EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Editors)

Pen-to-mime: A Pen-Based Interface for Interactive Control of a Human Figure

Masaki Oshita

Kyushu Institute of Technology

ABSTRACT

This paper presents a pen-based intuitive interface to control a virtual human figure interactively. Recent commercial pen devices can detect not only the pen positions but also the pressure and tilt of the pen. We utilize such information to make a human figure perform various types of motions in response to the pen movements manipulated by the user. A figure walks, runs, turns and steps along the trajectory and speed of the pen. The figure also bends, stretches and tilts in response to the tilt of the pen. Moreover, it ducks and jumps in response to the pen pressure. Using this interface, the user controls a virtual human figure intuitively as if he or she were holding a virtual puppet and playing with it.

In addition to the interface design, this paper describes a motion generation engine to produce various motions based on the parameters that are given by the pen interface. We take a motion blending approach and construct motion blending modules with a set of small number of motion capture data for each type of motions. Finally, we discuss about the effectiveness and limitations of the interface based on some preliminary experiments.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics] Methodology and Technology - Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation.

1. Introduction

There are many demands for an interactive motion control of a virtual character on desktop environments. However, this has been a difficult challenge in the computer graphics field. Since human figures have a large number of degrees of freedom (DOF) and their movements are complicated, it is not easy to control them through a common device that has only a small number of DOF such as a mouse or gamepad. Using a motion capture equipment, full body motion of a virtual character can be directly controlled by mapping the movements of an actor to the virtual character. However, motion capture is expensive and needs a large space. They are not suitable for a desktop application. Currently most of interactive applications such as computer games employ a traditional input device such as a mouse, keyboard or gamepad. Therefore, the range of control is very limited. Although it is possible to control complex motions by combining multiple gamepads and/or mouse, the user needs a practice to learn such an interface design since the mapping from the multiple input devices to the movements of a controlled figure may not be intuitive.

In this paper we present a pen-based intuitive interface to control a virtual human figure interactively. The key idea is that we use a pen as a metaphor of a figure and map pen moments to the figure motion (Figure 1). Recent commercial pen devices can detect not only the pen positions but also the pressure and tilt of the pen. We utilize such information to make a human figure perform various types of motions in response to the pen movements manipulated by the user. A figure walks, runs, turns and steps along the trajectory and speed of the pen. The figure also bends,

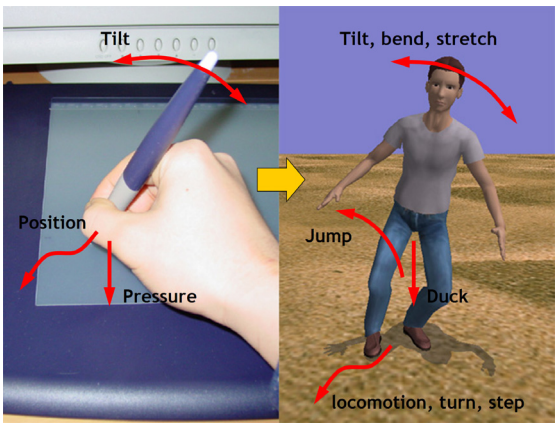

Figure 1: *The pen-based interface. The figure movement is associated with the pen manipulated by the use. The positions, pressure, and tilt of the pen is used to make the figure perform various motions.*

stretches and tilts in response to the tilt of the pen. In addition, it ducks and jumps in response to the pen pressure. Using this pen-based interface, the user controls a virtual human figure intuitively as if he or she were holding a virtual puppet and playing with it.

The main targets of this work are interactive applications that especially require various types of locomotion control such as telexistence [Tac03], virtual theaters [PG96], multi-user environments, computer games, etc. Current common gamepad devices are enough for a simple locomotion control. However the combination of multiple devices for complex locomotion including tilt, bend, duck, and jump is not intuitive. Our interface is suitable for such a complex control. For application-depending other tasks such as picking up an object or opening a door, we can use a button or voice interface with our system since these command does not require quantitative control. In addition, our interface is interesting and easy to use even for kids. The users can express their feelings by making a virtual figure move in their ways through our pen-based interface.

In addition to the interface design, this paper describes a motion generation engine to produce various motions based on the parameters that are given by the pen interface. We take a motion blending approach and construct motion blending modules with a set of small number of motion capture data for each type of motions: standing movements, locomotion, turn, step, and jump. We also introduce a motion transition scheme and foot constraints for generating continuous and natural motions. Finally, we introduce some preliminary experiments and discuss about the effectiveness and limitations of the interface.

This paper organized as follows. Section 2 reviews some related works. Section 3 introduces the pen-based interface from the user's viewpoint. Section 4, 5, and 6 explains system overview, interface implementation, and motion generation implementation, respectively. Finally, we discuss about our interface based on some user experiments in Section 7. Section 8 concludes this paper and shows future works.

2. Related work

In this section, we discuss related works from two points of view: motion control interface and locomotion generation.

2.1 Motion control interface

Many researchers have developed locomotion systems that generate a realistic walking or running motion along a user specified path [SM01][Gle01][KGP02][LCR*02]. On these systems, the user can draw a path using a mouse or pen device, a walking or running motion is then planned and executed based on the given trajectory. However, in most of the systems, the user can only specify the trajectory of locomotion but speed or styles. They focus on generating natural walking motion along a given curved path or/and on a curved terrain rather than sophisticated user interface or making various types of motions. Recently, Thorne et. al. [TRP04] developed an animation system that generates various types of motions based on "gestures" that are drawn by the user along with a locomotion path. In these systems, a motion is usually generated after an entire path is given. Therefore, the user cannot control the figure during a motion interactively.

