UDC 621.397.3:681.32

Development of Production Environment for Motion Contents for Humans, Animals, and Other Structures

●Fumio Nagashima

(Manuscript received June 17, 1999)

This paper proposes an efficient method for producing contents which describe the motions of complex structures, particularly humans and animals. Previously, the simulation of human and animal animation was a time-consuming task that had to be done by highly skilled people. To make this process faster and easier, the authors propose to use several motion dynamics techniques to generate smooth motions and mechanisms for integrating these techniques. This paper describes a system developed by the author that is based on this proposal and describes various key technologies developed by the author's recent research activities; namely, a generalized motion dynamics algorithm, an efficient collision analysis algorithm, and a high-speed component software technique.

1. Introduction

Commercial CG modeling software that supports the creation of advanced static CG contents fairly well is already available. It is, however, still difficult to create advanced dynamic CG contents. Specifically, it is very difficult to create the dynamic motions of humans and animals because these motions must be extremely smooth and follow the laws of motion.

It is expected that difficulties in creating dynamic CG contents will be solved through a dynamic simulation of natural motion based on quick-response kinematic/dynamic algorithms in which kinematic/dynamic concepts are applied to the static CG world and dynamic modeling. It is also expected that elaborate combinations of dynamic simulations with data obtained from actual human motions (derived from the latest measuring instruments, for example, a motion capture unit) will be able to create more natural motions.

2. Purpose

We decided to develop a PC-based production environment that makes it easier for users to produce contents that include the natural motions of humans and animals. Specifically, the project aims at developing software tools, including software packages, that enable the creation of natural motions by merging dynamic simulations based on quick response kinematic/dynamic algorithms with measurement data of actual motions. The production of motion content requires several kinds of technologies, a typical example of which is the combination of motion capture data and dynamics simulation. Because there is often a requirement for interactive 3-D motion contents, our production environment makes it possible to create motion contents in the form of an executable computer program. This paper describes the software framework, motion integration functions, general purpose dynamics functions, and collision analysis functions of the production environment.

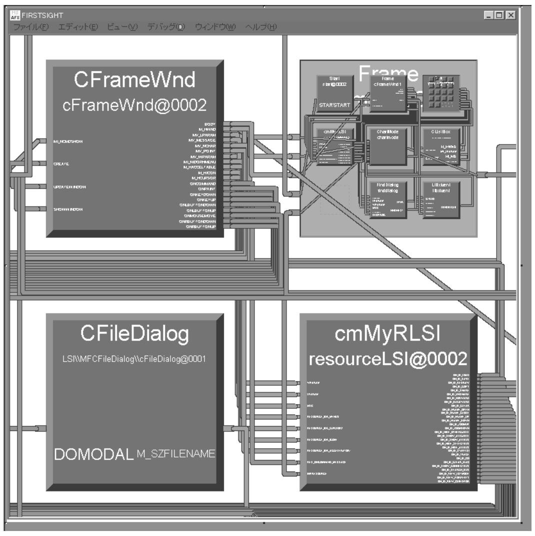

Figure 1
Software LSIs.

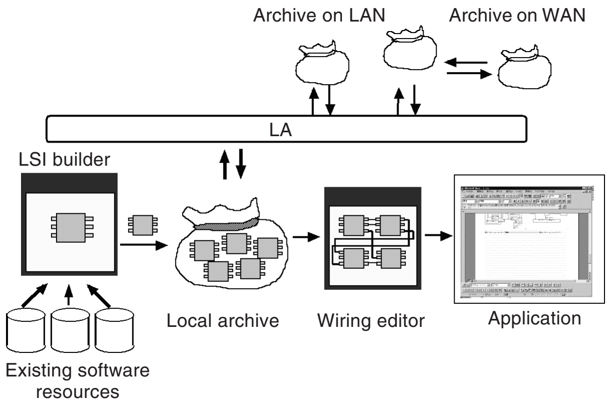

Figure 2
Concept of Firstsight.

