Puppet Playing: An Interactive Character Animation System with Hand Motion Control

Zhiqiang Luo, Chih-Chung Lin, I-Ming Chen, Song Huat Yeo, and Tsai-Yen Li

School of Mechanical & Aerospace Engineering
Nanyang Technological University, Singapore
{zqluo,michen,myeosh}@ntu.edu.sg
Computer Science Department
National Chengchi University
Taipei, Taiwan
{s9421,li}@cs.nccu.edu.tw

Abstract. Puppetry is a popular art form involving the process of animating the inanimate performing puppets. Puppet playing not only is controlled by artist's hand motions but also follows physical laws and engineering principles. To implement a puppet playing scenario in virtual environments, an interactive animation system is designed by taking into account both the user's hand motion and the constraints of puppet and environment. Here the hand motion is real-time captured and recognized through a new input device, namely SmartGlove. The animation system adapts IMHAP testbed for procedural animation and generates puppet animation based on both the procedural animation technique and motion capture data from SmartGlove. Thus the physical hand motion can either activate the designed procedural animation through motion recognition or tune the parameters of the existing procedural animation to generate new puppet motions. This system allows a user to directly control puppet animation while preserving the high accuracy of motion control. The animation results show that the association of hand motion can smoothly plan and generate each procedure animation and seamlessly blend the gap between key frames. The potential application and improvement of the current animation system are discussed.

Keywords: Character Animation, SmartGlove, Motion Planning, Puppetry.

1 Introduction

Character (specifically humanoid) animation is an oldest but still challenging research topic in computer animation. Two of challenges are emphasized here. First, people (e.g. viewers of animation) are most familiar with the characteristics (satisfying physics and biological requirements) of human motion in daily life. Thus the generation of natural and accurate motions should take into account of various parameters affecting human motion. Second, the character animation is usually predefined and is difficult to be generated and controlled in real time in order to adapt new tasks and environments. Interactive control of character animation can extend the adaption of predefined animation procedure and significantly save the time and effort to build a new animation. To

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XII, LNCS 6670, pp. 19–35, 2011.

respond to these two challenges, one interactive animation system is designed by employing both motion capture and procedure animation techniques. The interactive animation system can plan, generate, and control natural character animations in real time. One task scenario, puppet playing, is chosen to illustrate the background, procedure, and test results of our animation system.

Puppetry and Puppet Theater are popular art forms having a long and fascinating heritage in many cultures. The merit of the art form is the fascination with the inanimate object animated in a dramatic manner and humans' curiosity in producing an exact artificial copy of themselves. People are largely interested in the theatrical and artistic content of puppetry, though basic puppet fabrication and manipulation techniques follow physical laws and engineering principles. Most types of puppets in use today fall into four broad categories [1]. First, the glove puppet is used like a glove on a user's hand. Second, the rod puppet is held and moved by rods, usually from below but sometimes from above. Third, the marionette is a puppet on strings, suspended from a control mechanism held by the puppeteer. Last, the shadow puppets are usually flat cut-out figures held against a translucent, illuminated screen. In the present study, the first category, the glove puppet, will be animated in computer through the interaction with the user's hand.

Glove puppet animation is one type of character animation. Currently, the mainstream methods for generating character animations can be divided into three categories. The first method is to generate character animation by sampled data. Specifically, the sampled data can be obtained by the motion capture technology where the motions are performed by a real actor [6]. Animations made by this technology are more plausible. However, without knowing the meaning and structure of the motion, it is also difficult to modify a capture motion to fit the constraints of a new environment. The second method is to use the commercial 3D animation software to set the key frames of a character by animation professionals and interpolate between key frames to generate the final animations. However, even for good animators, producing an animation in this way is time consuming and the result is less realistic. The third method is to generate animations by simulation or planning procedures [2][11]. It is also called knowledge-based animation since it usually requires a dynamics model of the object under simulation or motion-specific knowledge to plan and generate the motions. Due to the complexity of the underlying computational model, the knowledge-based animation is usually generated in an off-line manner and displayed in real time. However, even along with a good knowledge or simulation system it is still difficult to generate the realistic animation by the computer procedural. It should be noted that the methods in above three categories are somewhat complementary to some degree and each of them may be more suitable for certain kinds of animations.

Being the intricate and prehensile parts of the human body, our hands are the primary agent for physical interaction with the external world. We use our hands in varies of aspects to perform our everyday activities. In order to capture the human hand's motion, researchers have developed many sensing methods in the past few decades. Besides the optical [4], acoustic and magnetic [10] sensing techniques, innovative techniques such as fiber-optic [3], strain gauge and hall-effect sensing are introduced. Nevertheless, these sensing methods still have rooms for improvement to achieve the stringent requirements from applications in various fields, such as the medicine, training, entertainment, and virtual reality. Some criteria for a glove to capture the

hand motion include the sensing accuracy (stability of the sensor signals, repeatability & reliability of movements), ease of wear and removal, rapid calibration, adaptation for different hand sizes, no electromagnetic interference, no temperature variation, and low cost. Furthermore, to manipulate the glove puppet investigated in the present study, it is crucial to accurately capture the finger motion in order to control the specific component of a puppet. A wearable glove-based multi-finger motion capture device, called the SmartGlove, is developed in the present study. The user's hand motion, especially the finger's motion, is captured by SmartGlove in order to drive the glove puppet animation implemented by the computer.

