ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2006 (ACE 2006), Hollywood, U.S.A., June 2006.

Motion-Capture-Based Avatar Control Framework in Third-Person View Virtual Environments

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka, Japan
+81-948-29-7718
oshita@ces.kyutech.ac.jp

ABSTRACT

This paper presents a motion-capture-based control framework for third-person view virtual reality applications. Using motion capture devices, a user can directly control the full body motion of an avatar in virtual environments. In addition, using a third-person view, in which the user watches himself as an avatar on the screen, the user can sense his own movements and interactions with other characters and objects visually. However, there are still a few fundamental problems. First, it is difficult to realize physical interactions from the environment to the avatar. Second, it is also difficult for the user to walk around virtual environments because the motion capture area is very small compared to the virtual environments. This paper proposes a novel framework to solve these problems. We propose a tracking control framework in which the avatar is controlled so as to track input motion from a motion capture device as well as system generated motion. When an impact is applied to the avatar, the system finds an appropriate reactive motion and controls the weights of two tracking controllers in order to realize realistic and also controllable reactions. In addition, when the user walks in position, the system generates a walking motion for the controller to track. The walking speed and turn angle are also controlled through the user's walking gestures. Using our framework, the system generates seamless transitions between user controlled motions and system generated motions. In this paper, we also introduce a prototype application including a simplified optical motion capture system.

Categories and Subject Descriptors

I.3.6 **[Computer Graphics]**: Methodology and Techniques - *Interaction Techniques*; I.3.7 **[Computer Graphics]**: Three-dimensional Graphics and Realism - *Animation*; I.3.7 **[Computer Graphics]**: Three-dimensional Graphics and Realism - *Virtual Reality*.

Keywords

Avatar, Motion Control, Interface, Motion Capture, Virtual Reality.

1. INTRODUCTION

Motion capture technology has been used in many areas, including movie making, motion analysis, and computer games. Although the most current main uses are for offline processes, motion capture is expected to become a major control device for computer entertainment. Using motion capture devices, a user can directly control the full body motion of a character in virtual environments. For such kinds of applications, we believe that the third-person view is most suitable (Figure 1). Most current virtual reality applications employ the first-person view with which the users can feel as if they are in the virtual environments. However, this view is not quite suited for some kinds of applications, such as fighting games in which the movements of the characters are important. In the real world, we can feel the movements of our own body and physical interactions with other people and objects through our senses. However, we cannot do such things in current virtual reality systems since we can only obtain visual information or very limited physical information from some haptic device. By using the third-person view, however, in which the user watches himself as an avatar, the user can sense such movements and interactions with other objects visually. This is the reason why we consider the third-person view to be more suitable for some entertainment applications. However, there are still a few fundamental issues to be dealt with in order to realize such kinds of applications. One problem is that it is difficult to

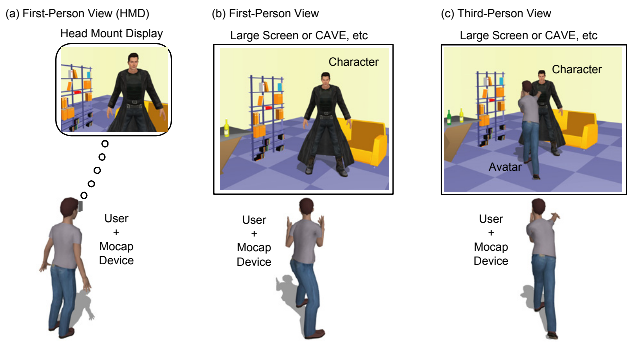

Figure 1. First-person view (a) (b) and third person view (c).

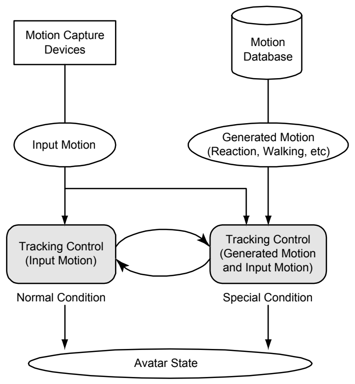
Motion Capture Devices
Motion Database
Input Motion
Generated Motion (Reaction, Walking, etc)
Tracking Control (Input Motion)
Tracking Control (Generated Motion and Input Motion)
Normal Condition
Special Condition
Avatar State

Figure 2. Tracking control approach. Under normal conditions the avatar tracks a captured motion. Under conditions in which multiple system generated motions are required, the avatar can track each of them seamlessly.

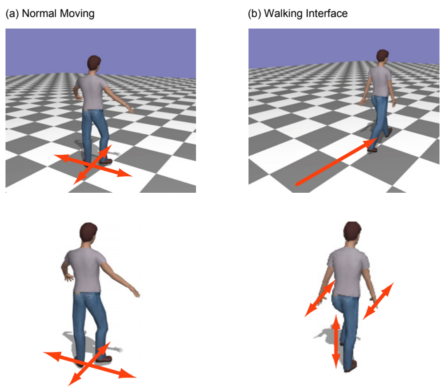

Figure 3. Walking interface. Basically the user can freely move around in the mocap area. If the user performs a walking gesture without horizontal movement, the system then generates a walking motion to be tracked.

realize physical interaction from the environment to the avatar because the avatar is controlled based on input from a motion capture device and because physical influences cannot be applied to the user in the real world. Another problem is that it is difficult for the user to walk around virtual environments because motion capture areas are very small compared to the size of virtual environments. Some existing virtual reality systems use an additional device such as a gamepad or a 3D pointing interface for navigation, which can be very difficult to use in motion-capture-based systems.

