therefore, if the subchains do not have connections with each others, we can process the assembly and disassembly computations in parallel. In Fig.1, for example, the two joints added or removed in steps 1, 2, 5 and 6 can be processed in parallel. Thus, although the kinematic chain contains 5 joints, the total computation time would be almost equivalent to handling 3 joints serially on one process.

Two quantities are computed in the assembly phase: the constraint force at the new joint and the acceleration of the points where new joints are to be added in the future. Note that the joints to be added in the future assembly steps are not considered at this stage. This is why we need the disassembly phase which computes the constraint forces in the target chain. Once we have all constraint forces, the link accelerations are easily computed by applying Newton and Euler equations of motion to each link. The accelerations of two neighboring links are then used to compute the acceleration of the joint between them.

The idea of assembling and disassembling links is similar to Divide-and-Conquer Algorithm (DCA) [9, 10] and Hybrid Direct/Iterative Algorithm[11], although the involved computations are different.

3 Preliminaries and Notations

3.1 Notations

Variables related to link k

$\boldsymbol{M}_k$	:	spatial inertial matrix ($\boldsymbol{R}^{6\times 6}$)
$\boldsymbol{c}_k$	:	centrifugal, colliolis, gravitational forces ($\boldsymbol{R}^6$)
$\dot{\boldsymbol{\theta}}_k$	:	spacial velocity ($\boldsymbol{R}^6$)
$\mathcal{J}_k$	:	set of joints connected to link k

Variables related to joint i

N_{Ci}	:	number of constraint conditions
N_{Fi}	=	$6 - N_{Ci}$, degrees of freedom
$\dot{\boldsymbol{q}}_i$	:	joint velocity ($\boldsymbol{R}^{N_{Fi}}$)
$\boldsymbol{\tau}_i$	:	joint torque ($\boldsymbol{R}^{N_{Fi}}$)
$\boldsymbol{f}_i$	:	constraint force ($\boldsymbol{R}^{N_{Ci}}$)
p_i	:	number of the parent-side link
c_i	:	number of the child-side link

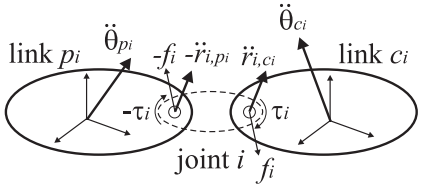

Fig.2: Frames attached to joints and links

$\boldsymbol{\tau}_i$ and $\boldsymbol{f}_i$ are defined as the force and/or moment applied to the link connected to the end-link side of joint i. The other link receives $-\boldsymbol{\tau}_i$ and $-\boldsymbol{f}_i$.

Variables related to subchain A

N_{LA}	:	number of links
N_{JA}	:	number of joints
N_{CA}	:	total number of constraints
N_{FA}	:	total number of DOF
$\dot{\boldsymbol{\theta}}_A$	:	vector composed of $\dot{\boldsymbol{\theta}}_k$ of all links ($\boldsymbol{R}^{6N_{LA}}$)
$\boldsymbol{c}_A$	:	vector composed of $\boldsymbol{c}_k$ of all links ($\boldsymbol{R}^{6N_{LA}}$)
$\boldsymbol{M}_A$	:	block diagonal matrix composed of $\boldsymbol{M}_k$ of all links ($\boldsymbol{R}^{6N_{LA}\times 6N_{LA}}$)
$\dot{\boldsymbol{q}}_A$	:	vector composed of $\boldsymbol{q}_k$ of all links ($\boldsymbol{R}^{N_{FA}}$)
$\mathcal{E}_A$	:	set of external joints connecting subchain A to other subchains
$\mathcal{L}_A$	:	set of links connected to joints in $\mathcal{E}_A$

3.2 Joint Constraints and Joint Variables

We attach several frames to each links and joints as shown in Fig.2, where link c_i is connected to link p_i towards the end link. One frame is attached to each link whose spatial velocity is denoted by $\dot{\boldsymbol{\theta}}_k$ as defined in the previous subsection. For joint i connecting two links p_i and c_i, we define two frames each fixed to one of the links, whose spatial velocities are denoted by $-\dot{\boldsymbol{r}}_{i,p_i} \in \boldsymbol{R}^6$ and $\dot{\boldsymbol{r}}_{i,c_i} \in \boldsymbol{R}^6$ respectively. Note that the direction of the velocity of link p_i at joint i is taken in the opposite direction to be consistent with the definition of $\boldsymbol{f}_i$ and $\boldsymbol{\tau}_i$.

We also give two matrices for each joint denoted by $\boldsymbol{K}_{Ci} \in \boldsymbol{R}^{N_{Ci}\times 6}$ and $\boldsymbol{K}_{Ji} \in \boldsymbol{R}^{N_{Fi}\times 6}$. $\boldsymbol{K}_{Ci}$ is given such that the constraint condition at joint i is expressed as

$$\boldsymbol{K}_{Ci}(\dot{\boldsymbol{r}}_{i,c_i} + \dot{\boldsymbol{r}}_{i,p_i}) = \boldsymbol{O}, \tag{1}$$

while $\boldsymbol{K}_{Ji}$ is given such that the joint velocity of joint i is expressed as

$$\dot{\boldsymbol{q}}_i = \boldsymbol{K}_{Ji}(\dot{\boldsymbol{r}}_{i,c_i} + \dot{\boldsymbol{r}}_{i,p_i}). \tag{2}$$

We assume $\boldsymbol{K}_{Ci}$ and $\boldsymbol{K}_{Ji}$ are constant for simplicity, which is the case for most practical joint types, although it is straightforward to include time-dependent $\boldsymbol{K}_{Ci}$ or $\boldsymbol{K}_{Ji}$.

