Multi-Touch Interface for Character Motion Control Using Example-Based Posture Synthesis

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka
820-8502, Japan
oshita@ces.kyutech.ac.jp

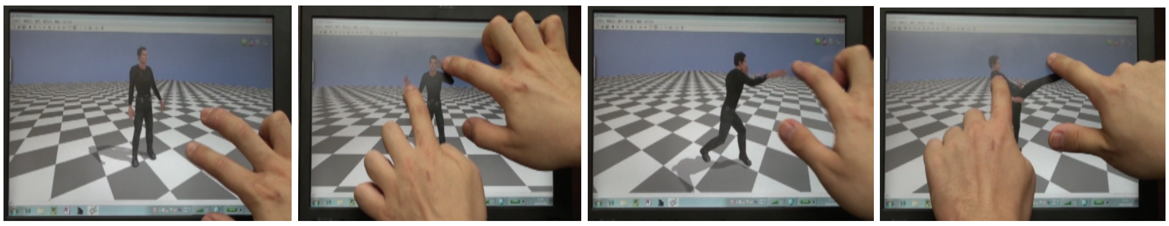
Figure 1. Example of the multi-touch interface for character motion control.

ABSTRACT

We propose a multi-touch interface for character motion control with which a user can control a character's pose freely and make the character perform various actions by simply dragging the character's body parts using a multi-touch input device. We use style-based inverse kinematics to synthesize a natural-looking pose that satisfies given constraints using a learned model of sample postures. However, the style-based inverse kinematics is suited to posture editing and not motion control. It cannot handle various types of actions and cannot generate continuous and physically valid motions. To overcome these limitations, we prepare different learned models for each action and choose an appropriate learned model according to the user's input. More specifically, we use a single learned model for posing and different learned models for each type of action. We also use different motion generation methods for posing and action controls. Furthermore, we introduce tracking control as a post-process after posing and action controls to generate continuous and physically plausible motion. We implemented the proposed method using a Windows 7 Touch API on a multi-touch-enabled personal computer, and demonstrated the power of our interface.

Keywords
Motion control, multi-touch interface, example-based posture synthesis, physics-based control.

1. INTRODUCTION

The flexibility of character motion control in interactive applications is currently very limited. Since common input devices such as a gamepad, mouse or keyboard have only a small number of degrees of freedom, the user can do nothing but simply select an action from a few pre-defined actions by pressing an associated button and make a character perform a fixed action such as walking, punching, or kicking. It is impossible for users to pose the character freely or make the character perform various actions in the user's own style. This is particularly a limitation in some interactive applications such as fighting games, dance animation and online communication using avatars. For example, users may want a character to perform various moves in fighting games and interactive dance animation or use various gestures in communication in a multi-user network environment.

We propose a multi-touch interface for character motion control with which a user can control a character's pose freely and make the character perform various actions by simply dragging the character's body parts using a multi-touch input device (Figure 1).

In theory, using inverse kinematics (IK), a user could control a character's pose in detail. However, with conventional input devices, the user can control only one end effector at a time. Moreover, it is difficult to realize natural-looking motion using IK even with a multi-touch device because multiple body parts must be controlled in a coordinated way to execute an action.

Our method uses style-based IK [GMH*04][SL06]. Style-based IK constructs a learned model of poses in advance by mapping a large number of sample poses onto a low-dimensional latent space. It then synthesizes a natural-looking pose that satisfies given constraints using the learned model. However, style-based IK has several problems. First, it cannot handle many types of actions with a single learned model because appropriate poses depend on the type of action to be expressed. Second, it is difficult to generate continuous motion from given touch strokes of body parts because there is no guarantee that a continuous trajectory in Cartesian space is mapped to a continuous trajectory in latent space. Third, resulting motions that are synthesized using style-based IK lack physical validity, since the character can take any pose at any speed without considering physical constraints. For these reasons, style-based IK cannot be simply used for the motion control of various types of actions. Basically, style-based IK is suitable for making a pose but not for interactive motion control.

To solve these problems, we prepare different learned models for each action and choose an appropriate learned model according to the user's input. More specifically, we use a single learned model for posing and different learned models for each kind of action. We also use different motion generation methods for posing and action controls. During posing control, a pose is generated simply using a learned model based on multi-touch inputs. When a touch stroke for a body part matches the initial trajectory of the primary body part of a prepared action model, the system switches to action control. During action control, we control time progression by considering conditions of sample postures with a learned model to generate continuous and natural-looking motion. We also introduce tracking control as a post-process for posing and action controls to generate continuous and physically plausible motions. Using physics simulation, physical effects such as the shaking of non-controlled body parts by controlled body parts are calculated and applied.

