Character Motion Control Interface with Hand Manipulation Inspired by Puppet Mechanism

Masaki Oshita* Yuta Senju Syun Morishige

Kyushu Institute of Technology

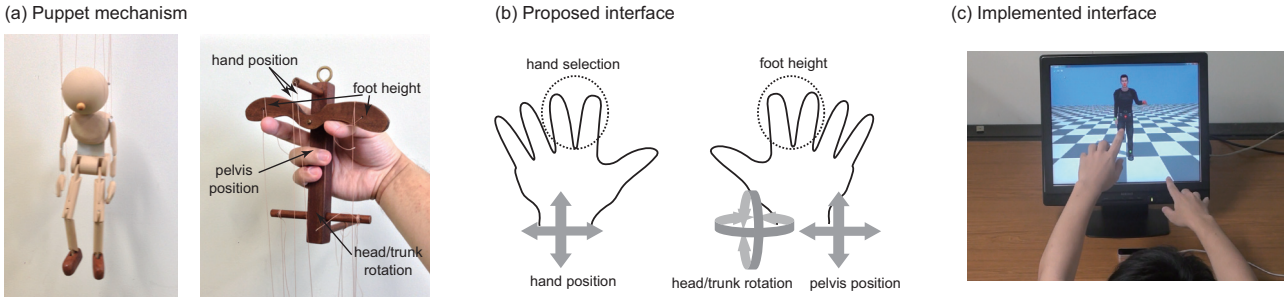

Figure 1: *Puppet-like motion control interface: (a) puppet mechanism, (b) proposed interface inspired by the puppet mechanism, and (c) picture of the implemented interface with a depth-camera-based hand motion sensing device.*

Abstract

In this paper, we propose an interactive motion control interface with hand manipulation. Using fingers and hands, the user can simultaneously control a large number of degrees of freedom. Our interface is inspired by the control mechanism of a real puppet, which is suspended by about 10 strings attached to different parts of the puppet's body and the controller. The puppeteer can move the puppet in a variety of ways by holding and moving the controller with the right hand and manipulating the strings with the left hand. Our interface was designed based on this puppet mechanism. The character's pelvis translation and head/body rotation are controlled by the user's right hand, while the character's legs are controlled by the right hand fingers. The character's arms are controlled by the user's left hand and fingers. We have developed a method to control a character's motion based on this interface design. In this research, we used a depth-camera-based hand motion sensing device to implement our prototype. Using such a camera-based device obviates the need for the user to wear any device. However, the chosen technology has some limitations in hand motion sensing. Our method has been designed to work well even under these constraints. In this paper, we present our interface design and implementation, as well as the experimental results and a discussion thereof.

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, Data Glove, Hand Manipulation

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

1 Introduction

Interactive control of character motion is important in real-time applications such as computer games and communication using avatars. However, the freedom of 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, among others, the user cannot do anything other than select pre-defined actions by pushing a button, touching the screen, or performing a gesture. Even though the user may sometimes wish to change a character's pose directly to make it perform an original motion, this is not possible using conventional interfaces.

In this paper, we propose an interactive character motion control interface using hand manipulation. With fingers and hands, the user can simultaneously control a large number of degrees of freedom. Our interface is inspired by the mechanism of a real puppet (Figure 1(a)), which is suspended by about 10 strings attached to different body parts and the controller. The puppeteer can move the puppet in a variety of ways by holding and moving the controller with the right hand and pulling some strings with the left hand. We designed our interface based on this puppet mechanism (Figure 1(b), (c)). The character's pelvis translation and head/body rotation are controlled by the user's right hand, while the character's legs are controlled by the right hand fingers. The character's arms are controlled by the user's left hand and fingers. We have developed a method to control a character's motion based on this interface design.

In this research, we make use of a depth-camera-based hand motion sensing device, Leap Motion [LeapMotion], to implement our prototype. Using such a camera-based device obviates the need for the user to wear any device. However, the selected technology has some limitations in hand motion sensing. Our method is designed to work well even under such constraints. In this paper, we present our interface design and implementation, as well as experimental results and a discussion thereof.

Because of its mechanism, there are restrictions on the range of possible motions for a real puppet; not all human motions are feasible. Our interface has the same limitation. Nevertheless, the motions that are possible can still be useful in actual applications. The target applications of our interface include computer games, communica-

tion via avatars, and interactive animation creation by interactive performance, amongst others.

In addition, in the same way that a puppeteer with a real puppet needs to practice before performing, our interface requires some training as well. We consider this to be an acceptable trade-off for the ability to control a character freely, something that is difficult to realize using conventional interfaces.

