Character Motion Control by Hands and Principal Component Analysis

Masaki Oshita* Hayato Oshima Yuta Senju Syun Morishige

Kyushu Institute of Technology

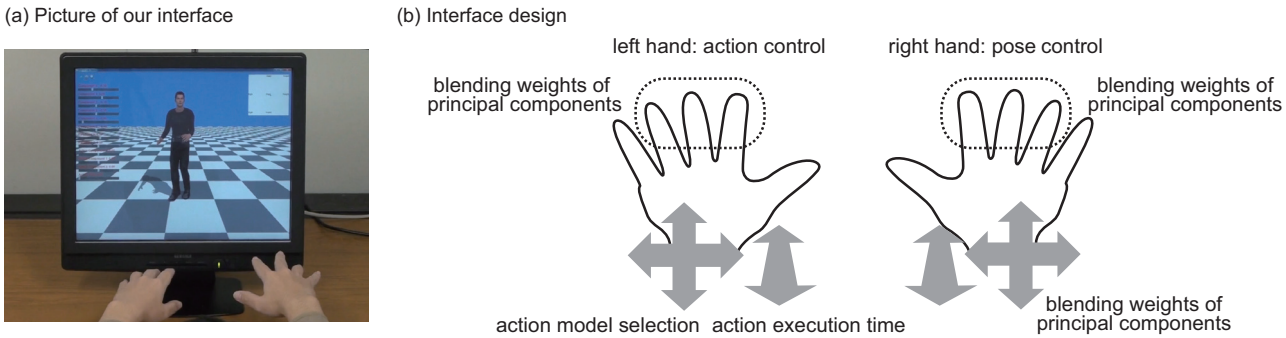

Figure 1: *Proposed interface. (a) Our interface. (b) Interface design. To control a pose, the blending weights of the principal components are varied by moving the right hand and bending/extending the fingers. To control an action, the primary key pose is altered by bending/extending fingers in the same way as the pose control. An action is executed by moving the left hand forward and controlling the action execution time. The current action model is selected by moving the left hand in the up-down and left-right directions.*

Abstract

In this paper, we propose an interactive character motion control interface that uses hands. Using their hands and fingers, the user can control a large number of degrees-of-freedom (DOFs) at the same time. We applied principal component analysis (PCA) to a set of sample poses, and assigned the extracted principal components to each DOF of the hands (such as the hand positions and finger bending/extending angles). The user can control the blending weights of the principal components and deform the character's pose by moving their hands and bending/extending their fingers. We introduced pose and action controls, so that we can alter the standing pose and perform various actions with deformations. So that various types of actions were possible, we constructed a number of action models in advance. We introduced action model selection and action execution mechanisms. We developed methods for computing the feature vector, for applying PCA, and for pose and action synthesis. We present our experimental results and demonstrate the effectiveness of our interface.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Keywords: Motion Control, Hand, Principal Component Analysis, Computer Animation, User Interface

*e-mail:oshita@ces.kyutech.ac.jp

1 Introduction

The interactive control of character motion is important in real-time applications such as computer games and communications using avatars. However, character control in current applications is very limited. Although there are various types of control devices such as the game pad, keyboard, and touch screen, the user can only select pre-defined actions by, e.g., pushing a button, touching the screen, or performing a gesture. Users sometimes want to change the pose and style of motions depending on the situation, but this is not possible with conventional interfaces.

In this paper, we propose an interactive character motion control interface for hands. Using hands and fingers, the user can control a large number of degrees-of-freedom (DOFs) at the same time. We applied PCA to a set of example poses, and assigned the extracted principal components to each DOF of the hands (such as the hand positions and finger bending/extending angles). The user can control the blending weights of the principal components and deform the character's pose by moving hands and bending/extending fingers. We introduce pose and action controls for altering the standing pose and performing various actions with deformations. So that various types of actions are possible, we constructed a number of action models in advance. We introduced action model selection and action execution mechanisms. We developed methods for computing the feature vector, applying PCA and pose and action synthesis. In this paper, we present our experimental results and demonstrate the effectiveness of our interface.

Using fingers and hands, the user can make the character take various poses and perform actions with various deformations. This is an advantage of our method, when compared with conventional interfaces in current applications. Our interface provides an interesting experience of freely controlling a character's pose and motion, using fingers and hands like a puppeteer.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes our interface and system design. Section 4 explains our pose method, and Section 5 explains our action synthesis method. Section 6 presents our experimental results and discussion. Finally, Section 7 concludes this paper.

2 Related Work

2.1 Principal Component Analysis (PCA)

PCA has been extensively applied to character motion processing. In general, PCA is a popular technique for dimension reduction. Because human poses and motions are high dimensional data, PCA can be used to reduce the dimensions and make processing more efficient. Safonova et al. [2004] applied PCA to increase the efficiency of computations in space-time motion optimization. Shin and Lee [2006] applied PCA to map example poses into a two-dimensional latent space, so that the user can synthesize a pose or a motion by selecting a point or drawing a trajectory in the latent space. Min et al. [2009] applied PCA to example motions instead of poses, to synthesize new motions based on given constraints. PCA is also often used for motion or gesture recognition [Ishigaki et al. 2009; Liang et al. 2009; Numaguchi et al. 2011]. However, these methods process the principal components internally, and do not provide a way for controlling the blending weights. One reason for this is that it is difficult to control multiple principal components using conventional input devices. This research allows users to interactively control the multiple components using a hand motion sensing device.