We implemented the proposed method using Windows 7 Touch API on a multi-touch-enabled personal computer, and thus demonstrated the power of our interface.

The rest of this paper is organized as follows. In Section 2, we review related works. Section 3 presents an overview of our method, while Sections 4, 5 and 6 describe posing, action and tracing controls respectively. Results and a discussion are presented in Section 7. Finally, Section 8 concludes the paper.

2. RELATED WORK

Multi-touch Interface for Motion Control

To our knowledge, there are only a few research works that use a multi-touch interface for motion control and generation. Krause et al. [KHS*08] applied conventional IK to a character model based on multi-touch inputs for animation. However, as explained in the previous section, it is difficult to realize complex motions with this approach. Moreover, animation takes a long time and interactive control is impossible. Kip and Nguyen [KN10] proposed a system to control one arm and hand of a character using a multi-touch interface by changing several parameters to blend arm and hand postures. However, their system is limited to the control of one arm and one hand and cannot be used to control full-body motion.

Trajectory-Based Motion Generation

Since the mouse and pen were common input devices well before multi-touch devices became available, there has been much research on the use of a single point or trajectory to control or generate character motion.

A common way to use trajectory for motion control is to use a trajectory to specify a locomotion path [PSS02]. Many animation systems have a function to generate walking and running motions according to a trajectory drawn on the ground. However, with this type of interface, the type of motion is limited to walking or running motion.

Throne et al. [TBvdP04] introduced gesture-based motion selection. Based on the gestures drawn along a trajectory, their system inserts predefined motions such as a jump or flip. Oshita [Osh05] proposed a stroke-based motion selection technique that chooses an appropriate action according to the initial and terminal points of a single stroke drawn on the screen. With these two systems, users can simply select actions by drawing a trajectory or stroke, but postures and the speed of actions are fixed and cannot be controlled.

Igarashi et al. [IMH05] proposed a spatial keyframing animation technique. By placing key poses in the three-dimensional space in advance, a new posture is synthesized by blending the key postures according to the distances between the current mouse position and the key postures. By drawing a trajectory on the screen (moving the mouse cursor), a continuous motion can be generated.

However, to generate natural-looking and intended motion, the key postures must be placed at appropriate positions and at appropriate distances and the user must move the mouse cursor properly. This requires training and careful design. It is also difficult to generate various types of action with a single set of key postures when using this method. Dontcheva et al. [DYP03] used a physical three-dimensional marker to specify body trajectories. However, since the user can control one body part at a time with their system, they need to repeat specifying trajectory for each body part. Their method is for off-line animation making and is not suitable for interactive control.

Style-based IK

As explained in Section 1, style-based IK generates a posture according to user input. The approach is not suitable for motion generation or control. Previous application of style-based IK is limited to posture editing [GMH*04] and motion generation based on trajectory in the latent space [SL06]. Motion generation based on trajectory in three-dimensional space or on a screen was not realized.

Min et al., [MCC09] introduced a statistics-based model of motion instead of postures. Using a given trajectory of a specified body part, sample motions are blended to synthesize new motion. The method can be used with multi-touch devices. However, since entire motions are blended, detailed control such as time and speed control and changing postures is not possible. Moreover, the model must be learned for a specific type of action and the system does not allow the execution of combinations of various types of actions.

Other motion control interfaces

Various devices have been used for interactive motion control by researchers. Some physical sensors such as Wii remote (acceleration sensors) can be used with gesture recognition technique [LLZ*09] [BNT08]. By performing a specific gesture, the user can select an action from predefined actions. However, this type of gesture-based interface simply substitutes for conventional interfaces such as game pad and keyboard. Control of the character's full-body motion and executing action with user's own style are not possible.

Some researches combine physics simulation with a control of limited number of degrees of freedom. Laszlo et al. [LZS05] used mouse to control a few joints and generated full-body motion by using physics simulation. Shiratori and Hodgins [SH08] used Wii remotes to determine a few parameters for physics-based controllers of several actions such as walking, running, and jumping. By using physics simulation, physically plausible motion can be generated and controlled. However, the physics-based controllers must be designed for each type of action in advance and realizing various types of actions and styles is difficult.