The remainder of this paper is organized as follows. Section 2 reviews related work, while Section 3 presents our interface design. Section 4 describes our implementation including system overview, input data processing, user interface, and motion control. Section 5 presents experimental results and a discussion thereof. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Motion Control Interface using a Data-Glove

Several character motion control interfaces using data-gloves for hand manipulation have been developed. Some of these introduced puppet-like control mechanisms, but they differ from our interface.

Okada [2003] introduced a metaphor for puppet-like control and virtual strings. He mapped each finger to the movement of a specific body part in an upward direction. He also introduced physical constraints of gravity and ground contact to generate natural motion. However, unlike a real puppet, a character's hands can be controlled only in an upward direction. This restriction stems from the fact that only one hand is used in this interface, and extension to two hands was not considered. As a result, the possible range of motion is quite limited.

Nik et al. [2009] used position-based control and mapped the user's hand positions to the end-effector positions of the character. Unlike a real puppet, the fingers are only used to switch the controlled effector. The interface in this study has restricted control of the pelvis and upper body. The authors introduced additional modes to control an arm and a leg together and to activate predefined motions using gestures. However, once again the possible range of motion is limited.

There are various other data-glove-based motion control interfaces that mimic different puppet control mechanisms. Komura and Lam [2006] controlled walking motion using a data-glove. They used a pre-created walking motion sequence and mapped the movement of two fingers to one-dimensional locomotion time through a training process. This interface is restricted to locomotion control. Nik and Oshita [2012] also used discrete hand positions to select a motion and finger angles to specify the motion blending parameters. However, with these interfaces, a user can control only the abstract parameters directly and not the pose of the character.

2.2 Motion Control Interface using Other Devices

There are also other motion control interfaces using various input devices that are manipulated by the user's hands. Taking inputs from such devices can be a way of controlling a character.

Some researchers [Lockwood and Singh 2012][Sugiura et al. 2009] have used multi-touch devices to detect fingertip positions from finger walking gestures to extract motion control parameters. These systems are restricted to the control of leg movements. Moreover, these multi-touch devices can capture fingertip positions only on contact with the device.

Oore [2002] used two bamboo sticks with a magnetic tracking sensor to control the body parts of a character. The bamboo stick can be used as a proxy of a body part such as the trunk, arm, or leg. However, because at most two body parts are controlled at the same time, it cannot be used for interactive motion control. Numaguchi et al. [2011] used a puppet as an input device for a motion search system. By holding the body parts of the puppet and making it mimic a motion, a set of similar motions can be searched in a database. Although it is difficult to make a puppet perform human-like natural motion in this way, it is possible to extract sufficient features to search for the intended motion. This system is not suited to interactive motion control.

Physical sensors such as the Wii remote (acceleration sensors) [Nintendo] can be used with a gesture recognition technique. By performing a specific gesture, the user can select an action from certain predefined actions. However, this type of gesture-based interface is simply a substitute for conventional interfaces like the game pad and keyboard. Controlling the character's pose and motion and performing actions according to the user's own style are not possible.

Other research combines physics simulation with control of a limited number of degrees of freedom. Using a mouse to control a few joints, Laszlo et al. [2005] generated full-body motion with physics simulation. Shiratori and Hodgins [2008] used Wii remotes to determine a few parameters for physics-based controllers for several actions such as walking, running, and jumping. By using physics simulation, physically plausible motion can be generated and controlled. However, since physics-based controllers must be designed in advance for each type of action, realizing a variety of different actions and styles is difficult.

Recently, multi-touch input devices such as tablet computers have been used for interactive character motion control. Krause et al. [2008] applied inverse kinematics (IK) to a character model based on multi-touch inputs for animation. Kipp and Nguyen [2010] proposed a system to control one arm and hand of a character using a multi-touch interface by changing several parameters to blend the arm and hand postures. These systems are restricted to the control of a single part of the body. Oshita [2012] applied a statistics-based IK (style-based IK [Grochow et al. 2004]) to control the whole body of the character. However, the synthesized poses are limited to certain given example poses. Oshita [2013] also combined multi-touch interface and motion control models to control a character's pose and motion without being limited to predetermined examples. One of the problems of multi-touch interfaces is that the inputs are 2-dimensional, and thus an additional mechanism is needed to control 3-dimensional positions and orientations of body parts.

Recently, low-priced markerless motion capture devices such as the Microsoft Kinect [Microsoft] have become available. Using such devices, the character's full body motion can be directly controlled [Ishigaki et al. 2009][Oshita 2006]. However, the device requires a larger workspace and to perform highly dynamic actions, the user must actually perform the action, which is difficult for untrained users. Although depth-camera-based hand motion sensing devices [LeapMotion ; Intel] have also become available and are currently receiving much attention from developers. To our best knowledge, there is no other system that uses such a device for character motion control yet.

