Using Motion Capture for Interactive Motion Editing

Masaki Oshita[*] Hiroyuki Muranaka

Kyushu Institute of Technology

Abstract

Motion capture technology has been widely used for creating character motions. Motion editing is usually also required to adjust captured motions. Because character poses which include joint rotations, body positions, and orientations are high-dimensional data, it is difficult to manipulate character poses through a conventional mouse-based interface. We propose a motion editing system that uses a motion capture device. Our system can capture a motion and edit the captured motion using the same motion capture device. Our motion-capture-based interface can specify motion editing parameters such as time period, body part selection, end-effector position, pose, and motion segment. We conducted a user study to compare our system and conventional mouse-based motion editing system. The results showed that our interface is more efficient than the conventional 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 Capture, Motion Editing, Computer Animation, User Interface

1 Introduction

Motion capture (mocap) technology has been widely used for creating character motion for movies and computer games, among other applications. Motion editing is usually required after motion capture because adjustments such as fixing a foot position or changing pose and timing are often necessary. However, this editing is not easy. Character poses, which include joint rotations, body positions, and orientations are high-dimensional data, and as such are difficult to manipulate though the conventional mouse-based interface. Because the motion editing process is done by an animator separately from the motion capture process, it is also difficult and often impossible to go back and redo a capture.

To solve these problems, we propose a motion editing system that uses a motion capture device (see Figure 1 and 2). The system user can capture a motion and edit the captured motion using the same motion capture device, going back and forth between the motion capture and editing processes. We developed a mocap-based interface for specifying motion editing parameters required for our motion editing methods such as time period, body part selection, end-effector position, pose, and motion segment. We designed an effective interface for each type of parameter. Poses input from the motion capture device are directly used to specify spatial parameters such as position, pose, and motion segment. On the other hand, simple parameters such as time period, body part selection, and mode selection are specified through combinations of gesture recognition, virtual time line, and menu by processing input poses from the motion capture device.

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

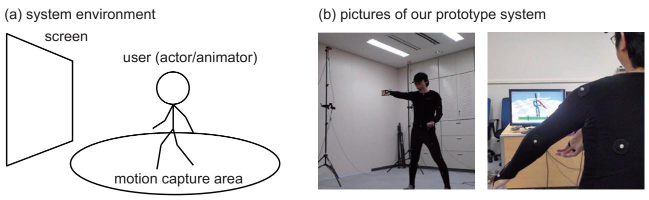

Figure 1: *Proposed system: using motion capture for interactive motion editing.*

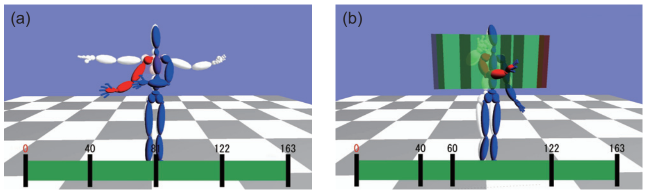

Figure 2: *Screen shots from our system.*

We conducted a user study to compare our system and a conventional mouse-based motion editing system. The results showed that our interface is more efficient than the conventional ones.

The remainder of this paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 describe the system overview and motion editing methods, respectively. In Section 5, the interface design for our mocap-based motion editing system is explained. Section 6 presents our experimental results. Section 7 concludes this paper.

2 Related Work

Motion editing software such as MotionBuilder, Maya, Max, and Softimage are widely used in animation production but need to be operated through a mouse-based interface. It is difficult to manipulate high-dimensional parameters such as position, orientation, and pose. Only trained animators can use these systems. It is difficult for motion capture actors to edit their own motions.

There are several systems that employ spatial sensors to utilize a high-dimensional input for motion editing. Dontchev et al. [2003] used a spatial marker to specify the movements of a character's body part. Oore et al. [2002] used two bamboo sticks with a magnetic tracking sensor to do the same, with a bamboo stick being used as a proxy for a body part such as the trunk or an arm or leg. However, because only one or two body parts are controlled at a time, the user needs to repeatedly specify the movements of each body part for motion editing.