Recently, low-price markerless motion capture devices such as Microsoft Kinect have become available. Using such a device, the character's full body motion can be freely controlled [IWZ*09] [Osh06]. However, such a device requires a larger workspace. Moreover, to perform highly dynamic actions, the user must actually perform the actions. This is difficult for non-trained users.

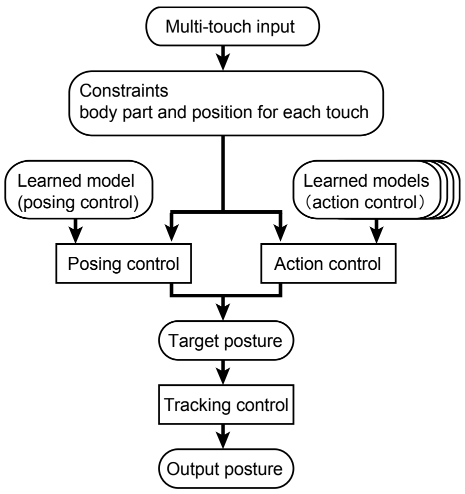

Figure 2. System overview.

3. SYSTEM OVERVIEW

The overview of our system is shown in Figure 2. The system generates a character's posture in each frame according to multi-touch inputs. Any type of multi-touch device can be used with our system. When the user touches a body part of the character and drags it on the screen, each touch input is handled as a constraint to control the character. Note that each constraint is a two-dimensional position on the screen; that is, a half-line in the three-dimensional space.

Our system uses posing and action control modes with different learned models. Under the initial condition, the posing control mode is used to change the character's posture according to multi-touch inputs. When the trajectory of the body part that is dragged by the user matches the trajectory of a predefined action, the system switches to the action control mode to generate an action that dynamically changes according to the multi-touch inputs. When the action finishes, the system switches back to the posing control mode.

In addition to posing and action controls, tracking control is applied as a post-process to change the

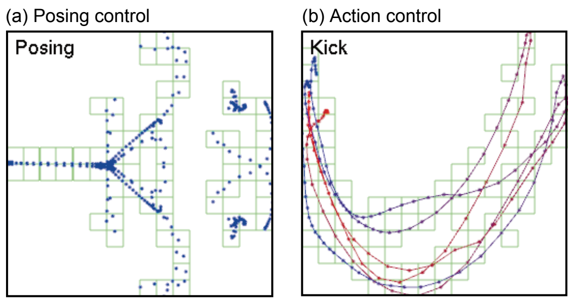

Figure 3. Example of latent space (two-dimensional).

posture generated by posing and action controls so that the generated motion is continuous and physically plausible.

Since our system handles only the touching and dragging of character's body parts, other types of multi-touch inputs such as touching and dragging in another area, and multi-touch gestures including pinching and swiping, can be used for other functions such as viewpoint control depending on the application.

4. POSING CONTROL

The posing control is used to generate character motions that have no particular form such as a change in the standing pose or the free movement of hands and feet. For posing control, we use a method similar to conventional style-based IK. The problems of style-based IK are solved by combining action and tracking controls. In our implementation, we use a method similar to that used by Shin et al. [SL06].

As sample postures for the learned model in posing control, we use postures extracted from motion data for various poses. In addition, the initial period of motion data for action control is also used, since initial parts of actions are realized by posing control before action control is initiated.

Learned model for posing control

To apply style-based IK, sample postures are mapped on a low- (typically two- or three-) dimensional latent space to construct a learned model in advance. There are various ways to map example postures onto a latent space [SL06]. In our implementation, we employed multi-dimensional scaling (MDS). The sample postures are mapped to a latent space so that the distances between sample postures are preserved in the latent space. The distance between any two postures is calculated according to the sum of distances between all pairs of corresponding joints after two postures are aligned according to their pelvis position and orientation in the same manner as in [KGP02].

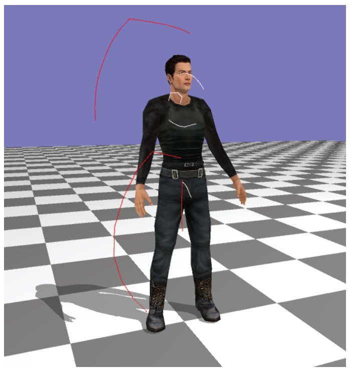

Figure 4. Example of trajectories of action models.

Posture synthesis during posing control

From the given constraints, which are two-dimensional positions of body parts, a posture is synthesized using the learned model by blending sample postures in the latent space.