3 User Interface Design

In this section, we describe our motion control interface with hand manipulation. First we explain the puppet mechanism and the characteristics of the input device. Then we present the interface design based on these.

3.1 Puppet Mechanism

Figure 1(a) illustrates the control mechanism of a standard puppet. The puppet is suspended by about 10 strings each of which is attached to a different body part and the controller. The puppeteer can move the puppet in a variety of ways by holding and moving the controller with the right hand and pulling some strings with the left hand.

Global translation of the puppet is controlled by moving the controller held in the right hand. By tilting the controller, the head and trunk of the puppet are also tilted. When the controller is tilted slightly, only the head of the puppet is tilted. Then tilting the controller more causes the trunk of the puppet to be tilted as well. In this way the head and trunk can be controlled simultaneously in a coordinated fashion.

There is a movable piece on the controller, either edge of which is connected to the strings from each of the puppet's legs, and this can be tilted right and left by moving the fingers on the right hand. By moving this piece, the puppet's leg is lifted upwards. Because this piece moves like a seesaw, when one of the puppet's legs is lifted, the other leg is automatically lowered. In this way both legs are controlled simultaneously in a coordinated way. However, this mechanism allows the puppet's legs to move only in one direction (upwards). While this is a limitation of the puppet mechanism, it is helpful in simplifying the control of a puppet.

To control the puppet's arms, the puppeteer holds the string attached to the hands with the left hand fingers and pulls the string in the direction that the puppet's arms should move. The string attached to each of the puppet's hands is connected through the controller; by moving a string towards the controller, the respective arm is lifted while the other arm is lowered. In this way both arms are controlled in a coordinated way.

This puppet mechanism has been well designed so that multiple body parts are controlled in a coordinated way using simple manipulation. Owing to this simple yet sophisticated mechanism, even a novice can quickly learn to control a puppet and make it perform simple motions. A demonstration of a puppet controlled by a novice is included in the accompanying video.

Although there are some variations, the control mechanism of a standard puppet is as described above. Our interface is designed according to this mechanism. More detailed explanations of puppet control mechanisms, methods and variations thereon, as well a history of puppetry can be found in the literature [Currell 1999; Bell 2000; Currell 2005].

3.2 Input Devices

Our interface works with any hand motion sensing device including wired data-gloves [5DT] and magnetic or optical tracking devices [Fastrak ; NaturalPoint]. However, as mentioned in Section 1, in this research we used a depth-camera-based hand motion sensing device, Leap Motion [LeapMotion], which was recently released. We designed our interface and method based on the characteristics of this device.

Depth-camera-based technology, which has been widely used in recent devices [LeapMotion ; Intel ; Microsoft], has both advantages and disadvantages to its use. The advantages are that the user does not have to wear any device and this approach is more accurate than standard or stereo cameras. The disadvantages are that it can suffer from recognition/tracking errors and occlusion problems.

The raw inputs from a depth-camera are 3-dimensional point clouds. Usually the depth-camera application programming interface (API) has the ability to recognize/track hands and fingers. By using such an API, the positions of the hands and fingers can be acquired once they have been recognized and tracked by the depth-camera [LeapMotion ; Intel].

Nevertheless, there is no guarantee that all hands and fingers will be recognized even if they are in front of the camera. If a finger is bent, it cannot be distinguished from the hand, and therefore cannot be recognized. For the same reason, thumbs are often not recognized even when they are extended. Although the other four fingers are relatively recognizable, when the fingers are close to each other, they may not be distinguishable even when extended. With respect to hand position and orientation, when the palm faces the camera, its orientation is reliable. However, when the palm is positioned perpendicular to the camera, its orientation becomes unstable and unreliable. In addition, when one hand is in front of the other, the hand behind is occluded and cannot be recognized. Although these APIs can estimate fingertip positions and hand position and orientation, they are not that accurate and stable because the hand position and orientation are influenced by finger movements. On the other hand, fingertip positions are relatively accurate if the fingers are extended and clearly recognized by the depth-camera.

These observations were obtained through experimentation with the depth-camera device and API. Based on these, in order to use finger movements for control, only those angles of a finger between fully extended and only slightly bent are reliable. Moreover, hand orientation is reliable only within a certain range. The positions of the right and left hands must be controlled in separate workspaces to avoid occlusion. These characteristics were considered when designing our interface.