The present study introduces an interactive animation system to generate, plan and control a glove puppet animation by using the SmartGlove. With the user's hand motion input from the SmartGlove, a user can directly control the puppet's two arms and head motions. Furthermore, the animation is created based on the method of the procedural animation. The procedural animation is used to design some motions on the puppet's feet and the body, which compensates the input from SmartGlove. The input from SmartGlove can in turn help to tune the parameters of the animation procedure. The seamless integration between the hand motion input and the procedural animation ensures the fluent motion of the puppet.

The rest of the paper will be organized as follows. The system architecture is first introduced to provide the big picture of the current glove puppet animation system. Then, the design of the SmartGlove and the puppet model will be described, respectively. After that, the technique for the motion recognition and the implementation of the procedural animation is presented, followed by the experimental results to test the animation. Finally, we will conclude the paper and list the future work in the last section.

2 System Architecture

The present puppet animation system, called the IMPuppet, is constructed based on an experimental testbed for procedural animation, IMHAP (Intelligent Media Lab's Humanoid Animation Platform, [7]). The system architecture of IMPuppet is shown in Fig. 1. The structure of the system is based on the model-view-control (MVC) design pattern. With the MVC design pattern, the system is divided into three main models: *model*, *view* and *controller*. Model is to encapsulate data; view is to interact with the user; and controller is the communication bridge between the former two. In Fig. 1, each square stands for a module, and the arrow between the modules means the data flow from one module to the other one. The modules of animation dispatcher and animation generator are the main components in IMPuppet, and play the role of "model" in MVC. The module of animation controller plays the role of "controller", receives events from the user, and controls how animation is played. The modules of user interface and 3D browser play the roles of "view" in MVC, which responds to interaction with the user and draws the scene on the screen. The function of each module is described in more details as follows.

The modules of animation dispatcher and animation generator are the core of the IMPuppet system. They are defined with abstract interfaces in order to perform experiments on animation procedures. The animation dispatcher functions as a global planner which directs one or more animation generators. Specifically, the animation

dispatcher receives high-level commands, such as a path that the animated character should follow, from the graphical user interface of the system or sequence of motion data from the data glove. Then, the animation dispatcher maps the high-level commands into a sequence of locomotion according to the kinematics and geometric models of the puppet character as well as the environment. The animation dispatcher defines the target of a motion generation and dispatches relevant commands and data to the appropriate animation generators. An animation generator is both a local planner in charge of generating animation motions with the predefined procedure and a motion editor utilizing and modifying the motion capture data.

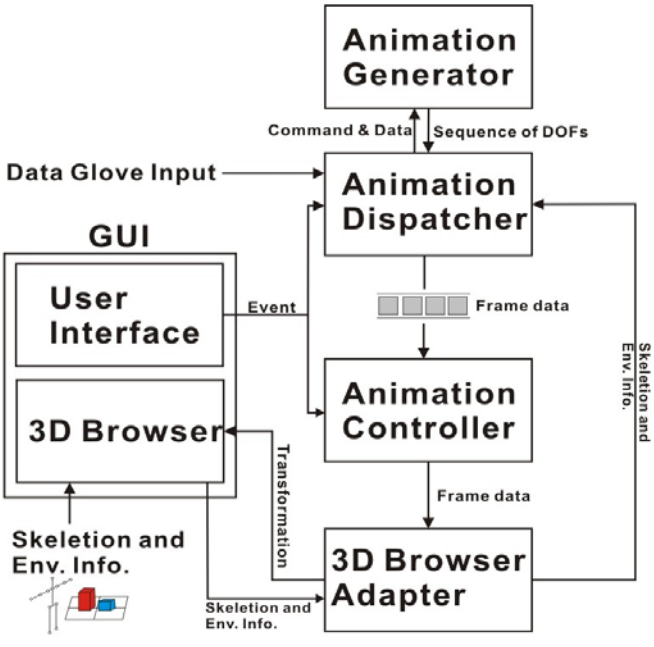


Fig. 1. IMPuppet system architecture

The detail structure of animation generator and animation dispatcher is shown in Fig. 2. The motion recognition system and planning system are put together in the animation dispatcher. Once a motion is recognized, the corresponding procedural motion can be sent to the frame data with the help of planning system. The animation generator module consists of all generators for an animation frame. The module of animation controller reads the queue of frames sent from the animation dispatcher and controls the playing of animation. The user controls the animation by either the SmartGlove (the main input) or the command (which is activated as an event) through the user interface.

The module of animation controller reads the queue of frames generated by the animation dispatcher and controls the playing of animation. The user controls the animation by accessing the given user interface and then the user interface calls the animation controller to invoke proper actions.

The 3D browser is treated as a plug-in in IMPuppet to read 3D models from files and provide External Authoring Interface (EAI) to other modules. EAI is a programming interface defined in most VRML browsers that allows a programmer to retrieve and set the attributes of an arbitrary object or skeleton joint, but not all 3D browsers support exactly this interface. To adapt to those browsers that support an interface with different naming or argument conventions, another module called "browser adapter" is designed to act as a wrapper of the 3D browser. It encapsulates the 3D browser and provides a uniform interface to other modules for the rest of the system. As a result, the 3D browser can be replaced easily without modifying the rest of the system as long as appropriate browser adapter has been implemented. In the present IMPuppet system, the open-source 3D browser JMonkey [5] is chosen as the 3D browser.