This paper presents a motion-capture-based control framework that is suitable for third-person view virtual reality applications. Our method solves the avatar control problems described above. In our system, an avatar is controlled so as to track an input motion from a motion capture device instead of just playing the input motion (Figure 2). When a physical interaction is applied to the avatar (e.g., a collision with other characters), the system automatically generates a reactive motion (e.g., falling down to the ground), and the avatar then tracks the reactive motion. Even during the reactive motion, the user can control the avatar slightly.

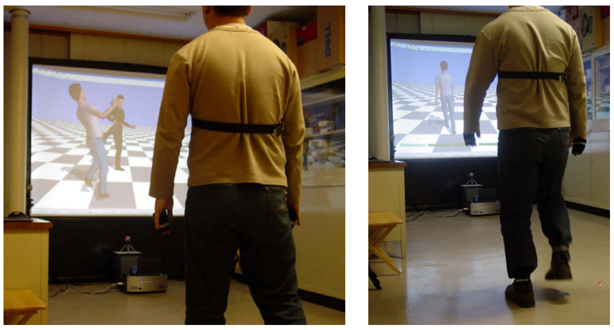

Figure 4. Pictures of the prototype system.

With this tracking approach, the system can generate seamless transitions between user controlled motions and system generated reactive motions. The user can experience the physical interactions visually through the reactions of the avatar in the third-person-view virtual environments without any special force-feedback devices. In addition, we propose a walking-gesture-based navigation interface (Figure 3). In this interface, when the user walks in position, in other words, performs a walking motion in one place without moving horizontally, the system generates a walking motion. The walking motion is controlled through the user's walking gestures. For example, if the user moves his legs and arms quickly, the system will generate running motion based on the motion speed. The user can control both the speed and direction of the motion very easily and intuitively. We developed a prototype of this system including an optical motion capture system (Figure 4).

The rest of this paper is organized as follows. Section 2 introduces related work and discuses the advantages of our work. Section 3 provides an overview of the proposed system framework and the basic data representations. Sections 4, 5, and 6 explain the system's tracking control, reactive motion control, and walking control, respectively. Section 7 then introduces the experimental results and evaluates the proposed method. Finally, Section 8 concludes this paper and describes future directions of research.

2. RELATED WORK

A number of studies have investigated motion-capture-based character control [14]. However, in most of these systems, the input motion is directly used to control the character's motion. Unfortunately, physical interaction between the character and the environment is therefore not considered. Hasegawa et al. [4] proposed a system similar to ours. They also used the third-person view and a tracking control system. However, they always tracked the input motion and did not generate automatic reactions. Therefore, the character may move unnaturally when a large impact is applied to the character. Hämäläinen et al. [3] used a profile third-person view in a martial arts combat game. They also used video images of the users captured in real-time with a camera instead of a combination of a motion capture system and computer generated human figure. While this is an interesting method, because of its approach the avatar cannot react to impacts from its enemies. Their system also addressed the motion area problem and exaggerated the horizontal movements of the users by simply scaling the horizontal position of the user's image.

Although this approach expands the user's reach, the users still cannot move around wide virtual environments.

Generating realistic reactive motion to a given physical impact is one of the most difficult challenges in computer animation. Since the human control mechanism is very complex and has yet to be well modeled, it is very difficult to generate realistic reactions. Some researchers have tackled this problem by using a tracking control approach and/or a motion database in a way similar to our methods. A method proposed by Zordan et al. [18] searched for appropriate reactive motions from a database based on the results of physics-based simulations after impacts were applied to a virtual character, and a method proposed by Komura et al. [7] searched for reactive motions based on the amount of impact that generated resulting reactions using a tracking control system that took into account the searched motion data and momentum variations during the reactive motions. Oshita et al. [12] generated reactive motions that did not use any reaction data and instead used a heuristic dynamic control that altered the output of simple tracking control while taking into account balance and stress controls. These methods were designed for the generation of realistic reactions without any user control. On the other hand, the control method proposed in this paper controls the avatar based on both input motion from the motion capture device and searched reactive motion from a database using an approach similar to those in [7] and [18]. Although we currently use searched motion from the database without any modification as the target motion of the tracking control, more advanced methods such as those described in [7] and [18] can also be used with our system.

There are many control interfaces that use an intuitive input device different from motion capture [16]. However, such interfaces do not consider interactions between the character and the environment, and it is also difficult to use them in our system as additional interfaces for navigation, etc., as our system needs the full body control of the user.

There are many approaches available for motion recognition [2]. However, the existing methods simply recognize what kind of motion is being executed. In our system, therefore, because walking motion should be controlled through the user's movements, we employ a fuzzy based state detection and state tree for computing walking speed.