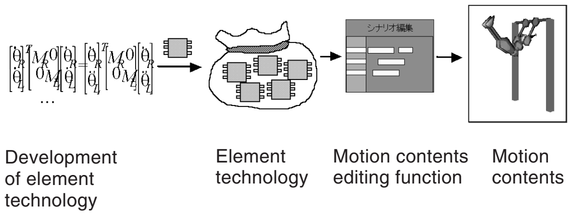
Development of element technology
Element technology
Motion contents editing function
Motion contents

Figure 3
Concept of developed software.

3. Framework

3.1 Firstsight[1)]

The production environment is based on Firstsight, which is a component software tool developed by Fujitsu. The basic concept of Firstsight is the software LSI, which is a new type of componentware based on the concepts of object orientation and data flow programming. Software LSIs bear many similarities to hardware LSIs, but they represent a totally new approach to constructing real-time applications.

There are two steps to creating an application using Firstsight. One is to develop an algorithm using the process description language C/C++ and wrapping that algorithm into a software LSI using a software LSI builder. The other is to build an application using software LSIs using a wiring editor. **Figure 1** shows a screen shot of software LSIs on a wiring editor, and **Figure 2** shows the concept of Firstsight. The developed software has the concept shown in **Figure 3**. A motion specialist creates software LSIs, and the user uses them to create motion contents.

3.2 Function framework

Because the production environment is based on Firstsight, it provides the user with software LSIs and LSI circuits. The functions of the production environment allow the user to construct motion contents very flexibly because Firstsight can combine functions by using software LSIs. Formally, the user could not combine functions in such a system. The outcome of the project will consist of six functions that are related to one another as shown in **Figure 4**. These functions are described below.

1) Motion integration functions

These functions will read mechanism data from a mechanism database as directed by the user. Also, they will create and update status data and scenario information related to the specified mechanism data and create an executable data file. The mechanism data describes structures of humans, animals, and other linked systems. The status data describes physical properties at a particular moment. These functions will also read the executable data file, update status data for each consecutive frame, and pass the updated status data to the motion display functions.

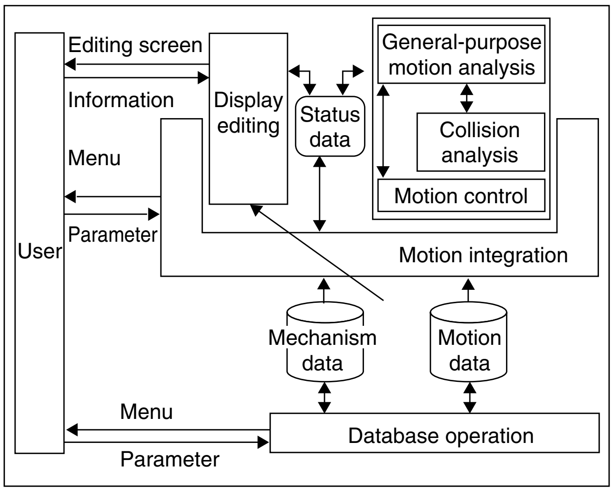

Figure 4
Relations between functions.

2) General-purpose motion analysis

These functions will read status data created with the motion contents editing functions described in Chapter 4 and calculate what the status data of each open or closed link (described in Chapter 5) will be after the period ΔT. This status will depend on the information in the status data that identifies open and closed links using the algorithms described in Chapters 4 and 5.

The general-purpose motion analysis functions will call the collision analysis functions and the motion control functions internally, and instuct them to calculate status data while taking into account the collision force and joint torques based on joint angle-oriented motion data.

3) Collision analysis

These functions will read status data updated by the general-purpose motion analysis functions to check for collisions between objects and calculate the collision forces according to the algorithm described in Chapter 6.

4) Motion control

These funcions will read status data updated by the general-purpose motion analysis functions, and convert the angular velocity-oriented motion data in the status data to joint torque-oriented motion data according to the open/closed link identification information in the status data using the algorithms described in Chapter 5. By making these conversions, these functions will update the status data so that it can be used by the general-purpose motion analysis functions.

5) Display editing

These funtions will read status data updated by the motion integration functions, draw and display three-dimensional CG images, and update or modify the attributes (including the location and color) of objects in a three-dimensional CG space.

6) Database operation

These functions will enable new data to be saved to the mechanism and motion databases.