Various researchers have studied use of spatial devices for three-dimensional positioning tasks [Bérard et al. 2009; Teather and Stuerzlinger 2007]. Designing such interfaces requires special care and sometimes conventional mouse-based interfaces outperform spatial devices. In our research, the user edits a character's pose and motion, which are high-dimensional data compared with simple positioning tasks. Using the user's own pose for specifying parameters is considered to be intuitive and effective compared with using a spatial device for general positioning tasks. Our experiments validate this assumption.

3 System Overview

A user of our system can capture a motion and edit the captured motion using the same motion capture device. Figure 1 shows a typical setup. The user wears a motion capture device and can operate the system while watching a large screen placed in the motion capture environment. Any type of motion capture device can be used with our system. For our prototype, we used the Optitrack optical motion capture system from NaturalPoint [NaturalPoint] with 12 cameras. Using the Optitrack software development kit (SDK), our system can record the user's current pose as a continuous stream (30 frames/second).

The typical workflow is as follows: The user captures a new motion first. Alternatively, an existing motion file can be imported for editing. The user can review and edit the captured motion on the screen by specifying parameters acquired through our motion-capture-based (mocap-based) interface. The process repeats until the desired motion is achieved. The edited motion can then be exported and used in any other animation system.

Figure 2 shows screen shots from our system. The user's current pose is displayed as a blue stick figure, while the current pose in the captured motion is displayed as a white stick figure. The timeline is also displayed on the screen. The current time is indicated on the timeline.

The character model must be created in advance and given to the system. Because the character's skeletal model is different from the actor's skeletal model, captured motions and editing parameters are automatically retargetted from the user's pose to the character's pose. We implemented an on-line retargetting process which is similar to [Shin et al. 2001; Lee and Shin 1999]. In our experiments, we used a character model that contains 50 joints. The user's skeletal model used in the motion capture system [NaturalPoint] had 23 joints.

A captured motion is represented as a series of poses. A pose is represented by the position and orientation of the pelvis and the rotations of all joints. In our implementation, each joint rotation is represented by a quaternion.

4 Motion Editing Method

This section describes the motion editing methods in our system. This is not the novel part of this research. We implemented standard motion editing methods using basic kinematics-based techniques such as pose interpolation and inverse kinematics [Witkin and Popović 1995; Lee and Shin 1999].

Table 1 shows the list of our motion editing methods and their required parameters. "Pose deformation" and "motion replacement" are used to change a part of the motion. "Fixing a body part" is used to immobilize its position for a certain time period. This editing method can be used to prevent foot sliding while the foot is contacting the ground, for example. All editing methods requires a time period parameter and target body parts. A time period parameter consists of four time values $t_{init}, t_{begin}, t_{end}, t_{term}$ where $t_{init} < t_{begin} \leq t_{end} < t_{term}$. In the case of pose deformation, the deformation is applied to the pose at a specified time ($t_{begin} = t_{end}$) and only three times are specified.

Table 1: *Motion editing methods in our system.*

method	parameters
pose deformation	time period, body parts, pose
fixing a body part	time period, body part, position
motion replacement	time period, body parts, motion

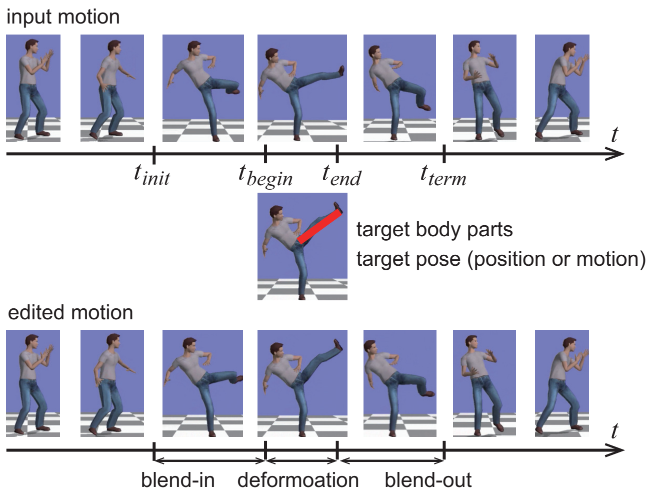

Figure 3: *Motion editing method.*

Figure 3 depicts the motion editing process. Given the target pose, position or motion is applied to the motion segment ($t_{begin} \sim t_{end}$). A smooth blend-in and blend-out are applied before and after the motion segment to generate a continuous motion.