Some researches employ input devices that have multiple DOF for animation. Oore et. al. [OTH02] used two bamboo tubes in which a 6 DOF magnetic tracker is embedded for each tube. Dontcheva et. al. [DYP03] used small widgets and a camera-based motion capture system. In these systems, the movements of a device are directly mapped to a part of the subject body. However, virtual figures have more DOF than such input devices. To solve this problem, they proposed layered editing techniques. By repeating specifying the movements for each body parts [OTH02] or from an abstract motion to detailed motions [DYP03], complex motions are composed interactively in their systems. These systems are motion editing tools rather than motion control interface. Since they need some iteration to generate one motion, they are not suitable for a motion control interface on an interactive application.

Laszlo et. al. [LPF00] proposed an animation system to control an articulated figure interactively through a mouse or keyboard by introducing a physics-based model. They mapped an input device to the key DOF of a figure, then the movements of all DOF are simulated based on physics-based model. As result, continuous and natural-looking motions are generated interactively using a common input device. However, this method is not suitable for generating complex motions such as various styles of locomotion since in such motions many DOF of a figure should be controlled cooperatively rather than just follow the physics low.

While above methods maps acquired data from some input devices to a particular DOF of a target articulated figure, our method use them as an abstract parameters (speed, angle, tilt, etc.) for motion generation modules that are constructed in advance. Our system is aimed at "motion control" rather than "motion editing". As the matter of the number of DOF, there are some devices that have more DOF than pen device. For example, a magnetic sensor has 6 DOF (position and rotation of sensor) and so does 3D mouse [Spa]. However, we think that pen device is more suitable for motion control of biped figure because pen device gives the user some intuitive physical feedback. When a pen is pressed on the tablet, it gives back reactive forces to the user. When it was tilted, the user feel the moments caused by the vertical axis of the pen as discussed in [OTH02]. These feedback forces help the user to sense the state of the figure. In addition to those benefits, pen devices are more convenient and inexpensive than other devices.

There are some techniques utilizes much higher DOF from a device such as foot pressure sensor pad [YP03] or silhouette image of a human figure from camera vision [LCR*02]. These systems use an input data to find an appropriate motion from a database using a similarity search rather than just use the input to control a figure's DOF directly. Therefore, it is difficult to control the resulting motion subtly through the systems. Moreover, they need a

large space as much as a motion capture. Monkey2 [Mon], a articulated figure device that has the same DOF with human body, is suitable for a key posture specification in an off-line animation editing but for an interactive motion control.

Davis et. al. [DAC*03] proposed a sketching animation system. This system extracts 3D keyframe postures of the skeleton from a sequence of 2D images that are drew by the user for each desired keyframe. Although their work and ours have a similarity that both use a pen device as interface, we aim at totally different technique. Our system uses a pen as an input device and as a metaphor of a human figure rather than a drawing tool. Lately many researchers use pen devices as an intuitive interface for graphic system [Iga03] and a research for the effectiveness of the pen pressure in 2D GUI is also reported [RBB04]. However, to author's knowledge, no previous method utilize multidimensional input that is acquired from pen device such as pressure and tilt for interactive motion control.

2.2 Locomotion generation

Locomotion is a complex motion and many motion generation techniques that are targeted for locomotion have been developed by researches. However, many existing methods can change resulting motions through very few parameters and lack controllability. Most of them [SM01][Gle01][KGP02] generate a walking motion based on just a given path. Therefore, these methods cannot be used in our system because we attempt to change motions based on multi-dimensional parameters that are given from a pen device (speed, angle, bend, tilt, and duck). In addition, we expect to generate motions of various character (e.g. man, woman, child, elder, fashion model, soldier, etc) by replacing the motion data set.

To address these problems, we need to take an appropriate approach first. Existing locomotion generation techniques are categorized into four approaches:

1. Procedural motion generation [BTT90]
2. Physics-based motion generation [BC89][LPF96]
3. Motion database or motion graph [Gle01][KGP02] [LCR*02]
4. Motion blending [PSS02][RCG98][SM01][TSNN01] [WH99]

First, the procedural approach is to generate walking motion based on some experimentally designed functions. It is difficult to adapt to various types of characters because locomotion models are carefully tuned and designed for a particular character and difficult to be modified. Second, the physics-based approach is similar to the procedural method except that it combines physics-based dynamic simulation. By controlling joint torques based on walking motion trajectories and using a dynamic simulation, physically-correct motion is generated. However, this method is more difficult to adapt to wide range of motions because the parameters of controllers should be tuned by hand. The third and fourth approach uses a set of motion capture data. Because they are based on actual human motion data, resulting motions are expected to be realistic. In addition, these methods can be adapted to various characters by changing the data set. The difference between two approaches is that motion blending methods use some motion data at the same time and generate a new motion by blending the multiple motion data while the other approach selects an appropriate motion segment from a data set and adjusts it. Lately, the motion graph approach is commonly used by many researches [Gle01][KGP02]. This approach seems to be easy to use since motion data do not have to be edited, aligned, and parameterized. However, if the parameter space is large, it is difficult to apply this approach because many motions are required. Moreover, using this method, it is difficult to satisfy a set of given parameters correctly. Therefore, we decided to take the motion blending approach.

The motion blending methods are further categorized into two methods: local and global blending. A local blending method [WH99][SM01][TSNN01] uses only n+1 example motions that are close to the given parameter in a n-dimensional parameter space. This method ensures the resulting motions are close to the original motions. However, this method needs much example motions especially when the dimension of the parameter space is large. A global blending method [RCG98][PSS02] uses the whole example motions to generate resulting motions. Although this approach does not need so much sample data, the sample data should be well scattered so that they represent the motion well over the parameter space. The local blending method allows only interpolation but exploration while the global blending method allows both. In addition, on the local blending method, it is difficult to change motion parameters during a motion since the motion set may be changed. Based on these features, we decided to use the global blending approach because we use a higher dimensional parameter space and a smaller numbers of example motions are desired.