3. SYSTEM FRAMEWORK

An overview of the proposed control framework is shown in Figure 6. The avatar state is updated based on the input motion from the motion capture in each step of the simulation. A tracking controller computes control accelerations based on the current state and the given motion using a PD servo so that the avatar tracks the input motion. When a large impact is applied to the avatar or when the user performs a walking gesture, the coordinator generates a target motion and controls the blending weights of the output of the tracking controllers. The integrator updates the avatar state based on the accelerations and the blending weights. The integrator also changes the avatar state when it collides with another character or an object.

The avatar basically tracks the input motion from the motion capture device. When a large impact is applied to the avatar, it mainly tracks a reactive motion that is searched for from a reaction database. When the user performs a walking gesture, a

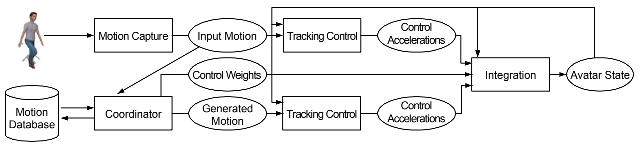

Figure 6. System overview.

walking motion is generated by using a motion blending technique, and the avatar accordingly tracks the walking motion. Even during reactive or walking motion, depending on the conditions the avatar is still controllable through the input motion. The details of this process are explained in the following sections. The coordinator determines the blending weights of the output of the two tracking controllers.

In our system, the tracking controllers compute accelerations based on the current state of the avatar and the target motion. Although in standard physics-based simulation systems [5][8] a tracking controller computes joint torques and a simulator then computes the resulting joint accelerations from the joint torques, we chose to use angular accelerations for controllability. In torque based control, the gain parameters of the tracking controller should be carefully tuned for the individual characters and target motions. They produce unstable motion especially when the system mixes multiple outputs from a number of tracking controllers. Therefore, we decided to control angular accelerations in a way similar to that described in [12] or [15]. Although torque based controls are generally thought to produce more physically valid motions, this is not necessary in our system since the input motion capture data and reactive motions in the database are based on human motion and therefore are considered to be already physically valid. Thus we use tracking control for blending multiple controls but not for producing physically correct motions.

3.1 Human Model

Virtual characters, including avatars and autonomous characters, are modeled as articulated figures. The human model of our prototype system has 20 segments and 19 joints (Figure 5), which is a typical model used for computer games.

The posture of a human model is represented as follows.

$$\mathbf{M} = \{\mathbf{p}_{\mathrm{root}}, \mathbf{q}_{\mathrm{root}}, \mathbf{q}_0, \cdots, \mathbf{q}_i, \cdots, \mathbf{q}_{n-1}\} . \quad (1)$$

This consists of the position $\mathbf{p}_{\mathrm{root}}$ and orientation $\mathbf{q}_{\mathrm{root}}$ of the

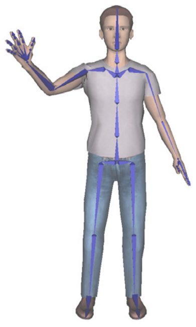

Figure 5. Human figure model.

pelvis (root) and the rotation of all joints $\mathbf{q}_i\,(i = 0 \ldots n-1)$, where n is the number of joints. In our method, pelvis orientation and joint rotation are represented by rotation vectors. The orientation of the vector represents the rotation axis in the local coordinates system of the parent segment, and the length of the vector represents the rotation angle around the rotation axis.

The velocities and accelerations of a human model are also represented in the same way as that described in (1). In our framework, the avatar state consists of posture $\mathbf{M}$ and velocity $\dot{\mathbf{M}}$. The tracking controller outputs angular accelerations $\ddot{\mathbf{M}}$. Changes in the posture and velocity of the avatar are computed from the angular accelerations using numerical integration on each simulation step. When the avatar collides with other characters or objects, the changes of velocity $\dot{\mathbf{M}}$ are computed by solving a linear system [9].

3.2 Optical Motion Capture System

Any kind of motion capture system can be used with our framework. For our prototype system, we have developed a simplified optical motion capture system. Current motion capture systems require some markers or sensors to be attached to the user. Although markerless motion capture [1] is desirable for our target applications, it is still difficult to capture human motion precisely and in real-time without using markers or sensors. Currently the most popular motion capture systems are optical-based systems. However, they are very expensive and require a lot of time to set up the required number of markers. Therefore, we have developed a simplified optical system which uses a relatively small number of markers. We use three OptiTrack FLEX3 infrared cameras [10] from Natural Point. Our system acquires only the position or both the position and orientation of the primary body parts. Inverse kinematics [14] is then used to compute the full body posture. For example, we computed the joint rotations of an arm from the position and orientation of the shoulder and the position of the hand. The orientation of the hand can be estimated from the elbow angle. In addition, the rotations of the elbow around the axis from the shoulder to the hand can be determined from an experimentally determined constant value [17].

4. AVATAR CONTROL

As explained in the previous section, we use a PD controller to compute tracking accelerations based on the current state of the avatar and the target state from the target motion.

$$\ddot{\mathbf{q}}_i = k\left(\mathbf{q}_{\text{target},\,i} - \mathbf{q}_i\right) + d\left(\dot{\mathbf{q}}_{\text{target},\,i} - \dot{\mathbf{q}}_i\right), \tag{2}$$

where $\ddot{\mathbf{q}}_i$ is the output joint rotational acceleration of the i-th joint, $\mathbf{q}_i, \dot{\mathbf{q}}_i$ are the current joint rotation and rotational velocity, respectively, and $\mathbf{q}_{\text{target},i}, \dot{\mathbf{q}}_{\text{target},i}$ are the target rotation and rotational velocity, respectively, that are acquired from the target motion. The gain parameter k and damping parameter d are set manually. Since we use angular accelerations instead of joint torques, we can use the same k and d for all joints. To control joint rotations in the PD controllers, we treat rotations and rotational velocities as rotation vectors.