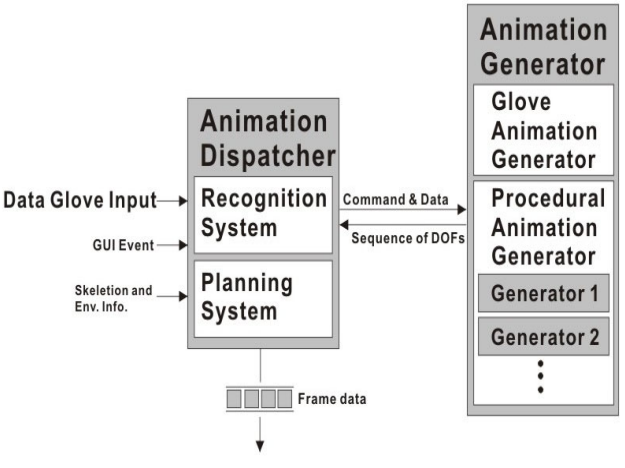


Fig. 2. Detail structure of animation generator (right) and animation dispatcher (left)

3 SmartGlove

The SmartGlove is aimed to achieve high performance hand/finger motion tracking and monitoring at an affordable cost for wide adoption. SmartGlove employs a new optical linear encoder (OLE) to capture the finger motion. The SmartGlove system consists of five multi-OLE strips (each includes two OLEs) and a microcontroller. The multi-OLE strips will capture and send the appropriate motion data of each finger joint to the microcontroller that synthesizes all the information sent to it from the multiple OLEs. Then, using a forward human hand kinematics model embedded into the gateway, the microcontroller will transmit the captured motion data to a remote robot, a virtual reality system or a computer for further analysis and application through wired or wireless communication. Compared to currently available hand capturing devices, the critical OLE sensing element is low-cost, compact, light-weighted, and immune to temperature or electromagnetic interferences.

3.1 Hand Kinematic Modeling

Human hand is modeled with a hierarchical tree structure that consists of rigid links and joints. This hierarchical structure is represented in Fig.3, and the position of each joint is described using the D-H transformation with reference to the heel of the hand (the world coordinate system ($x_0^{'}, y_0^{'}, z_0^{'}$)). [8] The posture of each finger ray (labeled as 1 to 5 from the thumb to the little finger as shown in Figure 3.) is represented under a local coordinate system. Now with the D-H transformation, the position of each joint can be transformed from the local coordinates to the world coordinates sequentially. There are 23 internal degrees of freedom (DOF's) located in the hand skeleton model [13][12].

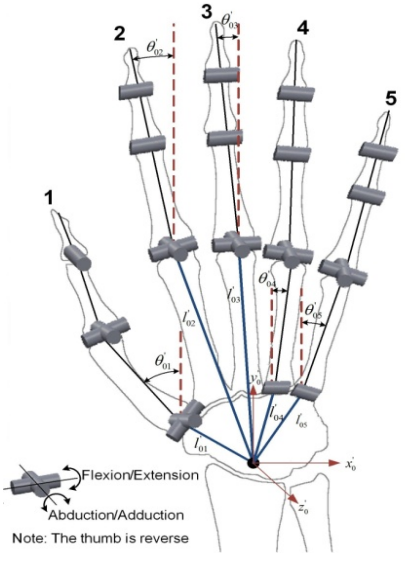


Fig. 3. The kinematic model of a user hand

As shown in Fig.5, the five finger rays can be divided into three different groups in terms of the kinematic structure. Each of the four fingers has four DOF's. The Distal Interphalangeal (DIP) joint and the Proximal Interphalangeal (PIP) joint both have one DOF and the remaining two DOF's are located at the Metacarpophalangeal (MCP) joint. Different from the four fingers, the thumb has five DOF's. There are two DOF's at the Trapeziometacarpal (TM) joint (also referred as Carpometacarpal (CMC) joint), and two DOF's at the Metacarpophalangeal (MCP) joint. The remaining one DOF of the thumb is located at the Interphalangeal (IP) joint. The basic flexion/extension (f/e) and abduction/adduction (a/a) motions of the thumb and fingers are performed by the articulation of the aforementioned 21 DOF's. The abduction/adduction motions only occur at each finger's MCP joint as well as the thumb's MCP and TM joints. Another two internal DOF's are located at the base of the 4th and 5th (ring and little finger's) metacarpals which performs the curve or fold actions of the palm. It should be noted that the present SmartGlove only captures fifteen DOF's of hand motion in the present project. The

motions of the end joints (near the finger tips) of index, middle, ring and little fingers were not captured as the range of the joint motion is quite small. The motions of the first joints of the thumb, ring and little fingers will be captured in the future system.

3.2 Multiple OLEs

The single-point OLE sensor detects joint flexion displacement through the 1-DOF linear movement of the sensor outputs [9]. Two or three OLE sensors put together can detect joint motions of multiple DOF's. The basic working principle of SmartGlove uses a variation of OLE principle by placing multiple OLE's in series on different finger segments to capture the displacements of different detecting points on a single encoder strip. This encoder strip passes through all OLE's on the same finger. Thus, it is termed as Multi-point OLE. As shown in Figure 4, three disks (from left to right) represent three in-line joints with radius of R1, R2 and R3, respectively. Denote their bending angles as φ1, φ2 and φ3, respectively. Three OLE's are placed and fixed at positions (A), (B) and (C), as shown in Fig.4. Assume that the displacement readings obtained by these three OLEs are D_1, D_2 and D_3, respectively.