Crochow et al. [2004] considered another dimension reduction technique, the scaled Gaussian process latent variable model (SGPLVM). They used it to map example poses onto a two- or three-dimensional latent space for pose synthesis. Although SGPLVM is more powerful than PCA, the latent space is nonlinear and its axes do not represent specific pose deformations. Therefore, SGPLVM does not fit with our interface, and we chose PCA.

2.2 Data Glove Interface

Researchers have previously used data gloves to control characters. Okada [2003] introduced a metaphor of a puppet-like control and a virtual string. He mapped each finger, so that it represents the upward movement of a specific body part. Komura and Lam [2006] controlled walking motions using a data glove. They mapped the movements of two fingers to one-dimensional locomotion time. Oshita et al. [2013] also developed a puppet-like control, which used both hands and their fingers in the same way as an actual puppet. These methods use the data glove to only control a part of the character's body. This is because there are fewer DOFs for a data glove than for a human pose. It is impossible to control the whole body using these kinds of mappings.

Nik and Oshita [2012] used a data glove to specify parameters for motion blending [Rose et al. 1998; Park et al. 2002]. With their interface, the user only controls abstract parameters and cannot directly control the character's pose.

2.3 Motion Control Interface

Various input devices are used as motion control interfaces.

There are many methods for controlling a character's motion using a conventional pointing device such as a mouse or pen. Generating locomotion along with a given trajectory is a common interface [Park et al. 2002]. However, with this type of interface, the type of motion is limited to walking or running. Thorne et al. [2004] introduced gesture-based motion selection. Based on the gestures drawn along a trajectory, their system inserts predefined motions such as a jump or flip. Oshita [2005] proposed a stroke-based motion selection technique that chooses an appropriate action according to the initial and terminal points of a single stroke drawn on the screen. With these systems [Thorne et al. 2004; Oshita 2005], users can simply select actions by drawing a trajectory or stroke. However, the executable motions are fixed and cannot be changed. Igarashi et al. [2005] proposed a spatial keyframing animation technique. This approach can continuously change a character's pose based on the cursor position. But to use this technique, the key poses must be placed at appropriate positions that are dependent on the specific action.

Multi-touch input devices such as tablet computers have recently become widely available. There are some multi-touch interfaces for interactive character motion control. Oshita [2012] applied a statistics-based inverse kinematics (style-based IK [Grochow et al. 2004]) to control the whole body of the character. However, the synthesized poses are limited to a given set of example poses. To utilize high dimensional inputs, a large number of example poses must be given. Our interface can generate various poses using the principal components extracted from given example motions. Additionally, the combination of hand positions and finger angles has more DOFs than multi-touch inputs.

Some researchers have combined physics simulations with controlling a limited number of DOFs. Laszlo et al. [2005] used a mouse to control a small number of joints, and generated the full-body motion using physics simulations. Shiratori and Hodgins [2008] used Wii remotes to determine a few parameters for physics-based controllers of several actions such as walking, running, and jumping. Physically plausible motion can be generated and controlled using physics simulations. However, physics-based controllers must be designed for each type of action in advance, and it can be hard to achieve various types of actions and styles.

Recently, low-price markerless motion capture devices such as Microsoft Kinect [Microsoft] have become available. A character's full body motion can be freely controlled using these devices [Ishigaki et al. 2009; Oshita 2006], but they require a large workspace. Moreover, the user must actually perform the highly dynamic actions, which is difficult for untrained users.

3 System Design

In this section, we describe our system design including input devices, interface design and system overview.

3.1 Input Devices

Our interface can work with any hand motion sensing device. In this research, we used a depth-camera-based hand motion sensing solution, which combines a PrimeSense camera [PrimeSense] (as used in a Microsoft Kinect [Microsoft]) and a third party hand motion sensing library [3Gear Systems]. Depth-camera-based technologies have recently been extensively applied [PrimeSense ; Leap Motion ; Intel]. They have the advantage that the user does not have to wear a device. Alternatively, we can use a device such as a data glove [5DT] to capture finger movements, and magnetic [Polhemus] or optical tracking devices [Natural Point] to capture hand movements. Using these hand motion sensing devices, the positions and orientations of two hands and the bending/extending angles of ten fingers (or positions of finger tips) can be measured in real time. Our method uses only the hand information given in Table 1.

