Intuitive Interfaces for Motion Generation and Search

Yoshihiro Okada, Hiroaki Etou, and Koichi Niijima

Graduate School of Information Science and Electrical Engineering, Kyushu University,
6-1 Kasuga-koen, Kasuga, Fukuoka, 816-8580, Japan
{okada,h-etou,niijima}@i.kyushu-u.ac.jp

Abstract. This paper treats intuitive interfaces for motion generation and motion search. These are research results on the interactive animation system achieved by the research group of the authors as the part of a research project "Intuitive Human Interface for Organizing and Accessing Intellectual Assets". For CG animation creation, the motion design of CG characters as well as the shape design is very laborious work. For the motion design, the authors have already proposed a component based motion editing environment, a real-time motion generation system using puppet/marionette metaphors and a motion database management system. These allow the user to create motions interactively, intuitively and make it easy to distribute and re-edit motions. In this paper, the authors introduce these motion generation and motion search systems.

1 Introduction

We have been interested in narrative database systems. Traditionally narrative data have been represented by the text media so far. Now narrative data would be possible to be represented by the CG animation because advances of recent computer hardware technologies have made it possible to create CG animations by very lower costs rather than ever. Towards the development of a narrative database system, we have been studying on an interactive animation system. Especially, this paper treats intuitive interfaces for motion generation and motion search. These are research results on the interactive animation system achieved by our research group as the part of a research project "Intuitive Human Interface for Organizing and Accessing Intellectual Assets".

For 3D animation creation, the character design is a very important factor but very hard work. Especially its motion design and shape modeling are very laborious work. For the motion design, we have already proposed a component based motion editing environment [1, 2]. There are many researches on motion generation for computer animation. Witkin and Kass proposed concept of *spacetime* constraints [3]. After that, many research papers based on *spacetime* constraints were published [4, 5]. IK (Inverse Kinematics) is one of the other popular methods for efficient motion generation. The motion path functionality is also a popular technique to intuitively define movement of a character's center of mass. However, most popular and traditional motion design is based on key-frame animation [6]. A motion is represented as a sequence of a number of poses those are automatically generated by interpolation of several key-poses. Each key-pose is defined by specifying the joints angles of an articulated figure model. Our proposed motion editing environment is also based on the key-frame

G. Grieser and Y. Tanaka (Eds.): Intuitive Human Interface 2004, LNAI 3359, pp. 49–67, 2004.

animation technology. This environment displays all sequential key-poses at the same time on a computer screen. Then by looking at those key-poses users can recognize a complete motion those key-poses mean and can edit each key-pose interactively and easily through comparing with its adjoining poses on a computer screen. We have developed such motion editing environment using *IntelligentBox* [7, 8], which is a component based 3D graphics software development system that provides functional components called *boxes*. The proposed motion editing environment is developed as a composite *box*, which includes user-defined motion information itself. Therefore, users can exchange their edited motions with each other through *Internet* by copy-and-transfer operations of such a composite *box*, and create new motions only by modifying the motions already defined by other users.

Besides the above motion editing environment, for motion generations, we have also proposed a real-time motion generation system using puppet/marionette metaphors [9, 10]. The puppet is a children toy and it is easy to manipulate for even children. For this reason, at first, we employed the puppet metaphor for real-time motion generation. We developed such a real-time motion generation system using *IntelligentBox* [9]. This system used the set of a data-glove and a magnetic motion sensor as its interfaces. A puppet motion is controlled using one-hand actions. Our data-glove generates only ten finger-joint angles. This number is not enough to fully control an articulated figure. So the system requested the user to prepare multiple mapping tables that defines association between ten finger-joint angles of a data-glove and 17 joint of an articulated figure. When the user performs his/her hand action to generate the motion of an articulated figure, he/she chooses an adequate mapping-table and changes the current one into the chosen one by additional keyboard manipulation. Indeed this was inconvenient. Therefore we introduced another metaphor, i.e., a marionette metaphor and especially introduced gravity field concept and ground contact constraint [10]. Actually in the real world, a human action is performed in the gravity field and usually on a floor or on the ground. Similarly a marionette action is also performed based on the physics of the gravity and the ground contact constraint. For this reason, we employed a marionette metaphor to generate the motions of an articulated figure in real time by one-hand actions. As related works, there are researches on the use of motion capture systems as a real-time input interface to control CG character interactively [11]. Noser and Thalmann proposed virtual tennis game environment using a full-body motion capture system as a real-time motion input interface [12]. Full-body motion capture systems have become common. However they are still very expensive and they cannot be used on the desktop. Laszlo, et al [13] proposed interactive control technique for physically-based animation using standard input devices, i.e., a mouse device and a keyboard. This technique requests the user to prepare many motion primitives. The use of a mouse device and a keyboard is convenient but they are not intuitive devices. Oore, et al proposed a desktop input device and interface for interactive 3D character animation [14, 15]. Our research system is very similar to theirs. They use two magnetic-based motion sensors while we use the set of a magnetic-based motion sensor and a data-grove. While our system generates motions of an articulated figure in real time by the user's one-hand actions, which are squat and jump, walking motion and so on, Oore's system can not generate a character animation in real time. This means that their system generates motions of some parts of a figure separately and later it has to compose a full body motion of them. Although the

quality of motions generated by Oore's system seems better than that of our system, the abstract motions generated by our system are able to be used for various applications, e.g., as initial motions for key-frame animation [1, 2], as query motions to search required motions from motion databases and so on.

The above motion editing environment and real-time motion generation system are main research results for motion generations of our interactive animation system research. Indeed, besides the above two topics, we have been studying motion database systems. The development of motion database management systems poses two main questions: which feature can be assumed to be the best similarity among motions, and what is the best way to specify the query. We employed new similarity feature based on the symbolic representation of motion data by the spatial quantization as the answer of the first question. Motion data is a sequence of several poses, each of which consists of joints angles data of an articulated figure model at a different frame time. To reduce the calculation cost for the motion search, we employ not all the joints but a small number of the joints, those mean semantically important joints called feature joints, of each pose of a complete motion data. Furthermore we employed new similarity measure based on the spatial occurrence probability. We divided the 3D space around a model into a small number of subspaces semantically. At each frame, the position of each feature joint is represented as the symbol value that corresponds one of the subspaces, in which the feature joint exists. This means spatial quantization. In this paper, we show two spatial quantization ways and their experimental results for the evaluation. We have already developed its prototype system using *IntelligentBox*. In this paper, we also propose an intuitive interface as the answer of the second question. This interface provides the facility allows the user to intuitively enter query motions for motion data searches. Researches on motion database systems are very few so far. As for motion generation aspects, Unuma et al. [16] proposed an algorithm for the motion regeneration by composing from several semantic primitive motions based on Fourier transform. Uehara group [17] proposed a method for the extraction of primitive motions represented symbolized values to recognize human motions by matching those symbolized sequential values. Lim et al. [18] also proposed an algorithm for the key-posture extraction out of human motion data based on the curve simplification algorithm. We have never found motion database management systems like our prototype that employs the symbolic representation for motion data to reduce the calculation cost of motion searches and that provides an intuitive interface for entering query motions.