3.3 Software framework

The developed software is based on Firstsight and consists of a motion editing program, motion execution program, and database program. The elementary software is built by LSI wrapper. The motion editing program generates an LSI circuit. The motion execution program is an LSI circuit. **Table 1** shows the framework of the developed software.

1) Motion contents editing program

Figure 5 shows the block diagram of the motion contents editing program. The program consists of an event dispatcher LSI class, motion data interpolation class, motion data retrieval LSI class, executable data generation LSI class, and integration data handling LSI class.

2) Motion contents execution program

Figure 6 shows the block diagram of a typical motion contents execution program. The program consists of a content execution LSI class, general purpose motion analysis LSI circuit, execution control LSI class, motion control LSI circuit, and collision analysis LSI circuit.

3) Database program

Figure 7 shows the block diagram of the database program. The program consists of sections for mechanism data interpolation, motion data combination, intermediate motion, and measurement data conversion.

Table 1
Software framework.

Program name		Purpose
Motion contents editing program		Generating motion contents for humans and animals
	Event control LSI class	Controlling the display and window system
	Motion data retrieval LSI class	Retrieving and expanding motion data for scenario
	Motion interpolation LSI class	Interpolating motion
	Executable generation LSI class	Generating an executable data file
	Integration data handling LSI class	Handling of integration file which contains entire information of contents
Motion contents execution program		Executing the executable file generated by motion contents editing program
	Executing LSI class	Executing LSI circuit
	Executable data retrieval LSI class	Retrieving and expanding the executable data to an LSI circuit
	Execution control LSI class	Controlling display and windows during execution
	Motion control LSI circuit	Controlling the motion
	General purpose dynamics LSI class	Calculating forward dynamics
	Collision analysis LSI class	Detecting collisions and calculating collision forces
Database program		Handling of motion contents data
	Mechanical data processing program	Combining two sets of mechanical data
	Motion data processing program	Generating motion data from other structure motion data
	Measured data processing program	Handling measured data

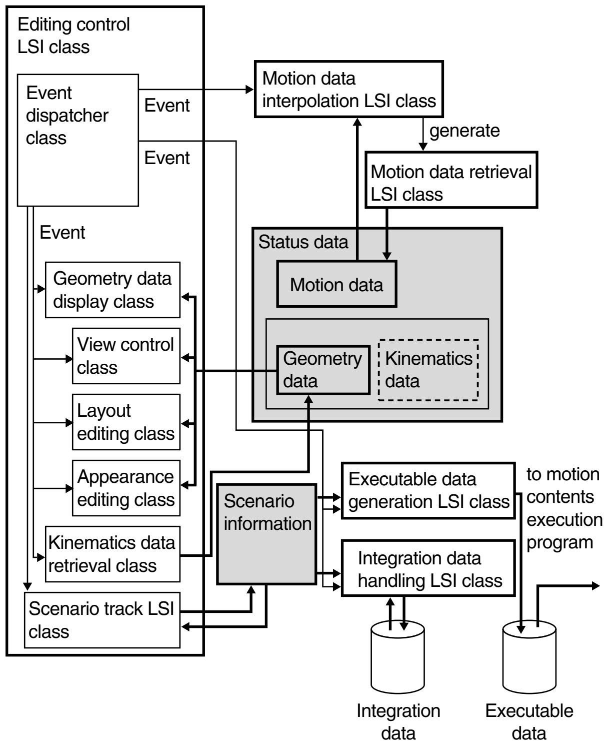
Editing control LSI class
Event dispatcher class
Event
Event
Event
Geometry data display class
View control class
Layout editing class
Appearance editing class
Kinematics data retrieval class
Scenario track LSI class
Motion data interpolation LSI class
generate
Motion data retrieval LSI class
Status data
Motion data
Geometry data
Kinematics data
Scenario information
Executable data generation LSI class
Integration data handling LSI class
to motion contents execution program
Integration data
Executable data

Figure 5
Motion contents editing program.