3. User interface

In this section, we describe the user interface design of our system from the user's view point. The algorithm for generating animation based on user input will be described in the following sections.

The interface was designed so that it enables many kinds of motion as much as possible and also is intuitive especially for novice users.

3.1 Tablet device

Among some pen and tablet devices available, we used a commercial product, intuos 2 from Wacom [Wac] in our system. Intuos 2 is one of the popular products and detects the status of various inputs from a pen as listed in Table 1. The position data are absolute values at where the pen is pointing in the tablet coordinates. When the pen touches the tablet, some positive pressure value is acquired. While the pen being in the air, the pressure value is zero. The altitude and azimuth are the tilt of pen. Unfortunately, only 2 DOF of pen rotation are available in Intuos 2 (Figure 2) and the direction that the figure facing cannot be controlled

Table 1: *Input data available from the tablet device.*

input type	resolutions	main use in our interface
x-position	0.01 mm	locomotion, turn, steps
y-position	0.01 mm	
pressure	1024	duck and jump
altitude	-90~90 (64)	bend, stretch, tilt
azimuth	0~360 (64)	
buttons	2	

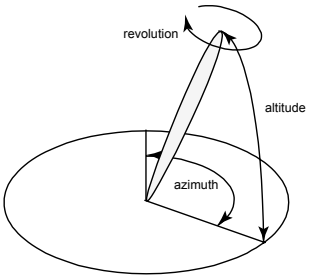

Figure 2: *The pen tilt information from intuous 2. Only altitude and azimuth are detectable but revolution.*

by using the revolution of the pen. Therefore, we introduce a scheme to determine whether the figure faces the direction that the pen is moved or keeps its direction while backward or lateral step as described in section 3.2.5. Although negative altitude values are acquired when the pen is inverted upside down, our system only uses positive values of altitude. The positions, altitude, azimuth and button values are acquired even while the pen does not touch the tablet unless the pen moves more than about 1 cm above the tablet. If the pen leaves far away from the tablet, no input data is captured. These input data from tablet device are easily captured in a user program using the Wacom tablet library [Wac].

Although our interface is designed for the input data that are available in intuos 2, any other kind of tablet device that detect the same kinds of input data will be used successfully in our system with small adjustments. Although some economical tablet device cannot detect the tilt of the pen, our interface works with them successfully except only some motions that relay on the pen tilt are omitted.

3.2 Pen-based interface for motion control

Since we attempt to use a pen as a metaphor of a human figure, the interface is designed so that a figure intuitively responds the movements of the pen manipulated by the user.

3.2.1 Locomotion

The main movement that is realized through this pen interface is locomotion which includes various styles of walking and running. The figure walks or runs in the same direction with the pen movements. The speed of locomotion changes based on the speed of the pen movements during motion. If the pen is moved slowly, the figure walks (Figure 4 (a)). On the other hand, if the pen is moved quickly, the figure runs (Figure 4 (b)). If the pen speed closes to zero, the figure stops at the end of current step.

The locomotion direction is changed based on the angle between the direction that the figure is facing and the direction of the pen movements. If the pen moves in the same direction that the figure is facing, the figure walks or runs straight forward. If the pen moves toward different direction from the figure orientation, a curved walking or running is generated (Figure 4 (c)). If the pen moves opposite direction with the figure, it turns around for the specified direction and then starts walking or running (Figure 3).

As explained here, the figure is moved based on the relative movements of the pen. We decided to take this approach since it is important in our interface design that the figure movements is directly associated to the pen movements. However, this approach does not ensure a precise control of a target position of locomotion. We will discuss about this issue in Section 7.

3.2.2 Bend, stretch and tilt

The tilt of the pen also affects the figure motion in our interface. The figure basically rotates its body to tilt, bend and stretch based on the direction for that the pen is tilted by the user (Figure 5 (a)-(c)). This works not only in standing posture but also during locomotion (Figure 6 (a)-(c)).

3.2.3 Duck

The pen pressure is used for two purposes in our interface. The first one is duck. If the pen is pressed onto the tablet, in other words the pen pressure becomes large, the figure ducks. The depth of duck depends on the size of pressure. As long as the pressure is being applied, the figure keeps ducking. This works both in standing (Figure 5 (c)) and locomotion (Figure 6 (c)).

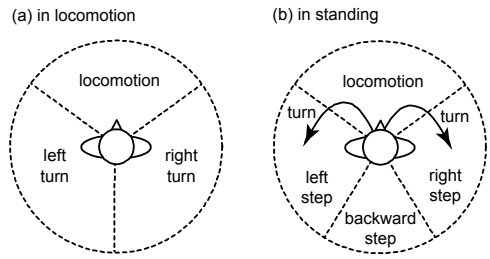

Figure 3: *Locomotion, turn, step control based on the orientation of the pen movements (a) in locomotion and (b) in standing.*

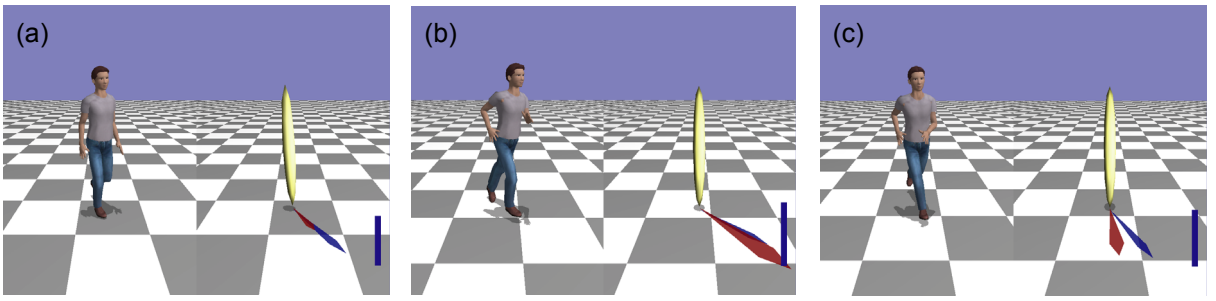

Figure 4: *Locomotion interface. The right view visualizes the movements of the pen that is manipulated by the user. The yellow stick shows pan tilt. The red arrow under the pen shows the orientation and velocity of the pen movements. The blue arrow shows the figure's orientation. And the left view shows the controlled figure. (a) The figure walks when the pen is moved slowly. (b) It runs in response to the speed of the pen movements. (c) When the directions of the pen and the figure is different, a curved locomotion is generated.*