Using the above quantities, we define the following Jacobian matrices and their derivatives:

$$\begin{aligned} \boldsymbol{J}_{i,k} &\triangleq \frac{\partial \boldsymbol{r}_{i,k}}{\partial \boldsymbol{\theta}_k} & \dot{\boldsymbol{J}}_{i,k} &\triangleq \dot{\boldsymbol{J}}_{i,k}\dot{\boldsymbol{\theta}}_k \\ \boldsymbol{H}_{i,k} &\triangleq \boldsymbol{K}_{Ci}\boldsymbol{J}_{i,k} & \boldsymbol{h}_{i,k} &\triangleq \boldsymbol{K}_{Ci}\dot{\boldsymbol{J}}_{i,k} \\ \boldsymbol{H}_{Ji,k} &\triangleq \boldsymbol{K}_{Ji}\boldsymbol{J}_{i,k} & \boldsymbol{h}_{Ji,k} &\triangleq \boldsymbol{K}_{Ji}\dot{\boldsymbol{J}}_{i,k}. \end{aligned}$$

3.3 Equation of Motion of a Subchain

Equation of motion of link k is described as

$$\boldsymbol{M}_k\ddot{\boldsymbol{\theta}}_k + \boldsymbol{c}_k = \sum_{m\in\mathcal{J}_k} (\boldsymbol{H}_{m,k}^T\boldsymbol{f}_m + \boldsymbol{H}_{Jm,k}^T\boldsymbol{\tau}_m). \quad (3)$$

The acceleration of the frame attached to link k side of joint i is computed by

$$\ddot{\boldsymbol{r}}_{i,k} = \boldsymbol{J}_{i,k}\ddot{\boldsymbol{\theta}}_k + \dot{\boldsymbol{J}}_{i,k}. \quad (4)$$

Using Eqs.(1) and (4), the constraint condition at joint i connecting links p_i and c_i is described as

$$\boldsymbol{H}_{i,c_i}\ddot{\boldsymbol{\theta}}_{c_i} + \boldsymbol{H}_{i,p_i}\ddot{\boldsymbol{\theta}}_{p_i} + \boldsymbol{h}_{i,c_i} + \boldsymbol{h}_{i,p_i} = \boldsymbol{O}. \quad (5)$$

Combining Eq.(5) of all joints in subchain A yields

$$\boldsymbol{H}_A\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_A = \boldsymbol{O} \quad (6)$$

where $\boldsymbol{H}_A \in \boldsymbol{R}^{N_{CA}\times 6N_{LA}}$ is a block matrix in the following form:

$$\boldsymbol{H}_A = \begin{array}{c} \\ i \end{array} \begin{pmatrix} & \vdots & & \vdots & \\ \cdots & \boldsymbol{H}_{i,p_i} & \cdots & \boldsymbol{H}_{i,c_i} & \cdots \\ & \vdots & & \vdots & \end{pmatrix} \begin{array}{c} p_i \\ \\ c_i \end{array} \quad (7)$$

and $\boldsymbol{h}_A \in \boldsymbol{R}^{N_{CA}}$ is a vector in the following form:

$$\boldsymbol{h}_A = \begin{array}{c} \\ i \end{array} \begin{pmatrix} \vdots \\ \boldsymbol{h}_{i,c_i} + \boldsymbol{h}_{i,p_i} \\ \vdots \end{pmatrix}. \quad (8)$$

Using $\boldsymbol{H}_A$, the global form of Eq.(3) is described as

$$\boldsymbol{M}_A\ddot{\boldsymbol{\theta}}_A + \boldsymbol{c}_A = \boldsymbol{H}_A^T\boldsymbol{f}_A + \boldsymbol{H}_{JA}^T\boldsymbol{\tau}_A. \quad (9)$$

Eqs.(6)(9) are solved in terms of $\boldsymbol{f}_A$ as

$$\boldsymbol{f}_A = \boldsymbol{S}_A^{-1}(-\boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{H}_{JA}^T\boldsymbol{\tau}_A + \boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{c}_A - \boldsymbol{h}_A) \quad (10)$$

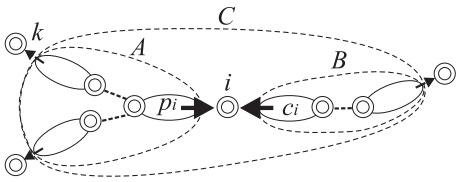

Fig.3: Assembling two subchains

where

$$\boldsymbol{S}_A \triangleq \boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{H}_A^T. \quad (11)$$

Evaluating Eq.(10) directly, as in most commercial softwares, leads to an $O(N^2)$ algorithm, while making use of the sparsity of $\boldsymbol{S}_A$ yields more efficient $O(N)$ solution[7].

Since $\ddot{\boldsymbol{\theta}}_k, \ddot{\boldsymbol{r}}_{i,k}$ and $\boldsymbol{f}_i$ change as the assembly phase proceeds, we denote the subchain when the value was computed by left-upper indices. Absence of the index means that the value is for the complete chain.

4 Details

4.1 Assembly Phase

Suppose we are going to assemble subchains A and B by connecting links p_i and c_i through joint i to build subchain C as illustrated in Fig.3.

Before adding joint i, the equations of motion and kinematic constraints of subchain X ($X = A, B$) are

$$\begin{aligned} \boldsymbol{M}_X{}^X\ddot{\boldsymbol{\theta}}_X + \boldsymbol{c}_X &= \boldsymbol{H}_X^{T\,X}\boldsymbol{f}_X + \boldsymbol{H}_{JX}^T\boldsymbol{\tau}_X & (12)\\ \boldsymbol{H}_X{}^X\ddot{\boldsymbol{\theta}}_X + \boldsymbol{h}_X &= \boldsymbol{O}. & (13) \end{aligned}$$