3.3 Proposed Interface

Based on the puppet mechanism and the characteristics of the input device described above, we designed our interface for controlling a character's motion as shown in Figure 1(b). In this subsection, we explain our control interface in detail.

3.3.1 Control of Pelvis, Head, and Trunk

The user's right hand position is assigned to the translation of the character's pelvis, while the orientation of this hand is assigned to the rotation of the character's head and trunk. Similar to the puppet mechanism, slight rotation of the user's hand causes only the head of the character to rotate. With greater rotation of the user's hand, the trunk of the character is also rotated, causing the character to bend, extend, and tilt its trunk as well.

The character is controlled based on the relative translation and rotation of the hand with respect to its initial position and orientation. The translation and rotation are properly scaled for application to the translation and rotation of the character's pelvis considering the different sizes of the workspaces. Using relative positions and orientations prevents the character making a large movement when the hand is recognized.

If the user's hand is not recognized by the sensing device, the corresponding body parts automatically revert to their natural state. The pelvis position returns to a balanced position near the center of the support polygon, while the head and trunk orientations revert to their natural orientations.

3.3.2 Control of Legs

The index and middle fingers of the user's right hand are assigned to controlling movement of the character's legs. When the finger is bent, the character's corresponding foot is lowered and remains on

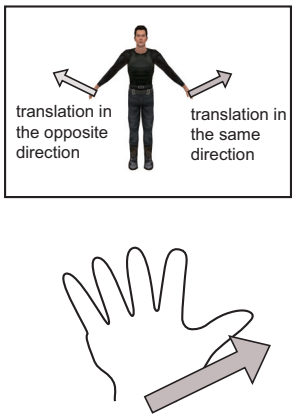

Figure 2: *Symmetrical control of both arms.*

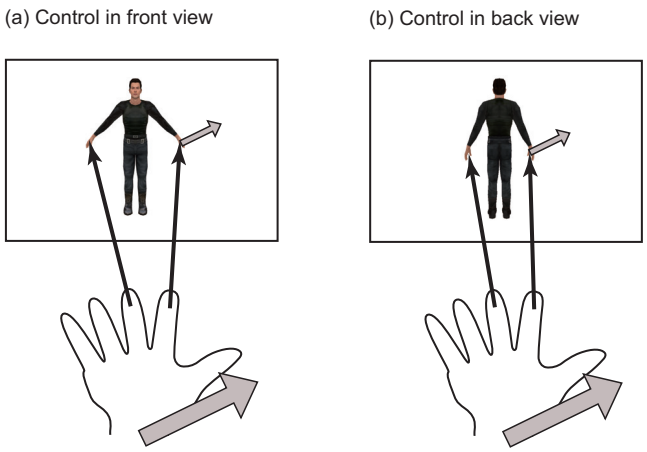

Figure 3: *Lateral direction of control.*

the ground. When the finger is extended and the fingertip moved upwards, the character lifts the corresponding foot upwards.

While a foot is flying, its horizontal position is controlled by the position of the pelvis so that the foot comes right below its hip joint. Although only the vertical position of the foot is controlled by the right hand finger, its horizontal position can be controlled indirectly by translating the pelvis. If the finger is not recognized by the sensing device, the character simply keeps its foot on the ground or lowers it if the foot was lifted.

When both the index and middle fingers are extended, only one foot is lifted because lifting both feet at the same time is impossible. In such a case, the foot corresponding to the finger that is extended most is lifted.

3.3.3 Control of Arms

The user's left hand and fingers are assigned to controlling movement of the character's arms. The user's left hand position is used to control the position of one of the character's hands in the same way that pelvis translation is controlled by the right hand position. The arm pose is automatically determined based on the controlled hand position. The index and middle fingers of the user's left hand are used to select which arm is controlled. When one of the fingers is extended, the corresponding arm is controlled. When the finger is bent, control of the corresponding arm terminates and the arm is automatically lowered.

If we were to allow the user exclusive selection of only one arm in the same way that the legs are controlled, only one arm could be controlled simultaneously as is the case with a puppet. However, unlike the legs, it is possible to raise both arms at the same time. To allow such motion, our interface allows the user to move both arms together. When either the index or middle finger is extended, the corresponding hand is moved based on the position of the left hand. When the other finger is also extended, while keeping the first finger extended, the other hand is moved based on the position of the left hand, but symmetrically as shown in Figure 2. For example, when the character's left hand is moved to the left, the right hand is symmetrically moved to the right. We introduced this symmetrical control mode because both arms often move symmetrically in human motions. Although our interface is restricted to this kind of simultaneous control of both arms and does not allow independent control of each arm, this additional mode allows an extended range of motions.