All input data from the hand motion sensing device (Table 1) are normalized between $-1 \sim 1$. When a finger is relaxed and slightly bent, its finger angle is zero. We allow some variation in the zero value range. When the finger is completely bent, the angle is 1. When the finger is fully extended, the angle is -1. The initial position of a hand is $(0, 0, 0)$, and it can vary between $(-1, -1, -1)$ and $(1, 1, 1)$. The initial position and operational range are acquired

Table 1: *Input parameters from the hand motion sensing device.*

Parameter	Dimension
finger angles (both hands)	$5 \times 2 = 10$
hand positions (both hands)	$3 \times 2 = 6$

Table 2: *Control parameters in our method.*

Parameter	Variable	Dimension
blending weights of principal components for pose control	θ	6 (up to 8)
blending weights of principal components for primary key pose for action control	ϕ	3 (up to 5)
action execution time	t	1
action model selection	$\mathbf{s}$	2

for each hand during a calibration process, and the input positions are scaled during run time. We also apply a low-pass filter to the finger angles and hand positions to reduce noise. In our implementation, we apply a moving average filter, which calculates the average of n data series.

3.2 Interface Design

The interface design of our method is shown in Figure 1. We have introduced pose and action controls. The right hand is used to control the pose, and the left is used to control the action.

To control the pose, we applied PCA to a set of example standing poses and assigned the extracted principal component to each DOF of the right hand (hand position and finger bending/extending angles). When the hand is in its initial position and all the fingers are relaxed, the character has a neutral standing pose. By moving the right hand and bending/extending its fingers, the blending weights of the principal components are controlled and the synthesized standing pose changes dynamically. Among the extracted principal components, the one that creates the largest movement of the center of mass in the x-axis direction was assigned to the x-axis position of the user's right hand. The same mapping was applied to the y- and z-axes positions. The other principal components with larger movements were assigned to the fingers of the right hand. This way the user can intuitively remember the mappings between the DOFs and principal components.

To control the action, we applied PCA to a set of primary key poses, using example motions for each type of action. The primary key pose is the most important (representative) key pose in the action (e.g., for the punching action, the pose with the arm extended the most). The primary key poses were manually specified in advance. The principal components extracted from the primary key poses were assigned to the finger bending/extending angles of the left hand. By bending/extending the fingers of the left hand, the user can control the blending weights of the principal components and alter the primary key pose of the action, in the same way as the pose control. The controlled key pose is visualized on the screen as shown in Figure 2. By moving the left hand forward, example motions are blended to realize the controlled key pose, and the synthesized action is performed. It is also possible to adjust the primary key pose by controlling the weights of the principal components when the action is being executed. We chose to use the primary key pose for action control, because it is important to represent the style of the action when it being executed . And it is

(a) During pose control

(b) During action control

Figure 2: *Screen shots of our prototype.*

impossible to control multiple poses, even with a large number of DOFs.

So that we can control various types of actions, a number of action models were constructed in advance. The current action model is selected by moving the left hand in the up-down and right-left directions. In general, the principal components depend on the type of action; PCA must be applied to a set of example poses for the same type of action. Therefore, separate action models were constructed for each type of action. Each action model was mapped into the model selection space in advance. The current action model is selected using the position of the left hand.

The control parameters for our method are summarized in Table 2. Although it is possible to capture the bending/extending angles of five fingers for each hand, we used only three fingers (the index, middle and ring fingers). This is because it is difficult to move the little finger independently from the ring finger, and it is hard to measure the movement of the thumb using the depth-camera.

Figure 2 contains screen shots of our prototype. The controlled parameters and the model selection space are shown on the left and right side on the screen for monitoring. However, this information is not necessary when using our interface. There are six models in the model selection space. The blue dot represents the current position in this space. When an action model is selected, the primary key pose synthesized from the current blending weights is visualized on the screen as a red stick figure.

As an alternative mouse-based interface, the user can control the parameters by dragging the sliders, select the model selection parameter by clicking a point on the model selection space, and exe-

cute the action by double clicking. We compared our hand motion interface with this alternative mouse-based interface in our experiments.

3.3 System Overview

Our system consists of two stages: set up and run time. During the set up process, one pose control model and a number of action control models are constructed by applying PCA to example poses from given example motions. A pose control model is constructed from a number of example standing poses. As explained above, we construct separate models for each type of action. For constructing an action control model, we need a number of example motions and their timings for the primary key pose. An example motion can be either a motion capture sequence (a series of poses) or a keyframe animation.

In our experiments, we constructed six action control models: punch, kick, avoid, guard, jump, and walk. Each action model is constructed using two to five example motions. The pose control model is constructed using approximately 15 example standing poses.

During run-time, the control parameters in Table 2 are computed based on the input from the device in Table 1 and are then used to control the poses and actions.

4 Pose Control

In this section, we describe our pose synthesis method, which blends the principal components extracted from a set of example standing poses.

4.1 Feature Vector

To apply PCA to a set of example poses, each example pose must be represented as a feature vector. The definition of a feature vector is important to PCA and the blending of the principal components.

In general, a character's pose is represented by the position and orientation of the root segment (pelvis) and the rotations of all joints. There are several ways to represent joint rotation such as combinations of rotational angles (Euler angles) or a 3 by 3 rotational matrix and quaternion. However, this conventional representation is not suitable for PCA, because independently blending joint rotations can unnaturally change the position of the end-effector (e.g., foot).