${}^A\ddot{\boldsymbol{r}}_{i,A}$ and ${}^B\ddot{\boldsymbol{r}}_{i,B}$ are computed by

$$\begin{aligned} {}^A\ddot{\boldsymbol{r}}_{i,A} &= \boldsymbol{J}_{i,A}{}^A\ddot{\boldsymbol{\theta}}_A + \dot{\boldsymbol{J}}_{i,A} & (14)\\ {}^B\ddot{\boldsymbol{r}}_{i,B} &= \boldsymbol{J}_{i,B}{}^B\ddot{\boldsymbol{\theta}}_B + \dot{\boldsymbol{J}}_{i,B}. & (15) \end{aligned}$$

where $\ddot{\boldsymbol{r}}_{i,A}$ and $\ddot{\boldsymbol{r}}_{i,B}$ denote the linear and angular accelerations of the frames of the subchain A and B side of joint i respectively. The Jacobian matrices $\boldsymbol{J}_{i,A} \in \boldsymbol{R}^{6\times 6N_{LA}}$ and $\boldsymbol{J}_{i,B} \in \boldsymbol{R}^{6\times 6N_{LB}}$ are written in the following forms:

$$\begin{aligned} \boldsymbol{J}_{i,A} &= (\ \boldsymbol{O}\ \cdots\ \boldsymbol{J}_{i,p_i}\ \cdots\ \boldsymbol{O}\ ) & (16)\\ \boldsymbol{J}_{i,B} &= (\ \boldsymbol{O}\ \cdots\ \boldsymbol{J}_{i,c_i}\ \cdots\ \boldsymbol{O}\ ). & (17) \end{aligned}$$

From Eqs.(12)(13), the constraint forces of subchain A are computed by

$${}^A\boldsymbol{f}_A = \boldsymbol{S}_A^{-1}(-\boldsymbol{K}_{CA}\boldsymbol{T}_A\boldsymbol{K}_{JA}^T\boldsymbol{\tau}_A + \boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{c}_A - \boldsymbol{h}_A) \quad (18)$$

where

$$\boldsymbol{S}_A \triangleq \boldsymbol{H}_A \boldsymbol{M}_A^{-1} \boldsymbol{H}_A^T \quad (19)$$

$$\boldsymbol{T}_A \triangleq \boldsymbol{J}_A \boldsymbol{M}_A^{-1} \boldsymbol{J}_A^T. \quad (20)$$

Similarly for subchain B, ${}^B\boldsymbol{f}_B$ is computed by

$${}^B\boldsymbol{f}_B = \boldsymbol{S}_B^{-1}(-\boldsymbol{K}_{CB}\boldsymbol{T}_B^T\boldsymbol{K}_{JB}^T\boldsymbol{\tau}_B + \boldsymbol{H}_B\boldsymbol{M}_B^{-1}\boldsymbol{c}_B - \boldsymbol{h}_B) \quad (21)$$

where $\boldsymbol{S}_B$ and $\boldsymbol{T}_B$ are defined in the same way.

When we connect subchains A and B through joint i, we have the equations of motion

$$\begin{aligned} \boldsymbol{M}_A {}^C\ddot{\boldsymbol{\theta}}_A + \boldsymbol{c}_A &= \boldsymbol{H}_A^{TC}\boldsymbol{f}_A + \boldsymbol{H}_{JA}^T\boldsymbol{\tau}_A \\ &\quad + \boldsymbol{H}_{i,A}^T {}^C\boldsymbol{f}_i + \boldsymbol{H}_{Ji,A}^T\boldsymbol{\tau}_i \quad (22) \\ \boldsymbol{M}_B {}^C\ddot{\boldsymbol{\theta}}_B + \boldsymbol{c}_B &= \boldsymbol{H}_B^{TC}\boldsymbol{f}_B + \boldsymbol{H}_{JB}^T\boldsymbol{\tau}_B \\ &\quad + \boldsymbol{H}_{i,B}^T {}^C\boldsymbol{f}_i + \boldsymbol{H}_{Ji,B}^T\boldsymbol{\tau}_i \quad (23) \end{aligned}$$

and constraint conditions

$$\begin{aligned} \boldsymbol{H}_A {}^C\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_A &= \boldsymbol{O} \quad (24) \\ \boldsymbol{H}_B {}^C\ddot{\boldsymbol{\theta}}_B + \boldsymbol{h}_B &= \boldsymbol{O} \quad (25) \\ \boldsymbol{H}_{i,A} {}^C\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_{i,A} + \boldsymbol{H}_{i,B} {}^C\ddot{\boldsymbol{\theta}}_B + \boldsymbol{h}_{i,B} &= \boldsymbol{O} \quad (26) \end{aligned}$$

where $\boldsymbol{H}_{i,A} \triangleq \boldsymbol{K}_{Ci}\boldsymbol{J}_{i,A}$ and $\boldsymbol{H}_{i,B} \triangleq \boldsymbol{K}_{Ci}\boldsymbol{J}_{i,B}$. Combining Eqs.(22)(23) and Eqs.(24)–(26), we obtain

$$\begin{aligned} \boldsymbol{M}_C {}^C\ddot{\boldsymbol{\theta}}_C + \boldsymbol{c}_C &= \boldsymbol{H}_C^{TC}\boldsymbol{f}_C + \boldsymbol{H}_{JC}^T\boldsymbol{\tau}_C \quad (27) \\ \boldsymbol{H}_C {}^C\ddot{\boldsymbol{\theta}}_C + \boldsymbol{h}_C &= \boldsymbol{O} \quad (28) \end{aligned}$$

where

$$\boldsymbol{M}_C \triangleq \begin{pmatrix} \boldsymbol{M}_A & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{M}_B \end{pmatrix} \quad (29)$$

$${}^C\ddot{\boldsymbol{\theta}}_C \triangleq \begin{pmatrix} {}^C\ddot{\boldsymbol{\theta}}_A \\ {}^C\ddot{\boldsymbol{\theta}}_B \end{pmatrix} \quad (30)$$

$$\boldsymbol{c}_C \triangleq \begin{pmatrix} \boldsymbol{c}_A \\ \boldsymbol{c}_B \end{pmatrix} \quad (31)$$

$${}^C\boldsymbol{f}_C \triangleq \begin{pmatrix} {}^C\boldsymbol{f}_A \\ {}^C\boldsymbol{f}_B \\ {}^C\boldsymbol{f}_i \end{pmatrix} \quad (32)$$

$$\boldsymbol{\tau}_C \triangleq \begin{pmatrix} \boldsymbol{\tau}_A \\ \boldsymbol{\tau}_B \\ \boldsymbol{\tau}_i \end{pmatrix} \quad (33)$$

$$\boldsymbol{H}_C \triangleq \begin{pmatrix} \boldsymbol{H}_A & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{H}_B \\ \boldsymbol{H}_{i,A} & \boldsymbol{H}_{i,B} \end{pmatrix} \quad (34)$$

$$\boldsymbol{h}_C \triangleq \begin{pmatrix} \boldsymbol{h}_A \\ \boldsymbol{h}_B \\ \boldsymbol{h}_{i,A} + \boldsymbol{h}_{i,B} \end{pmatrix} \quad (35)$$