The remainder of this paper is organized as follows: Section 2 describes the component based motion editing environment. Section 3 describes the real-time motion generation system using puppet/marionette metaphors. Section 4 treats the motion database system. Especially we describe symbolic representation of motion data for motion searches. Finally Section 5 concludes the paper.

2 Component Based Motion Editing Environment

This section describes the component based motion editing environment. Especially we explain what kinds of *boxes* of *IntelligentBox* are developed and used.

2.1 Component Structure of a Human-Like Model and Its One Pose Data

Fig. 2.1 shows components of a human-like model. This model is consisted of 17 joints. Each joint is *3DRotationBox*. The bottom *box* is *ArrayBox* that stores xyz-angle data of the all joints. Therefore, this *ArrayBox* keeps one pose data. This composite *box* illustrated in the figure is used as a unit for editing one pose.

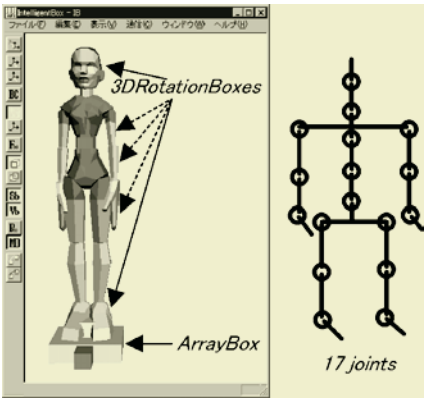

Fig. 2.1. Component structure of a human-like model.

2.2 Editing of Multiple Key-Poses

Fig. 2.2 (left) shows multiple key-poses those mean a walking motion. This motion is consisted of five key-poses. By looking at the five key-poses, it is possible to understand that motion is a walking motion. Figure 2.2 (center) shows another motion. This is a jump motion consisting of six different poses. As for the walking motion, the character's center of mass moves gradually in one direction. So it is not difficult to specify each pose by directly dragging the joints of the corresponding model on a computer screen. However as for the jump motion, the character's center of mass moves up and then down again. So it is difficult to specify each pose by directly dragging the joints of the corresponding model on a computer screen because some poses

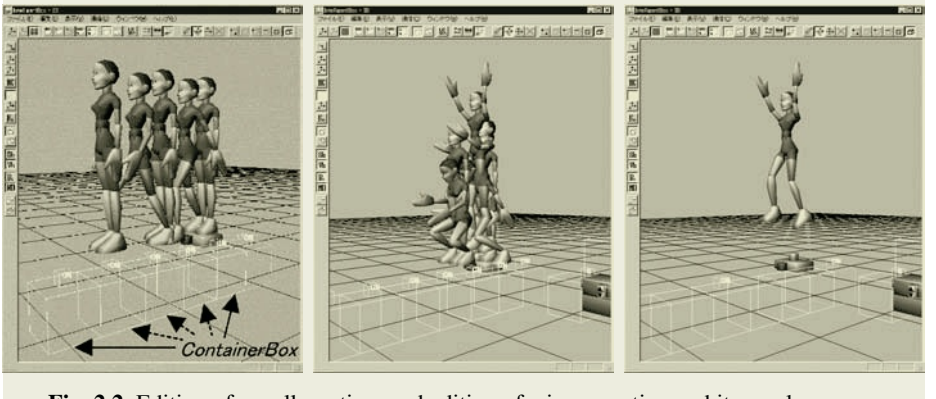

Fig. 2.2. Editing of a walk motion, and editing of a jump motion and its one key-pose.

are occluded. For this case, the system provides functionality to disappear other unnecessary poses except the one pose that the user is currently modifying. There is a particular *box* called *ContainerBox* to provide such functionality. Several *ContainerBoxes* are located below each model and assigned as its parent *box*. *ContainerBox* controls visibility of its descendant *boxes*. If a user clicks a mouse button on *ContainerBox*, its descendant *boxes* become invisible, and the user clicks again then its descendant *boxes* become visible. By interactively controlling their visibility, users can edit each pose with showing only one corresponding pose as shown in Figure 2.2 (right).

2.3 Mechanism of Motion Generation

Fig. 2.3 (left) shows data flow and component structure for concatenated motion generation. The upper part of the figure is component structure for one motion generation. There is *InterpolationBox*. *InterpolationBox* generates a motion as a complete sequence of poses generated by interpolation among several key-poses. The motion is stored in the *slot* of *ArrayBox*. Furthermore the lower part of the figure is component structure for concatenation of several motions. There is *MotionConcatenationBox*. *MotionConcatenationBox* generates one motion as a sequence of several motions. If there are two difference motions, i.e., a walking motion and a jump motion, and the walk motion is assigned a value of zero as its ID number and the jump motion is assigned a value of one. After the user specifies a sequence of ID number values like (0, 1, 1, 0) as the parameter of *MotionConcatenationBox*, the *MotionConcatenationBox* generates one concatenated motion. That motion acts in the order of a walk, a jump, a jump and a walk. Two sequential motions are concatenated smoothly by a linear combination of last *n* frames of the first motion and first *n* frames of the second motion. Strictly speaking, concatenation process generates a smooth motion, i.e., the first motion fades out and the second motion simultaneously fades in.

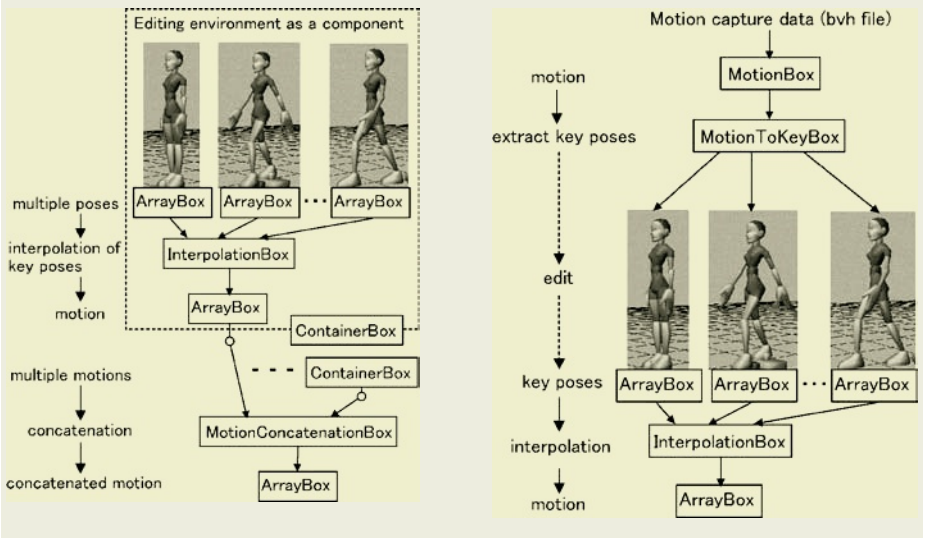

Fig. 2.3. Data flow of concatenated motion generation (left) and data flow to generate new motion by re-editing a motion capture data (right).