In this research, the feature vector consists of the position and orientation of the pelvis, positions and orientations of both feet, swivel angles of the knees, and joint rotational angles for all joints in the upper body. The positions and orientations are represented in the local coordinates of the example action, defined by the position of the base foot and the orientation of the body (as shown in Figure 3). The joint rotations are represented by their rotational angles.

The local coordinates of the example motion are defined so that its origin matches the position of the base foot, and its z-axis matches the body orientation. The body orientation represents the direction that the character is facing, and is sometimes different from the pelvis direction [Oshita 2008]. The base foot and body orientation are specified manually for each example motion. The y-axis is always the up vector, and the x-axis is the cross product of the z- and y-axes. We use these local coordinates to keep the positions of the feet during posture synthesis. Previous work that applied PCA to character poses [Shin and Lee 2006; Ishigaki et al. 2009; Liang et al. 2009; Numaguchi et al. 2011] discarded the global position and orientation of poses and processed them separately. However, it is important for our method to control poses and realize actions that include global translations and rotations such as walk, step, and jump.

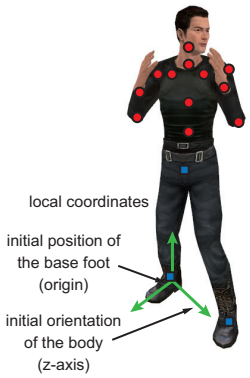

Figure 3: *Feature vector representation. The positions and orientations are represented by the local coordinates which are defined by the initial position of the base foot and the initial orientation of the body. A blue rectangle represents a position and a read circle represents a rotation (orientation).*

We use rotational angles (Euler angles) for the joint rotations, because other representations (such as matrix and quaternion) cannot be represented by a linear combination of the principal components.

A leg pose is represented by the position and orientation of the feet, and the swivel angle of the knee. The pose is reconstructed from the feature vector using analytical inverse kinematics [Tolani et al. 2000]. For arm poses, it is possible to use either the position or orientation of the hand, or the joint angles in the arm. We tested both implementations, and decided to use the joint angles, because position of the hand did not produce well synthesized poses. We believe that this is because the global hand positions are influenced by the whole body and the arm postures, and the relationships are strongly nonlinear.

In total, the feature vector has 52 DOFs.

4.2 Application of PCA

To apply PCA to a set of feature vectors $\mathbf{f}_i$, we first computed their average feature vector $\mathbf{f}_{average}$. Then, PCA was applied to the differences between the feature vectors and the average feature vector $\bar{\mathbf{f}}_i = \mathbf{f}_i - \mathbf{f}_{average}$.

The PCA is computed as follows. We first calculate the covariance matrix $\mathbf{C}$ of all the feature vectors $\bar{\mathbf{f}}_i$. $\mathbf{C}$ is a $n \times n$ matrix, where n is the dimension of the feature vector. Then, we compute the eigenvectors and eigenvalues of the covariance matrix $\mathbf{C}$. They are

$$\mathbf{P}^{-1}\mathbf{C}\mathbf{P} = \mathbf{D} \tag{1}$$

The matrix $\mathbf{P}$ consists of the eigenvectors $\mathbf{p}_j$, which represent the principal components. The matrix $\mathbf{D}$ is the diagonal matrix of eigenvalues λ_j, which represent the contributions of the principal components. Both $\mathbf{P}$ and $\mathbf{D}$ are $n \times n$ matrices. Because we have six DOFs (three for hand position and three for finger angles) to control the pose, we can control the weights of the six most influential principal components from the n principal components in $\mathbf{P}$. It is known that each kind of action can be effectively reproduced using between four and seven principal components [Safonova et al. 2004]. It is considered that it is sufficient to control approximately

six principal components. In our implementation, Equation 1 is computed using the Jacobi eigenvalue algorithm [Press et al. 1992].

4.3 Pose Synthesis

The feature vector of an output posture $\mathbf{f}$ is synthesized by blending the principal components with given weights $\mathbf{w}$. That is,

$$\mathbf{f} = \mathbf{w}^t\mathbf{P} + \mathbf{f}_{average} \tag{2}$$

As explained in Section 3.1, the input DOF θ_j is such that the hand position ($j = 1 \sim 3$) and finger angles ($j = 4 \sim 6$) are normalized between $-1 \sim 1$. The appropriate scale for each principal component varies. To determine them, we first calculate the sets of weights for realizing each example pose using Equation (2). We use the maximum and minimum weights $w_{max,j}, w_{min,j}$ to determine the range and scale of each principal component. That is,

$$\mathbf{w}_j = \begin{array}{ll} w_{max,j}\theta_j & (if \theta_j \geq 0) \\ -w_{min,j}\theta_j & (if \theta_j < 0) \end{array} \tag{3}$$

As result, the output feature vector $\mathbf{f}$ is computed from $\theta_\mathbf{j}$ ($j = 1 \sim 6$) using Equations (2)–(3). When all the fingers are relaxed, the pose from the average feature vector is synthesized. By moving the hand and bending/extending the fingers, the corresponding principal component is added to the synthesized pose. This allows the user to interactively change the pose.

