Real-time Character Motion Control Using Data Gloves

Nik Isrozaidi Nik Ismail
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
820-8502 Japan.
Phone: +81-948-29-7738

isrozaidi@cg.ces.kyutech.ac.jp

Katsuya Ishiguro
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
820-8502 Japan.
Phone: +81-948-29-7738

ishiguro@cg.ces.kyutech.ac.jp

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka,
820-8502 Japan.
Phone: +81-948-29-7718

oshita@ces.kyutech.ac.jp

Abstract

For computer games, communications using avatars, and real-time animation systems, users want to move a character freely in a virtual world. However, the flexibility of the current motion control interface is very limited because currently character motion is simply controlled with pre-defined motion data. In this paper, we present a motion control method that uses two data gloves as an input device, making a virtual character perform various motions. Each part of the character's body is controlled using input from data gloves. For example, a user can control the character's left arm and left leg using their left hand. However, there are limited degrees of freedom using data gloves to control all the character's body parts directly. Moreover, it is difficult for users to perform complex motions such as stepping or jumping because multiple body parts have to be controlled at the same time. In order to solve these problems, we introduce three novel ideas. First, we change the mapping between the user's hand and the character's body parts dynamically. For example, when the both hands are moving in the same direction, they are used to control the pelvis instead of arms or legs. Then, we introduce a manual switch between arms and legs. A hand is used for controlling the arms or the legs by switching the mode manually with a finger. Second, we use mechanisms of real puppets to control multiple body parts synchronously. Third, we combine the motion data based control with the position based control. When it is needed, a motion is selected from the pre-defined motion data, and is then executed. We use this approach for locomotive motion such as rightward step, leftward step, backward step, and jump. A motion is selected when the character's waist is moved more than certain distance in a given direction. Using this approach, complex motions can be performed easily.

Keywords

Character animation, interface, data glove

1. Introduction

Nowadays, real-time character motion control is widely studied for computer games, virtual reality, humanlike communications using avatars, and real-time animation systems. Character motion interfaces are an important part of these studies. There are thus demands for users to control characters freely in a virtual world.

In current applications, a user can make characters perform certain pre-defined motions. The only thing that a user can do is to choose which motion should be executed. The characters' motions do not change even if they are controlled by different players. Ideally, even the same type of motion should have differences depending on the users. For example, every person walks in their own way. Therefore, each user should be able to change the style or trajectories of motion. The performed motions should reflect the personality of the user. To solve this problem, real-time inputs from the user should be used to control characters.

In this paper, we propose a motion control method that uses data gloves as an input device. The goal of this research is to develop an interface that will allow users to control a virtual character freely. Using our system, the user can make the character perform various movements and change the style and trajectory of motions.

There are two problems in real-time character motion control using this kind of approach. The first problem is a lack of degree of freedom (DOF) from the input device. The degrees of freedom of data gloves are limited in terms of directly controlling all of the character's body parts. The second problem is that it is difficult for users to control a complex motion such as a punch or a kick because multiple body parts have to be controlled synchronously for such motions. In order to make a character perform these kinds of motions, users must control arms and legs in coordination.

In order to solve these two problems, we introduce three novel ideas.

First, in order to solve the first problem, we change the mapping between the user's hand and the character's body part dynamically. This means that users can change the focus on the character. In this position based control, the mapping is changed automatically or manually depending on the mode. The user can manually switch from leg control to arm control by bending their index finger. When the user moves both hands in the same direction, the system automatically switches to waist control instead of arm or leg control.

Second, in order to solve the problem of lack of freedom, we introduce a synchronous control inspired by the mechanisms of real puppets. In puppetry, both arms are connected by one string and the performer can move them simultaneously. Similarly, we introduce arm and leg symmetry based on the position of the user's hand. The user can switch to this mode by bending their thumb fingers.

Third, in order to solve the problem of complex motion, we combine the motion data based control with the position based control explained above. Using this method, a motion is selected and used in response to user's control when it is necessary. It is based on the position of the character's waist when the user moves both hand more than a certain distance. The user can also control the character's arms using position based control while motion data is being used. For now we have applied this to the walking motion only.