In addition to joint rotations, the pelvis and end-effectors that come into contact with the ground are also controlled. The system controls the position and orientation of the body parts and applies inverse kinematics to correct the joint rotations on the limb. For the inverse kinematics, we use the same analytical method [14] used in our optical motion system (Section 3.2). The position of the pelvis (end-effectors) $\mathbf{p}_j$ is controlled as follows.

$$\ddot{\mathbf{p}}_j = k_{\text{pelvis}}\left(\mathbf{p}_{\text{target},j} - \mathbf{p}_j\right) + d_{\text{pelvis}}\left(\dot{\mathbf{p}}_{\text{target},j} - \dot{\mathbf{p}}_j\right). \tag{3}$$

The orientation of the pelvis is controlled in the same way used for joint control shown in equation (2).

5. REACTIVE MOTIONS FOR IMPACTS

To realize realistic reaction to various kinds of impacts, we introduced a few types of control schemes. The controllable factors in the proposed framework include the choice of reactive motion and the weights for the two tracking controls of input motion and reactive motion.

Our method is based on the observation of real human movements, taking into consideration the fact that reactive human motions can be divided into three phases: active control, active control with protective steps, and passive control with falling down (Figure 7). When an impact is applied to a human, he first attempts to maintain his balance by moving his arms and upper body (active control phase). If the impact is small, he can recover to a stable state. If the impact is relatively large and he cannot recover during this phase, protective steps must be taken to keep from falling down (active control with protective steps phase). He may then recover after a few steps, or he may fall down. While falling down, he can barely control his body because the falling motion is basically caused by gravity (passive control phase). To sum up, there are three types of reactive motions depending on the number of phases involved (Figure 7 (a)-(c)).

Our control scheme realizes these reactive motions. First, the type of reaction is determined based on the avatar's state after an impact. Then, the weight values for the input motion and reactive motion are controlled during each phase based on the determined reaction type. Figure 7 shows the trajectories of the weight values for each reaction type.

When the changes of the avatar state are small after an impact is applied to the avatar (Figure 7 (a)), no reactive motion is used and only a small weight is used for the tracking control of the input motion in order to realize the reaction of the active control phase. When a large impact is applied to the avatar, however, an appropriate reactive motion is searched for from within the reaction database. Based on the chosen reactive motion, the system then determines whether the avatar will recover at the end of the motion (Figure 7 (b)) or fall down (Figure 7 (c)). At first a smaller weight is given to the tracking control of the input motion in the same way as in the case of a small impact. If the avatar is to recover, then the weight becomes larger during the second phase and the avatar goes back to the input motion. On the other hand, if the avatar is to fall down, then the weight becomes smaller during the next phase while the weight for the reactive motion becomes larger. During passive control, the weight for input motion is set to 0 and the avatar tracks the reactive motion.