The character's pose is reconstructed from the output feature vector $\mathbf{f}$, using the coordinates computed from the character's current position and orientation. The character's initial coordinates (initial position and orientation) are given first. They are updated as the character performs actions. For example, when the character walks, steps or jumps, the character's coordinates are updated accordingly.

Some previous methods that used a low-dimensional latent space such as PCA for pose synthesis [Min et al. 2009; Grochow et al. 2004] introduced additional post-processing for correcting the weights. This is because a combination of weights may produce an unnatural pose. However, our method does not use post-processing, because it would limit the synthesized pose to the given example poses. The purpose of our method is to synthesize new poses by adding some deviations (principal components), rather than choosing an existing example pose. If many example poses that include all possible deviations can be prepared, it may be effective to limit the synthesized pose within the example poses. However, this is not practical. Moreover, because the pose modification caused by each principal component is a linear deformation that the user can visualize during control, they can avoid unexpected poses by themselves.

We also did not introduce additional processes to maintain balance, for the same reasons. However, unbalanced poses are sometimes generated. There are various methods for changing a character pose for balancing [Macchietto et al. 2009; Jain et al. 2009]. However, it is difficult to introduce these methods into an interactive pose control system. Although this may be an interesting extension to this work, it is outside of the scope of this paper.

4.4 Assignment of Principal Components

As explained in Section 3.2, the principal components are assigned to each DOF based on the magnitude of their influence on the pose deformations. Let $\mathbf{C}(\mathbf{f})$ be the function that computes the center of mass of a pose $\mathbf{f}$. The maximum displacement of the center of mass for each principal component is

$$\mathbf{C}_j = \mathbf{C}(\mathbf{f}^{+j}) - \mathbf{C}(\mathbf{f}^{-j}) \tag{4}$$

where $\mathbf{f}^{+j}$ and $\mathbf{f}^{-j}$ are computed using $\theta_i^{+j} = 1(i = j), 0(i \neq j)$ and $\theta_i^{-j} = -1(i = j), 0(i \neq j)$, respectively. Among the extracted principal components, the one that results in the largest movement of the center of mass $\mathbf{C}_j$ in the x-axis direction is assigned to the x-axis position of the user's right hand. The same mapping is applied to the y- and z-axes positions. Among the rest of the principal components, the three that represent the largest movements are assigned to three fingers of the right hand. Because there is no criterion for this mapping, we simply assign the three principal components from the index finger to the ring finger.

5 Action Control

Action control model is constructed for each type of action. The primary key pose is synthesized by using the same method with the pose synthesis. Action synthesis is done by blending the example motions to realize the controlled primary key pose.

5.1 Application of PCA

A number of example motions must be given to construct an action model. The timing for the primary key pose must be specified for each example motion. PCA is applied to the primary key poses. The extracted principal components are assigned to the fingers of the left hand. In the same way as pose control, we use the principal components that represent larger movements of the center of mass. By bending and extending the fingers, the user can deform the primary key pose.

5.2 Action Synthesis

During action synthesis, the character's pose (feature vector) is computed by blending the poses (feature vectors) from the example motions with the weights $\mathbf{x}$ determined by the controlled primary key pose.

$$\mathbf{f}(t) = \sum_{i=1}^{m} x_i \mathbf{f}_i(t) \tag{5}$$

where $\mathbf{f}_i$ is the pose (feature vector) from i-th example motion at the normalized time t, and m is the number of example motions.

Let $\mathbf{w}_i$ be the weights of the principal components for generating the i-th key pose. The relationship between the weights of examples $\mathbf{x}$ and the weights of the principal components $\mathbf{w}$ is represented by

$$\{\mathbf{w}_0\mathbf{w}_1...\mathbf{w}_n\}\mathbf{x} = \mathbf{w} \tag{6}$$

$$\mathbf{W}\mathbf{x} = \mathbf{w} \tag{7}$$

Only the controllable elements are considered for the weights of the principal components. Therefore, the weights (n') have three dimensions. Assuming that $m > n'$, the weights of examples $\mathbf{x}$ are computed from the weights of principal components $\mathbf{w}$, using a pseudo inverse matrix $\mathbf{W}^+$. That is,

$$\mathbf{W}^+\mathbf{w} = \mathbf{x} \tag{8}$$

Because $\mathbf{W}$ is constant, $\mathbf{W}^+$ is only computed once.

When an action starts and finishes, there is a transition between pose and action synthesis. To realize smooth transitions, we must blend two poses from the pose and action synthesis.

5.3 Action Execution

The action execution time is controlled by moving the user's hand forward. When the forward-backward hand position (z-axis) is between $-1 \sim 0$, the action time is zero and the action is not performed. When it is between $0 \sim 1$, the action time is controlled and the action is executed. The z position represents the normalized action time t between $0 \sim 1$, where $t = 0$ is the initial pose of the action and $t = 1$ is the terminal pose. During execution, the pose at time t is generated by blending example motions using Equation (5).

However, if the example pose at a given normalized time t is simply synthesized, if the hand moves quickly, the action may be executed too quickly and look unnatural. Because the duration of the action varies depending on the type of action, it is difficult for the user to know the appropriate speed for the hand movement. Therefore, the time specified by the user's input is used as the target time. The execution time is advanced or reversed toward the target time at a reasonable speed.