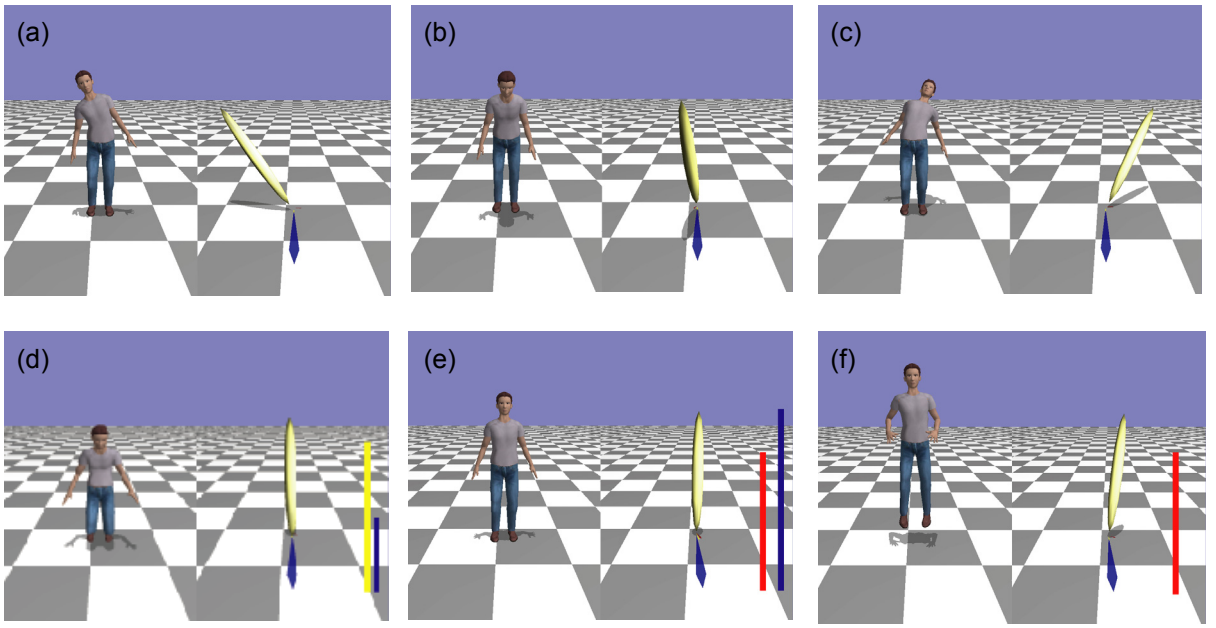

Figure 5: *The figure (a) tilts, (b)bends, and (c) stretches in response to the pen tilt. Duck and jump motion is performed based on the pen pressure. The blue bar in the left of the window shows pen pressure. The red bar next to it shows a high pass filtered pressure and the yellow bar for a low pass filtered. (d) If the pen is pressed onto the tablet for a while, the figure ducks. (e) If the pen is pressed and released quickly, (d) the figure jumps.*

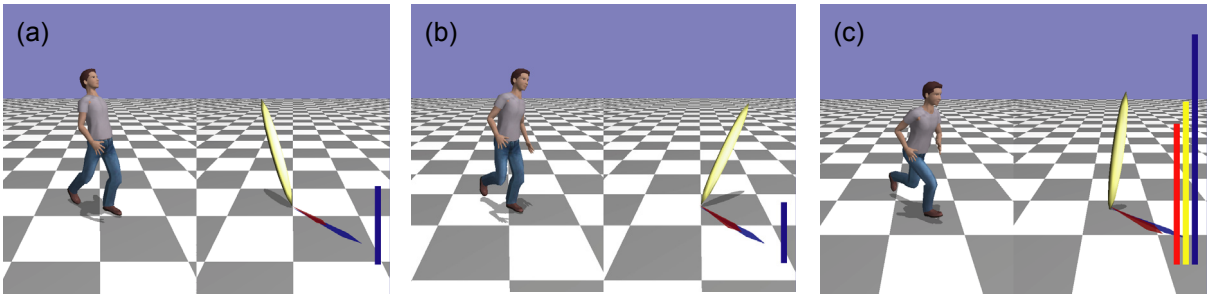

Figure 6: *The bend, stretch, tilt and duck control also works during locomotion. The images show examples of (a) stretch, (b) tilt, and (c)duck.*