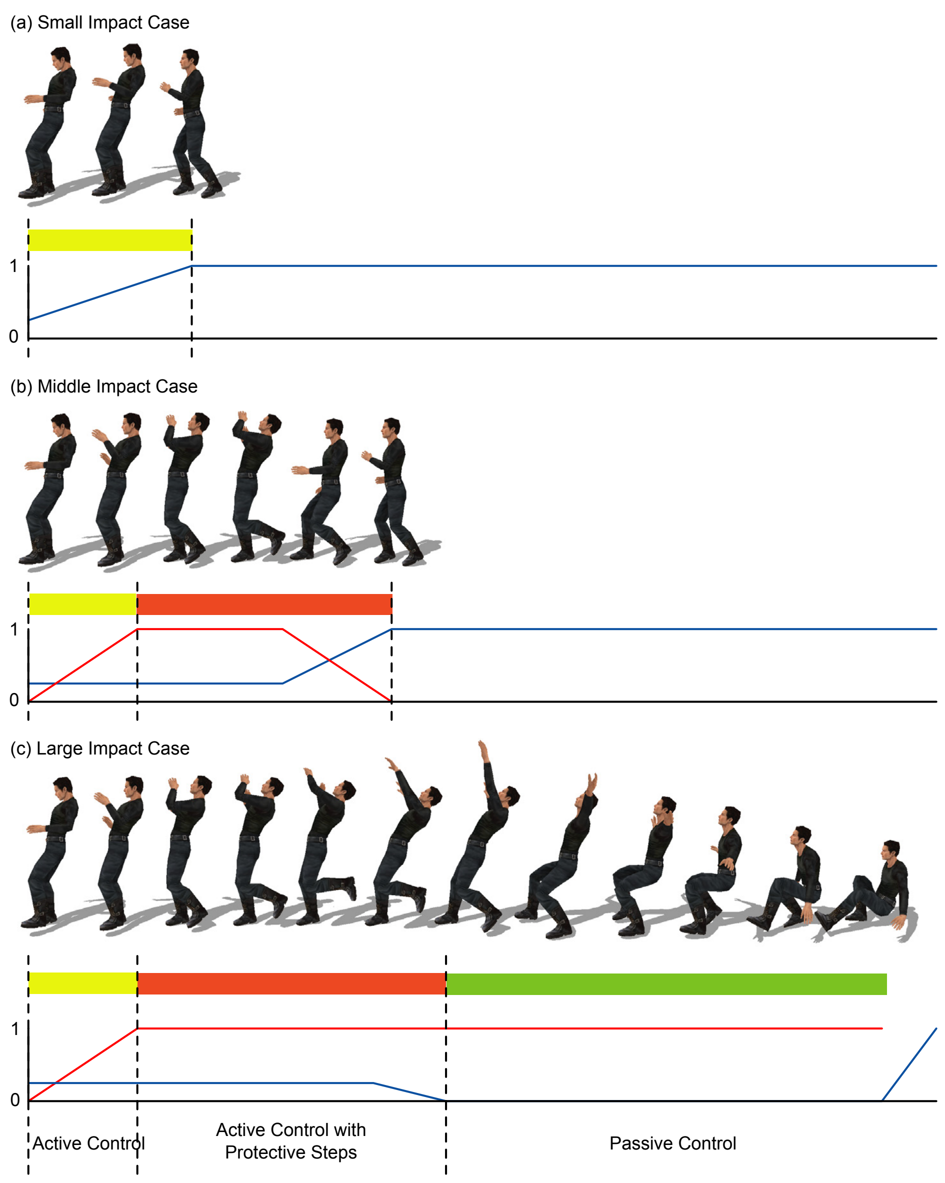

Figure 7. Reactive motions. We assume that reactive motions consist of three phases. Reactive motions are categorized based on in which phase the avatar recovers his balance. The red and blue trajectories show the weights of reactive motion and input motion, respectively.

After the avatar falls down, the avatar's posture and the user's posture may differ. If the avatar tracks the user's pose, an unnatural standing up motion is generated. In such a case, after a reactive motion has finished, a transition motion is executed to allow the avatar to reach a posture that is close to the user's posture. For this reason the avatar again reverts to the tracking control of the input motion. The execution of such transition motion is explained in Section 5.4.

5.1 Reaction Database

A number of reactive motions are stored in a database. When a large impact is applied to an avatar and the avatar loses his balance, the system searches for an appropriate reactive motion from the database and uses it as a target motion for tracking control. To find the best reaction, each reaction motion has an index key which is automatically computed in advance. The index key contains the balance and posture states of the avatar at the time when the reaction began. The direction and speed at which the figure is falling (balance state) have first priority here since these values determine the direction of the reaction. To take both the position and velocity of the center of mass into account, we used the idea of extrapolated center of mass (XcoM) proposed by Hof et al. [6] to evaluate the balance conditions of human movements. The XcoM is based on an inverted pendulum model. The position of XcoM is computed by

$$\mathbf{p}_{\mathrm{XcoM}} = \mathbf{p}_{\mathrm{CoM}} + \sqrt{l/g}\,\dot{\mathbf{p}}_{\mathrm{CoM}}\,, \tag{4}$$

where $\mathbf{p}_{\mathrm{CoM}}, \dot{\mathbf{p}}_{\mathrm{CoM}}$ are the position and velocity of the center of mass, respectively, l is the leg length, and g is the acceleration of gravity. Though here we could use the position and velocity of the center of mass as separate variables and evaluate them with some arbitrary weights, in this case we would have to tune the most appropriate set of weights. Thus we decided to use an established prediction model instead of going through this process.

In addition to the center of mass (XcoM), the posture and velocities of the avatar are also important factors when choosing a reactive motion. However, the use of the position and velocity of all joints or segments is redundant. Therefore, the positions and velocities of only the end-effectors are considered. Since the velocities of a human figure tend to be large when the figure is losing its balance, we cannot ignore the affect of these velocities. Based on the same concept of XcoM, we use equation (4) for the end-effectors also in order to simultaneously take the positions and velocities into account. We call these values the extrapolated end-effectors (XEE), and they are represented in the local coordinates with their origins in the root of the limbs. Further, the front vector is the figure orientation, and the up vector is the world Y-axis. As a result, 15 dimensional vectors are used for the index keys (Figure 8).

The index key of a reactive motion is computed in advance based on the character's state at the time when the magnitude of XcoX becomes largest just after the impact is applied to the character. The system automatically searches the time around the specified first key time within a certain time range (0.2 sec).

5.2 Searching for Reactive Motion

When an impact is applied to the avatar, the system first supposes that the impact is small and simply starts to control the weight of the tracking control without any reaction data, as shown in Figure 7 (a). Next, the balance condition (XcoM and XEEs) is computed at each frame. If the XcoM exceeds the support polygon, then the figure is considered to be starting to lose its balance [6]. The reactive motion that has the closest index key to the balance condition is then used. The support polygon is computed from the faces of the figure model that are in contact with the ground using a convex hull computation [11].