2.4 Motion Capture Data Support

As shown in Fig. 2.3 (right), *IntelligentBox* provides a particular *box* called *MotionBox*. This *box* reads a motion capture data file and generates a motion as a sequence of several poses. Currently this *box* supports *BioVision* Inc. BVH file format. In the figure, there is another *box* called *MotionToKeyBox* under *MotionBo*x. This *box* automatically extracts multiple key-poses from the motion data generated by the *MotionBox*. Once several poses are extracted as key-poses, the user can create new motions by reediting those key-poses.

3 Real-Time Motion Generation Using Puppet/Marionette Metaphors

This section describes the real-time motion generation method proposed in this research project. Especially we explain how the user creates motions interactively by the intuitive interface of the proposed system.

3.1 Motion Generation Using Puppet Metaphor

Before explaining the mechanism of motion generation using marionette metaphor, this subsection explains the mechanism of motion generation based on the puppet metaphor. We use the term of 'puppet' to indicate an articulated figure that is directly controlled by the user hand action. On the other hand, we use the term of 'marionette' to indicate a string hung articulated figure. This 'marionette' case has to simulate physical effects, i.e., gravity field and ground contact constraint besides direct control by the user hand action.

3.1.1 Component Structure of a Puppet Model

Fig. 3.1 (left) shows components of a typical puppet model. This model consists of 17 joints. Each joint has three DOF (Degrees Of Freedom) and then it rotates along x-, y- and z-axes. The system receives finger-joint angle data sent from a data-glove device. We use a data-glove named Super Glove Jr. produced by Nissho Electronics Corporation [19]. This device generates ten finger-joint angles data. Each of these angles is applied to some specific joints of the puppet. Then the real-hand action controls the puppet motion in real time. The system also receives one set of position and orientation data sent from a magnetic-based motion sensor, Polhemus Inc. 3SPACE ISOTRACK II [20]. The position and orientation of the puppet change according to this data.

As mentioned above, the puppet model, a human-like model, has 17 joints. However, the data-glove generates only ten finger-joint angles data. To control the puppet motion by only one-hand action it needs a certain mapping scheme between 17 joints of the puppet and ten angles data of the data-glove as shown in Fig. 3.1.

3.1.2 Mapping Scheme for Puppet Motion Control

First of all, we assume that each joint angle p_i of a puppet is represented as the linear combination of ten-angle data of a data-glove. It is calculated using the next equation.

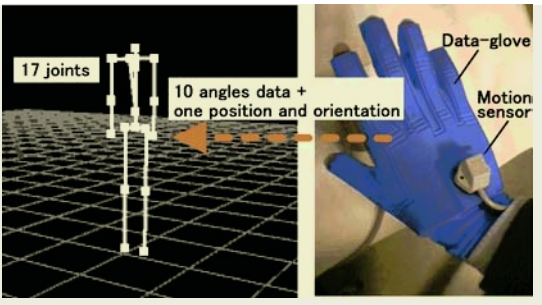

Fig. 3.1. Puppet/marionette model and its control interface.

$$p_i = a_{i,0}h_0 + a_{i,1}h_1 + \cdots + a_{i,9}h_9 + b_i \tag{3.1}$$

where, p_i is the angle of *i*-th joint of a puppet, $h_0, h_1, \cdots, h_9$ are ten-angle data of a data-glove and b_i is the initial angle of *i*-th joint of a puppet. $a_{i,0}, a_{i,1}, \cdots, a_{i,9}$ are coefficients arbitrarily specified by the user.

As for all joints of a puppet, equation (3.1) is transformed into the matrix expression.

$$P = \prod \times H \tag{3.2}$$

where, P is a vector $[p0, p1, p2, \cdots p16]^T$, whose element p_i is also a vector of the x-, y-, and z-angle values of each puppet joint. H is a vector $[h0, h1, \cdots, h9,1]^T$, whose element is the angle value of each hand joint. Finally, $\prod$ is a matrix that means a mapping table and initial pose information as the following matrix.

$$\prod = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,9} & b_0 \\ a_{1,0} & a_{1,1} & \cdots & a_{1,9} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{16,0} & a_{16,1} & \cdots & a_{16,9} & b_{16} \end{bmatrix} \tag{3.3}$$

A vector $[b_0, b_1, \cdots, b_{16}]^T$ specifies an initial pose. Its element means a vector of the initial x-, y- and z-angle values of each puppet joint.

Fig. 3.2 shows four poses of the hand and their four corresponding poses of a puppet. We define Pose 1 is an initial pose since the paper shape of a hand seems more natural rather than the stone shape. Using only one mapping table, by the one-hand poses in Pose2, 3 and 4, the poses of a puppet shown in the right group of Fig 3.2 are obtained for instance. This mapping is adequately specified by the user to make it easier for his/her hand to control the puppet.

3.2 Motion Generation Using Marionette Metaphor

As mentioned in Section 1, one mapping table allows us to make very few kinds of motions and it is insufficient in the practical use. So the system requests the user to

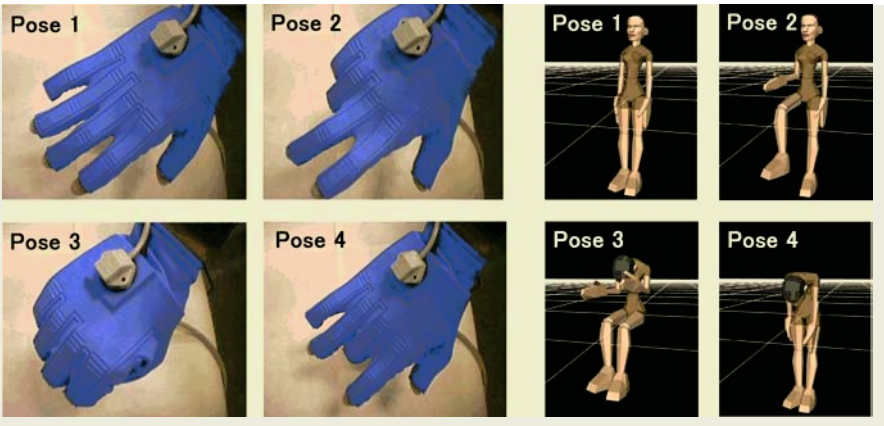

Fig. 3.2. Four one-hand poses and their corresponding puppet poses.

prepare multiple mapping tables and to choose one of them by the keyboard input properly. However, it was thought that this is inconvenient and more intuitive interface is needed. We introduced another real-time motion generation method using a marionette metaphor. This subsection treats motion effects of marionette metaphor, i.e., gravity field and ground contact constraints.