5 Interface Design

In our system, the user first selects which motion editing method is to be applied and then specifies all necessary parameters in any order. Once all parameters are given, the motion editing method is applied and the deformed motion is displayed. The user can change the parameters if necessary. Finally, the user can either confirm the deformed motion or cancel it. This process is repeated until a desired motion is achieved.

Figure 4 shows the states and transitions in the motion editing interface. A transition is initiated by performing a gesture command. Motion capture or motion editing mode is selected after menu activation. When a capture is complete, the interface reverts back to the previous state as shown by the dotted arrows.

We designed our mocap-based interface to include gesture command input, mode selection, time specification, body part selection, pose input, position input, and motion segment input. These interfaces are explained in the subsections below. As explained in Section 4, our system has three motion editing methods: pose deformation, fixing a body part, and motion segment replacement. Time selection and body section are used for all the motion editing methods. Pose input, position input, and motion segment input are used only for pose deformation, fixing a body part, and motion segment replacement, respectively.

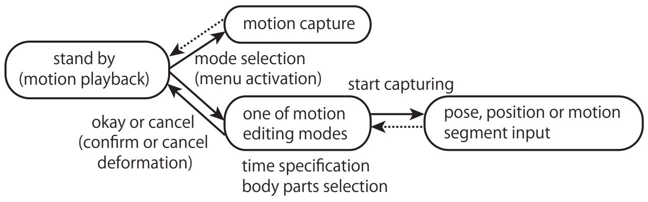

Figure 4: *States and transitions in the motion editing interface.*

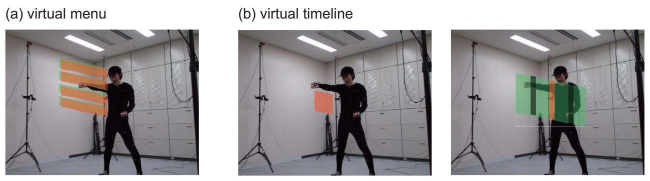

Figure 5: *Virtual menu and timeline. Users can operate on the menu and timline as if they exist in front of them.*

5.1 Gesture Command

We introduced four gesture commands for okay, cancel, menu activation, and start capturing. Initially, we considered assigning different gestures to each motion editing method so that the user can select an editing method by performing the corresponding gesture. However, we found it impractical for users to remember all the different gestures. On the other hand, using only a virtual menu for all commands is also impractical because the user may inadvertently touch the virtual menu when performing capture and edit operations. Therefore, we decided to combine gestures for fundamental commands and a virtual menu for selecting a motion editing method.

Although any gesture can be assigned to these commands, we assigned the simple gestures as follows:

- okay: waving the right hand from left to right
- cancel: waving the right hand from right to left
- menu activation: waving the right hand from up to down
- start capturing: waving the right hand from down to up

Any of the various methods for gesture recognition can be used. We used an existing method [Oshita and Matsunaga 2010], which worked well for our experiments.

5.2 Method Selection Using Virtual Menu

The motion editing method that the user will apply is selected from a virtual menu. When the user performs the menu activation gesture, a virtual menu is placed in the space in front of the user as shown in Figure 5(a) and displayed on the screen. Similar to using a mouse, the user can select a menu item by moving the right hand up and down. The selected menu item is displayed in a different color. By moving the right hand forward, the user can then select the menu item. We determined the appropriate size and depth of the menu selection space empirically.