When the figure starts to lose its balance, meaning that the XcoM moves outside the support polygon, a reactive motion is searched for from within the database based on the distance between the index key of each reaction data set and the search key computed from the state of the figure at that particular time. The distance of the *i*-th data in the database is computed as

$$E_i = w_{\mathrm{XcoM}} \left| \mathbf{p}^i_{\mathrm{XcoM}} - \mathbf{p}_{\mathrm{XcoM}} \right| + \sum w_j \left| \mathbf{p}^i_j - \mathbf{p}_j \right|\,, \tag{5}$$

where $\mathbf{p}^i_{\mathrm{XcoM}}, \mathbf{p}^i_j$ are the position of XcoM and the *j*-th end-effectors (hands, foots and head), respectively, and w_{XcoM}, w_j are the weights for XcoM and the end-effectors, respectively. We currently use $w_{\mathrm{XcoM}} = 5.0, w_j = 1.0$ in our implementation. Since the XcoM distance may be small at first even when a large impact is applied to the figure, the system keeps computing the balance condition after the XcoM exceeds the support polygon during a fixed duration. Then the conditions of the largest balance values are used for searching the reaction data.

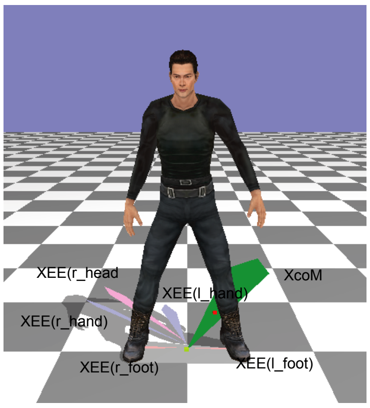

Figure 8. Index keys of a reactive motion. The appropriate reactive motion is searched for based on the eXtrapolated Center of Mass (XcoM) and the Extended End Effectors (XEE).

5.3 Control of Weights

The weight values for the input motion and the reactive motion are determined from the reaction type, the current phase, and the current time, as shown in Figure 7. For each reactive motion, we assign keytimes that indicate the respective segment of each phase in advance. During the first phase, the weights change linearly. During the second phase, the weights change linearly before a fixed duration from the keytime.

In addition to these linear functions, the weights for input motion are changed based on the velocities of the upper body of the input motion. If only the weights in Figure 7 were used and the user did not move his upper body much, the produced motion would become less reactive compared to the original reactive motion. To address this, we multiply the normalized velocities of the hands and head with the weight function when the sum of the velocities is smaller than threshold v.

$$w_{\mathrm{input}}(t) = \begin{cases} f(t) & \mathit{if}\left(\left|\mathbf{v}_{\mathrm{right\_hand}}\right| + \left|\mathbf{v}_{\mathrm{left\_hand}}\right| + \left|\mathbf{v}_{\mathrm{head}}\right|\right) > v \\ f(t) \times \left(\left|\mathbf{v}_{\mathrm{right\_hand}}\right| + \left|\mathbf{v}_{\mathrm{left\_hand}}\right| + \left|\mathbf{v}_{\mathrm{head}}\right|\right) / v & \text{otherwise.} \end{cases} \tag{6}$$

After the reaction type and reactive motion are determined, the system simply controls the weights. After the execution of the reactive motion and transition motion (if required) have finished, the system then goes back to normal control.

5.4 Transition to Input Motion

As explained in Section 5, after the avatar falls down, the system executes a transition motion in order to prevent an unnatural transition. We assume that the user is always standing, and therefore a standing up motion is executed for transition when the avatar falls down. Further, though we can choose the appropriate motion based on the input state (the user's state) and the current state (the avatar's state), in our current implementation we simply assign a standing up motion to each reactive motion manually.

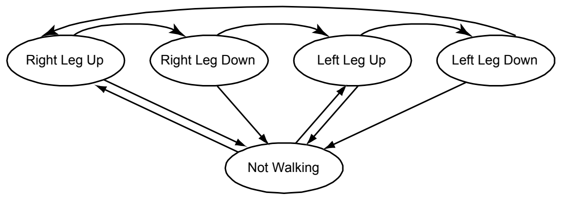

Figure 9. Walking states. While walking the avatar transits through the above four states.

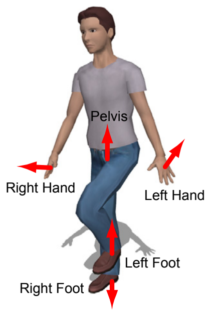

Figure 10. Velocities used for detecting walking gestures.

6. WALKING CONTROL

The system generates walking or running motion as a target motion for the tracking controllers when the user performs a walking gesture (Figure 3). In order to make the avatar start to walk, the user has to move his legs and arms in a synchronized way as we normally do during walking. Once a walking motion starts, the avatar keeps walking while the user keeps moving his legs. The user can move his upper body freely after the walking motion starts. For example, the avatar can wave its hand or hit other character while walking. To realize this kind of interface, the system recognizes both the full body gesture (starting condition) and the lower body gesture (maintaining condition). In addition, the tracking weights for the input motion and walking motion are controlled based on the recognition results.

6.1 Generation of Walking Motions

Currently we use a motion blending technique [13] for the generation of walking motion so that the speed and turn angle of walking are controlled based on the user's motion. Of course, any other kind of motion generation technique (e.g., motion graphs, dynamic simulations, etc.) can also be used with our interface.