Due to the accumulated displacement at the distal joints, we use the following equations to calculate the angle of each finger joint.

$$D_1 = L_1 = \frac{2\pi R_1 j_1}{360} \tag{4}$$

$$D_2 = L_1 + L_2 = \frac{2\pi R_1 \varphi_1}{360} + \frac{2\pi R_2 \varphi_2}{360} \tag{5}$$

$$D_3 = L_1 + L_2 + L_3 = \frac{2\pi R_1 \varphi_1}{360} + \frac{2\pi R_2 \varphi_2}{360} + \frac{2\pi R_3 \varphi_3}{360} \tag{6}$$

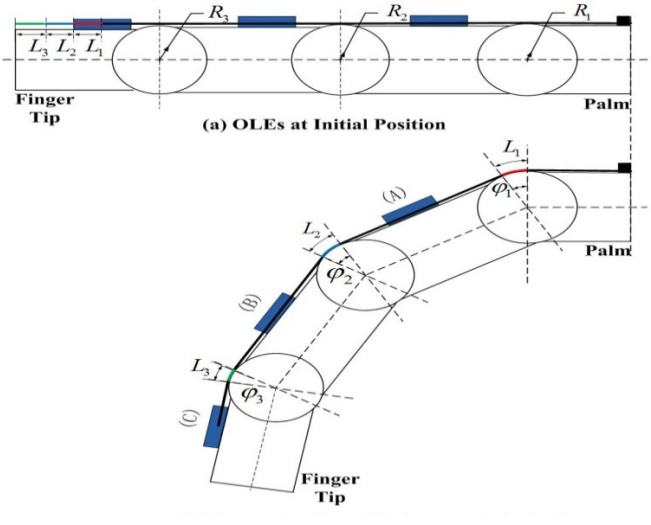


Fig. 4. Multi-point sensing principle for multiple OLEs

Because of the natural arches of a hand, the multi-point sensing can be adopted in finger motion capture. As introduced in the hand kinematics above, there is at least 14 joints' FE motions need to be captured in order to perform basic multi-finger sensing, and all these 14 joints are all within the five longitudinal arches. Hence, by introducing one strip for each longitudinal finger arch, we are able to use the multi-point sensing method to capture the finger's movement.

3.3 Prototype

The hardware of SmartGlove comprises two main components, the OLE module and the microcontroller. All these hardware components are mounted on a glove [14].

As shown in Fig. 5, the OLE module is the sensing module in the system which includes three basic units: *the sensing unit* (sensor and lens), *the interface unit* (the customized PCB board), and *the housing unit* (the customized base plate & strip). The sensing unit is fixed in the housing unit to obtain the displacement of strip and to communicate with the microcontroller through the interface unit.

The sensor used in OLE is Avago's optical mouse sensor product ADNS-3530, which is based on Optical Navigation Technology that measures changes in position by optically acquiring sequential surface images (frames) and mathematically determining the direction and magnitude of movement. In order to make the size of the OLE compact, the ADNS-3530 is designed for surface mounting on a PCB board.

The housing unit is the holder for the optical navigation sensor and the moving strip made of DelrinTM. According to the performance of ADNS-3530, the distance between the lens and the moving strip determines the resolution of the result. Based on the datasheet, in order to get high resolution of the sensor, the distance should be within the range of 0.77mm to 0.97mm. Furthermore, the surface material of the strip also affects the sensor's resolution. To make sure that the strip slides smoothly in the housing unit, there must be a gap between the strip and the base plate. Consequently, for the stable readout, white Formica is the ideal choice for surface material of the strip because the mean resolution is very stable within the pre-defined range.

SmartGlove uses the Arduino Diecimila/Bluetooth which is based on the Atmega168. It is an open-source physical computing platform based on a simple I/O board. The programming language for the Arduino microcontroller is an implementation of Wiring/Processing language. The microcontroller communicates with the OLE's via SPI protocol and sending out all the motion data to PC via USB/Bluetooth.

For the ease of the replacement and maintenance of the sensors, the OLE's are mounted onto the glove using Velcro, and the microcontroller connects OLE's by ribbon wires. Thus, the glove can be separated from the OLE's and all the hardware for cleaning. This feature takes a big leap toward using such data gloves in common daily living. Several photos of the SmartGlove prototype are shown in Fig.5.

Furthermore, in order to make the glove type OLE's sensitive, the glove should fit nicely on the human hand. On the other hand, the glove should not hinder free motion of the hand. Therefore, soft and stretchable fabric is used for the SmartGlove. In this project, two different fabrics are used: the semi-stretching fabric, which can be stretched only in a single direction, and the stretching fabric, which stretches in all directions. The glove uses stretching fabric for backside of the MCP joints and semi-stretching fabric for the palm side to avoid stretching along the finger direction. Thus, the glove has good elasticity to fit the hand of the users.

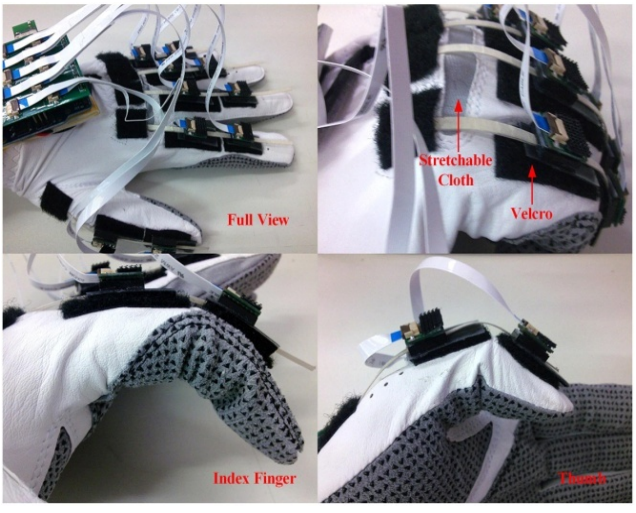


Fig. 5. SmartGlove prototype

3.4 Experimental Test

The Grip Test (uses a gripped hand position) and the Flat Test (uses a flat hand position) are carried out to analyze the repeatability and reliability. Data is collected from five healthy male students aged 22-27 years with comparable hand size and no hand movement disorders. Five sets of measurement are performed in each test on each subject and each set of measurement includes ten grip/release actions.