3.2.1 Motion Effect of Gravity Field

Similarly to a real marionette, the body of our marionette consisting of a waist, left and right hips, a chest, and left and right collars is treated as one rigid part. Fig. 3.3 shows five kinds of hand positions and poses in the upper group, and their corresponding marionette positions and poses in the lower group. First one (Pose 1) is a normal position and pose. Other four have a different orientation but their positions are almost the same. Second figure (Pose 2) and third figure (Pose 3) show that the hand fall down forward and that the hand go up backward respectively. These rotations of the marionette body correspond to pitch of the Euler Transformation. Fourth figure (Pose 4) and fifth figure (Pose 5) show the rotations of the marionette body correspond to roll of the Euler Transformation. Although there are no figures, the

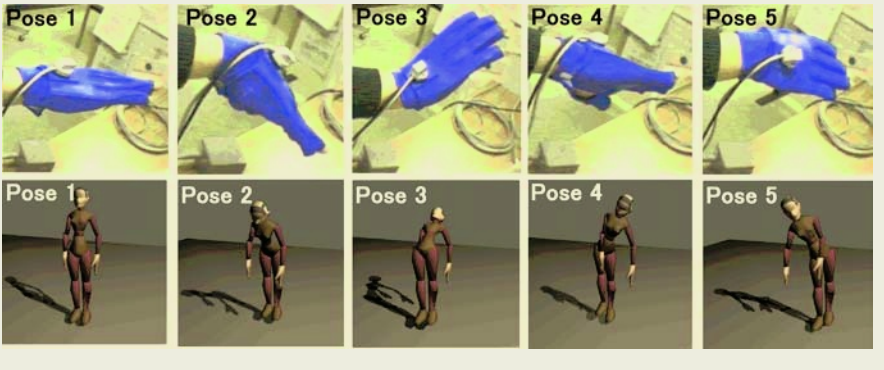

Fig. 3.3. Normal and four one-hand orientations and their corresponding marionette poses with gravity effect.

rotations of the marionette body correspond to head of the Euler Transformation is also possible. The marionette exists in the gravity field so its two arms and two legs always hang down. In this case, the positions of the hand is enough high so the ground contact constraint does not exist. Only gravity effect exists.

3.2.2 Motion Effect Based on Marionette Metaphor

Actually the arms of a real marionette are controlled by the strings connected to each of their hand, and the legs of the marionette are controlled by the strings connected to each of their knee. Fig. 3.4 shows another set of example poses those describe this effect. We used a mapping table similar to the one used to generate the poses of Fig. 3.2, that is, the thumb and little finger of the user hand control the marionette's legs and the index finger and third finger control the marionette's arms. Pose 1 means that the marionette right hand is pulled up by a virtual, invisible string. This virtual string is controlled by the hand action of the user, strictly speaking, by the action of the user's third finger. Pose 2 is almost the same as Pose 1. In this case, the marionette left hand is pulled up by the action of the user's index finger. In the case of Pose 3, the marionette both left and right hands are pulled up.

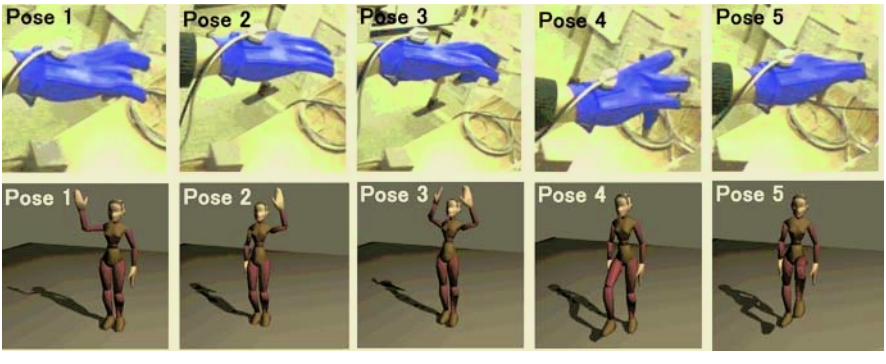

Fig. 3.4. Five one-hand poses and their corresponding strung marionette poses.

As for these poses, the inverse kinematics is partially used to determine the position of the intermediate joint, i.e., the elbows because the user's hand actions only control the position of each of the marionette's two hands. To carry out this, we introduced 2-links Inverse Kinematics. For its more detail, see the paper [10]. As for the remainder of Fig. 3.4, Pose 4 means that the marionette right knee is pulled up by a virtual, invisible string. This virtual string is controlled by the action of the user's little finger. Pose 5 is almost the same as Pose 4. In this case, the marionette left knee is pulled up by the action of the user's thumb. In the both cases, lower legs fall down due to the gravity effect.

3.2.3 Motion Effect of Ground Contact Constraint

As for a real marionette, ground contact constraint plays a significant role to effectively generate various motions. Fig. 3.5 shows another set of example poses concerning ground contact constraint. Left figure (Pose 1) is an initial pose. In this case, neither two feet nor two hands touch the ground. Center figure (Pose 2) means that two

feet touch the ground due to the lower position of the waist since the user's hand position becomes lower. In this case, positions of both left and right knee are automatically calculated using 2-links Inverse Kinematics. Furthermore, right figure (Pose 3) of Fig. 3.5 shows a pose with both feet and hands contacting the ground.

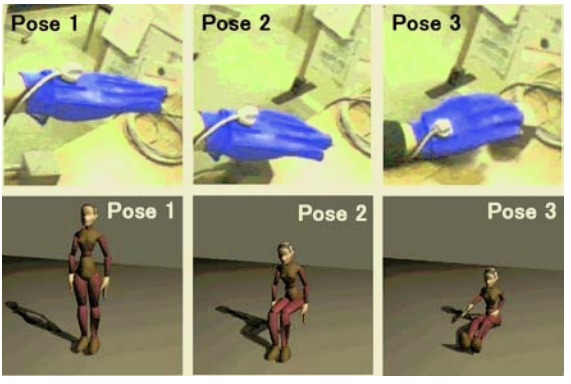

Fig. 3.5. Three one-hand poses and their corresponding marionette poses with ground contact constraints.