Solving Eqs.(22)–(26) in terms of ${}^C\boldsymbol{f}_i$ and simplifying it using Eqs.(14)(15)(18)(21) yields

$${}^C\boldsymbol{f}_i = -\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{K}_{Ci}({}^A\ddot{\boldsymbol{r}}_{i,A} + {}^B\ddot{\boldsymbol{r}}_{i,B} + \boldsymbol{P}_{i,i}\boldsymbol{K}_{Ji}^T\boldsymbol{\tau}_i) \quad (36)$$

where

$$\begin{aligned} \boldsymbol{\Gamma}_{i,i} &= \boldsymbol{K}_{Ci}\boldsymbol{P}_{i,i}\boldsymbol{K}_{Ci}^T \\ \boldsymbol{P}_{i,i} &= \boldsymbol{\Lambda}_{Ai,i} + \boldsymbol{\Lambda}_{Bi,i} \\ \boldsymbol{\Lambda}_{Xi,i} &= \boldsymbol{J}_{i,X}\boldsymbol{\Phi}_X\boldsymbol{J}_{i,X}^T \\ \boldsymbol{\Phi}_X &= \boldsymbol{M}_X^{-1} - \boldsymbol{M}_X^{-1}\boldsymbol{H}_X^T\boldsymbol{S}_X^{-1}\boldsymbol{H}_X\boldsymbol{M}_X^{-1} \\ \boldsymbol{S}_X &= \boldsymbol{H}_X\boldsymbol{M}_X^{-1}\boldsymbol{H}_X^T. \end{aligned}$$

Since ${}^*\ddot{\boldsymbol{r}}_{i*}$ and $\boldsymbol{\Lambda}_{*i,i}(* = A, B)$ of the subchains are required to compute ${}^C\boldsymbol{f}_i$, we have to compute ${}^C\ddot{\boldsymbol{r}}_{k,C}$ $(k \in \mathcal{E}_C)$ and $\boldsymbol{\Lambda}_{Cj,j}$ for the coming assembly computations, where j is the joint going to be added next. First, using the relationship

$$\ddot{\boldsymbol{r}}_{k,C} = \boldsymbol{J}_{k,C}\ddot{\boldsymbol{\theta}}_C + \boldsymbol{j}_{k,C}, \quad (37)$$

${}^C\ddot{\boldsymbol{r}}_{k,C}$ is computed by

$${}^C\ddot{\boldsymbol{r}}_{k,C} = {}^X\ddot{\boldsymbol{r}}_{k,X} + \boldsymbol{\Lambda}_{Xk,i}(\boldsymbol{K}_{Ci}^T {}^C\boldsymbol{f}_i + \boldsymbol{K}_{Ji}^T\boldsymbol{\tau}_i) \quad (38)$$

where $\boldsymbol{\Lambda}_{Xk,i} = \boldsymbol{J}_{k,X}\boldsymbol{\Phi}_X\boldsymbol{J}_{i,X}^T$ and $X \in \{A, B\}$ is selected so that k is also included in $\mathcal{E}_X$.

Next, we compute $\boldsymbol{\Lambda}_{Cm,j}(m \in \mathcal{E}_C)$ which includes all of the $\boldsymbol{\Lambda}_C$ matrices required for the next assembly. Now we have

$$\begin{aligned} \boldsymbol{\Lambda}_{Cm,j} &= \boldsymbol{J}_{m,C}\boldsymbol{\Phi}_C\boldsymbol{J}_{j,C}^T \quad (39) \\ \boldsymbol{\Phi}_C &= \boldsymbol{M}_C^{-1} - \boldsymbol{M}_C^{-1}\boldsymbol{H}_C^T\boldsymbol{S}_C^{-1}\boldsymbol{H}_C\boldsymbol{M}_C^{-1} \quad (40) \\ \boldsymbol{S}_C &= \boldsymbol{H}_C\boldsymbol{M}_C^{-1}\boldsymbol{H}_C^T. \quad (41) \end{aligned}$$

We first yield a simplified expression of $\boldsymbol{\Phi}_C$. Using Eqs.(34)(29) and (41), $\boldsymbol{S}_C$ is written as

$$\boldsymbol{S}_C = \begin{pmatrix} \boldsymbol{S}_A & \boldsymbol{O} & \boldsymbol{S}_{Ai} \\ \boldsymbol{O} & \boldsymbol{S}_B & \boldsymbol{S}_{Bi} \\ \boldsymbol{S}_{Ai}^T & \boldsymbol{S}_{Bi}^T & \boldsymbol{S}_{ii} \end{pmatrix} \quad (42)$$

where

$$\begin{aligned} \boldsymbol{S}_{Ai} &\triangleq \boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{H}_{i,A}^T \\ \boldsymbol{S}_{Bi} &\triangleq -\boldsymbol{H}_B\boldsymbol{M}_B^{-1}\boldsymbol{H}_{i,B}^T \\ \boldsymbol{S}_{ii} &\triangleq \boldsymbol{H}_{i,A}\boldsymbol{M}_A^{-1}\boldsymbol{H}_{i,A}^T + \boldsymbol{H}_{i,B}\boldsymbol{M}_B^{-1}\boldsymbol{H}_{i,B}^T. \end{aligned}$$

which yields

$$\boldsymbol{S}_C^{-1} = \begin{pmatrix} \boldsymbol{S}_{CAA} & \boldsymbol{S}_{CAB} & \boldsymbol{S}_{CAi} \\ \boldsymbol{S}_{CAB}^T & \boldsymbol{S}_{CBB} & \boldsymbol{S}_{CBi} \\ \boldsymbol{S}_{CAi}^T & \boldsymbol{S}_{CBi}^T & \boldsymbol{S}_{Cii} \end{pmatrix} \quad (43)$$