To use motion blending, we first prepare example walking motions. The keytimes are set to the example motions by hand. This keytime information is essential to the synchronization of the example motions and to making the motion blending work. In addition, the speed and turn angles of each example motion are computed in advance. Using this information, the blending weights for the example motions are computed when a set of walking parameters (speed and turn angle) are given. A resulting walking motion is then computed by blending the example motions with the blending weights. More details regarding the motion blending method can be found in [13].

6.2 Detection of Walking Gestures

To control the parameters of walking motion, in addition to whether or not the user is walking, the speed and turn angle parameters are also required. The walking orientation (turn angle) is easily computed from the orientation of the user's pelvis. To compute the walking speed, we divide walking motion into four phases (right leg up, right leg down, left leg up, left leg down), as shown in Figure 9, and detect in which state the user is currently. Based on the state transition speed, the walking speed is then estimated.

The walking state is determined based on the velocity of the pelvis, feet, and hands (Figure 10). We use a heuristic method based on fuzzy rules.

To detect the full body walking states, we use the velocity of the pelvis, feet, and hands. For example, during the right-leg-up state, the vertical velocity of the right foot $v_y^{\text{right\_foot}}$ must be a large positive value. The velocity of the left hand $v_z^{\text{left\_hand}}$ must be frontward, and the velocity of the right hand $v_z^{\text{right\_hand}}$ must be backward. In addition, the horizontal velocity of the pelvis $v_{|x+z|}^{\text{pelvis}}$ must be always a small value during walking. We evaluate the scores of these conditions based on a membership function (Figure 11), which is represented by the average and dispersion. For example, the evaluation (0 to 1) for the right foot $v_y^{\text{right\_foot}}$ is computed as follows.

$$e_y^{\text{right\_foot}} = \begin{cases} \left(v_y^{\text{right\_foot}} - a^{\text{right\_foot}}\right) / d^{\text{right\_foot}} & if \left|v_y^{\text{right\_foot}} - a^{\text{right\_foot}}\right| < d^{\text{right\_foot}} \\ 0 & if \left|v_y^{\text{right\_foot}} - a^{\text{right\_foot}}\right| \geq d^{\text{right\_foot}} . \end{cases} \quad (7)$$

Based on evaluations computed in the same way, the probability of the state is then computed by

$$e^{\text{right\_leg\_up (full)}} = \min\left\{e_y^{\text{right\_foot}}, e_{|x+z|}^{\text{pelvis}}, e_z^{\text{right\_hand}}, e_z^{\text{left\_hand}}\right\}. \quad (8)$$

To detect the lower body walking state, we need consider only the velocities of the pelvis and feet.

$$e^{\text{right\_leg\_up (lower)}} = \min\left\{e_y^{\text{right\_foot}}, e_y^{\text{left\_foot}}, e_{|x+z|}^{\text{pelvis}}\right\}. \quad (9)$$

For the other three states, we use the same functions.

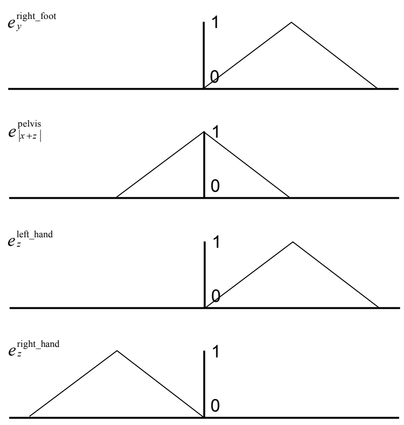

Figure 11. Membership functions for evaluating the conditions of "right leg up."

When the probabilities of $e^{\text{right\_leg\_up (full)}}$ or $e^{\text{left\_leg\_up (full)}}$ exceed a threshold, the avatar starts to walk. While walking, if the probability of the next state ($e^{\text{right\_leg\_down (lower)}}$ in the case of "right leg up," for example) exceeds the threshold, it then transits to the next state.

When the walking state transits from one state to the next, the velocity of the end-effectors crosses zero, and the state is not recognized correctly in the instance. To address this, we wait for a fixed duration until a stop in the walking motion. When the next state is detected, the walking state then transits to the next state. The state transition speed is computed based on the previous transition times. In addition, based on the state transition speed, the generic time (0 to 1) of the walking motion is also computed.

6.3 Control of Walking Motions

During walking motion, the weight for the lower body including the pelvis is set to 0. For the upper body, the weight is set to 0 when either the user moves his arms in the walking way or when the user doesn't move his arms. On the other hand, a larger value is set to the upper body when the user moves his arms in the same manner as given in equation (6). As a result, after a walking motion starts, the user can control his arm movements.

7. EXPERIMENTS AND DISCUSSION

We developed the proposed framework and controllers as well as a prototype system in which the user can control the avatar through a motion capture system and interact with an autonomous character. Figure 4 shows some pictures of the prototype system. Currently, the motions of the character are either randomly chosen from a motion set or are controlled by another user through a keyboard interface.