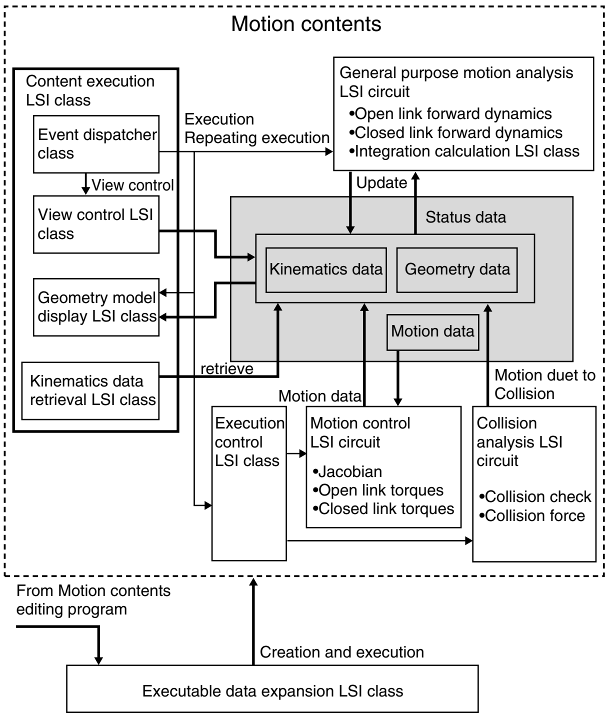
Motion contents
Content execution LSI class
Event dispatcher class
View control
View control LSI class
Geometry model display LSI class
Kinematics data retrieval LSI class
Execution
Repeating execution
General purpose motion analysis LSI circuit
•Open link forward dynamics
•Closed link forward dynamics
•Integration calculation LSI class
Update
Status data
Kinematics data
Geometry data
Motion data
retrieve
Motion data
Motion duet to Collision
Execution control LSI class
Motion control LSI circuit
•Jacobian
•Open link torques
•Closed link torques
Collision analysis LSI circuit
•Collision check
•Collision force
From Motion contents editing program
Creation and execution
Executable data expansion LSI class

Figure 6
Motion contents execution program.

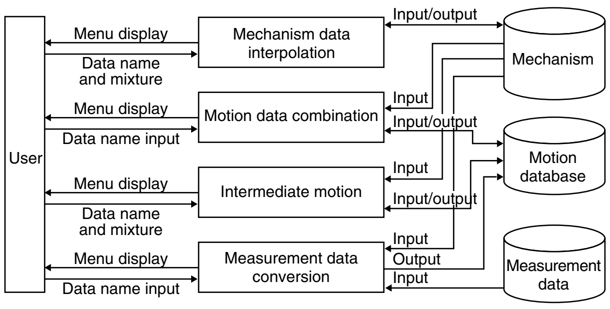

Figure 7
Database program.

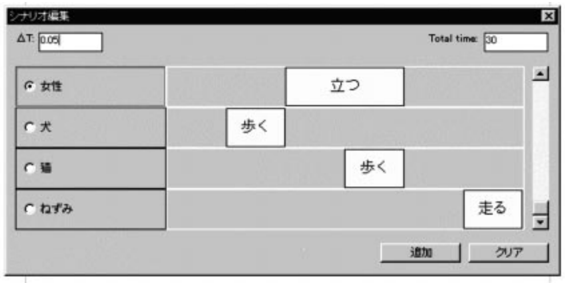

Figure 8
Screen of scenario editing function.

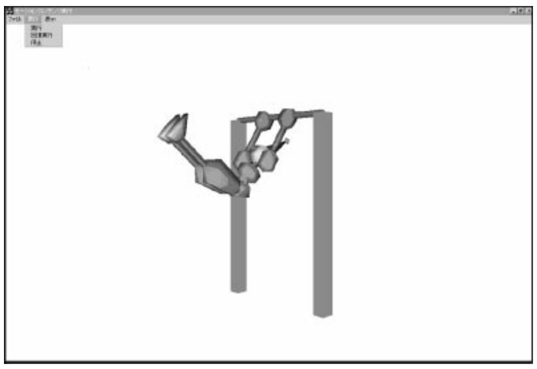

Figure 9
Screen of motion contents execution function.