where

$$\begin{aligned}
\boldsymbol{S}_{CAA} &= \boldsymbol{S}_A^{-1} + \boldsymbol{S}_A^{-1}\boldsymbol{S}_{Ai}\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{S}_{Ai}^T\boldsymbol{S}_A^{-1} \\
\boldsymbol{S}_{CAB} &= \boldsymbol{S}_A^{-1}\boldsymbol{S}_{Ai}\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{S}_{Bi}^T\boldsymbol{S}_B^{-1} \\
\boldsymbol{S}_{CAi} &= -\boldsymbol{S}_A^{-1}\boldsymbol{S}_{Ai}\boldsymbol{\Gamma}_{i,i}^{-1} \\
\boldsymbol{S}_{CBB} &= \boldsymbol{S}_B^{-1} + \boldsymbol{S}_B^{-1}\boldsymbol{S}_{Bi}\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{S}_{Bi}^T\boldsymbol{S}_B^{-1} \\
\boldsymbol{S}_{CBi} &= -\boldsymbol{S}_B^{-1}\boldsymbol{S}_{Bi}\boldsymbol{\Gamma}_{i,i}^{-1} \\
\boldsymbol{S}_{Cii} &= \boldsymbol{\Gamma}_{i,i}^{-1}.
\end{aligned}$$

Substituting Eq.(43) into Eq.(40) and simplifying it using $\boldsymbol{\Phi}_A$ and $\boldsymbol{\Phi}_B$, we obtain

$$\boldsymbol{\Phi}_C = \begin{pmatrix} \boldsymbol{\Phi}_{CAA} & \boldsymbol{\Phi}_{CAB} \\ \boldsymbol{\Phi}_{CAB}^T & \boldsymbol{\Phi}_{CBB} \end{pmatrix} \tag{44}$$

where

$$\boldsymbol{\Phi}_{CAA} = \boldsymbol{\Phi}_A - \boldsymbol{\Phi}_A\boldsymbol{H}_{i,A}^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{H}_{i,A}\boldsymbol{\Phi}_A \tag{45}$$

$$\boldsymbol{\Phi}_{CAB} = -\boldsymbol{\Phi}_A\boldsymbol{H}_{i,A}^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{H}_{i,B}\boldsymbol{\Phi}_B \tag{46}$$

$$\boldsymbol{\Phi}_{CBB} = \boldsymbol{\Phi}_B - \boldsymbol{\Phi}_B\boldsymbol{H}_{i,B}^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{H}_{i,B}\boldsymbol{\Phi}_B. \tag{47}$$

Computing all elements of $\boldsymbol{\Phi}_C$ requires $O(N^2)$ computations. Our final goal is, however, to evaluate $\boldsymbol{\Lambda}_{Cm,j}$ that requires only selected blocks of $\boldsymbol{\Phi}_C$ because of the sparsity of $\boldsymbol{J}_{m,C}$:

$$\boldsymbol{J}_{m,C} = \begin{pmatrix} \boldsymbol{O} & \cdots & \boldsymbol{J}_{m,p_m} & \cdots & \boldsymbol{O} \end{pmatrix} \tag{48}$$

where $p_m \in \mathcal{L}_C$ is the link in subchain C that is connected to joint m. We only need to compute the (p_m, p_j)-elements of $\boldsymbol{\Phi}_C$ since $\boldsymbol{\Lambda}_{Cm,j}$ is computed by

$$\boldsymbol{\Lambda}_{Cm,j} = \boldsymbol{J}_{m,p_m}\boldsymbol{\Phi}_{Cp_m,p_j}\boldsymbol{J}_{j,p_j}. \tag{49}$$

m and j may be in either subchain A or subchain B. In case $m \in \mathcal{E}_A$ and $j \in \mathcal{E}_A$, for example, we use the upper-left block of Eq.(44):

$$\begin{aligned}
\boldsymbol{\Lambda}_{Cm,j} &= \boldsymbol{J}_{m,p_m}(\boldsymbol{\Phi}_A - \boldsymbol{\Phi}_A\boldsymbol{H}_{i,A}^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{H}_{i,A}\boldsymbol{\Phi}_A)\boldsymbol{J}_{j,p_j} \\
&= \boldsymbol{J}_{m,p_m}\boldsymbol{\Phi}_A\boldsymbol{J}_{j,p_j} \\
&\quad -\boldsymbol{J}_{m,p_m}\boldsymbol{\Phi}_A\boldsymbol{J}_{i,A}^T\boldsymbol{K}_i^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{K}_i\boldsymbol{J}_{i,A}\boldsymbol{\Phi}_A\boldsymbol{J}_{j,p_j} \\
&= \boldsymbol{\Lambda}_{Am,j} - \boldsymbol{\Lambda}_{Am,i}\boldsymbol{K}_i^T\boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{K}_i\boldsymbol{\Lambda}_{Ai,j}
\end{aligned} \tag{50}$$

which only uses the quantities computed in the assembly process for constructing subchain A. Other cases are also handled in similar ways.

Computations for assembling subchain C through joint i consist of folowing four steps:

1. compute $\boldsymbol{P}_{i,i}$ and $\boldsymbol{\Gamma}_{i,i}^{-1}$,
2. compute ${}^C\boldsymbol{f}_i$,
3. compute $\boldsymbol{\Lambda}_{Cm,n}$ $(m, n \in \mathcal{E}_C)$, and
4. compute ${}^C\ddot{\boldsymbol{r}}_{m,C}$ $(m \in \mathcal{E}_C)$.

4.2 Disassembly Phase

The constraint forces computed in the assembly phase are valid only in the corresponding subchains. After the completion of the assembly phase, the values might have changed due to the effects of joints added afterward. The disassembly phase computes the constraint forces in the completed chain by disassembling the subchains in the reverse order of the assembly phase. When a joint is removed, its final constraint force is computed, which in turn can be regarded as an external force for the two subchains which the joint had connected.