5.3 Time Specification

A small rectangle is placed in front of the user, as shown in Figure 5(b). It moves automatically to maintain a position relative to the user. By touching the rectangle, the time specification is activated and the user can operate on the time line, as shown in Figure 2(b). The virtual menu can also be used to specify $t_{init}, t_{begin}, t_{end}, t_{term}$ by touching the virtual time line. By touching a key time on the timeline and moving it to right and left, the selected key time is altered.

5.4 Body Parts Selection

By touching their own body parts, the user can select which joints the motion editing method is applied to. The selected joints are drawn in a different color, as shown in Figure 2(a). By touching the selected joints again, the body parts selection is canceled. The interface determines a touch has occurred when the distance between the hand position and joint position falls below a threshold we previously set.

For pose deformation and motion segment replacement, multiple joints can be selected, while only one body part can be selected for fixing a body part. To facilitate specifying multiple joints, we introduced an additional interface tool. By touching two adjacent joints, a group of joints are selected. For example, by selecting the right shoulder and then the right elbow, all joints in the right arm are selected. By touching these joints in the reverse order, the selection of joints in the right arm is canceled. In a similar way, by touching the pelvis and chest, all joints in the upper body are selected.

5.5 Pose Input

A user's pose is captured and used as input for pose deformation. Although capturing a pose is simple, there are several issues to be addressed.

First is the timing of pose capturing. Because our system receives the user's current pose as a continuous stream, to take one input pose, the timing of pose capturing needs to be determined. To accomplish this, we introduce a countdown mechanism. When the user performs the start capturing gesture, as explained in Section 5.1, the system shows a countdown timer on the screen. The user can effect a pose while the timer is counting down. When the timer reaches zero, the user's current pose is taken as the pose input. We set the countdown timer to a default three seconds.

The second issue is placement of the captured pose. Because it is possible to replace any pose in the original motion, the position and orientation of the original pose may not match that of the captured pose. This may cause a large gap in the position and orientation of the deformed pose. To prevent this, the captured pose must be aligned to the position and orientation of the original pose in the original motion, although a small change of position needs to be allowed for if the user intends it. To realize this, the captured pose is represented by coordinates based on the user's pose when the user started capturing. The pose is then conformed to the coordinates based on the original pose at time t_{begin} in the original motion.

5.6 Position Input

The same basic methods used for pose input are used to handle the timing of pose capture and placement of the captured pose. The position of the selected body part is acquired and conformed to the coordinates based on the original pose at time t_{begin} in the original motion.

5.7 Motion Segment Input

To capture a motion segment, the timing for starting and ending motion capture need to be determined. To specify end timing, a

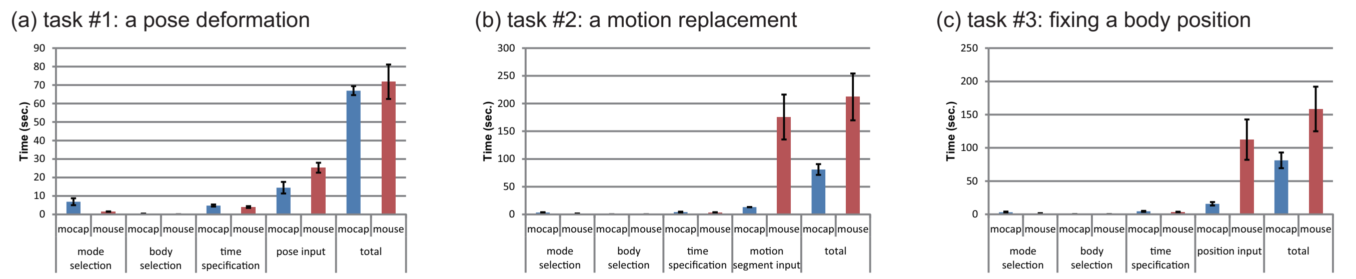

Figure 6: *Results from our experiments. Average required times for each step on each task by our mocap-based (blue) and alternative mouse-based (red) interfaces. Note that the vertical scales are different on each graph.*

different mechanism than that for starting is required because the user cannot perform a gesture during motion capture.