Using our method, users can control each of the character's body parts separately. Users can also control the character's legs and arms simultaneously. Through our interface, locomotive motions such as rightward step, leftward step, backward step, walk, jump, and duck can be realized.

2. Previous Work

In this section, we discuss related works from the perspective of input devices and control methods.

Traditional input devices such as a mouse, gamepad, joystick, or keyboard are commonly used for controlling character motion. However, there is the lack of DOF with these traditional input devices is a problem. Therefore, the control methods that map inputs from the mouse to the character's motion are important. Laszlo and colleagues [1] developed an interactive system to control physically simulated planar characters by controlling joint torque with mouse movements. Motions of the two joints are controlled by linearly mapping the mouse position to the two desired joint angles. Then, the joint torque is computed using a proportional-derivative (PD) controller. A spatial key framing technique has been proposed by Igarashi and colleagues [2]. Using this technique, users set up a correspondence between a 3D position and the character's posture. The users dragged a 3D marker to create a performance by interpolating among the corresponding character poses. In order to perform different kinds of motion, they would need to change the motion data. Using our interface, users can perform and control various types of motions because we map the user's hand to the character's body parts dynamically.

The full body motion of a user acquired with a camera input device has also been used to control characters. Freeman and colleagues [3] used image moments and optical flow to detect the user's movement and allowing the user's motion to be directly mapped to the character's motion. However, they didn't control each part of the character's body. They just made the character perform pre-defined motions according to the user's gestures, such as leaning or walking in different directions. Using our interface, each part of the character's body parts can be controlled. A sparse marker set worn by the user was built by Chai and Hodgins [4]. They construct a motion graph [5] from pre-recorded motion data, and search for motion segments that are close to the current marker positions to reconstruct the full-body motion of the user. However, a large space is required to operate these systems. Furthermore, the motion capture equipment is expensive. Using our interface, only a small space is required. Our equipment is also cheaper than theirs.

Data glove based interfaces have also been researched. Komura and Lam [6] proposed a method to control the walking motion using the joints of the index and middle fingers and the position of a data glove. They proposed a new method to map the finger motion of the user to the locomotion of 3D characters using preprocess data. However, other motions and whole bodies cannot be controlled. Using our interface, users can control each part of the character's body. Okada [7] proposed a motion generation method using a puppet mechanism. This method uses a data-glove and a magnetic sensor as an input device. A user creates the motion of an articulated figure by his/her finger action like controlling a puppet. He also introduces the physical constraint of gravity and ground contact to generate the natural motion of an articulated figure. However, the hands, feet, and neck of the character can only be moved in one direction. Using our interface, users can control each of the character's body parts in three dimensions. Users can also control the character's legs and arms synchronously.

3. User Interface

3.1 Input Device

We used P5 data gloves from Essential Reality Co. Ltd. [8] as an input device in our implementation. The infrared sensors detect the position and orientation of the wrist. Our method is applicable to any other data glove device.

3.2 Interface Design

On our interface design, each part of the character's body is controlled using corresponding hand positions from the data gloves. Figure 1 shows the mapping between a user's hands and their character’s body parts. The character’s arms and legs are controlled by the corresponding hand position of the user. The user’s right hand controls the right leg and right arm of the character (labeled as 1 in figure 1), while the user’s left hand controls the left leg and left arm of the character (labeled as 2 in figure 1). The waist is also controlled by hand positions (labeled as 3 in figure 1). Switching modes between arm control, leg control and synchronous control is labeled as 4 in figure 1. These modes will be explained below.