Suppose we are about to remove joint i. Joints in $\mathcal{E}_C$ were assembled *after* joint i; therefore, they are removed *before* joint i in the disassembly phase and we already know the final constraint forces of joint $k \in \mathcal{E}_C$, denoted by $\boldsymbol{f}_k$.

Regarding $\boldsymbol{f}_k$ $(k \in \mathcal{E}_C)$ as external forces, we have the new equations of motions for subchains A and B:

$$\begin{aligned}
\boldsymbol{M}_A\ddot{\boldsymbol{\theta}}_A + \boldsymbol{c}_A &= \boldsymbol{H}_{i,A}^T\boldsymbol{f}_i + \boldsymbol{H}_A^T\boldsymbol{f}_A \\
&\quad +\boldsymbol{H}_{Ji,A}^T\boldsymbol{\tau}_i + \boldsymbol{H}_{JA}^T\boldsymbol{\tau}_A \\
&\quad + \sum_{k\in\mathcal{E}_A}(\boldsymbol{H}_{k,A}^T\boldsymbol{f}_k + \boldsymbol{H}_{Jk,A}^T\boldsymbol{\tau}_k) \\
\boldsymbol{M}_B\ddot{\boldsymbol{\theta}}_B + \boldsymbol{c}_B &= \boldsymbol{H}_{i,B}^T\boldsymbol{f}_i + \boldsymbol{H}_B^T\boldsymbol{f}_B \\
&\quad +\boldsymbol{H}_{Ji,B}^T\boldsymbol{\tau}_i + \boldsymbol{H}_{JB}^T\boldsymbol{\tau}_B \\
&\quad + \sum_{k\in\mathcal{E}_B}(\boldsymbol{H}_{k,B}^T\boldsymbol{f}_k + \boldsymbol{H}_{Jk,B}^T\boldsymbol{\tau}_k)
\end{aligned}$$

and the equations of constraints:

$$\begin{aligned}
\boldsymbol{H}_A\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_A &= \boldsymbol{O} \\
\boldsymbol{H}_B\ddot{\boldsymbol{\theta}}_B + \boldsymbol{h}_B &= \boldsymbol{O} \\
\boldsymbol{H}_{i,A}\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_{i,A} &= \boldsymbol{H}_{i,B}\ddot{\boldsymbol{\theta}}_B + \boldsymbol{h}_{i,B}
\end{aligned}$$

where the unknowns are $\boldsymbol{f}_i, \boldsymbol{f}_A, \boldsymbol{f}_B, \ddot{\boldsymbol{\theta}}_A$ and $\ddot{\boldsymbol{\theta}}_B$. Solving these equations in terms of $\boldsymbol{f}_i$, the final constraint force is computed by

$$\boldsymbol{f}_i = {}^C\boldsymbol{f}_i - \boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{K}_{Ci}\sum\boldsymbol{\Lambda}_{Xk,i}^T(\boldsymbol{K}_{Ck}^T\boldsymbol{f}_k + \boldsymbol{K}_{Jk}^T\boldsymbol{\tau}_k). \tag{51}$$

Once the constraint force is computed, the accelerations of the both sides of the joint are computed by

$$\begin{aligned}
\ddot{\boldsymbol{r}}_{i,X} &= {}^X\ddot{\boldsymbol{r}}_{i,X} + \boldsymbol{\Lambda}_{Xi,i}(\boldsymbol{K}_{Ci}^T\boldsymbol{f}_i + \boldsymbol{K}_{Ji}^T\boldsymbol{\tau}_i) \\
&\quad + \sum\boldsymbol{\Lambda}_{Xk,i}^T(\boldsymbol{K}_{Ck}^T\boldsymbol{f}_k + \boldsymbol{K}_{Jk}^T\boldsymbol{\tau}_k)
\end{aligned} \tag{52}$$

where $X = A, B$. Finally, the joint acceleration is computed by

$$\ddot{\boldsymbol{q}}_i = \boldsymbol{K}_{Ji}(\ddot{\boldsymbol{r}}_{i,B} - \ddot{\boldsymbol{r}}_{i,A}). \tag{53}$$

All the quantities except for $\boldsymbol{f}_k$ $(k \in \mathcal{E}_C)$ used to compute $\ddot{\boldsymbol{q}}_i$ are already computed in the assembly phase.

4.3 Closed Kinematic Chains

We have two options to assemble a closed kinematic chain in our algorithm: (1) add a joint to connect two links in the same subchain, or (2) add multiple joints simultaneously to connect two different subchains. Since the latter approach is a straightforward extension of the method described above, we present the equations for the former approach.

Suppose we are going to add joint i to connect links m and n, both in subchain A, and construct subchain C. The equations of motion and constraint of subchain A are exactly the same as Eqs.(12)(13). After adding joint i, we have the following equations:

$$\begin{aligned} \boldsymbol{M}_A{}^C\ddot{\boldsymbol{\theta}}_A + \boldsymbol{c}_A &= \boldsymbol{H}_{i,A}^T{}^C\boldsymbol{f}_i + \boldsymbol{H}_{Ji,A}^T\boldsymbol{\tau}_i \\ &\quad + \boldsymbol{H}_A^{TC}\boldsymbol{f}_A + \boldsymbol{H}_{JA}^T\boldsymbol{\tau}_A \quad (54) \\ \boldsymbol{H}_{i,A}{}^C\ddot{\boldsymbol{\theta}}_A + \boldsymbol{h}_{i,A} &= \boldsymbol{O} \quad (55) \end{aligned}$$

where $\boldsymbol{H}_{i,A} \triangleq \boldsymbol{K}_{Ci}\boldsymbol{J}_{i,A}$ and $\boldsymbol{J}_{i,A}$ has the following form:

$$\boldsymbol{J}_{i,A} = \begin{pmatrix} \cdots & \boldsymbol{J}_{i,m} & \cdots & \boldsymbol{J}_{i,n} & \cdots \end{pmatrix}. \quad (56)$$