We divide the latent space into a grid and assign a representative sample posture to each grid in advance as shown in Figure 3, where each dot represents a sample posture. By comparing the representative sample postures with given constraints, the corresponding cell can be found efficiently. Since the constraints consist of a two-dimensional half line of selected body parts, the distance between given constraints and a sample posture can be calculated from the average of the distance between the half line of the constraints and the position of the corresponding body part in the sample posture. Once the corresponding cell is determined, the distance between each sample posture in the cell and given constraints is calculated to determine the blending weight of each sample posture. By blending the sample postures with the weights, a posture is synthesized. For posture blending, rotation of each join in the sample postures is blended using quaternions [PSS02]. For pelvis position and orientation, the relative position and orientation taken from the initial state in original motion data are blended. The blended position and orientation are added to the character's original position and orientation when the posing control begins.

The synthesized posture is generated by blending sample postures so that the posture satisfies the given constraints as much as possible. However, since the number of sample postures is limited compared with the number of possible posture configurations, there is no guarantee that the constraints are satisfied. Therefore, we further apply numerical IK to the synthesized posture. However, if body parts are moved a large distance, the modified posture may become unnatural. Therefore, we limit the distance to within a certain distance (0.2 m). In addition, the foot positions of the synthesized posture may not match the current posture. We also apply conventional IK to fix the foot position when the foot is on the ground. In our implementation, we use cyclic coordinate descent (CCD) IK [Wel93].

5. ACTION CONTROL

Action control is used to generate a character's motion with a particular form such as punching, kicking, or jumping. We prepare a specific learned model for each action. An action model is selected depending on user inputs. During action control, we control time progression by considering conditions of sample postures to generate continuous and natural-looking motions.

Executing condition for action control

The executing condition for an action model is determined according to the trajectory of a body part on the screen when the user drags the body part.

Each action model has a primary body part (e.g., the right hand for a right-hand punching action) and its average trajectory, which is calculated from the motion data used to train the action model. The primary body part is manually specified. The trajectory of the primary body part taken from each action model is projected onto the screen and compared with the input trajectory.

When the primary body part is dragged by the user, the distance between the user input trajectory and the trajectory of each action model is calculated to determine whether the action is to be initiated. A trajectory is represented by a series of pairs of the two-dimensional screen point and time. To calculate the distance between two trajectories, corresponding points from two trajectories are first determined using a dynamic programming algorithm. The average distance is then calculated from the difference in positions and difference in relative times for each pair of corresponding points:

$$D = \sum_{(i,j)\in S} \left| \mathbf{t}_{\text{input},i} - \mathbf{t}_{\text{action},j} \right| + h \sum_{(i,j)\in S} \left| t_{\text{input},i} - t_{\text{action},j} \right|, \quad (1)$$

where $\mathbf{t}_{\text{input},i}, t_{\text{input},i}, \mathbf{t}_{\text{action},j} t_{\text{action},j}$ are the positions and times of the points for the input and action trajectories, S is the set of corresponding points for two trajectories, and h is a scaling parameter. When the computed distance D is lower than a threshold, the action is initiated. This process is applied only when the input trajectory has certain length and S covers more than half the action trajectory.

In our preliminary user test, we found that users tend to drag a body part quickly when they want to execute an action while they drag body parts slowly for pose control. Therefore, we change the threshold depending on the speed of the touch input. When body parts are moved quickly, an action is initiated even if the distance between the trajectories is large.

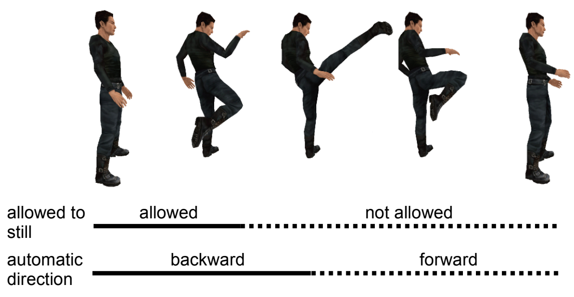

Figure 5. Conditions during an action.