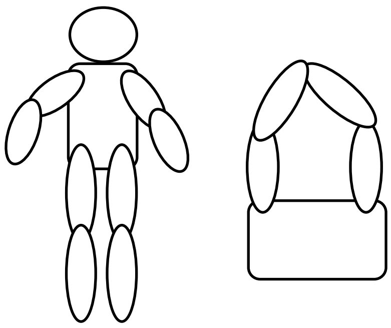

Figure 10
Open and closed links.

4. Motion integration functions

The motion integration functions create motion contents in the form of software LSIs. These functions are as follows.

1) Motion contents editing functions

These functions will read mechanism and motion data from the mechanism and motion databases to create an executable data file for the motion contents execution function.

2) Scenario editing function

This function will create scenario information to generate an executable data file. The user can assign a start and end time for each character motion. A scenario consists of a group of motions specified for the characters being animated. The motion specified for an individual character is called an editing track. All operations are made using a mouse. **Figure 8** shows an example screen of this function.

3) Motion contents execution function

The execution of motion contents is done using a Firstsight LSI circuit. The Firstsight interpreter reads an executable data file and executes it. Firstsight uses a general purpose dynamics function, collision analysis function, motion control function, and display function that are realized with one or more LSIs. **Figure 9** shows an example screen of the motion contents execution function.

5. General purpose dynamics function

This function determines the accelerations of joints from their torques. The positions and attitudes of links are calculated by the motion integration function. The general purpose dynamics function can generate a natural motion without a full description of the motion. Formally, the motion contents creator had to give a full description of a motion being generated.

There are two types of link system: an open link system and a closed link system (see **Figure 10**). An open link system does not have a

Table 2
Link structure pointers.

Pointer name	Function
Parent pointer	Represents the highest-hierarchy link. This link is nearest to the base link.
Child pointer	Represents the lowest-hierarchy link. This link is farthest from the base link.
Brother pointer	Represents the brother link connection.
Virtual pointer	Distinguishes a virtual link from a real link.

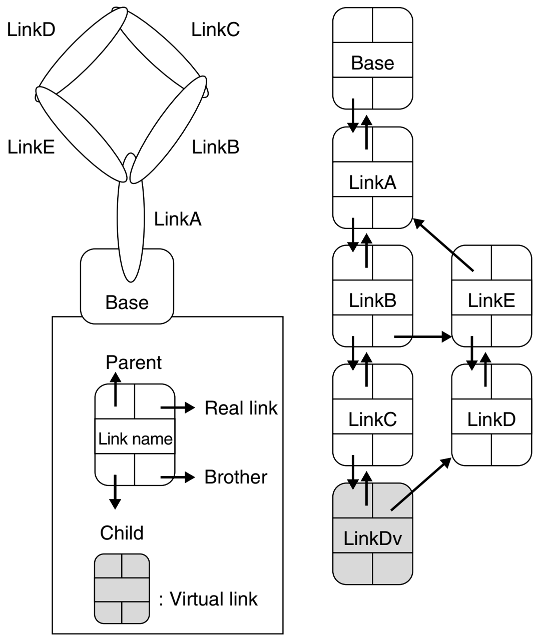

Figure 11
Example link system.

loop of links. Many industrial robots belong to this group. A closed link system has one or more loop links. Some example of members of this group are the playground swing and certain parallel-mechanism industrial robots. In motion contents, there are many closed link systems. For example, when a human grabs something by both hands, the link system containing the hands is a closed link system. The treatment of closed link systems is therefore important when creating a motion content.

Open link system dynamics have been studied in detail[2),3)] and many simulation systems can calculate their behavior. Closed link systems however have not been studied well and many simulation systems cannot calculate their behavior efficiently. Our algorithm for closed link systems, which is highly efficient, is described below.

1) Expression of link system

A link is defined using four pointers. **Table 2** shows the functions of the pointers. **Figure 11** shows an example system of links. The virtual link in Figure 11 is used for closed links.