3.3.4 Lateral Direction of Control

The mapping between the right-left side of the user's hands and that of the translation of the controlled body parts is changed based on the view as shown in Figure 3. If the camera is behind the character and both the user and the character are facing the same direction (Figure 3(b), they are mapped in the same direction. For example, if the hand moves to the left, the corresponding body part moves to the left. The index and middle fingers of the user's left hand are assigned to the right and left arms of the character, respectively. On the other hand, if the camera is in front of the character and the character faces the opposite direction (Figure 3(a)), they are mapped in a symmetrical direction. For example, when the hand moves to the left, the corresponding body part moves to the right. The index and middle fingers of the user's left hand are assigned to the left and right arms of the character, respectively. If the camera is to the side of the character, the previous mapping of front or back is retained until the view is changed either to the front or to the back.

While controlling a puppet, the puppeteer always faces the same direction as the puppet. However, during character motion control in interactive applications, the user can change the view and sometimes may want to see the character from the front. Therefore, we introduced this switch in lateral direction to facilitate control even when the view is in front of the character.

4 Implementation

In this section, we describe our implementation of the proposed interface. The system is divided into three modules: input data processing, user interface, and motion control as illustrated in Figure 4. The input data processing module processes inputs from a hand motion sensing device. The user interface module interprets the processed input data and determines the control parameters for the motion control module. The motion control module updates the character's pose based on the given control parameters and the previous state of the character.

The input, output and intermediate data are summarized in Table 1. The first group ('input') represents input data from the hand motion sensing device. In this step, hands and fingers are not identified. Recognition data for the hands and fingers are retrieved. The second group ('processed data') represents the results of input data processing for use in the user interface module. The third group ('control parameter') represents the intermediate parameters from the interface module to the motion control module. The fourth

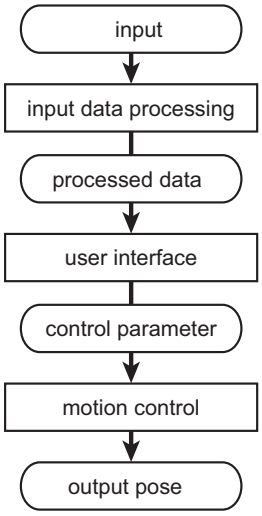

Figure 4: *System structure.*

Table 1: *Input, output, and intermediate data at each step.*

Step	Data
input	position and orientation of recognized hands
	position of recognized fingertips for each hand
processed data	existence of data for each hand and finger
	position and orientation of each hand (if visible)
	flexion angle of each finger (if visible)
control parameter	character's pelvis position
	character's head and trunk orientation
	character's foot position (foot height)
	character's hand position (hand height)
output	character's pelvis position and orientation
	character's joint rotations

group ('output') represents the output character pose, which is a common form of representing a character's pose. The character's skeletal model is given to the system in advance.

4.1 Input Data Processing

This process takes inputs from a hand motion sensing device and generates the necessary data for the user interface module.

4.1.1 Hand and Finger Identification

As explained in Section 3.2, the depth-camera device and its API can recognize the hand positions and orientations and the fingertip positions. However, it cannot determine which hand (right or left) or fingers (thumb, index, middle, ring, or little) have been found.

Identification of hands is not that difficult. Assuming that the device is placed in the center in front of the user, it is easy to determine whether the hand seen is the right or left hand based on the initial lateral position of the hand.

Identification of fingers is relatively difficult. To do this, the fingertip position is converted to a relative position in the local coordinates of the corresponding hand. Based on its lateral position, it is possible to estimate which fingers they are. However, because there is no guarantee that all fingers have been recognized and because fingers can be moved in a lateral direction, identification can be complicated.

To solve this, we manually define probability functions for the five fingers based on their lateral positions. We represent each probability function by a radial basis function, defined by its center position and radius. When a number of fingers (at most five) are recognized, the probabilities of each recognized finger being the thumb, index, middle, ring, or little finger are computed. Of the combinations, the one with the highest probability is repeatedly chosen until all the fingers have been identified.

4.1.2 Calculation of Hand Position and Orientation

As explained in Section 3.2, the relative position and orientation of the hand with respect to its initial position and orientation are used in our user interface.

However, if the position and orientation at the time that the hand is recognized by the depth-camera are used as the initial position and orientation, it is possible that recognition of the hand will be lost when the hand is moved away from the camera even only slightly. To prevent this from happening, we implemented a short delay before the hand position and orientation are initialized after being recognized by the depth-camera. In our implementation, the interval is set to 1.0 sec.