The three-dimensional trajectory of each action model is projected onto a screen. Therefore, determining action execution is view-dependent. Depending on the current viewpoint, the trajectory may be normal to the screen and the projected trajectory may be too short. This makes the execution of action difficult and introduces recognition errors. For example, it is difficult to draw the trajectory of the hand for a punching action when the character is facing the viewpoint. To solve this problem, we compute a three-dimensional vector that represents the trajectory of the action model in advance and calculate the angle between the representative vector and viewpoint vector (eye vector). If the angle is smaller than a threshold (45 degrees in our implementation), the action is excluded from the action execution process. For example, when the user wants a character to perform a punching action, the user must control the viewpoint so that the user can see the character from the side or top before the user drags the character's hand, because a punching action cannot be executed when the view is toward the front or back of the character. We believe that this is a reasonable constraint because it is impossible to draw a trajectory of punching action from the front or back anyway, and it is natural for the user to change viewpoint so that the user can draw a trajectory properly. Figure 4 shows the trajectories of action models. The red trajectories represent executable actions while the pink trajectories represent non-executable actions from the current viewpoint.

Conditions on sample postures

We also use style-based IK as a fundamental technique for action control. During action control, the generated motion must follow the particular form of action. For example, when the user drags the character's right hand forward quickly during a punching action, the character should not suddenly move the hand to the specified position but should perform the punching motion until the right hand reaches the specified position. Moreover, action is not simply executed forward; it can also be stopped or executed backward depending on the user's control.

However, a stopping action should not be allowed when it results in a physically impossible motion. For example, a real human cannot stop still during a kick or jump. Even when the user stops controlling (i.e., all fingers are removed from the screen), a proper motion should be generated. The action should be either cancelled by being executed backward or finished by being executed forward automatically.

To realize such control, each sample posture has two independent conditions:

- a stillness condition that indicates whether the character is allowed to become still when the touch input stops moving but stays on the screen, and
- an automatic execution direction that indicates whether motion is to be executed forward or backward when the touch input stops and leaves from the screen.

To set these conditions, we specify the time segments of conditions on each motion as shown in Figure 5. For all motions associated with an action, the same types and order of segments are specified. Each sample posture is expressed in generalized time (0 to 1). In Figure 3(b), the color of the dot represents the generalized time of the sample posture (blue to red). Although it may be possible to determine these conditions automatically, we currently chose to set this information manually, since it does not take much time.

The learned model for each action is constructed in the same way as the model for posing control. Postures derived from motion data for the action model are used. In our implementation, we use a few (one to four) motions for each action model. An example of latent space for an action (kicking) is shown in Figure 3(b), where series of postures (dots) are connected and the color of the dots represents the generalized time (red to blue).

Posture synthesis during action control

During action control, the continuity of generated postures is considered. At first, in the same way as for posing control, the latent space coordinates are determined from given inputs (constraints on body parts). The constraints can include those on the primary body part and those on other body parts. Using a multi-touch device, positions or trajectories of several body parts can be specified. At the same time, the generalized time is computed from the sample postures near the coordinates and weights. If the generalized time for the synthesized posture is close to the current time, the synthesized posture is used as output. Otherwise, the current time is forwarded or rewound toward the synthesized time in a certain time interval. The interval is determined according to the execution frame rate (normally about 1/30 seconds). The same weights for sample postures taken from sample motions are used to generate a synthesized posture with the updated time.

If the state at the current generalized time does not allow stillness, the time keeps being forwarded or rewound, even if the user's finger is not moving on the screen. If the user's finger is lifted from the screen, the time is forwarded or rewound depending on the state at the current time. During such automatic forwarding or rewinding, the last weights for sample motions are kept and used for posture synthesis.

When the action control reaches the end of motion, the system switches back to the posing control. The discontinuity from switching control is fixed using tracking control.

6. TRACKING CONTROL

Tracking control is used as a post-process after posing or action control to generate continuous and physically plausible motion. We employ acceleration-based tracking to generate continuous motion and physics-based tracking to add a physical effect to the generated motion.

Physics-based tracking control has been widely used in previous research [ACS*07][ZMC*05][HP97]. Proportional derivative (PD) control is a simple and popular method that calculates an output torque for each joint according to the current and target states of the character. A physics simulation then updates the character's state according to the calculated torques. However, this approach has several problems. For PD controllers to work, appropriate gain parameters must be given for each joint. Appropriate parameters vary depending on motion and posture. Even though there are several methods for determining gain parameters semi-automatically [ACS*07][ZMC*05][HP97], it remains difficult to realize stable control, especially for highly dynamic motions such as jumping and kicking. Another problem is that the motions generated with this approach lag the target motion and are too slow and smooth. It is difficult to realize a target motion correctly while keeping the details of the target motion. To solve these problems, we only extract physical effects such as non-controlled body parts being shaken by controlled body parts from the result of physics-based tracking control. We use an acceleration-based method [Osh06] for more accurate and faster tracking control. By combining the two tracking controls, continuous and physically plausible motions are generated.