The user can normally control the execution time by using their hand position to advance or reverse it. However, at certain points in an action, reversing will appear unnatural. For example, consider a kick. It is possible first lift the foot up and down. However, after the kick action starts, it is difficult to pull the foot back. To prevent this, we specified the time of no return for each action in advance. After the action execution time passes this time, it is automatically advanced and cannot be changed by the user.

As explained in Section 5.2, during the initial and terminal periods of an action, the output poses from the pose and action controls are blended to create a smooth transition. As explained in Section 4.1, the character's coordinates are updated when the action is completed.

5.4 Action Model Selection

The user can select the current action model from a number of prepared action models. The action models are placed on the model selection space (two-dimensional space). The user can specify a position on the space $\mathbf{s} = \{s_0, s_1\}$ using the x and y (left-right and up-down) hand positions. The action model is selected based on the specified position.

We assigned a radial basis function (RBF) to each action model. RBFs are commonly used for computing blending weights from given parameters [Park et al. 2002]. The distances from each action model d_k are computed using the RBF. The radius of an RBF can be specified manually or automatically tuned based on the distance to the nearby action models. The distance from an action model is computed using

$$d_k = R(\mathbf{s}) = B(|\mathbf{s} - \mathbf{s}_k|/r_k) \tag{9}$$

where $\mathbf{s}_k$ and r_k represent the position and radius for each action model, respectively. The action model with the largest distance (d_k) is selected as the current action model.

When the action is being executed, the system does not allow changes to the current action model. When a different model is selected, the synthesized poses from the action execution and the selected model are blended to create a smooth transition during the terminal period of the execution.

6 Results and Discussion

In this section, we present our experimental results and some discussion. The accompanying video includes a demonstration of our interface. Examples of the principal components are shown in Figure 4 and 5.

6.1 Evaluation

We asked six undergraduate and graduate students in computer science to participate in our evaluation. They have a basic knowledge of computer animation. We also exhibited our system at a university event, and demonstrated it to over 100 visitors (high school students and parents). Some of them observed the demonstration, and some tried the system for a short time. We collected feedback from all these sessions.

Basically, the participants recognized that our interface has the unique advantage that the user can deform the character pose using their fingers. However, they also commented that some training would be required before using the interface.

The effectiveness of our interface for controlling poses depends on the extracted principal components. To use our interface, the user must remember the mappings between each DOF and the corresponding pose deformation. As shown in Figure 4, in our experiments, effective principal components were extracted and we realized an intuitive mapping. In this case, the principal component for the x position also included the movements along the z-axis, because there were no principal components that only represented movements along the x-axis. Nevertheless, the three principal components for the hand position approximately represented movements in each direction, and we considered that the mapping was intuitive. For the three principal components represented by finger angles, one was used to open/close the arms, one adjusted the foot and pelvis positions, and one rotated the back. These were also considered effective for altering the standing pose. Figure 5 shows the principal components for the punch action control model. We cannot associate them with positions (directions) in the action control. However, because we extracted the effective principal components and there are three principal components, the user can easily understand how to alter the primary key pose of the action using the principal components.

We found that fingers resulted in an intuitive control interface, but were inaccurate because of problems in the sensors and the abilities of the users. It is difficult to independently operate multiple fingers. When a finger is moved, there are small unintentional movements in the adjacent fingers. This limitation sometimes caused unexpected pose deformations. Some finger movement combinations are difficult to perform, for example, bending one finger and extending the next. We also noticed that it is difficult to keep fingers at certain angles. It is easy to move the hands to select the models and execute actions. However, the hand can sometimes move unintentionally, which changes the primary key pose. It was a little difficult to keep the hand still, and this created fatigue in the arms.

In our experiments, the system has six action models. Our framework can handle any number of action models, although we did not test this. However, if there are too many action models, it is difficult to select one from the model selection space because the recognizable range of hand positions is limited.