Moreover, when hand movement terminates, if the hand is simply moved away from the depth-camera range, the hand motion of being moved away is also captured and the character's body is moved unintentionally. To prevent this from happening, when the hand is moved in a backward direction at a higher speed than the given threshold, the hand movement is discarded. In our implementation, the threshold is set to 0.2 m/sec.

4.1.3 Calculation of Finger Flexion Angle

When a finger is recognized, its normalized finger flexion angle $\theta = -1.0 \sim 1.0$ is calculated based on the relative up-down position of the fingertip in the local coordinates of the hand. If the angle is -1.0, the finger is bent to the maximum, whereas an angle of 1.0 denotes that the finger is extended to the maximum. An angle of 0.0 means that the finger is relaxed and slightly bent.

The finger angle θ_{finger} is computed by scaling the up-down position of the user's fingertip $\mathbf{p}'_{u\_finger,y}$ based on the given parameters. In our implementation, we apply a simple linear scaling. A fingertip position between y_{range_min} and $y_{neutral\_min}$ is mapped to $\theta_{finger} = -1.0 \sim 0.0$. A fingertip position between y_{range_max} and $y_{neutral\_max}$ is mapped to $\theta_{finger} = 1.0 \sim 0.0$. Scaling parameters y_{range_min}, $y_{neutral\_min}$, $y_{neutral\_max}$, and y_{range_max} are manually set for each finger. Alternatively, they can be acquired through some interactive calibration process.

4.1.4 Application of Low-Pass Filters

As explained in Section 3.2, the hand position and orientation estimated from the depth-camera and its API are not that stable, whereas the fingertip positions are relatively stable. Therefore, we apply low-pass filters to the hand position and orientation and finger angles. In our implementation, we apply a moving average filter, which calculates the average of a series of n data. We also experimented with a more advanced filter such as a Kalman filter, but no significant difference was found.

4.2 User Interface

Based on the processed inputs, the pelvis position, head/body orientation, hand positions, and foot positions are determined. All output

positions and orientations are specified in the global coordinates of the character's space.

For the pelvis and hand positions, the user's relative hand position in the global coordinates of the user's workspace must be converted to a position in the global coordinates of the character's workspace. Since the acquired positions of the user's hands are specified as the relative translation from their initial positions, by applying the relative translation to the initial position of the character's corresponding body part, the position of the character's body part is computed as follows:

$$\mathbf{p}_{c\_pelvis} = \mathbf{M}\mathbf{S}\mathbf{p}_{u\_rhand} + \mathbf{p}_{c\_pelvis}^{initial} \quad (1)$$

where $\mathbf{p}_{c\_pelvis}$ is the position of the character's body part (the pelvis position in this case), $\mathbf{p}_{u\_rhand}$ is the user's relative hand position (the right hand position in this case), $\mathbf{M}$ is the transformation matrix from the user's workspace to the character's global coordinates computed from the character's initial position, and $\mathbf{S}$ is a diagonal scaling matrix to map the movements of the user's hand to the movements of the character's body part. Because the character's hand can be moved in a larger space than the character's pelvis, a larger scaling parameter is applied to the character's hand. $\mathbf{p}_{c\_pelvis}^{initial}$ is the initial position of the specific character's body part when control is initiated.

To obtain the foot positions, the foot height is computed from the flexion angles of the user's fingers. The horizontal foot position starts from its initial position $\mathbf{p}_{c\_rfoot}^{initial}$ (in the case of the right foot). As the foot is lifted, its horizontal position moves closer to the target position $\mathbf{p}_{c\_rfoot}^{target}$ determined based on the current pelvis (hip joint) position. Both $\mathbf{p}_{c\_rfoot}^{initial}$ and $\mathbf{p}_{c\_rfoot}^{target}$ are points on the ground. Thus, the foot position $\mathbf{p}_{c\_rfoot}$ (the right foot in this case) is computed as follows:

$$\mathbf{p}_{c\_rfoot} = (0, \theta_{rfoot} h_{foot}, 0) + (1-s)\mathbf{p}_{c\_rfoot}^{initial} + s\mathbf{p}_{c\_rfoot}^{target} \quad (2)$$

where $\theta_{rfoot}(0.0 \sim -1.0)$ is the finger flexion angle of the corresponding finger and h_{foot} is the maximum height of the foot. In our implementation, h_{foot} is set to 90% of the leg length of the character and $s(0.0 \sim 1.0)$ is the parameter for controlling its horizontal position. s increases as θ_{rfoot} increases. Once s reaches 1.0, it is fixed. As a result, the foot is put down on $\mathbf{p}_{c\_rfoot}^{target}$.