Acceleration-based tracking

Acceleration-based tracking uses PD controllers to calculate rotational acceleration according to the character's current state and the target posture calculated from posing or action control. The angular acceleration for each joint is calculated as

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

where $\ddot{\mathbf{q}}_i, \dot{\mathbf{q}}_i, \mathbf{q}_i$ are the joint rotational acceleration, velocity and rotation, respectively, $\mathbf{q}_{\text{target},\,i}$ is the target joint rotation, and k and d are gain and damping parameters. We use a quaternion to represent joint rotations. The minus symbol denotes the calculated difference between two rotations. The scaling operation scales the rotation of quaternion. Since we use rotational accelerations instead of joint torques, we can use the same k and d for all joints.

In addition to joint rotations, the spatial acceleration of the root segment (pelvis) is calculated:

$$\ddot{\mathbf{p}}_{root} = k\left(\mathbf{p}_{\text{target},\,root} - \mathbf{p}_{root}\right) - d\dot{\mathbf{p}}_{root}\,, \tag{3}$$

where $\ddot{\mathbf{p}}_{root}, \dot{\mathbf{p}}_{root}, \mathbf{p}_{root}$ are the pelvis acceleration, velocity and position, respectively.

From the calculated acceleration, the character's joint rotations and rotational velocity are updated:

$$\dot{\mathbf{q}}_i = \Delta t\ \ddot{\mathbf{q}}_i + \dot{\mathbf{q}}_i{}',\ \mathbf{q}_i = \Delta t\ \dot{\mathbf{q}}_i + \mathbf{q}_i{}', \tag{4}$$

where $\dot{\mathbf{q}}_i{}', \mathbf{q}_i{}'$ are the values for the previous frame.

Combining physics-based tracking

In parallel with acceleration-based tracking control, physics-based tracking control is carried out. Physics-based tracking uses PD controllers and physics simulation at first. In our implementation, we use Open Dynamics Engine (ODE) for physics simulation. For tracking control, we apply forces to each body segment according to the difference between current and target positions:

$$\mathbf{f}_i = k\left(\mathbf{p}_{\text{target},i} - \mathbf{p}_i\right) - d\dot{\mathbf{p}}_i\,, \tag{5}$$

where $\mathbf{p}_i, \dot{\mathbf{p}}_i$ are the position and velocity of the i-th segment, $\mathbf{p}_{\text{target},i}$ is the target position, and $\mathbf{f}_i$ is the force applied to the segment. The positions and velocities are updated according to the forces calculated in physics simulation. Other physical properties such as gravity and contact forces between the feet and ground are considered during physics simulation. These forces are automatically handled by the simulation engine.

As explained above, we extract physical effects from the results. Instead of using the posture generated from physics-based control, we extract velocities and apply the accumulation of the velocities with attenuation as physical effects. The physical effects are calculated as

$$\Delta\mathbf{q}_i = s_1 \Delta\mathbf{q}_i{}' + s_2\left(\mathbf{q}_{\text{sim},i} - \mathbf{q}_{\text{sim},i}{}'\right), \tag{6}$$

where $\Delta\mathbf{q}_i$ is the relative rotation for the physical effect, $\mathbf{q}_{\text{sim},i}$ is the joint rotation calculated in the physics simulation, $\Delta\mathbf{q}_i{}', \mathbf{q}_{\text{sim},i}{}'$ are the values for the previous frame, and s_1, s_2 are attenuation parameters (we use $s_1 = 0.2\Delta t$, $s_2 = 0.5\Delta t$). Physical effects such as vibration caused by the movement of other joints are calculated using equation (6).

We apply this physical effect to the body parts that are not controlled by the user and are not in contact with the ground. We divide the character's body into five segments: two legs, two arms and the torso. For each segment, the above condition is evaluated to determine whether the physical effects are applied as:

$$\mathbf{q}_{\text{output},i} = \mathbf{q}_i + \Delta\mathbf{q}_i\,, \tag{7}$$

where $\mathbf{q}_{\text{output},i}$ is the output joint rotation, $\mathbf{q}_i$ is the joint rotation calculated by the acceleration-based tracking control (equation (4)), and $\Delta\mathbf{q}_i$ is the relative joint rotation calculated by the physics-based tracking control (equation (6)). The same process is applied to the pelvis position and rotation. Afterward, for the legs in contact with the ground, we apply IK to fix the foot positions according to the changed pelvis position and orientation.