3.2.4 Walking Motion Generation

Our system also generates the walking motion of an articulated figure. In this case, the center position of the figure is calculated by following equations:

$$
\begin{aligned}
Z_{i+1}^{C} &= \frac{Z_i^{RF} + Z_i^{LF}}{2} + Z^{MS} \times \cos\theta_i^{Y}, \\
Y_{i+1}^{C} &= Y^{MS}, \\
X_{i+1}^{C} &= \frac{X_i^{RF} + X_i^{LF}}{2} + Z^{MS} \times \sin\theta_i^{Y}, \\
\theta_{i+1}^{Z} &= \theta_{MS}^{Z} \\
\theta_{i+1}^{Y} &= \theta_i^{Y} + \theta_{MS}^{Y}, \\
\theta_{i+1}^{X} &= \theta_{MS}^{X}.
\end{aligned}
\tag{3.4}
$$

Here, $X_{i+1}^{C}, Y_{i+1}^{C}, Z_{i+1}^{C}$ are the x, y, z component of the center position at i+1-th frame. X_i^{RF}, Z_i^{RF} are the x, z component of the right foot position at i-th frame, as well, X_i^{LF}, Z_i^{LF} are the x, z component of the left foot position. Y^{MS}, Z^{MS} are the y, z component of the position data sent by a motion sensor device. Its x component is not used in this case. $\theta_{i+1}^{X}, \theta_{i+1}^{Y}, \theta_{i+1}^{Z}$ are the rotation angles of the center along x-axis, y-axis, and z-axis respectively at i+1-th frame. θ_i^{Y} is the rotation angle along y-axis at i-th frame. $\theta_{MS}^{X}, \theta_{MS}^{Y}, \theta_{MS}^{Z}$ are the rotation angles data along x-axis, y-axis, and z-axis respectively, sent from a motion sensor device. It is possible to make a turn by changing θ_{MS}^{Y} and also control its speed by the change of Z^{MS}.

3.3 Discussion

Even if using conventional computer animation software, creation of the motions shown in Fig. 3.3, 3.4 and 3.5 is not easy, especially the motion of Fig. 3.5 is very difficult. However, using our system, the user can create those motions by his/her one-hand actions interactively in real time.

The main mathematical factor of our system is only 2-links Inverse Kinematics. This is very simple so its calculation cost is very low. Therefore, our system generates the motions demonstrated in this subsection in real time. As for the performance of the system, frame rate is around 18 fps using a standard PC, 850MHz Pentium III CPU, 640MB memory, and GeForce3 graphics. This value is satisfactory for interactive animation systems.

4 Motion Database System Using Symbolic Representation of Motion Data for Motion Searches

This section treats a motion database system using symbolic representation of motion data for motion searches. Especially we describe a system overview, our similarity measure of motion data and its evaluation results.

4.1 System Overview

Fig. 4.1 shows the system configuration of our motion database system. The system has original motion data as primary information and their symbolic representation data as secondary information in its motion database. When the user enters a query motion for similarity motion searches, the system generates the symbolic representation data from the query motion in order to compare it to each data in the secondary information and to output similar motions as the comparison result.

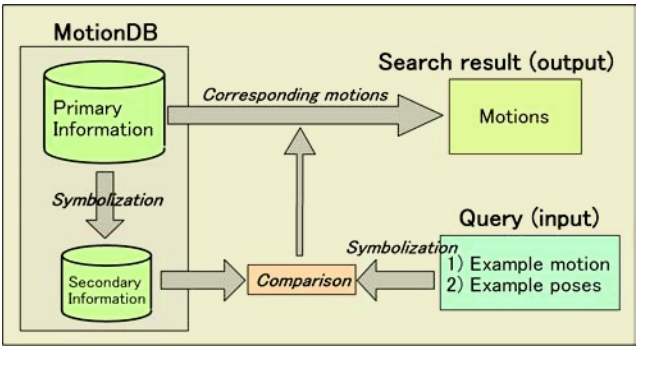

Fig. 4.1. System configuration.

4.2 Similarity Search of Motion Data

This subsection describes how to generate secondary information, i.e., symbolic representation data extracted from motion data by spatial quantization. This spatial quantization is applied to a few joints of an articulated figure model, which are semanti-

cally important joints called feature joints. First of all, we define feature joints in next sub-subsection. After that, we explain two spatial quantization ways, and introduce our similarity measure for similarity motion searches based on the spatial occurrence probability.

4.2.1 Feature Joints

If the quantization process to obtain secondary information is applied to all the joints of an articulated figure model, its calculation cost and the secondary information size become large. Then, if it is possible to determine small number of joints, those are semantically important or enough to characterize the corresponding motion, the quantization process should be applied to such a small number of joints. We decided four joints, i.e., a left wrist, right wrist, left ankle and right ankle, as feature joints because we assume that the positions of those joints are calculated using their parent joints information and then these joints have much significant information rather than the other joints. In the following sub- and sub-subsections, feature joints mean four joints, i.e., a left wrist, right wrist, left ankle and right ankle.

4.2.2 Spatial Quantization for Symbolic Representation of Motion Data

Each motion data consists of multiple sequential pose data. Our spatial quantization process is applied to each pose data and consequently multiple sequential symbolic representation data will be obtained after that. Strictly speaking, the symbolic representation data of each pose contains four symbol values, each of which means the location information of the corresponding feature joint. We treat two spatial quantization ways as shown in Fig. 4.2.