2) Forward dynamics

The equation for the motion of a link system is as follows:

$$\tau = A(q)\ddot{q} + b(q, \dot{q}), \tag{1}$$

where τ is the torque in the joint, q is the general coordinates of the link system, $A(\cdot)$ is the inertia matrix of the system, and $b(\cdot)$ is a nonlinear term. To determine $\ddot{q}$ using τ requires forward dynamics. However, because the inverse dynamics are generally easier to solve than the forward dynamics, the forward dynamics are calculated by the inverse solution method using inverse dynamics. Then, we determine $\ddot{q}$ using τ by inversely solving the inverse dynamics. The forward dynamics are therefore solved as follows:

a) Obtain $b(\cdot)$ by setting the acceleration of all joints to zero.

b) Obtain each column of $A(\cdot)$ by setting the acceleration of all joints to unity.

c) Obtain $\ddot{q}$ using the following equation:

$$\ddot{q} = A^{-1}(q)\left(\tau - b(q, \dot{q})\right). \tag{2}$$

Because this method can be applied to open links and to closed links, it is not necessary to distinguish between an open link and closed link system.

3) Inverse dynamics of closed link system

Because the inverse dynamics of an open link system have been established in previous papers,[4)] we will show a way to solve the inverse dynamics of a closed link system. Nakamura's method is known as an efficient method for solving the behavior of a closed link system.[5),6)] This method proceeds as follows:

a) Determine the general coordinate system θ_G to express the state of the closed link system.

b) Cut some joints to transform the system to an open link system, and name the cut joints θ_{Open}.

c) Calculate the Jacobian W between θ_G and θ_{Open}.

d) Calculate the Jacobian S between θ_G and θ_{Act}, where θ_{Act} is the real joint value in the closed system.

e) Obtain general force τ_G using the following equation:

$$\tau_G = W^T \tau_{Open}. \tag{3}$$

f) Calculate driving force τ_{Act} using the following equation:

$$\tau_G = S^T \tau_{Act}. \tag{4}$$

The method for determining the Jacobian W and S is not known except for a simple plane figure structure and certain other structures. However, natural structures, including humans and animals, are not simple. We therefore need a general method to determine the Jacobian W and S. The method used in our software to determine the Jacobian W and S is described below.

4) Calculation of W and S

Our method for calculating W and S is as follows:

a) Calculate the matrix which represents the constraint condition of the closed link system. This matrix is determined by the equal condition of virtual link velocities and real ones.

b) Select the independent row from the above matrix, and calculate the degrees of freedom.

c) Select the joints which are numerically equal to the degrees of freedom, name them using general coordinate system θ_G, and name the other joints using coordinate system θ_S.

d) Decompose the matrix calculated in a) to the corresponding general coordinate system θ_G and name it J_G, and name the other matrix J_S.

e) The Jacobian between all joints angles and the general coordinate system is as follows:

$$H = \left(\frac{E}{-J_S^{-1} J_G} \right). \tag{5}$$

f) Calculate W by taking the virtual cutting joints assuming that the joints between virtual links are cut.

g) Calculate S by selecting the part of H which corresponds to the driving joints.

Because this method does not need special information about the degrees of freedom or the characteristics of structures, it is truly a general method.

We can obtain natural motion using the forward dynamics of open link and closed link systems. The open link forward dynamics are well known, and the closed link forward dynamics have been described above.

6. Collision analysis

Motion is changed by the forces of collisions. The collision analysis function watches for collisions and monitors the distances between the links. A collision is assumed to have occurred when the distance between two links is smaller than an infinitesimal amount. When a collision occurs, this function calculates the force of the collision and the change of motion of links for the motion analysis function.

The system uses two methods of collision detection and a mechanism for switching between these two methods. The two methods are the successive Gilbert method and the bubble collision method. The collision force and the changes of the links' motions are calculated using the equation of momentum and energy. Formally, collision detection was very slow or provided only partial results. Because of the switching mechanism, however, the algorithm in this system is fast and fully functional.

1) Preprocessing for collision detection

a) Processing for creating a convex hull[7),8)]

First, it is necessary to create a convex hull for each link because the successive Gilbert method can handle only convex objects. The vertex point inside the link geometry is ignored in this process.

b) Creating the proximity list

This process creates the proximity list of the vertex by assuming that the link geometry is expressed by a triangular patch. It creates the connections between the vertexes of the triangle and outputs them to a file for use by the successive Gilbert method.

c) Creating hierarchical spheres of links

This process creates spheres which cover a polygon hierarchically. The centers of these spheres are on the polygon surface. The spheres are used by the bubble collision method.