7. RESULTS AND DISCUSSION

We implemented the proposed method using a Windows 7 Touch API on a multi-touch-enabled PC. We are also developing an iOS (iPad/iPhone/iPod touch) version. Currently, our system has about 10 action models such as models for punching, kicking, bowing, waving a hand, jumping, and walking. The accompanying video shows various motion controls using our interface.

In a preliminarily user test, we asked a professional animator to trial our system to determine if our system can be used for animation. He understood the concept of our system and was able to use the system immediately, including the requirement of changing the viewpoint depending on the action. He noted the limitations mentioned below and that the generated posture was sometimes unnatural. This is basically due to a lack of postures that we used for learning models, and the system can be improved by adding more motion data. The professional animator agreed that a multi-touch interface is useful and noted especially that he was able to fix body parts by continuing to touch them. He commented that it might be difficult to create final motions with our system in an interactive way because the creation requires editing that is more precise, but our system is a promising tool with which to generate initial motions to be edited later with other animation systems.

In the trial, we found out that a single touch is almost enough to control various actions. In general, as several body parts must be moved in a coordinated way to realize an action, multi-touch control is considered to be necessary. However, because we introduced novel methods to realize actions such as techniques for the switching of actions and action control, various actions can be easily realized by moving a single body part. Nevertheless, multi-touch control is useful when the user wishes to add changes to postures during action by moving secondary body parts. It is also necessary to change posture specifically during posing control by imposing constraints on multiple body parts. The user sometimes wishes to fix some body parts during posing or action control, and our multi-touch interface can be used not only to move body parts but also to fix them. The beauty of our method is that it integrates single-touch and multi-touch interfaces. Many types of motion generation are possible with a single touch while detailed control is possible with multiple touching. Users can thus take advantage of both types of touch.

Similarly to other pen-based or touch-screen-based interfaces [IMH05][MCC09][Osh05][PSS02][TBvdP04], our interface has several problems relating to the input device. First, when touching the screen, the screen is occluded by the user's hands. Another problem is that the user cannot give a command for the next action while the character is still executing the previous action, since it is difficult to touch and drag a moving body part. These are fundamental problems that are difficult to solve. However, the problems can be reduced by introducing additional interfaces and methods. For example, the use of multiple screens can solve the occlusion problem. By displaying a virtual future posture on the screen and allowing input to the posture, the problem of input delay can be solved.

A limitation of our method is that users can execute only predefined actions. By adding more motion data for new actions, our system can handle other types of action. However, it is difficult to combine two actions such as waving a hand while walking. To realize this type of combination, an additional action model for each combination must be given in advance. Blending of multiple action models during action control is future work. Another issue is the control of walking and running. As shown in the accompanying video, a cycle of walking can be executed in our current framework. However, to realize accurate control of continuous walking directions and paths, an additional method is required. There are animation systems that generate walking motions according to a given trajectory [PSS02], and they can be integrated with our system.

Since there is no alternative system that allows various controls as our interface does, it is difficult to make comparisons with other methods. However, conducting a user test to determine whether novice and professional users are able to use our interface easily is future work. Application of our methods to software that uses full-body motion control of a character and testing the effectiveness of our method are also future work.

8. CONCLUSION

We proposed a multi-touch interface for character motion control. Unlike conventional interfaces with which the user can only select an action from pre-created actions, our interface allows free posture control and the performing of various predefined actions with the user's own styles. Our methods can be used in interactive applications such as computer games and online communication. Using our interface, users will be able to express themselves by moving in their own style. In addition, our methods can be used for animation. Recently, multi-touch devices such as smart phones, tablet computers, and LCD monitors with multi-touch sensors have become popular. We believe that our methods will be a powerful interface for applications on such multi-touch devices.

ACKNOWLEDGMENTS

This work was supported in part by Research Grant from Hayao Nakayama Foundation as well as Grants-in-Aid for Scientific Research (24500238) from the Japan Society for the Promotion of Science (JSPS).

REFERENCES

[ACS*07] Brian Allen, Derek Chu, Ari Shapiro and Petros Faloutsos. On the Beat! Timing and Tension for Dynamic Characters. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2007, pp.239-247, 2007.