Repeatability is indicated by the range and standard deviation (SD). The average range and SD obtained from each OLE across Subjects 1 to 5 for each test are shown in the histogram in Fig.6.

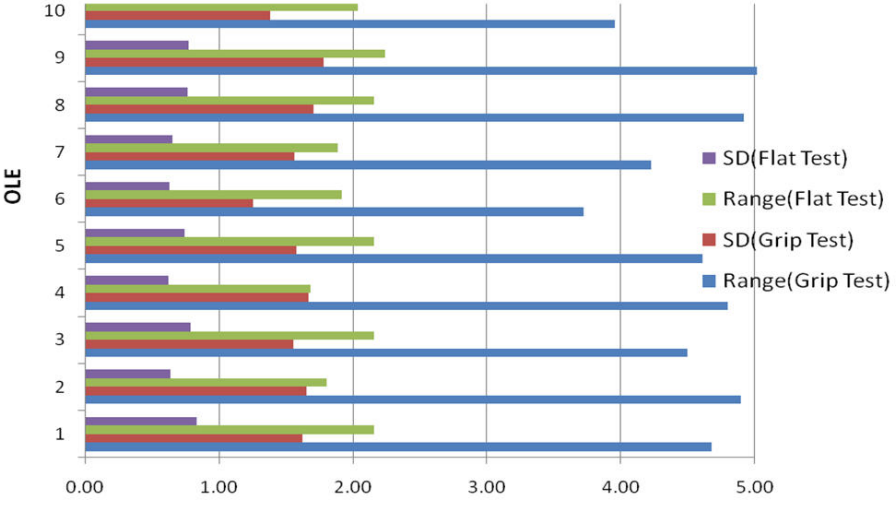


Fig. 6. Histogram of averaged rang and SD for each OLE

Reliability is indicated by the intraclass correlation coefficient (ICC). [15] The ICC analysis is performed for each test and for each OLE individually (ICC is calculated using Excel). The ICC values in Table 1 show consistency from one data block to another with no particular OLE showing significant lower reliability than the overall mean.

Table 1. ICC of reliability

	thumb		index		middle		ring		little		Aver -age
	MCP	IP	MCP	PIP	MCP	PIP	MCP	PIP	MCP	PIP	
Grip Test	0.937	0.954	0.882	0.963	0.913	0.987	0.948	0.957	0.969	0.964	0.947
Flat Test	0.955	0.968	0.893	0.966	0.908	0.976	0.955	0.968	0.958	0.979	0.953
Over -all											0.950

4 Puppet Model

The puppet model we use in our system is based on the traditional Chinese puppet. There are three reasons that we chose traditional Chinese puppet rather than other kinds of puppets. The first reason is that it fit our desired application for SmartGlove due to its manipulation method. The second reason is the rich performance that the traditional Chinese puppet has, compared to other kinds of puppetry. This performing art not only combines elements of Chinese opera, music, delicate costumes and props but also skillful manipulation of the puppet. This also makes the design of our system a great challenge. The third reason is that the motion of traditional Chinese puppet is suitable for procedural modeling. Because of the elements from the Chinese opera, most motions of traditional Chinese puppet follow strict rules about how it should be performed. These rules make it possible to design the motions in a procedural way.

4.1 Kinematics Model

The kinematics structure of a puppet is similar to a human figure with minor differences as show in Fig. 6. For example, the major active joints of a puppet are the neck, shoulder, left/right hips, and pelvis. The joints of elbows and knees are usually ignored. The motions of the legs are triggered either by another hand or by the gravity during swinging. The clothes are the passive part that moves with the legs as they swing. As show in Fig. 7, the additional links and joints are attached to the clothes at the front and back of the legs. Thus the clothes can moved when the legs are lifted.

4.2 Control of Puppet

SmartGlove can be used to directly control the puppet's hand and head movements. As shown in Fig. 8, the position of point O on the hand corresponds to the center of puppet. Currently SmartGlove cannot capture the abduction/adduction (a/a) motions between each finger. The angles of the two joints in Fig. 8, θ_1 andθ_2, represent the angle

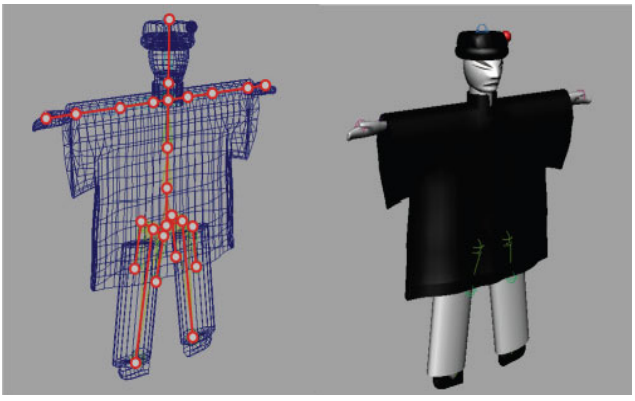


Fig. 7. Kinematics model of the hand puppet including the additional linkages with clothes