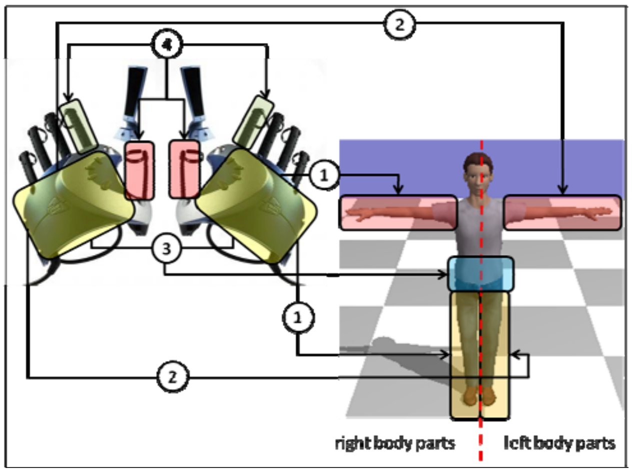

Figure 1 Mapping scheme between the user’s hands and the character’s body parts

We introduce control modes based on the ideas explained in section 1. The control mode can be changed manually or automatically. Index fingers and thumbs are used for mode switch control. Figure 2 shows the relationship between character body parts and the control modes. In this figure, the small boxes represents character body parts and the mode for the controlled body part, while the outer boxes represent modes for the control method. Transitions between character body parts control and control modes are represented with arrows and labeled with A, B, C and D. Each of these control modes is explained below.

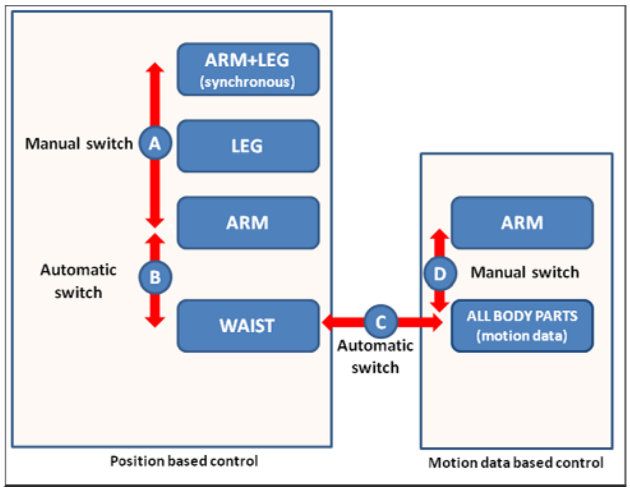

Figure 2 Control mode

During the position based control mode, users can control the character’s arm or leg by changing one of their hand positions. The right hand and the left hand are used independently. The right arm and leg are controlled with the user’s right hand, while the left arm and leg are controlled with the user’s left hand. The user can switch from leg control to arm control by bending their index fingers (labeled as A in figure 2). Using this interface, the user can control the character’s legs and arms separately.

As shown in figure 2 (labeled as synchronous), users can also control the character’s legs and arms simultaneously in the position based control mode. This is inspired by the puppet mechanism as explained in section 1. Users can switch to synchronous character control manually by bending their thumbs (labeled as A in figure 2). During this operation, the character's arms and legs both move based on the position of the user’s hands. Arm movements are synchronized with the opposite leg. The left leg moves symmetrically with the user’s right hand and the right leg moves symmetrically with the user’s left hand. We choose to move them symmetrically because human beings usually move so that the body balance is maintained, so it will look natural. We introduce this mode because we want to control the character’s leg and arm in the same time. Using this mode, motions such as marching and walking in place can be realized.

In position based control mode, the transition to waist control is done automatically (labeled as B in figure 2) when the user moves both hands in the same direction. The position of the character’s waist is controlled in response to the position and direction of both the user’s hands. Using this waist control, a waist bending motion such as waist bending front, waist bending back, waist tilt right, and waist tilt left can be realized.

In addition to the position based control modes explained above, we introduce the motion data based control mode. A motion is selected and executed automatically based on user’s control when it is necessary (labeled as C in figure 2). In our current implementation, we introduce six motions, which are the rightward step, leftward step, backward step, walking, jumping, and ducking. These motions are initiated based on the character’s waist position. Using this mode, stepping, walking, jumping, and ducking motions can be realized. In addition, position based control is also applicable during motion data based control. The user can activate arm control manually by bending their index fingers (labeled as D in figure 2). A waving hand during a walking motion can be realized using the combination of these controls.

4. Implementation

In this section, we explain the implementation of our system. A character posture is represented using joint angles with the position and orientation of the waist. The character used in our system has 50 joints. The main two control modes are position based control and motion data based control.

4.1 Position Based Control

There are four control modes in the position based control as shown in figure 2 (labeled as position based control). For arm and leg control including synchronous control, we use inverse kinematics for posture calculation. For waist control, we use posture blending.