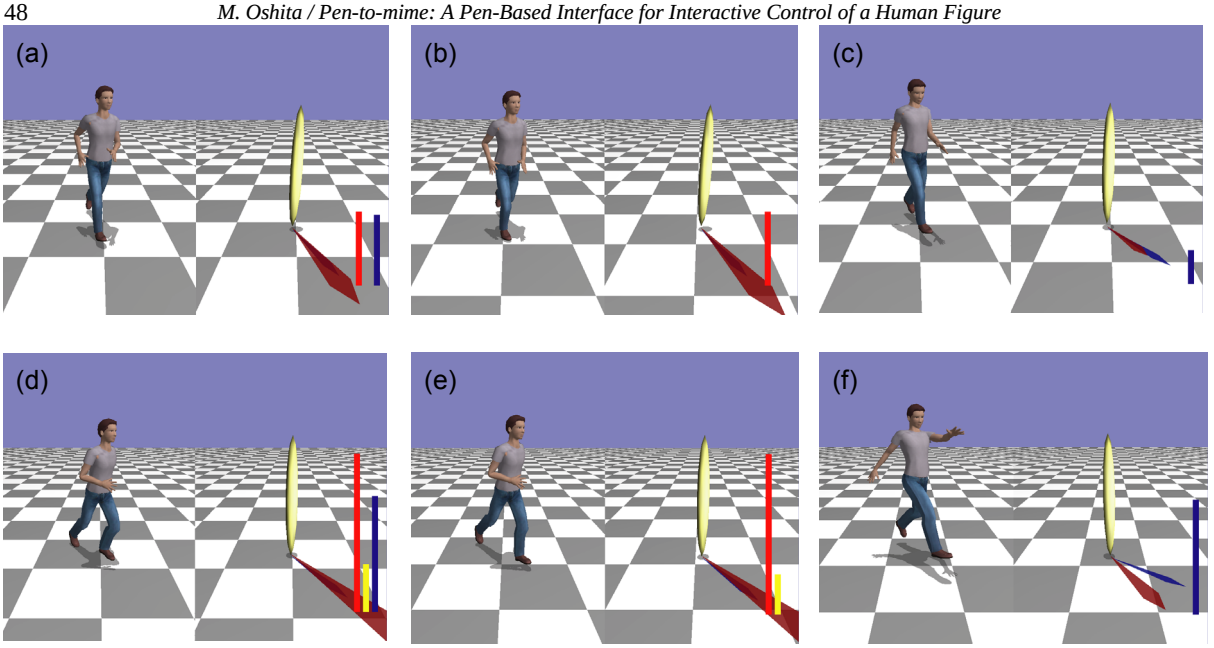

Figure 7: *Examples of jump motions during locomotion. When (a) the pen is pressed quickly and (b) then released, (c) a small jump is performed. (d)(e)(f) shows an example of high jump.*

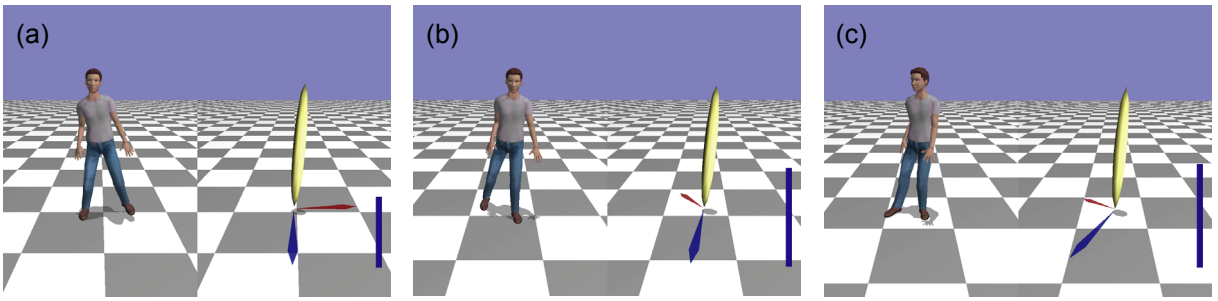

Figure 8: *Step and turn interface. The figure takes (a) lateral or (b) backward step while the figure is not walking. (c) In order to make a turn, the pen should be moved slightly frontward and then toward a desired turn direction.*

3.2.4 Jump

We also use pen pressures to make the figure jump. If the pen is pressed slowly and kept being pressed, the figure ducks. On the other hand, if the pen is pressed and released quickly, the figure jumps. The jump height is determined based on the pressure at the moment when the pen is released (Figure 7). For jump orientation and speed, the locomotion orientation and speed are automatically used when the jump is started. If the figure is not walking, the figure just jumps and lands there. Jump height (vertical movement) and orientation and speed (horizontal movement) are handled as independent parameters.

As described above, a jump motion is determined when it starts. So is the flying duration and distance of the jump. Therefore, the user cannot control the motion at all during the figure is flying. However, the system keeps detecting the pen movements during jump and it affects the motion right after the figure lands. If the user stops moving the pen after a jump starts, the figure stops when the jump motion is finished. Otherwise, the figure start walking or running again based on the pen speed. The produced motions in our system are smooth so that the user does not care the discontinuous between a jump and following locomotion.

3.2.5 Lateral and backward step

While the figure is standing, it moves differently. If the pen moves side or backward direction against the direction the figure is facing, the figure takes lateral or backward step instead of turn (Figure 8 (a)-(b)). If a user want to make the figure turn while it is standing, the user have to move the pen frontward slightly and then keep moving the pen toward a desired direction as shown in Figure 3 and Figure 8 (c). We designed this interface because we wanted to make figure both turn with orientation change and step without orientation change.

4. System overview

The structure of our system is shown in Figure 9. The system consists of the interface module and the motion generation module. The interface module interprets input data from a tablet device and sends motion parameters or motion transition command to the motion generator. The motion generation module is composed of submodules for each type of motion. Based on a current motion, one of submodules takes charge of generating motion. When a motion transition signal is sent from the interface module, the main submodule switches one to another.

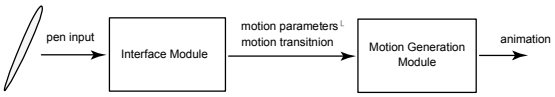

Figure 9: *System Overview.*

4.1 Motion generation modules

The motion submodules are developed based on a global motion blending method as discussed is Section 2.2 with a small numbers of example motion data. To ensure smooth blending over examples, every example in a motion module has similar patterns and the same sequence of keyframes, for example, left foot up or right foot down. Because of this reason, we separate standing jump (vertical jump) and walking jump, right turn and left turn, and right step and left step, respectively.