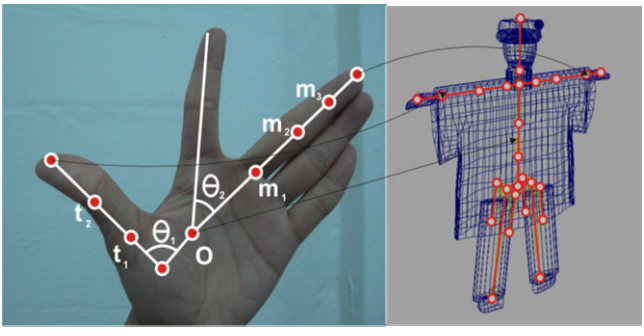


Fig. 8. Use thumb and middle finger to control the two arms of a puppet

of the joint between the thumb and the index finger and the angle of the joint between the index finger and the middle finger, respectively. Beside these two joints, SmartGlove can capture the bending angles of the other four joints, t_1, t_2, m_1 and m_2. According to our experiments, we found that it is reasonable to assume the bending angle of joint m_3 be equal to the angle of m_2. Then the end-point position of the two fingers can be computed by using forward kinematic based on the above joint angle and the length between the two joints. The end-point positions for the fingers of the puppeteer can then be used to update the hand positions of the puppet. Finally, by using inverse kinematic, the rotation matrix of the puppet's shoulder and elbow on the puppet's two hands can be determined. Similarly, SmartGlove can be used to detect the flexion/extension (f/e) of the index finger to control the head of the puppet.

With the method described above, we can move the puppet's hands and head in a way that reflects the physical motion of the user's fingers. And this is exactly the way that we control traditional Chinese puppet in most cases. However, it is not enough to only use this method to control and generate puppet motion if we want our virtual puppet to have full features of motions as the real one does. Hence, a computer procedure that can help on generating the modified motions is highly desirable. The idea

behind the design of our computer-assisted procedure can be divided into two steps. The first step is motion recognition and the second step is motion generation. In the motion recognition step, the computer is used to recognize the motion that a user intends to perform. If the motion can be recognized and an appropriate animation procedure is also available, then the motion will be generated with the corresponding procedure. Otherwise, the motion mapped directly to the hand gesture will be generated. These two steps will be described in more details in the following subsections.

5 Motion Recognition

The motion recognition module here is used to recognize the motion that the user intends to perform. With this module, the system can ask for the help of the motion generation procedure whenever needed. For example, when the user performs a walking motion with SmartGlove, the computer can be used to automatically generate the motion of swinging legs that are not controlled by the user. Or when a walking motion with the glove is recognized, we can use the procedural animation generator to generate a walking puppet motion for a given set of parameters. However, recognizing a puppet's motion such as walking and jumping by using hand gestures detected by SmartGlove is difficult. Instead, we recognize a puppet's motion based on the sequence of common puppet poses. In the present study, we find that a meaningful motion of a puppet usually consists of a continuous motion connecting two distinct key poses. For example, in shown in Fig. 9, the puppet moves to the right, to the left and with both hands, facing left and right and nodding.

To recognize the feature of a key pose, the rotation value on each joint is inputted into a finite state machine. The rotation value on each joint may trigger the state change of the state machine. When a state machine reaches its final state, the feature is recognized and the values are recorded.

Two types of motions can be recognized by the present recognition system. The first type of motion can be recognized right after a motion is inputted. This type of motion, such as walking, cannot be recognized until a complete sequence of the motion has been perceived. The second type of motion can be recognized after only parts of the motion are inputted. For example, jumping is an example. Once a fast movement from nodding down to nodding up is detected, jumping procedural is followed.

A basic motion for a puppet can be divided into parts. Assume that each part is represented by an alphabet. For example, moving right hand can be divided into parts such as moving right hand in and moving right hand out. We can use 'a' to stand for the motion segment of the right hand in, and 'b' stands for the motion of moving right out. Through this method, each motion can be represented by a finite string (e.g. 'acbd' or 'abde'). Then we can define a similarity function to compute the similarity between the input motion and the motion in the database. The similarity function is computed by comparing the two strings which represents the input and the stored motions. If the similarity value is larger than the pre-defined threshold, the current input motion is substituted by the motion previously stored in the database.

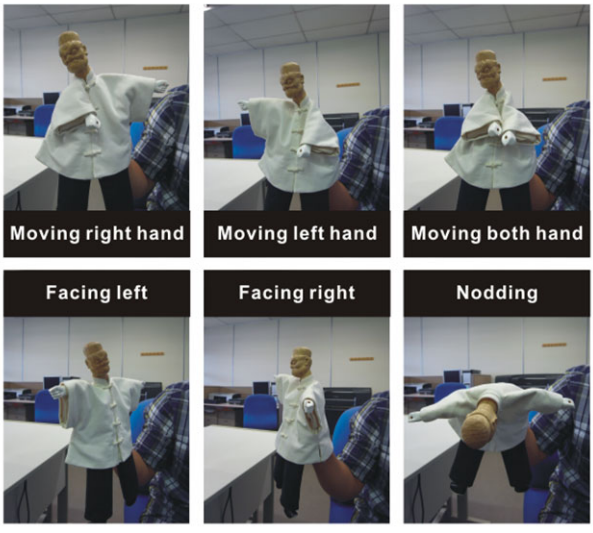


Fig. 9. Six base pose for motion recognition

6 Puppet Procedural Animation

Procedural animation is one type of the knowledge-based animation since it uses the knowledge of glove puppet manipulation to design animation procedures. A simple motion can be divided into multiple phases delimited by key frames with distinct spatial constraints in each phase. Then the motion between key frames can be computed by appropriate interpolation functions satisfying either the temporal or spatial constraints or both. Animations produced by this approach have the advantages of being flexible and computationally efficient for real-time environments. Nevertheless, it is also a great challenge to design a procedure with appropriate parameters that can produce natural and accurate motions.