The space division is performed based on the relative position of each joint to the center of mass of an articulated figure model. That is, the center of mass of the model becomes the origin of a local coordinate system around the model. This spatial quantization is not influenced by the direction of the model in a motion. The left figure of Fig. 4.2 shows one spatial quantization way that the space is divided into eight subspaces and one of the eight unique numbers, zero to seven, is assigned to each of them as its region number. For each pose in a motion, each feature joint is represented as the region number where the joint is located. For instance, in the case of the left figure, the right ankle is located in the subspace numbered by zero and then it is repre-

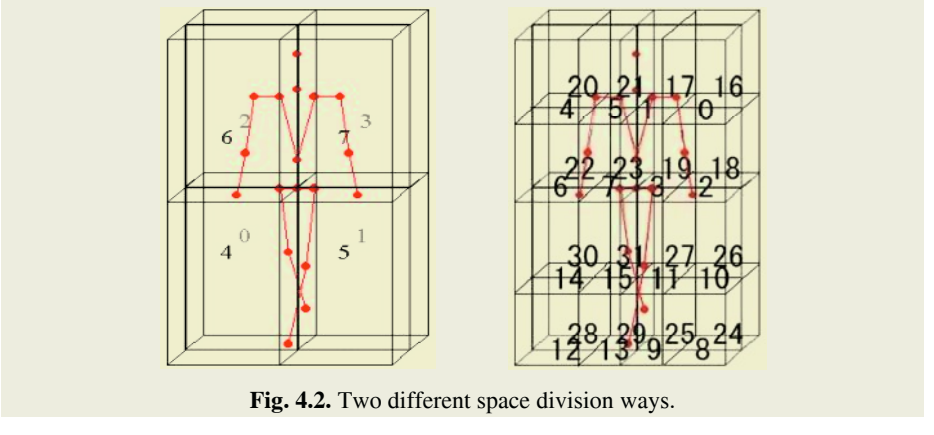

Fig. 4.2. Two different space division ways.

sented as the number 0, and the left ankle is located in the subspace numbered by 5 and then it is represented as the number 5. This is our symbolization process by the spatial quantization. Symbol values assigned to feature joints do not represent their strict positions, that is, there is ambiguity so that the secondary information consisting of such symbolic representation data seems available as the feature information of motions for similarity searches.

The papers [21, 22] proposed the same space division as the right figure of Fig. 4.2. In this paper, we propose another space division way shown in the right figure of Fig. 4.2 because this could obtain better results than the other several space division ways actually we tried. In the first trial, we decided to divide the space into 27 subspaces equally. In this case, we had to calculate the maximum space reachable for feature joints from the center of mass of a model. Then, we divided the space into 27 subspaces by dividing each X-, Y- and Z-direction into equally three segments. However, this division way is not good because in many motions, feature joints exist close to the center of mass so that outer regions of them are almost useless. As the second trial, we divided the same maximum space into 64 subspaces by dividing each direction equally into four segments. However, in this case, the number of regions is too many and most regions are useless. Especially outer regions are useless due to the same reason of the 27 division case. Finally we found that the space division shown in the right figure of Fig. 4.2 is the most efficient way in our trials. The space is divided into four segments in the both X- and Y-direction, and divided into two segments in the Z-direction. Totally the space is divided into 32 subspaces. Symbol values assigned to the subspaces are determined to satisfy that the difference of two symbol values of two adjoining subspaces is only one bit. Actual division points are decided as follows:

1) Three division points in X-direction: the origin, the midpoint between a left wrist and a left ankle, the midpoint between a right wrist and a right ankle.
2) Three division points in Y-direction: the origin, the midpoint between a shoulder and an elbow, the midpoint between a knee and an ankle.
3) One division point in Z-direction: the origin.

In the case of the right figure of Fig. 4.2, the positions of a left ankle, right ankle, left wrist and right wrist are symbolized with 9, 29, 26 and 14 respectively. This is the symbolization for one pose, and one pose data is represented as one set of symbol values like {9, 29, 26, 14}. This symbolization is applied to all pose data, and the corresponding motion data is represented as the sequence of sets of such symbol values. This symbolized data of a motion is used to generate the spatial occurrence probability explained in next sub-subsection.

4.2.3 Spatial Occurrence Probability

As mentioned previously, each motion data is a set of continuous pose data. By the symbolization process, the pose data at each frame time is represented as a set of four symbol values of feature joints. Contrarily, the motion of each feature joint in a whole motion is represented as a set of continuous symbol values. From the set, it is possible to calculate the spatial occurrence probability of the corresponding feature joint. Strictly speaking, the spatial occurrence probability means a probability distribution of subspaces in which the corresponding feature joint exist over the whole motion.

From one symbolized motion data, one set of four spatial occurrence probability data, i.e., four histograms shown in Fig. 4.3,.will be obtained. We use it for our similarity search as the feature information. For *i*-th feature joint, its occurrence probability of *j*-th region is calculated by the next equation.

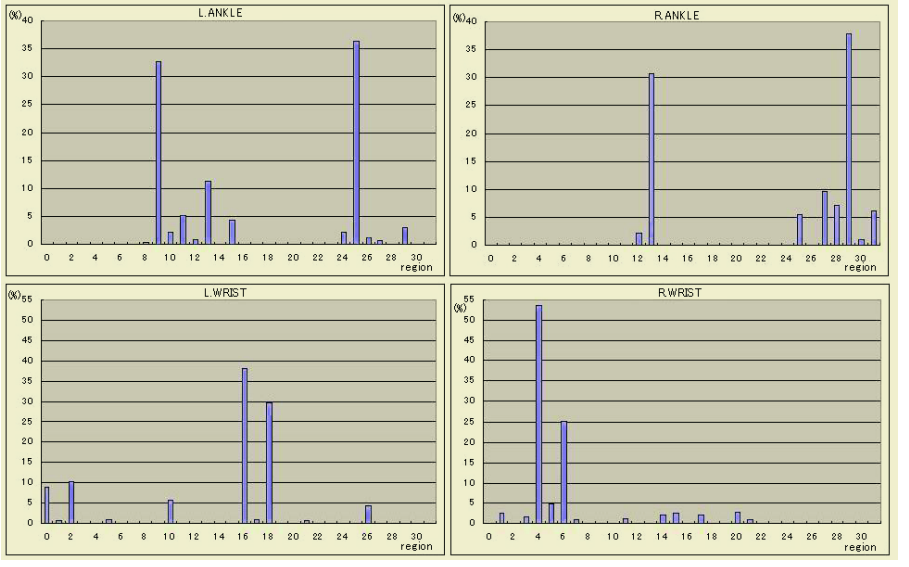

Fig. 4.3. Four histograms of a left wrist, right wrist, left ankle and right ankle.

$$p(i,j) = \frac{count(i,j)}{N} \tag{4.1}$$

Here, N is the number of frames in a symbolized motion data, $count(i,j)$ is the number of *j*-th region symbol values concerning *i*-th feature joint in a symbolized motion data.