To obtain the head and trunk orientations $\mathbf{q}_{c\_head} and \mathbf{q}_{c\_trunk}$, respectively, the user's hand orientation $\mathbf{q}_{u\_rhand}$ is distributed to them. First the head/trunk orientation in the character's global coordinates is computed by applying the transformation. Then the rotation is decomposed into three rotational angles around each axis of the character. When the rotational angle is within a given range, it is applied only to the head. If it exceeds the given range, the rotational angles over the range are applied to the trunk. In our implementation, we manually set maximum rotation angles (yaw, altitude, roll) for the head and trunk.

4.3 Motion Control

There are various ways of controlling a character based on given constraints such as pelvis position, head and body orientations, and hand and foot positions. The main contribution of this research is our interface design inspired by the puppet control mechanism. Any motion control method can be combined with our interface.

In this research, we introduced a kinematics-based simple motion control method, which computes a character's pose using inverse kinematics based on the given constraints, which is similar to an existing method [Oshita 2013].

Given any combination of pelvis position, head and body orientations, and hand and foot positions, the character's pose is updated according to the following steps.

1. If the pelvis position is given, the character's pelvis is set accordingly.
2. If the trunk and head orientations are given, they are applied to the corresponding back and neck joints.
3. If the foot and hand positions are given, inverse kinematics is applied to each limb to compute the joint rotations of the limb.

In addition, to generate natural poses and motion we introduce additional controls such as tracking, relaxing, balancing, and self-collision avoidance. The tracking control realizes smooth motion by adjusting the given target positions and orientations considering velocity. The relaxing control automatically lowers the arms and legs of the character when there is no input to either hand and foot. The balancing control moves the pelvis so that the center of mass of the character remains on the support polygon of the character. Self-collision avoidance prevents a body part from penetrating into the character. In particular, the arm is moved outwards if it penetrates the character's trunk until the penetration is resolved. These tracking, relaxing, and self-collision avoidance control models can reproduce the behavior of a real puppet, arising from physics and its mechanical model. Such human-like control models and algorithms have been developed in the computer animation field [Oshita 2013; Macchietto et al. 2009; Jain et al. 2009]. Further details of the implementation of our control models can be found in [Oshita 2013].

Using physics simulation is another way of generating motion. By constructing models of a virtual puppet, controller, and strings using a physics simulation engine, puppet motion can be simulated. However, we did not adopt this approach because the physical mechanism of a puppet is fairly complex and constructing a working model is not easy. Although such an approach may be an interesting future study, it is beyond the scope of this paper.

5 Results and Discussion

In this section, we present our experimental results and a discussion thereof. The accompanying video includes a demonstration of our interface, as well as a real puppet performance for comparison. Images of our interface are also depicted in Figure 5. In these images, inputs from the hand motion sensing device are visualized on the screen.

Ten undergraduate and graduate students, some of whom had basic knowledge of computer animation, were asked to participate in our experiment. Subjects were requested to try both a real puppet and our interface and provide us with feedback. Having first explained how to use both interfaces, we asked them to use each interface in a different order.

Basically our interface worked as intended. However, there was a problem with errors in tracking and hand pose and orientation estimation. Since the finger flexion angle calculation relies on the result of estimating the hand position and orientation, this calculation is also unstable. This is particularly problematic because the user does not know if both hands and all fingers have been recognized during control. As shown in Figure 5 and the accompanying video, we displayed spheres on the character's body parts when they were being controlled so that the user would know if their corresponding inputs had been recognized by the system.

Figure 5: *Images from a demonstration of our interface. Inputs from the hand motion sensing device are visualized on the screen. The rectangles represent the user's hands (red: right hand, blue: left hand). The spheres represent the user's fingertips (green: finger bent, red or blue: finger extended). The spheres on the character are controlled points (red: controlled by right hand or fingers, blue: controlled by left hand or fingers, green: not controlled, but the corresponding finger is recognized).*

The subjects commented that manipulation of the right hand was particularly difficult because three different tasks are assigned to the right hand: pelvis translation, head and trunk orientation, and leg translation. Basically the same procedure is required for a puppet and the subjects also had difficulty in using their right hands to manipulate the puppet. However, they felt that it was more difficult with our interface, probably because of the errors in tracking and estimation. The recognizable range of hand positions and orientations of the depth-camera-based device is narrower. Moreover, since estimation of the hand position and orientation and finger flexion angles are dependent on each other, controlling them separately is difficult.