In procedural animation, different motions require different animation generators. Here the walking motion is analyzed to illustrate how the motions are generated in a procedural manner. Nevertheless, walking can be performed in various ways. As advised by our puppet master, we have chosen the normal walking as an example, as shown in Fig. 10. In this walking motion, the adjustable high-level parameters for the animation procedure include step length, step height, and body swing angle. Four key frames are defined according to the given motion parameters.

The process of a walking animation procedure is carried out by three steps. First, the motions performed by a master puppeteer are recorded and the motion parameters are extracted to describe the characteristics of the motion. Second, the motion is decomposed into several phases by defining key frames to separate these phases according to the above motion parameters. For example, walking animation illustrated in Fig.9 is divided into four phases based on the three aforementioned motion parameters. Phase 1 is determined by the body swing angle; phase 2 is by the step height, phase 3 is by the step length, and phase 4 is by the body swing angle too. Lastly, the procedure for

interpolation between key frames is defined. More specifically, two types of interpolations are used to produce plausible motions. The first one is to interpolate along a curve for trajectory following, and the second defines how a motion is timed. In the current system, linear interpolation is used for simple motions while Bezier curve are used for sophisticate motions.

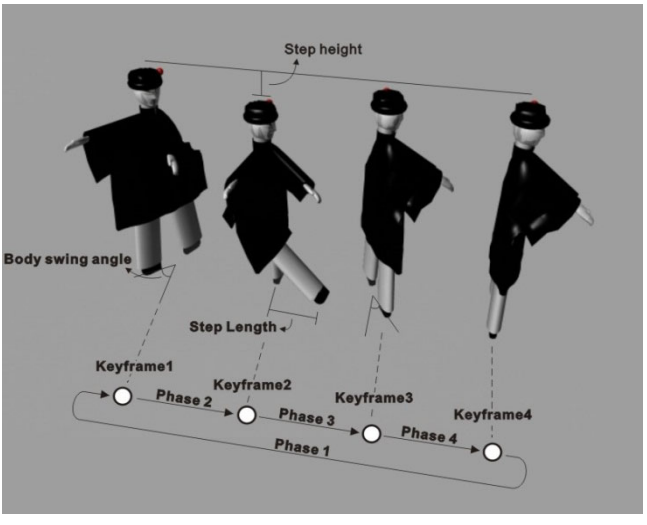
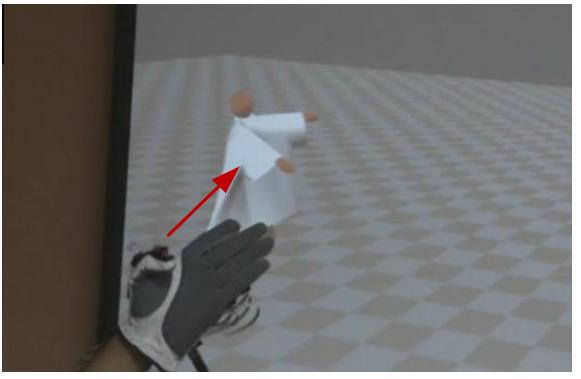


Fig. 10. The keyframes and the procedural parameters for the walking

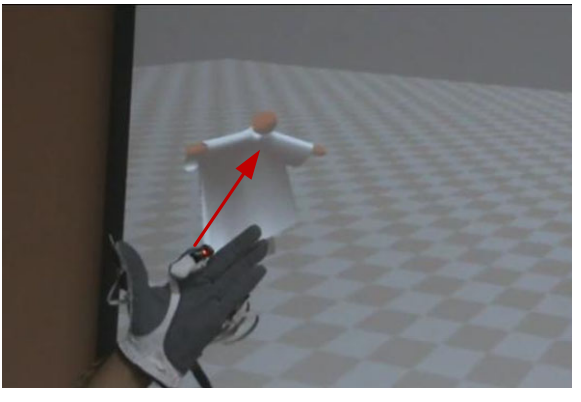
7 Animation Results

As mentioned in Section 2, the system is constructed based on an IMHAP experimental testbed for procedural animation. The 3D browser is JMonkey. The programming language is Java. The geometry of the virtual characters was modeled with the Alias Maya package. In Fig. 11, the animation controlled by the motion of three fingers is illustrated. Specifically, the thumb controls the animation for the right arm of the character (see Fig. 11(a)); the index finger controls the head animation of the character (see Fig. 11(b)); and the middle fingers control the left arm animation of the character(see Fig. 11(c)). These three controls lead to three featured motions recognized by using the real puppet (see Fig. 9). Each finger motion is recognized by the system and the three corresponding motions are implemented by the procedural animation.