4.1.1 Arm and Leg Control

In position based control, we use an analytical inverse kinematics (IK) [9] to calculate the character’s arm and leg postures based on the user’s hand position. For example, the angles of the joints in an arm (wrist, elbow, and shoulder) are calculated based on hand position and orientation. The hand position is computed from the user’s hand position. On our current implementation, the hand orientation is computed so that wrist joint angles are zero. This is because it is difficult for the user to change the orientation of their hand. The angles of the joints in a leg (ankle, knee, and hip) are computed from the foot position, which is computed from the user’s hand position in the same way.

In order to calculate the character’s hand position, the system converts the user’s hands positions to the character’s hand position. Usually, the character’s range of motion is larger than the user’s data glove workspace. Therefore, we scale the user’s hands positions from the workspace to the range of motion. We manually specify the sizes of the workspace and the range of motion. Based on these parameters, the position of the user’s hand is converted to a position in the character’s range of motion.

Similarly, we can compute a leg posture based on the feet position, which is computed from the user’s hand. However, for leg control, only one leg can be controlled at a time. This constraint is necessary because at least one of the character legs should be touching the ground to make it look natural. This constraint is applied when the user tries to move both legs using leg control. The system automatically determines which leg to be controlled and which leg to be fixed based on which hand moves more than the other.

In synchronous control, we compute an arm and opposite leg based on the hand position. The character’s arm and leg posture are computed differently as mentioned above. For the leg posture, the lateral translation is inverted from the hand position because the leg moves on the opposite side.

4.1.2 Waist Control

During the position based control mode, transition to waist control from arm or leg control is done automatically when the user moves both hands in the same direction. The position of the character’s waist is controlled in response to an average of both hand positions. The average position of the user’s hands is scaled from the workspace to the range of the character’s waist using the size parameters of the workspace and the range of motion explained above.

In this mode, the character’s postures are controlled using posture blending. We used posture blending to change the character’s whole body making the posture look natural. In order to use posture blending, key postures were prepared in advance. Our implementation currently uses four key postures created using a motion capture device. These key postures are waist bending front, waist bending back, waist tilt right, and waist tilt left.

The blending weight of each key posture $\boldsymbol{w_i}$, is based on the waist position. Each key posture has an associated waist position $\boldsymbol{q_i}$. The weight for each posture $\boldsymbol{w_i}$, is based on the distance between $\boldsymbol{q_i}$ and the current waist posture. The weights are then normalized so that they satisfy $\sum w_i = 1$. Blended posture $\boldsymbol{p}$, is computed as the weight sum of key postures expressed by $p = \sum w_i p_i$.

4.2 Motion Data Based Control

The system switches to motion data based control mode when a complex motion should be executed based on the user’s control. In this mode, a motion is selected and executed automatically. Currently, our system has six motions as shown in table 1. We obtained these motions from an existing motion data library.

During this mode, postures are computed from the selected motion data and the character's pelvis position and orientation just before executing the motion. The joint angles are acquired from motion data. The pelvis position and orientation from the motion data are transformed so that the initial posture of motion data matches the character's initial posture using a transformation matrix.

While the motion data is being performed, users can cancel the motion by moving the current waist position back to the original position. Subsequently, the motion data will be played backward from its current time to its beginning time. In order to prevent unnatural motion, there are conditions for canceling the motion based on the time when the character started to move as shown in table 1. These cancellation times depend on motion data. After the cancellation time is exceeded, the user cannot cancel the motion. After finishing the motion, the system returns back to the position based control mode. However, the walking motion can be repeated when the user keeps their hands in the same position and they satisfy the conditions in table 1. We allow only cyclic motion can be repeated. For other motions, in order to repeat them, the users have to move the waist again.

Table 1 Relationship between character motion, waist condition and cancellation time

Motion	**Condition**	**Cancellation Time**
Rightward step	Position of waist (*x*) > 0.3 m	0.3 s
Leftward step	Position of waist (*x*) < - 0.3 m	0.3 s
Backward step	Position of waist (*z*) < - 0.3 m	0.3 s
Jump	Position of waist (*y*) > 1.1 m	0.1 s
Walk	Position of waist (*z*) > 0.3 m	0.1 s
Duck	Position of waist (*y*) < 0.3 m	0.1 s

The user can control the character's arms using the position based control in the motion data based control mode. When the motion data is being played, the arm posture is overridden by arm control in position based control. Basically, the user can control the character's arms during any motion. However, we only allow arm control during the walking motion because during motions such as the step, or ducking motion, it is difficult to perform an arm movement. This is because the motion duration is so short. In addition, for other motions such as jumping, which involves arm movements, activating arm control distorts the motion, and the generated motion looks unnatural.