To accomplish this, we introduced a still detection mechanism. When the user stops the motion and stays still for a period of time, the system finishes the capturing motion. A determination of motion is based on velocities as calculated by differences in body part positions on consecutive frames of the capture. When velocity falls below our set threshold of $0.1m/s$, it is determined that the user is not moving. At this time, motion capture will cease after three seconds by default.

6 Experiments and Discussion

We implemented our prototype for motion-capture-based motion editing. The accompanying video includes a demonstration of our system.

We conducted a user test to evaluate the effectiveness of our approach compared to conventional mouse-based motion editing systems. We also implemented a mouse-based interface of our motion editing system for comparison. With the mouse-based interface, the user can select a motion editing method by clicking the menu displayed on the corner of the screen. The body part selection and time selection can be performed by clicking the body parts and time line on the screen. To change a pose and body part position, we implemented a pose editing interface. The user can control the rotation or position of a body part by dragging a handle that is displayed near the selected joint. This is a typical pose editing interface implemented in existing motion editing systems.

We compared the required time for motion editing process with our mocap-based interface to that of the mouse-based interface. Three graduate students participated in our experiment. They had fundamental knowledge of motion editing but were not experts in animation editing. They were asked to complete given motion editing tasks using the mocap-based interface and mouse-based interface. Each used both interfaces in a different order. For each trial the participant was given an original motion and a motion editing goal. The elapsed time for each task was measured.

Each experiment included three tasks:

1. pose deformation (changing the right arm pose to strike a lower target when the hand is extended in a punching motion)
2. motion replacement (changing the right arm motion from a punch to an uppercut motion)
3. fixing a body position (fixing the position of the other hand during punch)

Figure 6 shows the experiment results. The average time and deviation is listed for each step of the operation. The mocap-based interface achieved better results compared with the mouse-based interface. As we expected, the mocap-based interface worked efficiently for specifying high-dimensional parameters such as pose, motion, and position. For specifying simple parameters such as body part, time, and mode selection, the mocap-based interface required a little extra time but was almost as efficient as the mouse-based interface.

One of our concerns about the mocap-based interface was that it may not be suitable for precise control compared to the mouse-based interface. To evaluate this, we compared the difference between the controlled parameter and the parameters from the target motion and deemed the operation complete when this difference fell below a threshold. To evaluate the precision of the two different interfaces, we experimented with several different thresholds for time specification, pose input, and position input. Although detailed results are not presented in this paper, to summarize the results, it can be said that both interfaces required more time as the threshold became smaller. However, the overall difference between the results of the two interfaces did not change regardless of the threshold suggesting that the mocap-based interface is in fact capable of precise control.

A user of our system can work as both actor and animator. These tasks are currently separated in animation production because the editing process is difficult to use for actors. Using our system, actors can edit their own motions. This can be an interesting approach to animation production. Alternatively, two users (actor and animator) may use our system together by taking turns performing motion capture and motion editing. We are investigating future practical applications for our system.

7 Conclusion

In this paper, we proposed a motion editing system that uses a motion capture device. The results show that our mocap-based interface is more efficient than the conventional mouse-based interface.

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

BÉRARD, F., IP, J., BENOVOY, M., EL-SHIMY, D., BLUM, J. R., AND COOPERSTOCK, J. R. 2009. Did ”minority report” get it wrong? superiority of the mouse over 3d input devices in a 3d placement task. In *12th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT) 2009*, 400–414.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layerd acting for character animation. In *SIGGRAPH 2003*, 409–416.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99*, 39–48.

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

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., AND MATSUNAGA, T. 2010. Automatic learning of gesture recognition model using SOM and SVM. In *6th International Symposium on Visual Computing 2010*, 751–760.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2, 67–94.

TEATHER, R. J., AND STUERZLINGER, W. 2007. Guidelines for 3d positioning techniques. In *Future Play 2007*, 61–68.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *SIGGRAPH '95*, 105–108.