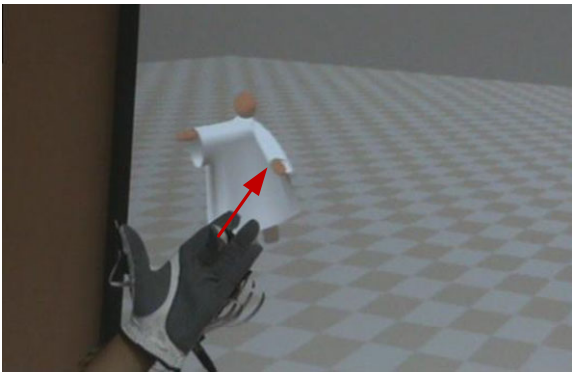
Fig. 12 shows the result of procedural animation by taking into account the input of the finger motions. The user attaches the thumb and the middle finger together and the system recognizes the activation of a specific motion: “moving puppet hand together.” After that, the animation system continues to generate the walking motion by swinging the puppet’s legs with appropriate motion parameters for each key frame. The user can then abducts the thumb in order to generate the specific animation: “waving right hand”. As shown in Fig. 12, the animations for different parts are performed simultaneously to synthesize the final animation.



(a) Controlling the right arm animation through the thumb motion



(b) Controlling the head animation through the index finger motion



(c) Controlling the left arm animation through the middle finger motion

Fig. 11. Control of the puppet motion by three fingers, thumb (a), index finger (b) and the middle finger (c)

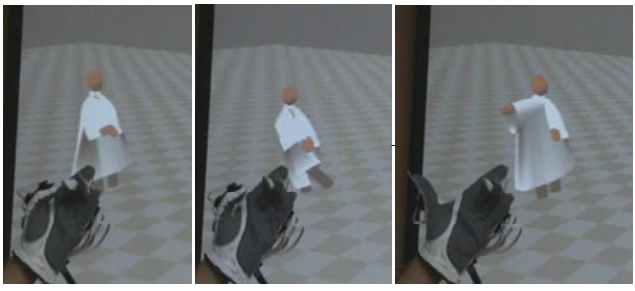


Fig. 12. Walking with hand motion input

8 Conclusion

The present study proposes an interactive character animation system to integrate the procedural animation technique and the input of motion data from the user's hand. The input of the motion data from SmartGlove can help user generate and control the puppet animation in real time. One efficient way is to generate the desired procedural animation is by recognizing the motion based on the stored motion database. When the motion is not recognized, the motion input can still be used to control the specific puppet joint to build the puppet animation or tune the parameters of the existing procedural animation. The gaps between key frames generated by different methods are blended by additional blending frames between motions.

The future work is two-fold. The immediate work is to build a glove puppet animation with personality. This can be done by automatically adjusting the parameters of the motion procedures, such as setting up the dynamic timing of animation to fit the motion feature data. Another work is to help a puppet learner practice during their training. The puppet learner can use SmartGlove to practice their manipulation skills virtually (without wearing the puppet), get the animation feedback, and then try the real puppet after the virtual practice.

Acknowledgement

This work was supported by Media Development Authority, Singapore under NRF IDM004-005 Grant.

References

1. Currell, D.: Puppets and Puppet Theatre. Crowood Press, Wiltshire (1999)
2. Chen, F., Li, T.Y.: Generating Humanoid Lower-Body Motions with Real-time Planning. In: Proc. of Computer Graphics Workshop, Taiwan (2002)
3. Fifth Dimension Technologies, `http://www.5dt.com`
4. Goebl, W., Palmer, C.: Anticipatory Motion in Piano Performance. J. of the Acoustical Society of America 120(5), 3004 (2006)
5. jMonkey Engine, `http://www.jmonkeyengine.com`

6. Kovar, L., Gleicher, M., Pighin, F.: Motion Graphs. In: Proc. of ACM SIGGRAPH (2002)
7. Liang, C.H., Tao, P.C., Li, T.Y.: IMHAP – An Experimental Platform for Humanoid Procedural Animation. In: Proc. of the Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Tainan, Taiwan (2007)
8. Lin, J., Wu, Y., Huang, T.S.: Modeling the Constraints of Human Hand Motion. In: Proc. of the Workshop on Human Motion, pp. 121–126. IEEE, Los Alamitos (2000)
9. Lim, K.Y., Goh, Y.K., Dong, W., Nguyen, K.D., Chen, I.-M., Yeo, S.H., Duh, H.B.L., Kim, C.G.: A Wearable, Self-Calibrating, Wireless Sensor Network for Body Motion Processing. In: Proc. of IEEE ICRA, Pasadena, California, pp. 1017–1022 (2008)
10. Mitobe, K., Kaiga, T., Yukawa, T., Miura, T., Tamamoto, H., Rodger, A., Yoshimura, N.: Development of a Motion Capture System for a Hand Using a Magnetic Three-Dimensional Position Sensor. In: ACM SIGGRAPH Research Posters: Motion Capture and Editing, Boston, USA (2006)
11. Perlin, K.: Real Time Responsive Animation with Personality. IEEE Transactions on Visualization and Computer Graphics 1(1), 1–16 (1995)
12. Rhee, T., Neumann, U., Lewis, J.P.: Human Hand Modeling from Surface anatomy. In: Proc. of the 2006 Symposium on Interactive 3D Graphics and Games, Redwood City, California, USA, pp. 27–34 (2006)
13. Wu, Y., Huang, T.S.: Human Hand Modeling, Analysis and Animation in the Context of HCI. In: Proc. of the International Conference on Image Processing., vol. 3, pp. 6–10 (1999)
14. Li, K., Chen, I.-M., Yeo, S.H.: Design and Validation of a Multi-Finger Sensing Device based on Optical Linear Encoder. In: Proc. of ICRA, pp. 3629–3634 (2010)
15. Dipietro, L., Sabatini, A.M., Dario, P.: Evaluation of an Instrumented Glove for Hand Movement Acquisition. J. of Rehabilitation Research and Development 40(2), 179–190 (2003)