As explained in Section 3, the possible range of motion for the puppet is limited and all kinds of human motions cannot be reproduced. Our interface basically has the same restriction. Complex motions such as fighting motions (e.g., punching and kicking), acrobatic movements (e.g., flip jumps and rolls), and dancing are not possible. Feasible motion includes all kinds of leg, arm, and upper body movements while standing. However, since puppets are used in theater performances and for expressing characters' feelings, this range of motion is quite adequate for a number of applications, including online communication using avatars and interactive animation creation.

One of the limitations of our interface is implementing walking motion. Owing to the small recognizable range, which is smaller than the workspace of a typical puppet, it is difficult to make a character walk around in a large space, although it is possible to perform various small steps in a small space. Moreover, although the trunk direction can be changed slightly, the character's orientation is fixed in our interface and cannot be changed. We may need to introduce an additional interface for such walking control.

Regarding the comparison with the real puppet, even though the design of our interface is similar to that of the puppet and the test subjects understood the usage of both quickly, the experience of using each was different. One reason for this is that there is no physical feedback with our interface, whereas the puppet provides physical feedback via the controller and strings. Such physical feedback is useful while controlling. However, some subjects commented that our system is capable of a wider range of motion for hand positions and trunk orientation, because there are no restrictions due to strings and thus the position and orientation of the hand can be controlled freely.

In terms of required training, it seemed that the subjects took more time getting used to our system, since they needed some practice in adapting to the unstable recognition and learning the recognizable range of hand movements.

Another difference is view control. Because of its mechanism, when controlling a puppet the puppeteer always views it from the top, and it is difficult to see its motion. On the other hand, with our interface, the user can see the character and its motion from any viewpoint. In fact, some of the subjects preferred this approach.

Unfortunately, we were unable to find a subject who was a professional puppeteer to test our interface. We expect that such a subject would be able to utilize our interface to a greater extent than an average subject could. Further user studies with professional puppeteers is left as a future work.

Our future work also includes enhancing our interface and method to solve the problems described in this paper. Combining our interface with other types of hand motion sensing devices is also a possible avenue for future work. To make it possible to perform a wider range of motions, a large change from the puppet mechanism may be required. The primary motivation for this project, however, was to implement puppet-like control, even if the range of motions was limited to that of a real puppet. Enhancing the interface and method without deviating too much from the puppet mechanism is a possible future direction for this research.

6 Conclusion

In this paper, we proposed an interactive motion control interface with hand manipulation. We designed our interface inspired by the puppet control mechanism. We described our interface design and implementation. Recently, depth-camera-based hand motion sensing devices [LeapMotion ; Intel] have become available and are currently receiving much attention from developers. We believe that our interface may be a good application for these new 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

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

BELL, J. 2000. *Strings, Hands, Shadows: A Modern Puppet History*. Detroit Inst of Arts.

CURRELL, D. 1999. *Puppets and Puppet Theatre*. Crowood Press.

CURRELL, D. 2005. *Making and Manipulating Marionettes*. Crowood Press.

FASTRAK. Polhemus fastrak. www.polhemus.com.

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

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.

KIPP, M., AND NGUYEN, Q. 2010. Multitouch puppetry: Creating coordinated 3d motion for an articulated arm. In *ACM International Conference on Interactive Tabletops and Surfaces 2010*, 147–156.

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

KRAUSE, M., HERRLICH, M., SCHWARTEN, L., TEICHERT, J., AND WALTHER-FRANKS, B. 2008. Multitouch motion capturing. In *ACM International Conference on Interactive Tabletops and Surfaces 2008*, 2.

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.

LEAPMOTION. http://www.leapmotion.com/.

LOCKWOOD, N., AND SINGH, K. 2012. Finger walking: Motion editing with contact-based hand performance. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2012*, 43–52.

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.

NATURALPOINT. Optitrack. www.naturalpoint.com/optitrack/.

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 2012 (IEVC 2012)*, 6.

NIK, I. N. I., ISHIGURO, K., AND OSHITA, M. 2009. Real-time character motion control using data gloves. In *International Conference on Computer Games, Multimedia and Allied Technology (CGAT) 2009*, 307–314.

NINTENDO. Wii remote. http://www.nintendo.com/wii/.

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.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. A desktop input device and interface for interactive 3d character animation. In *Graphics Interface 2002*, 133–140.

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 2006*, Article No. 2.

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.

OSHITA, M. 2013. Multi-touch interface for character motion control using model-based approach. In *Cyberworlds 2013*, 8 pages.

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.

SUGIURA, Y., KAKEHI, G., WITHANA, A., FERNANDO, C., SAKAMOTO, D., INAMI, M., AND IGARASHI, T. 2009. An operating method for a bipedal walking robot for entertainment. In *ACM SIGGRAPH Asia 2009 Emerging Technologies*, 79–79.