2) Algorithm for collision detection

a) Successive Gilbert method

This process calculates the distance between two links using the previous calculation's information for efficiency. The inner products of all vertexes are calculated at each collision calculation using the ordinary Gilbert method.[9),10)] We assume that the nearest point of contact between any two links is near the nearest point that was calculated in the previous time step. The inner products of vertexes are calculated near this previously calculated point. This is an efficient method for finding the nearest point of contact between two links by using only the near-point inner products using the proximity list file.

b) Bubble collision method[11)]

Bubble collision is a method for finding the lowest layer sphere pair and measuring the distance between them using the ordinary Gilbert method.

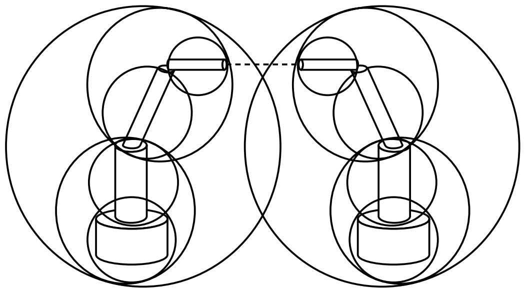

Figure 12
Example of superior tree.

c) Method used in developed software[12)]

This method switches between the successive Gilbert method and the bubble collision method according to the distance between the convex hulls of two links. It can calculate the distance between a group of links in the same way that distances between links in kinematics models are calculated. This method uses a superior tree to quickly find the nearest pair of links. An example of a superior tree is shown in **Figure 12**.

3) Collision force

The method for calculating the collision force and resultant change of motion is shown below. **Table 3** explains the symbols used in the method.

The motion equation of the link system R is:

$$M_R\ddot{\theta}_R + B_R = J_R^T F\ . \tag{6}$$

The motion equation of the link system L is:

$$M_L\ddot{\theta}_L + B_L = J_L^T(-F)\ . \tag{7}$$

Table 3
Symbols.

Symbol	Meaning
M_R	Inertia matrix of link system R
M_L	Inertia matrix of link system L
B_R	Bias vector of link system R
B_L	Bias vector of link system L
J_R	Jacobian matrix of link system R at collision point
J_L	Jacobian matrix of link system L at collision point
$\ddot{\theta}_R$	Joint acceleration of link system R at collision instant
$\ddot{\theta}_L$	Joint acceleration of link system L at collision instant
$\dot{\theta}_R$	Joint velocity of link system R before collision instant
$\dot{\theta}_L$	Joint velocity of link system L before collision instant
$\dot{\theta}_R'$	Joint velocity of link system R after collision instant
$\dot{\theta}_L'$	Joint velocity of link system L after collision instant
F	Collision force vector
f	Collision force
r	Unit vector of collision force
dt	Infinitesimal time
Δt	Integration time step

The energy balance equation for before and after the collision is:

$$\begin{bmatrix}\dot{\theta}_R\\ \dot{\theta}_L\end{bmatrix}^T\begin{bmatrix}M_R & 0\\ 0 & M_L\end{bmatrix}\begin{bmatrix}\dot{\theta}_R\\ \dot{\theta}_L\end{bmatrix}=\begin{bmatrix}\ddot{\theta}'_R\\ \ddot{\theta}'_L\end{bmatrix}^T\begin{bmatrix}M_R & 0\\ 0 & M_L\end{bmatrix}\begin{bmatrix}\dot{\theta}'_R\\ \dot{\theta}'_L\end{bmatrix},\quad(8)$$

where

$$\ddot{\theta}_R=\frac{\dot{\theta}'_R-\dot{\theta}_R}{dt}\;,\quad(9)$$

$$\ddot{\theta}_L=\frac{\dot{\theta}'_L-\dot{\theta}_L}{dt}\;,\quad(10)$$

$$F=f\cdot r\;.\quad(11)$$

The impulse is:

$$f\cdot dt=-\frac{b}{a}\;,\quad(12)$$

where

$$a=MJ^T_R M_R MJ_R+MJ^T_L M_L MJ_L\;,\quad(13)$$

$$b=MJ^T_R M_R\dot{\theta}_R+\dot{\theta}^T_R M_R MJ_R+MJ^T_L M_L\dot{\theta}_L+\dot{\theta}^T_L M_L MJ_L\;,\quad(14)$$

$$MJ_R=M_R^{-1}J^T_R r\;,\quad(15)$$

$$MJ_L=-M_L^{-1}J^T_L r\;.\quad(16)$$

Assuming that the collision force is constant over the integration time step, the collision force is calculated by:

$$F=\frac{f\cdot dt}{\Delta t}\,r\;.\quad(17)$$

And the joint velocities after collision are:

$$\dot{\theta}'_R=MJ_R\;f\cdot dt+\dot{\theta}_R\;,\quad(18)$$

$$\dot{\theta}'_L=MJ_L\;f\cdot dt+\dot{\theta}_L\;.\quad(19)$$

7. Conclusion

We have proposed an efficient method for producing contents which contain the motions of complex objects, particularly those of humans and animals. We have developed a production environment for motion contents, including motion contents for humans and animals, using the proposed method. This environment helps the user to create a complete motion content. Several elementary technologies are used together to edit a motion content, simulate a dynamic system, interpolate a motion, and execute a motion content. Some elementary technologies have been developed as a result of the author's recent studies.

This environment can be used to easily create motion contents in a short time without special skills. It is based on software LSIs, general motion analysis, and high-speed collision checking. General motion analysis can analyze a complex structure, especially the structures of humans and animals. The collision detection function is fast and accurate. The software LSI technology provides interfaces between key technologies.

References

1) F. Nagashima, K. Suzuki, and T. Maruyama: High Speed CG and Simulation Application Development Environment. *Fujitsu Sci. Tech. J.*, **33**, 2, pp.160-169 (1997).

2) M. W. Walker, and D. E. Orin: Efficient dynamic computer simulation of robot mechanisms. *Journal of Dynamic Systems, Measurement, Control*, 104, pp.205-211 (1982).

3) J. Y. S. Luh, M. W. Walker, and R. P. C. Paul: Resolved acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, **25**, 3, pp.468-474 (1980).

4) F. Nagashima and Y. Nakamura: Efficient computer scheme for the kinematics and inverse dynamics of a satellite based manipulator. Proceedings of IEEE International Conference on Robotics and Automation, pp.905-911, 1992.

5) Y. Nakamura: Dynamics Computation of

Closed Link Robots and Optimization of Actuational Redundancy. *Transactions of the Society of Instrument and Control Engineers,* **25**, 5, pp.600-607 (1989).

6) Y. Nakamura: Dynamics of Parallel Mechanism. *Journal of the Robotics Society of Japan*, **10**, 6, pp.13-15 (1992).

7) F. P. Preparata and M. I. Shamos: Computational Geometry - an Introduction. Springer-Verlag, New York, 1985.

8) H. Edelsbrunner: Algorithms in Combinatorial Geometry. Springer-Verlag, New York, 1987.

9) E. G. Gilbert, D. W. Johnson, and S. S. Keerthi: A Fast Procedure for Computing the Distance Between Complex Objects in Three Dimensional Space. *IEEE Journal of Robotics and Automation*, **4**, 2, pp.193-203 (1988).

10) E. G. Gilbert and C. P. Foo: Computing the Distance Between General Convex Objects in Three-Dimensional Space. *IEEE Transactions on Robotics and Automation*, **6**, 1, pp.53-61 (1990).

11) S. Quinlan: Efficient Distance Computation between Non-Convex Objects. Proceedings of the 1994 IEEE International Conference on Robotics and Automation, **4**, pp.3324-3329 (1994).

12) Y. Sato, M. Hirata, T. Maruyama, and Y. Arita: Efficient Collision Detection using Fast Distance-Calculation Algorithms for Convex and Non-Convex Objects. Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, pp.771-778, 1996.

Fumio Nagashima received the Dr. degree in Mechanical Engineering from Keio University, Tokyo, Japan in 1989. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1989 and has been engaged in research and development of software simulation tools. He is a member of the Japan Society of Mechanical Engineers (JSME).