4.2.4 Similarity Measure

In the case of the similarity search, our motion database system outputs motion data similar to the query motion entered by the user. Strictly speaking, the system calculates dissimilarity between a query motion and each motion of the motion database and then outputs motion data having the smaller dissimilarity value one by one. The dissimilarity value between two motions Q and T is calculated as Euclidean norm using the next equation.

$$E(Q,T) = \frac{1}{M}\sum_{i=0}^{M-1} \| Q(i) - T(i) \| = \frac{1}{M}\sum_{i=0}^{M-1} \sqrt{\sum_{j=0}^{N-1} (Q(i,j) - T(i,j))^2} \tag{4.2}$$

Here, $Q(i)$ and $T(i)$ are the histograms concerning feature joint i of two motions Q and T respectively. M is the number of feature joints. M is always 4. $Q(i,j)$ and $T(i,j)$ are the occurrence probabilities of region j concerning feature joint i of two

motions Q and T respectively. N is the number of regions. In the case of the right figure of Fig. 4.2, N is 32.

4.3 Prototype Systems

Figure 4.4 shows the snapshot of our motion database system developed using *IntelligentBox*. This system searches motions from motion database, similar to the example motion chosen by the user. The system also searches motions including sequential key poses specified by the user as the query. This means exact match search. There are five same articulated figure models in the figure. The most left one is prepared for the preview of searched motions. One of the searched motions is also used as the query motion of the next similarity motion search. The other four models are prepared to specify query poses for the exact match motion search.

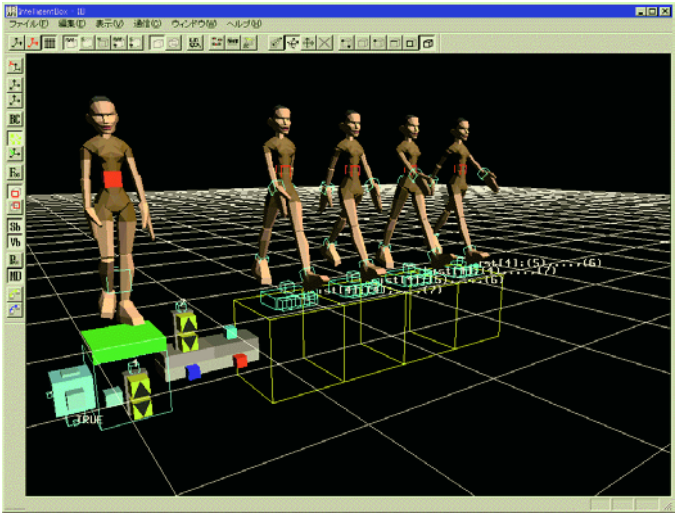

Fig. 4.4. Motion database system developed using *IntelligentBox*.

The system calculates the positions of feature joints to obtain the corresponding symbol values per each of the four poses and it obtains their symbolic representation data. Hence, the system retrieves motions whose symbolic representation data includes the symbolic representation data calculated from the four query poses. Although there are four models to specify query poses in the figure, it is possible to change the number of models by means of making their copies or deleting some of them interactively.

4.4 Experiments

This subsection presents experimental results of similarity searches for the evaluation of our proposed method. Before showing experimental results, we describe our motion database and evaluation measures.

4.4.1 Motion Database

For the evaluation of our motion search method, we had to prepare a motion database. We used a commercial product called "RIKIYA" [23]. It contains 300 motion data created by recording real human motions using a motion capture system. Unfortunately these 300 motions are composite motions like the sequence of primitive motions, e.g., the sequence of "walk", "tumble", "rise", and "walk" again. In the practical use, the user usually enters a primitive motion, e.g., "walk", "jump" etc., to search its similar motions. Therefore, we made seven classes of primitive motions, totally 61 primitive motions, from the 300 composite motions of "RIKIYA". Its details will be indicated in Table 4.1 and Table 4.2.

4.4.2 Evaluation Measures

We use the same three evaluation measures described in [24]. They are *First tier*, *Second tier* and *Nearest neighbor* as follows.

First Tier: This criteria means the percentage of top (k–1) matches (excluding the query) from the query's class, where k is the number of motions in the class.

Second Tier: This criteria is the same type of result, but for the top 2(k–1) matches.

Nearest Neighbor: This criteria means the percentage of test in which the top match was from the query's class.

4.4.3 Experimental Results

As described above, we prepared 61 primitive motions by manually cutting out from original motion data. There are seven classes, i.e., "walk", "jump", "tumble", "rise", "sit down a chair", "kick by right leg", and "throw by right hand". We calculated the above evaluation measures for the similarity searches on these seven classes of primitive motions. Table 4.1 and 4.2 show evaluation results of the 8 division case and those of the 32 division case. These values are averages of the same class motions and all motion classes. The total search times to generate the results on 61 motions for the 8 division case and the 32 division case are around 4 seconds and around 5 seconds using a standard PC, Pentium III 500MHz CPU and 256 MB memory. Search times for one query motion in the 8 division case and in the 32 division case become 0.065 second and 0.082 second respectively. Most results of the 32 division case are equivalent or better than those of the 8 division case. For the accuracy, the motion classes "kick by right leg" and "throw by right hand" indicate not good value. One reason for this seems that these motions have high locality so that we should use only right leg or right hand as feature joints. Another reason may be that the number of motions in each of these classes is too small.

4.5 Discussion

Our motion database does not have a large number and many kinds of motions because we had to prepare it by manually cutting out primitive motions from composite motions of a commercial product database. For precise evaluations of the availability of our motion database system, we will have to prepare a motion database consisting of a large number and many kinds of motions. Currently we have been studying algorithms that help us to extract primitive motions from composite motions, and we have

already proposed new method based on hierarchical curve simplification [25]. We will make a satisfactory motion database by the help of that method.

Table 4.1. Evaluation results on 32 division

Motion class	1st measure	2nd measure	3rd measure	# of motions
walk	0.905229	0.996732	1.000000	18
jump	0.393939	0.621212	0.666667	12
tumble	0.400000	0.555556	0.800000	10
rise	0.523810	0.666667	1.000000	7
sit down a chair	0.533333	0.633333	0.833333	6
kick by right leg	0.416667	0.666667	0.250000	4
throw by right hand	0.250000	0.333333	0.250000	4
All motion classes	0.566471	0.711770	0.786885	61

Table 4.2. Evaluation results on 8 division

Motion class	1st measure	2nd measure	3rd measure	# of motions
walk	0.937908	1.000000	1.000000	18
jump	0.295455	0.575758	0.500000	12
tumble	0.377778	0.688889	0.700000	10
rise	0.380952	0.547619	0.428571	7
sit down a chair	0.666667	0.666667	0.833333	6
kick by right leg	0.250000	0.333333	0.250000	4
throw by right hand	0.166667	0.250000	0.000000	4
All motion classes	0.533425	0.687945	0.655738	61

Moreover, as described in Section 3, we have proposed a real-time motion generation system using puppet/marionette metaphors. Using this system, the user can generate coarse motions of an articulated figure model by his/her hand actions in real time. Those motions are possible to be used as query motions for similarity motion searches. We are supposed to integrate the two systems to develop an intelligent motion management system.