Although our optical motion capture system is influenced by camera noises and the captured motion is somewhat unstable since we use a heuristic labeling algorithm similar to that described in [17], the detection of walking gestures basically works well. We have not experienced any errors, for example, in which the avatar starts walking based on a misjudgment. We did, however, experience walking motions that sometimes stopped contrary to the user's intention. Currently, if one walking step is passed over due to motion capture noise or some other factor, the walking motion stops even if the user maintains his walking motion. To avoid such situations, we need to refine our method to be more stable. Another problem of the current walking interface is that it is difficult to walk backward in our system. This is because if the user turns back he cannot look at the screen. Actually, this difficulty of changing one's orientation is a common problem of the third-person view. Using multiple screens that show an identically rendered image is one possible solution [3]. Other problems related to using the third-person view include the difficultly for users to see small objects or the faces of characters clearly in front of the avatar because of the camera distance. To solve such problems, application-dependent automatic camera positioning and orientation control or dynamic switching views between the first-person view and the third-person view may be useful.

In addition, we expect that the walking-gesture-based interface can be applied to other kinds of actions, such as climbing, swimming, or using props. However, since some actions require more complex gestures, a more generic and robust method may be necessary.

Moreover, our algorithm for reactive motions works well. However, some users did not recognize the differences between reactive motions and the effects of the user's movements. Currently, the user's movements during reactive motion only affect the visual appearance of the avatar but not the result of the motion. For example, no matter how the user tries to recover his balance by moving his arms, it does not affect the results of the motion (falling or recovering). If we can change the results of the motions based on the user's control, the reactive motions will be more interesting to the users. Moreover, currently we only handle instant collisions between characters, and therefore the system cannot handle constant contact situations, such as wrestling. To address these problems, we may need to introduce physics-based algorithms into the controllers.

8. CONCLUSION AND FUTURE WORK

We have proposed a control framework for third-person view virtual environments. We combine multiple tracking controllers for both captured motion and system generated motions. Although the current prototype system is a very simple application, it is very interesting to be able to control an avatar through a motion capture system and to interact with a virtual character. In the near future, it is expected that computer entertainment incorporating this kind of control framework will be available for us to play.

9. REFERENCES

[1] Cheung, G., Baker, S., Kanade, T. Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture. In Proc. of Computer Vision and Pattern Recognition 2003.

[2] Emering, L., Boulic, R., Thalmann, D. Live Participant's Action Recognition for Virtual Reality Interactions. In Proc. of Pacific Graphics, 15-21, 1997.

[3] Hämäläinen, P., Ilmonen, T., Höysniemi, J., Lindholm, M., Nykänen, A. Martial Arts in Artificial Reality, Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 005), 781-790, 2005.

[4] Hasegawa, S., Ishikawa, T., Naoki Hashimoto, N. Human Scale Haptic Interaction with a Reactive Virtual Human in a Realtime Physics Simulator. In Proc. of Advances in Computer Entertainment Technology 2005.

[5] Hodgins,, J. K., Wooten, W. L., Brogan, D. C. and O'Brien, J. F. Animating Human Athletes. SIGGRAPH '95 Proceedings, 71-78, 1995.

[6] Hof A. L., Gazendam, M. G. J., Sinke, W. E. The condition for dynamic stability. Journal of Biomechanics, 38, 1-8, 2005.

[7] Komura T., Ho E. S.L., Lau R. W.H. Lau. Animating Reactive Motion Using Momentum-based Inverse Kinematics. The Journal of Computer Animation and Virtual Worlds, 16(3-4), 213-223, Wiley, 2005.

[8] Laszlo J., van de Panne M., Fiume E. Limit Cycle Control and Its Application to the Animation of Balancing and Walking. In Proc. of SIGGRAPH 1996, 155-162, 1996.

[9] Moore, M., and Wilhelms, J. Collision Detection and Response for Computer Animation. Computer Graphics (SIGGRAPH '88 Proceedings), 22(3), 289-298, 1988.

[10] Natural Point Inc. http://www.naturalpoint.com/optitrack/.

[11] O'Rourke, J. Computational Geometry in C (Cambridge Tracts in Theoretical Computer Science), Cam-bridge University Press, 1998.

[12] Oshita, M., Makinouchi, A. A Dynamic Motion Control Technique for Human-like Articulated Figures. Computer Graphics Forum (EUROGRAPHICS 2001), 20(3), 192-202, 2001.

[13] Park S. I., Shin H. J., Shin S. Y. On-line Locomotion Generation Based on Motion Blending. In Proc. of ACM SIGGRAPH Symposium on Computer Animation 2002, 113-120, 2002.

[14] Shin, H.J., Lee, J., Gleicher, M., Shin, S.Y. Computer Puppetry: An Importance-Based Approach, ACM Transactions on Graphics. 20 (2), 67-94, 2001.

[15] Yamane, K., Nakamura, Y. Dynamics Filter - Concept and Implementation of Online Motion Generator for Human Figures. IEEE Transactions on Robotics and Automation, 19(3), 2003.

[16] Yin, K. K., Pai, D. K. FootSee: an Interactive Animation System. In Proc. of ACM SIGGRAPH / Eurapraphics Sym-posium on Computer Animation 2003, 329-338, 2003.

[17] Yonemoto, S., Arita, D., Taniguchi, R. Real-time Human Motion Analysis and IK-based Human Figure Control. In Proc of Workshop of Human Motion 2000, 149-154, 2000.

[18] Zordan, V. B., Chiu, B., Majkowska, A., Fast, M. Dynamic Response for Motion Capture Animation. ACM Transactions of Graphics (Proc. of SIGGRAPH 2005), 24(3), 697-701, 2005.