[BNT08] Ronan Billon, Alexis Nedelec, Jacques Tisseau. Gesture Recognition In Flow based on PCA Analysis using Multiagent System. ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2008 (ACE 2008), pp. 139-146, 2008.

[DYP03] Mira Dontcheva, Gray Yngve, Zoran Popovic: Layered Acting for Character Animation. ACM Transactions of Graphics

(SIGGRAPH 2003), Vol. 22, Issue 3, pp. 409-416, 2003.

[GMH*04] Keith Grochow, Steven L. Martin, Aaron Hertzmann, Zoran Popović. Style-based Inverse Kine-matics. ACM Transactions on Graphics, Vol. 23, Issue 3, pp. 522-531, 2004.

[HP97] Jessica K. Hodgins, and Nancy S. Pollard. Adapting Simulated Behaviors For New Characters. SIGGRAPH 1997, pp.153-162, 1997.

[IMH05] Takeo Igarashi, Tomer Moscovich, John F. Hughes. Spatial Keyframing for Performance-driven Animation. ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2005, pp. 253-258, 2005.

[IWZ*09] Satoru Ishigaki, Timothy White, Victor Zordan, C. Karen Liu. Performance-Based Control Interface for Character Animation. ACM Transactions of Graphics (SIGGRAPH 2009), 28(3), Article No. 61, 2009.

[KGP02] Kucas Kovar, Michael Gleicher, Fedderic Pighin. Motion Graphs. ACM Transactions on Graphics (SIGGRAPH 2002), Vol. 21, Issue 3, pp. 473-482, 2002

[KN10] Michael Kipp, Quan Nguyen. Multitouch Puppetry: Creating coordinated 3D motion for an articulated arm. ACM International Conference on Interactive Tabletops and Surfaces 2010, pp. 147-156, 2010.

[KHS*08] Markus Krause, Marc Herrlich, Lasse Schwarten, Jens Teichert, Benjamin Walther-Franks. Multitouch Motion Capturing. ACM International Conference on Interactive Tabletops and Surfaces 2008, 2 pages, 2008.

[LNS05] Joe Laszlo, Michael Neff, Karan Singh. Predictive Feedback for Interactive Control of Physics-based Characters. Computer Graphics Forum (EUROGRAPHICS 2005), Vol. 24, No. 3, pp. 257-265, 2005.

[LLZ*09] Xiubo Liang, Qilei Li, Xiang Zhang, Shun Zhang, Weidong Geng, Performance-Driven Motion Choreographing with Accelerometers, Computer Animation and Virtual Worlds (CASA 2009), Volume 20, Issue 2-3, pp. 89-99, 2009.

[MCC09] Jianyuan Min, Yen-Lin Chen, Jinxiang Chai. Interactive Generation of Human Animation with Deformable Motion Models. ACM Transactions on Graphics, Vol. 29, Issue 1, Article No. 9, 2009.

[Osh05] Masaki Oshita. Motion Control with Strokes. Computer Animation and Virtual Worlds, Vol. 16, Issues 3-4, pp. 237-244, 2005.

[Osh06] Masaki Oshita. Motion-Capture-Based Avatar Control Framework in Third-Person View Virtual Environments. ACM SIGCHI International Conference on Advances in Computer Entertainment Technology 2006 (ACE 2006), 9 pages, 2006.

[PSS02] Sang Il Park, Hyun Joon Shin, Sung Yong Shin. On-line locomotion generation based on motion blending. ACM SIGGRAPH Symposium on Computer Animation 2002, pp. 105-111, 2002.

[SL06] Hyun Joon Shin, Jehee Lee. Motion Synthesis and Editing in Low-Dimensional Spaces. Computer Animation and Virtual Worlds (CASA 2006), Volume 17, Issue 3-4, pp. 219-227, 2006.

[TBvdP04] M. Thorne, D. Burke. M. van De Panne. Motion Doodles: An Interface for Sketching Character Motion. ACM Transactions of Graphics (SIG-GRAPH 2004), Vol. 23, Issue 3, pp. 424-431, 2004.

[Wel93] Chris Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. M.Sc Thesis, Simon Fraser University, 1993.

[ZMC*05] Victor B. Zordan, Anna Majkowska, Bill Chiu, Matthew Fast. Dynamic Response for Motion Capture Animation. ACM Transactions of Graphics (SIGGRAPH 2005), Vol. 24, Issue 3, pp. 697-701, 2005.