The parameters that each motion module takes are shown in Table 2. Some motion modules such as locomotion is designed to generate one cycle of a continuous motion, e.g. moving the right leg first and the left leg. By repeating a same motion with varying parameters, continuous motions are generated. The parameters are allowed to be changed during motion except jump as described in section 3.2.4. However, currently we do not terminate a motion before a cycle of motion has finished in order to avoid discontinuous between motions. A motion transition command from the interface module is queued in the motion generation module and is executed after the previous motion has finished. Although the rules may cause some delay between user input and figure reaction, it is not so much problem since each motion used in our system is short (approximately 1 second) and transition to next motion is done smoothly.

Table 2: *Motion modules.*

motion	parameters	# examples
locomotion	speed, angle, tilt, bend-stretch, duck	12
standing motions	tilt, bend-stretch, duck	6
jump	speed, height	4
standing jump	height	2
right turn	angle	4
left turn	angle	4
right step	angle	3
left step	angle	3
backward step	angle	3

4.2 Example motions

Motion parameters are computed from the motion data using simple approximations. Motion speed and angle are computed from the difference between the first frame and last frame of the example motion. Jump height and tilt, bend, and stretch angle are calculated from the maximum height and spine angle during the motion. To apply global motion blending, a same sequence of keyframe is manually assigned to all motions in a motion set in advance.

5. User interface implementation

This section describes the implementations of the user interface. Based on the interface design described in section 3 , motion parameters of the current motion are calculated on each animation step. The conditions for motion transition from current motion to another motion are also evaluated. In addition, if next motion is determined, the parameters for next motion are also computed to realize smooth transitions between motions. The motion transition method is described in the next section.

5.1 Filtering input data

The input data acquired from the tablet library is filtered first to reduce noise and to prevent the figure responding sensitively. For this purpose, we employ a low pass filter used in 12,

$$y_i = y_{i-1} + (1-\alpha)(y_{i-1} - y_{i-2}) + \alpha(x_i - x_{i-1}), \qquad (1)$$

where y_i is the output value at i-th frame and x_i is the measured value at i-th frame and α is a mixing parameters. This filter works to preserve the velocity change continuously. The parameter α was determined through some trial. Currently we use $\alpha = 0.1$ for input data except for the pressure that is discussed in the next subsection.

5.2 Motion parameters

The motion parameters are computed from the filtered input values. The speed and angle are computed from the velocity vector of the pen movements. Tilt, bend and stretch angles are simply computed from the pen tilt. However, the computation of duck height and jump height is not a trivial problem. As explained in section 3.2.4, when a pen pressure is quickly given and released, the figure is expected to jump instead of duck. To address this, we use two separate filters and the difference between them. Using two different α , we calculate two pressure values through two separate filters: responsive and non-responsive ones. The pressure value that is filtered by non-responsive one is

used as the duck height. The difference between responsive and non-responsive ones is used for jump height. This combination works as a kind of high pass filter. Computed parameters are used to determine the motion blending weights immediately.

5.3 Conditions for motion transition

The type of the motion that is executed after the current motion is also determined based on the motion parameters during the current motion. The system keeps computing the pen speed, angle and jump height for this purpose. First if the pen is released during the motion and jump height parameter is positive, jump motion is initiated. Second if the speed exceeds the threshold, the figure waks, turns, or steps based on the angle. Otherwise, the figure just keeps standing.

6. Motion generation

As discussed in section 2.2 , we decided to take a global motion blending approach with small modification. When a set of parameters is given, weights of all example motions are determined first. By blending example motions based on the weights, a desired motion is then generated. In addition to this motion blending, we use motion transition and constraints methods for generating continuous motions.

6.1 Motion blending

Our implementation is based on the algorithm by Park et. al. [PSS02] and Rose et. al. [RCB98]. A typical method for weigh computation is to combine linear coefficients and nonlinear coefficients for a radial basis function.

$$\mathbf{w} = \mathbf{A}\mathbf{p} + \mathbf{B}\mathbf{R}(\mathbf{p}), \quad (2)$$

where $\mathbf{w}$ is a n-dimensional vector represents the weights for example motions and n is the total number of them; $\mathbf{p}$ is a m-dimensional parameters and m is the dimension of the parameter space. First term is for blending example motions linearly with coefficients matrix $\mathbf{A}$ and the second term is for reducing error using a radial basis function.

Ignoring the second term of equation (2) and considering the conditions that $\mathbf{w}_i = 1$ is satisfied when $\mathbf{p} = \mathbf{p}_i$ where $\mathbf{p}_i$ is the parameter vector of i-th motion, each row of $\mathbf{A}$ is computed using a least square method.

A radial basis function $\mathbf{R}(\mathbf{p})$ is used for the second term. The i-th column of $\mathbf{R}(\mathbf{p})$ is computed using a faction and distance between $|\mathbf{p} - \mathbf{p}_i|$. For a radial basis function, we use the cubic B-spline function in the same way with Park et. al.13. Using the same conditions for the first term, each row of the radial basis coefficients matrix $\mathbf{B}$ to reduce the error between the actual weights and computed weights calculated using the linear term is determined by solving the linear problem:

$$\mathbf{B}\mathbf{R}(\mathbf{p}_i) = \mathbf{w} - \mathbf{A}\mathbf{p}_i . \quad (3)$$

Using equation (2) and coefficients matrix $\mathbf{A}$ and $\mathbf{B}$, weights of example motions are determined based on the given parameters.