5. Experiments and Discussion

We have implemented our method and tried out the proposed interface. Although we have not done a user study, the following are some impressions that we had from the trial.

Using the position based control mode, simple arm and leg motions can be performed. For example, we could make a character wave their hand and perform a small kick. It is possible to move the character's arm and leg immediately. We felt that the arm and leg controls were responsive and intuitive. However, we could not perform a hand clap using palms because it did not look natural. This is because the palm orientation does not change, thus the right palm could not touch the left palm.

This problem can be resolved by controlling joint angles of the character’s wrist according to the direction of the user’s hand.

The synchronous control of the arms and legs was responsive. We could make the character perform motions such as walking and marching. However, this was limited to walking and marching in place only. We felt that the arms and legs using the same trajectory did not look natural. This problem can be solved by using a different trajectory for the arms and legs.

In the motion data control mode, all prepared motions were executed intuitively. Cancellation of the motion data did not work well because the time to cancel the motion was too short. Adjustment of the cancellation time, or another cancellation approach is necessary. Repeating the walking motion sometimes did not work well because we failed to keep our hands in the same position for a certain period of time. . We also performed a waving hand motion during the walking motion using position based mode control. This motion is easy to perform but it did not look natural because the palm orientation does not change. This problem can be resolved using the same approached as the hand clapping solution mentioned above.

6. Conclusion

In this paper, we presented a data glove based interface for controlling real-time character motion. We introduced dynamical mapping between a user’s hand and the character’s body parts. We also combined motion data based control with position based control. Currently, our proposed interface can perform various types of character motion such as moving around, hand waving, small kicks, small leg movement, and locomotive motion.

With the current interface, arm and leg motions do not look natural, though this problem can be resolved by controlling joint angles of the character’s wrists and feet according to the direction of the user’s wrists. This will generally improve the naturalness. Some complex motions such as punching or kicking in a fighting game cannot currently be performed; however, this condition can be resolved using data based control by specifying these kinds of motions. As a future work, we also intend to create natural character posture changes that accord with the user’s motion during the motion data based control mode.

References

[1] Laszlo,J., Van de Panne,M., and Fiume,E., Interactive control for physically-based animation, In Proc. ACM SIGGRAPH 2000, (2000), 201–208.

[2] Igrashi,T.,Moscovich,T., and Huges,J.F., Spatial keyframing for performance-driven animation, In Proc. ACM SIGGRAPH/ Eurographics Symp. on Computer Animation, (2005), 107–115.

[3] Freeman, W. T.,Anderson, D. B.,Beardsley, P. A.,Dodge, C. N., Roth, M., Weissman, C. D., Yerazunis, W. S., Kage, H., Kyuma, K., Miyake, Y., and Ichitanaka, K., Computer vision for interactive computer graphics, IEEE Computer Graphics and Applications 18, 3(1998), 42–53.

[4] Chai, J.,and Hodgins, J. K., Performance animation from low-dimensional control signals, ACM Trans. on Graphics 24, 3(2005), 686–696.

[5] Kovar, L., Gleicher, M., and Pighin, F.. Motion graphs. ACM Trans. on Graphics 21, 3 (2002), 473–482.

[6] Taku Komura and Wai Chun Lam, Real-time Locomotion Control by Sensing Gloves , Computer Animation and Virtual Worlds, Volume 17, Number 5, (December 2006), 513-525.

[7] Yoshihiro Okada, Real-time Motion Generation of Articulated Figures Using Puppet/Marionette Metaphor for Interactive Animation Systems, Proc. of the 3rd IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP03), (September 2003), 13-18.

[8] P5 Data glove http://www.vrealities.com/P5.html

[9] Jehee Lee and Sung Yong Shin, A Hierarchical Approach to Interactive Motion Editing for Human-like Characters, SIGGRAPH 99, (August 1999), 39-48.