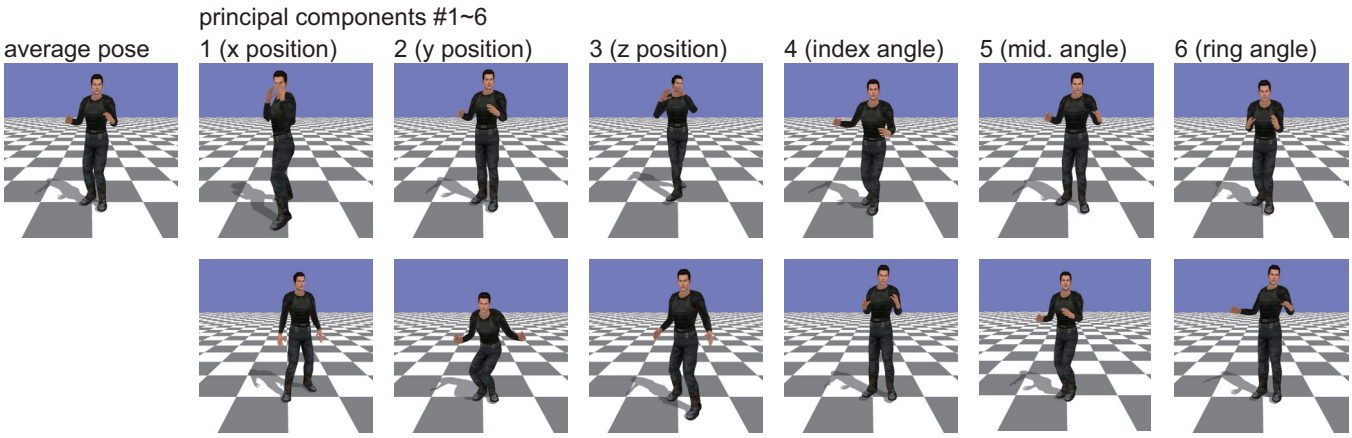

Figure 4: *An example of the principal components for the pose control model. The poses from the maximum and minimum weights of each principal component are synthesized.*

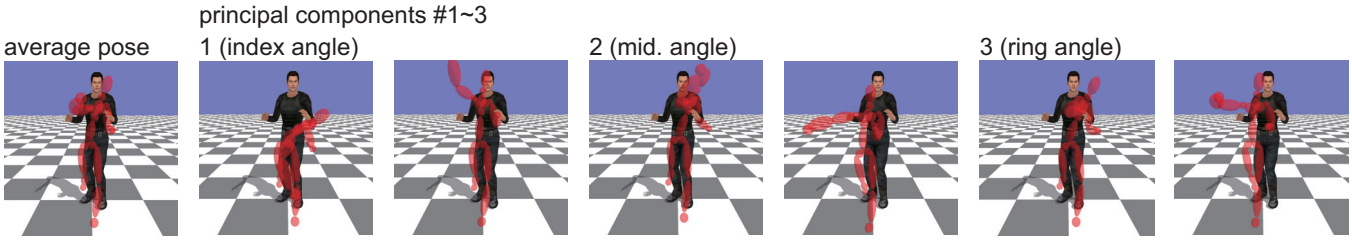

Figure 5: *An example of the principal components for the punch action control model. The primary key poses from the maximum and minimum weights of each principal component are synthesized.*

6.2 Comparison

We compared our interface with the alternative mouse-based interface using sliders, as described in Section 3.2. If the user simply wants to select an action model from the model selection space and execute an action, the mouse-based interface was much easier to use. However, it is obviously difficult to control multiple weights at the same time, and interactively deform the character pose. There is a trade-off between the DOFs and ease of use.

We conducted a user test to show the efficiency of our interface compared with a conventional interface. Because there is no alternative common interface that can interactively control a character's actions with deformations, it is difficult to compare our interface for action control tasks. Therefore, we evaluated our interface for posing control tasks. We compared our interface using hands or mouse (using sliders) with a conventional mouse-based pose editing interface. In this conventional technique, a joint rotation or position can be manipulated using handles displayed on the screen, as shown in Figure 6. The blue and red spheres on the character represent controllable end-effectors and joints, respectively. This kind of interface is commonly used in commercial animation systems such as Maya, 3ds Max, and Softimage . Although only one body part can be controlled at a time, its position and rotation can be controlled precisely.

The six subjects described in Section 6.1 participated in the user test. During the experiment, the subject was shown a target pose and he or she was asked to make the same pose using one of the interfaces. The system determines that the task is completed when the maximum distance between the positions of all joints in the target and controlled poses is below a threshold (10 cm). Different types of target poses were generated with random weights using our pose and action synthesis methods. Some target poses were generated from the pose control model, whereas some target poses were generated from the primary key poses of the action control models. We used random weights for one to three DOFs, and zero weights for the other DOFs. The results were measured for each type of target pose. When using our interface, if the primary key pose of an action was displayed as the target pose, the subject needed to execute an action after selecting an appropriate action model and controlling the primary key pose. When the pose editing interface was used, there were no differences between the target poses from the pose and action control models. We recorded the time required to reach each target pose from the initial pose. Each subject used all three interfaces in different orders, after a short training period.

The average times for the posing tasks are shown in Table 3. For all target poses, our hand interface was more efficient than our mouse interface and the pose editing interface. These results indicate that principal components are an efficient way of controlling poses with a given set of sample poses. They also demonstrate that hands and fingers can be used to effectively control the weights of the principal components, and are more efficient than a mouse and sliders.

7 Conclusion

In this paper, we proposed an interactive motion control interface based on hands. Using their hands and fingers, the user can interactively deform the character's pose by controlling the blending weights of the principal components. The user can execute various actions by altering the primary key pose. Although our interface requires some training, the user can control the character in a way that is not possible using conventional interfaces.