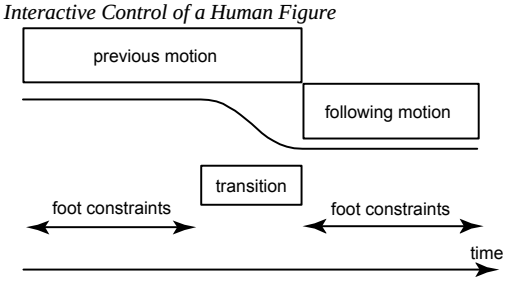

Figure 10: *Motion transition and foot constraints. The initial posture of the following motion is blended with the previous motion as the (n+1)th example motion.*

Once the weights are determined, a blended motion is computed by timewarping the example motions based on the sequential keyframes that are given in advance. For smooth blending we employ an incremental timewarping and sphere-vector space orientation blending technique that are presented by Park et. al. [PSS02].

The major differences between the original method [PSS02] and our method are follows. We use incremental technique also for the horizontal position of root instead of simply blending the positions of example motions, because blending of long step motion and short step motion may causes reverse movements of the root position. Therefore, we blend the velocities of example motion instead of positions of example motions. The height of root is computed based on the example motions and the ground height. In addition to this, we use the initial posture of the following motion as (n+1)th example for motion blending in order to realize smooth transition as discussed next (Figure 10).

6.2 Motion transition

We need to make a transition one motion to next motion. This includes repeating a same motion, such as locomotion cycle. For smooth transitions, we simply blend the terminal segment of the previous motion and the initial posture of the following motion (Figure 10). We employ the simple technique since the types of motions in our system are limited and the differences between motions are small.

6.3 Foot constraints

Constraints between the figure foot and the ground are applied. We introduced a kind of importance based end-effectors constraints [SLGS01]. Once a foot contacts the ground, a filter tries to keep it on the same position on the ground. The leg postures are computed based on the filtered end-effectors position using an analytical inverse kinematics. However, we do not apply the constraints during a motion transition phase, because the following motion sometimes become unnatural when two transition postures are different and the initial posture of the following motion is not established because of the constraints. Although our system currently generates motions on an even terrain, it is easy to retarget them on a curved terrain by controlling the heights of the foot constraints.

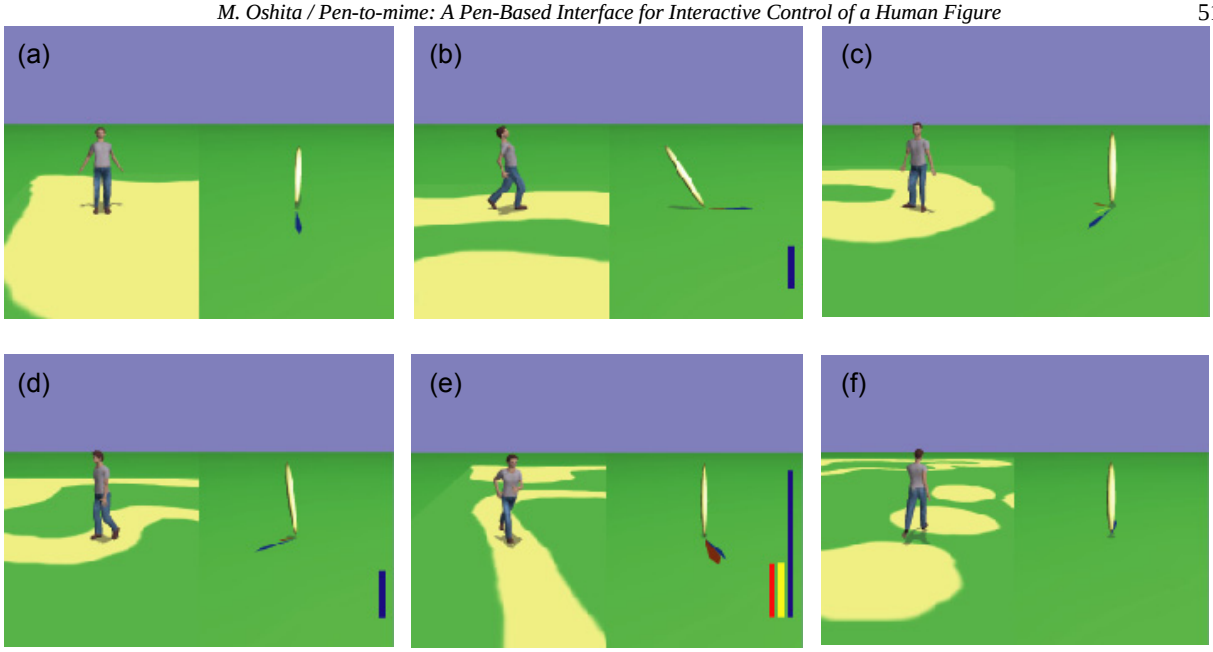

Figure 11: *Images form an example animation of pen-based interactive motion control.*

7. User experiments and discussion

We have implemented the proposed interface as a Windows application with MS Visual C++ and Wacom Tablet Library [Wac]. The program works on over 60 fps on a standard computer. Figure 11 shows an example animation. In the animation, the user was making the figure walk along the path in many styles. The camera was simply moved so that the figure was displayed on the center of the view. The orientation of the camera was fixed during the animation in order to keep the relationship between the pen direction and figure direction constant.

We asked some subjects to try our pen-based interface. Although they found that our interface design was very interesting and enjoyed it, at the first time, they have some difficulties with the control. The major reason is that our interface utilizes the multidimensional movements of the pen such as positions, pressure and tilt. For example, a user who tries to make the figure walk did not only move the pen but also tilt or press it on the tablet unconsciously. After five to ten minutes training, the users learned how to control the multidimensional input independently.

As discussed in Section 3.2.1, we currently use relative movements of a pen in order to make the figure locomotion. This interface works well when the user wants to the figure move in the specified direction. However, it was difficult to move the figure to a specified position, for example, to make it walk in a narrow route. This problem also occurred when we make the figure jump to a specific position. Since currently we cannot control the jump distance and duration, a precise control of jump is difficult, in the place as shown Figure 11 (f). To make the control efficiently on such a situation, we need to introduce some control scheme based on both absolute and relative positional information.