5 Concluding Remarks

In this paper, we described intuitive interfaces for motion generation and motion search. These are research results on the interactive animation system achieved by our research group as the part of a research project “Intuitive Human Interface for Organizing and Accessing Intellectual Assets”. We did these researches toward the development of a narrative database system in which narrative data would be represented as CG animations. As a result, we studied on intuitive interfaces for CG animation creation. For the motion design, we have proposed a component based motion editing environment, a real-time motion generation system using puppet/marionette meta-

phors and a motion database system by symbolic representation of motion data. In this paper, we mainly treated these topics. On the other hand, for the shape design, we have also proposed a polygonal model database management system that accepts hand sketch images as the query and outputs corresponding polygonal models by the silhouette image matching [26-28]. As well, we have other research results [29, 30]. Due to the page number limitation, we did not describe them in this paper. See each paper for its detail.

As future works, we will integrate all the research results mentioned in this paper in order to develop an interactive animation system using *IntelligentBox* towards the development of a narrative database system.

References

1. Okada, Y.: Intuitive Motion Editing Environment for Interactive Animation Systems, Proc. of Symposium on Visual Computing/Graphics and CAD, pp. 109-114, June 2001, (in Japanese).
2. Okada, Y., Component Based Motion Editing Environment for Game Character Design, Proc. of Second International Conference on Intelligent Games and Simulation, SCS Publication, pp. 22-26, 2001.
3. Witkin, A. and Kass, K.: Spacetime constraints, Proc. of SIGGRAPH'88, pp. 159-168, 1988.
4. Gleicher, M.: Motion editing with spacetime constraints, Proc. of SIGGRAPH'97, pp. 139-148, 1997.
5. Lee, J. and Shin, S.-Y.: A hierarchical approach to interactive motion editing for human-like figures, Proc. of SIGGRAPH'99, pp. 39-48, 1999.
6. Life Forms™, http://www.credo-interactive.com/products/lifeforms/lf_4-0_studio.html
7. Okada, Y. and Tanaka, Y., 1995: IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications, Proc. of Computer Animation '95, IEEE Computer Society Press, pp. 114-125.
8. Okada, Y. and Tanaka, Y., 1998: Collaborative Environments in IntelligentBox for Distributed 3D Graphic Applications, The Visual Computer (CGS special issue), Vol. 14, No. 4, pp. 140-152.
9. Okada, Y., Real-time character animation using puppet metaphor, Workshop Note of the First International Workshop on Entertainment Computing (IWEC2002), pp. 86-93, 2002.
10. Okada, Y.: Real-time Motion Generation of Articulated Figures Using Puppet/Marionette Metaphor for Interactive Animation Systems, Proc. of the 3rd IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP03), ACTA Press, pp. 13-18, Benalmadena, SPAIN, September 2003.
11. David J. Sturman, Computer puppetry., IEEE Computer Graphics and Applications, 18(1):38-45, January/February 1998.
12. Noser, H. and Thalmann, D., Sensor Based Synthetic Actors in a Tennis Game Simulation, Proc. of Computer Graphics International '97, IEEE Computer Society Press, pp.189-198, 1997.
13. Laszlo, J., Panne, M.van de, and Fiume, E., Interactive Control For Physically-Based Animation, SIGGRAPH2000, pp.201-208, 2000.
14. Oore, S. Terzopoulos, D. and Hinton, G. ,A Desktop Input Device and Interface for Interactive 3D Character Animation, Proc. of Graphics Interface 2002, 2002.
15. Oore, S. Terzopoulos, D. and Hinton, G. ,Local Physical Models for Interactive Character Animation, Computer Graphics Forum, Volume 21, Number 3, Proceedings of Eurographics 2002.

16. Unuma, M., Anjyo, K. and Takeuchi, R.: Fourier Principles for Emotion-based Human Figure Animation, Proc. SIGGRAPH95, ACM SIGGRAPH, pp. 91-96, 1995.
17. Osaki, R., Shimada, M. and Uehara, K.: A Motion Recognition Method by Using Primitive Motions, Proc. of the Fifth Working Conference on Visual Database Systems (VDB5), pp. 117-128, 2000.
18. Lim, Ik. S. and Thalmann, D.: Key-posture Extraction out of Human Motion Data by Curve Simplification, Proc. of 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC2001), vol. 2, pp. 1167-1169, 2001.
19. http://www.nissho-ele.co.jp/3d/
20. http://www.polhemus.com/
21. Watanabe, R., Okada, Y. and Niijima, K.: A motion search technique based on symbolized expression of pose data, Proc. of IPSJ 63rd all Japanese domestic conference, pp. 225-226, September 2001, (in Japanese).
22. Watanabe, R. Okada, Y. and Niijima, K.: A motion search system based on symbolized expression of pose data, Proc. of IPSJ 64th all Japanese domestic conference, pp. 49-50, March 2002, (in Japanese).
23. Viewworks, http://www.viewworks.co.jp/
24. Osada, R, et al: Matching 3D Models with Shape Distributions, Shape Modeling International, May 2001.
25. Etou, H., Okada, Y. and Niijima, K.: Feature Preserving Motion Compression Based on Curve Simplification, CD-ROM Proc. of ICME2004, 2004.
26. Okada, Y.: 3D Model Matching Based On Silhouette Image Matching, Proc. of CSCC2002 (Recent Advances in Circuits, Systems and Signal Processing), WSEAS Press, pp. 380-385, Rethimno Greece, July 2002.
27. Okada, Y.: 3D MODEL DATABASE SYSTEM BY HAND SKETCH QUERY, Proc. of IEEE International Conference on Multimedia and Expo, Vol. I, pp. 889-892, Lausanne, Switzerland, August 2002.
28. Okada, Y.: 3D Model Database System by Hand Sketch Query and Its Intuitive Interface, to appear in 13th European-Japanese Conference on Information Modeling and Knowledge Bases (13EJC), Kitakyushu, Japan, June 2003.
29. Akazawa, Y., Okada, Y. and Niijima, K.: REAL-TIME VIDEO BASED MOTION CAPTURE SYSTEM AS INTUITIVE 3D GAME INTERFACE, Proc. of Third International Conference on Intelligent Games and Simulation (GAME-ON2002), SCS Publication, pp. 22-28, London UK, November 2002.
30. Tanaka, Y., Okada, Y. and Niijima, K.: Treecube: 3D Visualization Tool for Hierarchical Information, to appear in 13th European-Japanese Conference on Information Modeling and Knowledge Bases (13EJC), Kitakyushu, Japan, June 2003.