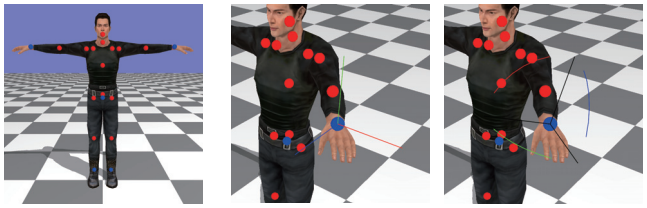

Figure 6: *Mouse-based interface for comparison. The position and orientation of a joint or end-effector can be controlled by dragging a handle.*

Table 3: *Results from the user test. Average time for making each target pose.*

target type	our interface (hand)	our interface (mouse)	pose editing (mouse)
pose (1 DOF)	15.0 s	19.7 s	95.1 s
pose (2 DOF)	22.3 s	22.4 s	63.9 s
pose (3 DOF)	19.2 s	24.6 s	95.0 s
action (1 DOF)	28.0 s	32.4 s	98.0 s
action (2 DOF)	30.5 s	36.5 s	138.9 s

New technologies for sensing hand motions [Leap Motion ; Intel] are becoming common. We believe that our interface can be a good application of these technologies.

Acknowledgment

This work was supported in part by a Grant-in-Aid for Scientific Research (No. 24500238) from the Japan Society for the Promotion of Science (JSPS).

References

3GEAR SYSTEMS. 3gear devkit. http://www.threegear.com/.

5DT. Data-glove. http://www.5dt.com/.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3, 522–531.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. Spatial keyframing for performance-driven animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005*, 253–258.

INTEL. Perceptual computing sdk. http://software.intel.com/en-us/vcsource/tools/perceptual-computing-sdk.

ISHIGAKI, S., WHITE, T., ZORDAN, V., AND LIU, C. K. 2009. Performance-based control interface for character animation. *ACM Transactions of Graphics (SIGGRAPH 2009) 28*, 3, Article No. 61.

JAIN, S., YE, Y., AND LIU, C. K. 2009. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics 28*, 1, Article No. 10.

KOMURA, T., AND LAM, W. C. 2006. Real-time locomotion control by sensing gloves. *Computer Animation and Virtual Worlds 17*, 5, 513–525.

LASZLO, J., NEFF, M., AND SINGH, K. 2005. Predictive feedback for interactive control of physics-based characters. *Computer Graphics Forum (EUROGRAPHICS 2005) 24*, 3, 257–265.

LEAP MOTION. http://www.leapmotion.com/.

LIANG, X., LI, Q., ZHANG, X., ZHANG, S., AND GENG, W. 2009. Performance-driven motion choreographing with accelerometers. *Computer Animation and Virtual Worlds (CASA 2009) 20*, 2- 3, 89–99.

MACCHIETTO, A., ZORDAN, V., AND SHELTON, C. R. 2009. Momentum control for balance. *ACM Transactions of Graphics (SIGGRAPH 2009) 28*, 3, Article No. 80.

MICROSOFT. Kinect. http://www.xbox.com/en-US/kinect.

MIN, J., CHEN, Y.-L., AND CHAI, J. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics 29*, 1, Article No. 9.

NATURAL POINT. Optitrack. http://www.naturalpoint.com/.

NIK, I. N. I., AND OSHITA, M. 2012. Evaluation of a data-glove-based interface for motion selection and style control. In *IIEEJ Image Electronics and Visual Computing Workshop (IEVC) 2012)*, 6.

NUMAGUCHI, N., NAKAZAWA, A., SHIRATORI, T., AND HODGINS, J. K. 2011. A puppet interface for retrieval of motion capture data. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2011*, 157–166.

OKADA, Y. 2003. Real-time motion generation of articulated figures using puppet/marionette metaphor for interactive animation systems. In *3rd IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP03)*, 13–18.

OSHITA, M., SENJU, Y., AND MORISHIGE, S. 2013. Character motion control interface with hand manipulation inspired by puppet mechanism. In *12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI) 2013*, 131–138.

OSHITA, M. 2005. Motion control with strokes. *Computer Animation and Virtual Worlds 16*, 3-4, 237–244.

OSHITA, M. 2006. Motion-capture-based avatar control framework in third-person view virtual environments. In *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE) 2006*, Article No. 2.

OSHITA, M. 2008. Smart motion synthesis. *Computer Graphics Forum (Pacific Graphics 2008) 27*, 7, 1909–1918.

OSHITA, M. 2012. Multi-touch interface for character motion control using example-based posture synthesis. In *International Conference on Computer Graphics, Visualization and Computer Vision (WSCG) 2012*, 213–222.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *ACM SIGGRAPH Symposium on Computer Animation 2002*, 105–111.

POLHEMUS. Fastrak. www.polhemus.com.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA.

PRIMESENSE. Carmine. http://www.primesense.com/.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–40.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3, 514–521.

SHIN, H. J., AND LEE, J. 2006. Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds 17*, 3-4, 219–227.

SHIRATORI, T., AND HODGINS, J. K. 2008. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics (SIGGRAPH Asia 2008) 27*, 5, Article No. 123.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Transactions of Graphics (SIGGRAPH 2004) 23*, 3, 424–431.

TOLANI, D., GOSWAMI, A., AND BADLER, N. I. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models and Image Processing 62*, 5, 353–388.