Solving Eqs.(54)(55) yields a result similar to Eq.(36):

$${}^C\boldsymbol{f}_i = \boldsymbol{\Gamma}_{i,i}^{-1}\boldsymbol{K}_{Ci}({}^A\ddot{\boldsymbol{r}}_{i,n} + {}^A\ddot{\boldsymbol{r}}_{i,m} - \boldsymbol{P}_{i,i}\boldsymbol{K}_{Ji}^T\boldsymbol{\tau}_i) \quad (57)$$

where

$$\begin{aligned} \boldsymbol{\Gamma}_{i,i} &= \boldsymbol{K}_{Ci}\boldsymbol{P}_{i,i}\boldsymbol{K}_{Ci}^T & (58) \\ \boldsymbol{P}_{i,i} &= \boldsymbol{\Lambda}_{Ai,i} & (59) \\ \boldsymbol{\Lambda}_{Ai,i} &= \boldsymbol{J}_{i,A}\boldsymbol{\Phi}_A\boldsymbol{J}_{i,A}^T & (60) \\ \boldsymbol{\Phi}_A &= \boldsymbol{M}_A^{-1} - \boldsymbol{M}_A^{-1}\boldsymbol{H}_A^T\boldsymbol{S}_A^{-1}\boldsymbol{H}_A\boldsymbol{M}_A^{-1} & (61) \\ \boldsymbol{S}_A &= \boldsymbol{H}_A\boldsymbol{M}_A^{-1}\boldsymbol{H}_A^T. & (62) \end{aligned}$$

In closed kinematic chains, the invertibility of $\boldsymbol{\Gamma}_{i,i}$ is not guranteed. A singular $\boldsymbol{\Gamma}_{i,i}$ indicates indeterminate constraint forces or inconsistent constraints.

4.4 Complexity

As is obvious from the algorithm described in this section, the complexity of the algorithm depends on the number of elements in $\mathcal{E}$ of each subchain. This relationship is similar to that of the number of handles in DCA and its complexity[10]. In [10], detailed discussion is made on reducing the number of handles and balancing the assembly tree. The conclusions of the discussion are summarized by the following points:

- The branching factor of the kinematic tree is limited up to three by link splitting.
- Therefore, it is possible to limit the number of handles to three.
- Although there exist cases where the number of handles grow infinitely when we try to obtain a balanced tree, we have several options to maintain $O(\log N)$ time complexity.

From these points, we can conclude that our algorithm also maintains $O(N)$ and $O(\log N)$ asymtotic complexity for serial and parallel computation respectively for most practical kinematic chains.

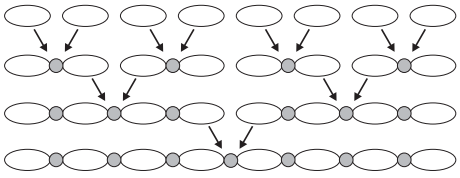

Fig.4: A schedule for an 8-link serial chain

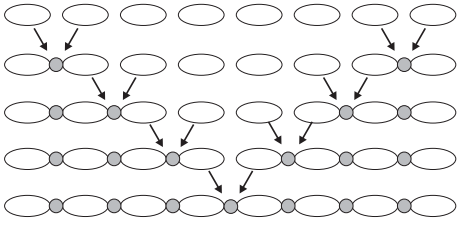

Fig.5: Another possible schedule

5 Parallel Computation

5.1 Scheduling

The method described in the previous section assumes nothing about the order of adding and removing joints in assembly and disassemly phases. In fact, the parallelism and total computational cost are determined by the scheduling. If we try to increase the parallelism, the total computational cost grows up and vise versa; therefore, it is less efficient to apply a schedule intended for larger number of processes than available. One of the advantages of our approach is that we can customize the parallelism and the total computational cost only by changing the schedule.

Figures 4 and 5 show two possible schedules for assembling an 8-link serial chain. It is obvious that the former one has higher parallelism, since it allows four processes to run in parallel at the first step and requires only three steps in total. The latter one, on the other hand, allows only two parallel processes and requires four steps in total. Therefore, if we have more than four processors, it is better to apply the former

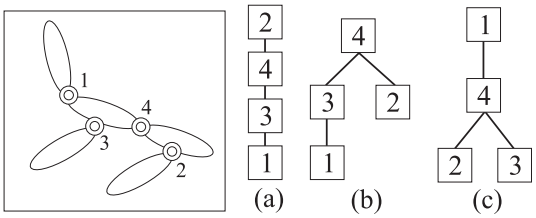

Fig.6: Different schedules for a kinematic chain

schedule. Its total computational cost, however, is worse than the latter. Therefore, if we don't have that many processors, it is better to use the latter order.

The issue here is how to determine the schedule of the assembly and disassembly phases that makes the best use of the available processors. We show an intuitive strategy to obtain the optimal scheduling for a given kinematic chain and the number of processors. It is difficult to give an algorithmic strategy for general cases and would be included in future works.

A schedule can be expressed by a binary tree like (a)–(c) in Fig.6, which represent different schedules for the kinematic chain in the left-hand side of Fig.6. Each node represents a subchain and labeled by the number of the last joint added to assemble the subchain. Two edges starting from a node point the two subchains connected by the joint. Null-pointing edges mean that the corresponding subchain consists of a single link.

A binary tree gives an intuitive idea of the parallelism and efficiency of the schedule: the number of leaf nodes indicates the parallelism, and the depth is nearly proportional to the computation time when there exists more processors than the number of the leaf nodes. Two strategies are derived from these facts:

1. the number of leaf nodes should be close to the number of the processors, and
2. the depth of leaf nodes should be even.

5.2 Communication

Suppose subchains A and C in Fig.3 are processed in different processes p_a and p_c. The following values should be passed between the two processes:

1. In assembly phase, send the following data from p_a to p_c:
   (a) $\mathbf{\Lambda}_{Am,i} (m \in \mathcal{E}_A)$
   (b) ${}^A\ddot{\boldsymbol{r}}_{m,A} (m \in \mathcal{E}_A)$
   (c) required blocks of $\boldsymbol{S}_A^{-1}$.
2. In disassembly phase, send $\boldsymbol{f}_i$ from p_c to p_a.