The system causes some delay during motion transition as explained in Section 4.1. However, this was not a serious problem since the delay is very short (approximately less than 1 sec). On the other hand, the motion parameters such as speed, angle, tilt, and duck are immediately applied during motion. Therefore, it is quite responsive.

We are now going to start more close experiments and comparison with a traditional interface. By developing a gamepad-based interface that can control the same kinds of motions with our interface using a gamepad that has multiple-sticks, we will be able to compare the two interfaces. We expect that our pen-based interface is more suitable especially for precise control and combination of multiple movements, e.g. walking with tilt.

8. Conclusion

We presented a pen-based interface for interactive motion control. Although it still had some difficulties for novice users in our experiments, the interface was intuitive and users found it very interesting. We think that our interface is also suitable for kids. Pen is a very common tool in our life and has potential for an interactive motion control interface. In addition to the intuitiveness, we expect more controllability by improving our interface and by utilizing the multidimensional input from pen including relative and absolute values.

In addition to the direct control based on the pen movements, more intelligent movements such as avoiding obstacles or working with other character will be required on some applications. We also intend to develop reaction models that generate believable reactions based on the environments in addition to input from the pen. For example, a dance application in which two figures dance coop-

eratively guided by two separate pens that are manipulated by two users is an interesting application.

Acknowledgements

The author would like to thank the anonymous reviewers for their helpful comments. This research was partially supported by the Grant-in-Aid for Scientific Research (16650022) from the Ministry of Education, Science, Sports and Culture.

References

[BTT90] BOULIC R., MAGNENAT-THALMANN N., THALMANN D.: A Global Human Walking Model with Real-Time Kinematic Personification. *The Visual Computer*, 6, 344-358, 1990.

[BC89] BRUDERLIN A., CALVERT T. W.: Goad-Directed, Dynamic Animation of Human Walking. *Computer Graphics (Proc. of SIGGRAPH 1989)*, 23, 3, 233-242, 1989.

[DAC*03] DAVIS J., AGRAWALA M., CHUANG E., POPOVIĆ Z., SALESIN D.: A Sketching Interface for Articulated Figure Animation. In *Proc. of ACM SIGGRAPH / Eurapraphics Symposium on Computer Animation 2003*, 320-328, 2003.

[DYP03] DONTCHEVA M., YNGVE G. POPOVIĆ Z.: Layered Acting for Character Animation. *ACM Transactions of Graphics (Proc. of SIGGRAPH 2003)*, 22, 3, 409-416, 2003.

[Gle01] GLEICHER M.: Motion Path Editing. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics 2001*, 195-202, 2001.

[Iga03] IGARASHI T.: Freeform User Interfaces for Graphical Computing. I*n Proc. of 3rd International Symposium on Smart Graphics*, 39-48, 2003.

[KGP02] KOVAR L., GLEICHER M., and PIGHIN F.: Motion Graphs. *ACM Transactions of Graphics (Proc. of SIGGRAPH 2002)*, 31, 3, 473-482, 2002.

[LPF96] 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.

[LPF00] LASZLO J., VAN DE PANNE M., and FIUME E.: Interactive Control For Physically-Based Animation. In *Proc. of SIGGRAPH 2000*, 201-208, 2000.

[LCR*02] LEE J. CHAI J., REISTMA P., HODGINS J., POLLARD N.: Interactive Control of Avatars Animated with Human Motion Data. *ACM Transactions of Graphics (Proc. of SIGGRAPH 2002)*, 22, 3, 491-500, 2002.

[Mon] Monkey2, Digital Image Design Inc. http://www.didi.com/www/areas/products /monkey2/

[OTH02] OORE S., TERZOPOULOS D. HINTON G.: A Desktop Input Device and Interface for Interactive 3D Character Animation. In *Proc. of Graphics Interface 2002*, 133-140, 2002.

[PSS02] 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.

[PG96] PERLIN K., GOLDBERG A.: Improv: A System for Scripting Interactive Actors in Virtual Worlds. In *Proc. of SIGGRAPH 1996*, 205-216, 1996.

[RBB04] RAMOS G., BOULOS M., BALAKRISHNAN R.: Pressure Widgets. *ACM CHI Conference on Human Factors in Computing Systems 2004, ACM CHI Letters*, 6, 1, 487-494, 2004.

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and Adverbs: Multidimensional Motion Interpolation. *IEEE Computer Graphics and Applications*, 18, 5, 32-40, 1998.

[SLGS01] 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.

[SM01] SUN H. C., METAXAS D. N.: Automating Gait Generation. In *Proc. of SIGGRAPH 2001*, 261-270, 2001.

[Spa] Spaceball. 3Dconnexion, http://www.3dconnexion.com

[Tac03] TACHI S.: *Telecommunication, Teleimmersion and Telexistence*, Ohmusha, IOS Press, 2003.

[TRP04] THORNE M., BURKE D. VAN DE PANNE M.: Motion Doodles: An Interface for Sketching Character Motion, *ACM Transactions of Graphics (Proc. of SIGGRAPH 2004)*, 23, 3, 2004.

[TSNN01] TSUMURA T., SOHIZUKA T., NOJIRINO T., NOMA T.: T4: A Motion-Capture-Based Goal-Directed Realtime Responsive Locomotion Engine. In *Proc. of Computer Animation 2001*, 52-60, 2001.

[Wac] WACOM Technology Co. http://www.wacom.com

[WH99] WILEY D. J., HAHN J. K.: Interpolation Synthesis of Articulated Figure Motion. *IEEE Computer Graphics and Applications*, 17, 6, 39-45, 1999.

[YP03] YIN K. K., PAI D. K.: FootSee: an Interactive Animation System. In *Proc. of ACM SIGGRAPH / Eurapraphics Symposium on Computer Animation 2003*, 329-338, 2003.