Table 1: Computation time for serial chains [ms]

links	8	16	32
1 procs	1.31	2.75	6.08
2 procs	0.984	1.87	3.93
4 procs	0.897	1.70	3.39
8 procs	—	1.57	2.90
4 procs*	—	1.58	3.16

Table 2: Computation time for human figures [ms]

DOF	34	48
1 procs	3.66	4.85
2 procs	2.49	2.93
4 procs	2.22	2.49

6 Simulation Examples

The presented algorithm was implemented on a cluster of 8 workstations with PentiumIII 1GHz processor. The nodes are connected by Myrinet and have parallel computation environment SCore[13] installed.

6.1 Computation Time

Table 1 shows the computation time for serial kinematic chains of 8 to 32 links connected by 3DOF spherical joints. The links are distributed to all processes evenly, except for the last case (marked * in the table), where we used 4 processes but the numbers of links assigned to them are not even. When a serial chain is divided into 4 subchains processed by 4 processes, the middle two subchains have two joints in $\mathcal{E}$, while those at the end have only one. Therefore, the computational costs of the processes handling the middle two subchains are slightly larger than the other two even if the numbers of the assigned links are the same. In the last case we moved one link each from the middle subchains to the other two and tried to make the computational costs, rather than the number of links, of the processes even. As a result, the computation time was reduced greatly from the case of 4 processes each with the same number of links.

Table 2 shows the computation time for free-flying human figures of 34 and 48DOF on 1 to 4 processes.

6.2 Dynamics Simulation of Human Figures

Figure 7 shows snapshots from dynamics simulation of a human figure including structure changes. The figure initially held the environment by the both hands, and then released the right and left hands. The computation time in serial computation varies from 3.4 to 5.0[ms] depending on the number of connections between the hands and the environment.

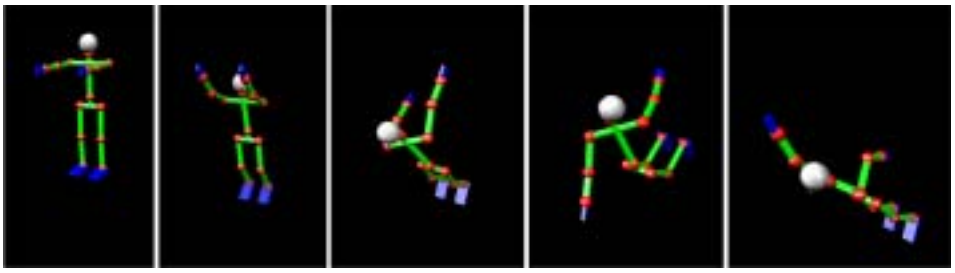

Fig.7: Dynamics simulation of structure-varying kinematic chain

7 Conclusion

We conclude this paper by emphasizing the following three points:

- An efficient parallel algorithm for forward dynamics of human figures was developed and implemented. Its asymptotic complexity is $O(N)$ for serial computation and $O(\log N)$ for parallel computation on $O(N)$ processors for most practical kinematic chains.
- The parallelism and total computational cost are customized for parallel computation on any number of processors only by optimizing the scheduling of assemly and disassembly phases. A novelty of this algorithm is in the fact that there are no algorithmic differences in serial and parallel computation. This enables us to use the same program on any systems from a single PC system to a system of large PC cluster.
- Simulation results showed that the computation time is effectively reduced by parallel computation. Forward dynamics computation of 48DOF human figure takes less than 2.5ms on four processors.

The first author acknowledges the support by the Japan Society for the Promotion of Science. This research was supported by CREST program, the Japan Science and Technology Corporation.

References

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

[2] Y. Nakamura, H. Hirukawa, and K. Yamane et al.: "Humanoid Robot Simulator for the METI HRP Project," *Robotics and Autonomous Systems*, vol.37, pp.101–114, 2001.

[3] R. Featherstone. *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Boston, MA, 1987.

[4] D.S. Bae and E.J. Haug: "A Recursive Formulation for Constrained Mechanical System Dynamics: PartI. Open Loop Systems," *Mechanics of Structures and Machines*, vol.15, no.3, pp.359–382, 1987.

[5] D.S. Bae and E.J. Haug: "A Recursive Formulation for Constrained Mechanical System Dynamics: PartII. Closed Loop Systems," *Mechanics of Structures and Machines*, vol.15, no.4, pp.481–506, 1987-88.

[6] D.E. Rosenthal: "An Order n Formulation for Robotic Systems," *The Journal of the Astronautical Sciences*, vol.38, no.4, pp.511–529, 1990.

[7] David Baraff: "Linear-Time Dynamics Using Lagrange Multipliers," In *Proceedings of SIGGRAPH*, pp.137–146, 1996.

[8] A. Fijany, I. Sharf, and G.M.T. D'Eleuterio: "Parallel $O(\log N)$ Algorithms for Computation of Manipulator Forward Dynamics," *IEEE Transactions on Robotics and Automation*, vol.11, no.3, pp.389–400, 1995.

[9] R. Featherstone: "A Divide-and-Conquer Articulated-Body Algorithm for Parallel $O(\log(n))$ Calculation of Rigid-Body Dynamics. Part1: Basic Algorithm," *International Journal of Robotics Research*, vol.18, no.9, pp.867–875, 1999.

[10] R. Featherstone: "A Divide-and-Conquer Articulated-Body Algorithm for Parallel $O(\log(n))$ Calculation of Rigid-Body Dynamics. Part2: Trees, Loops, and Accuracy," *International Journal of Robotics Research*, vol.18, no.9, pp.876–892, 1999.

[11] K.S. Anderson and S. Duan: "Highly Parallelizable Low-Order Dynamics Simulation Algorithm for Multi-Rigid-Body Systems," *AIAA Journal on Guidance, Control and Dynamics*, vol.23, no.2, pp.355–364, 2000.

[12] F. Faure: "Fast Iterative Refinement of Articulated Solid Dynamics," *IEEE Transactions on Visualization and Computer Graphics*, vol.5, no.3, pp.268–276, 1999.

[13] Parallel and Distributed System Software Laboratory. *SCore*. http://pdswww.rwcp.or.jp/home.html, 2001.