Building Hand Motion-Based Character Animation: The Case of Puppetry

Zhiqiang Luo I-Ming Chen Song Huat Yeo
School of Mechanical & Aerospace Engineering
Nanyang Technological University
Singapore
{zqluo, michen, myeosh}@ntu.edu.sg

Chih-Chung Lin Tsai-Yen Li
Computer Science Department
National Chengchi University
Taipei, Taiwan
{s9421, li}@nccu.edu.tw

Abstract— **Automatic motion generation for digital character under the real-time user control is a challenging problem for computer graphic research and virtual environment applications such as on-line games. The present study introduces a methodology to generate a glove puppet animation which is controlled by a new input device, called the SmartGlove, capturing the hand motion. An animation system is proposed to generate the puppet animation based on the procedural animation and motion capture data from SmartGlove. As the control of the puppet character in the animation takes into account the design of SmartGlove and the operation of the puppet in reality, the physical hand motion can either activate the designed procedural animation through motion recognition or tune the parameters of the procedural animation to build the new puppet motion, which allows the direct user control on the animation. The potential application and improvement of the current animation system are also discussed.**

Keywords-Motion Capture; Character Animation; Motion Planning

I. INTRODUCTION

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 [2]. First, the glove puppet is used like a glove on the 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 glove puppet will be animated in computer through the interaction with the user's hand.

The human hands are the main body part for the physical interaction with the world. People use hands in varies of aspects to perform the 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 signal, 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.

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 [1][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 a significant and each of them may be more suitable for certain kinds of animations.

The present study introduces a new and efficient methodology to generate 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 implement some motions on the puppet's feet or the body, which compensates the input from SmartGlove. The input from SmartGlove can in turn help to tune the parameter 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.

II. 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 as Fig. 1. The structure of the system is base on the model-view-control (MVC) design pattern. In MVC design pattern, the system is divided into three main models: model, view and controller. Model is used for encapsulate data; view is used for interactive 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 between each module. 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 the animation is played. The modules of user interface and 3D browser play the roles of "view" in MVC, which response to interact with the user and draw the scene on the screen. The function of each module will be described in more details below.

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 the animation 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

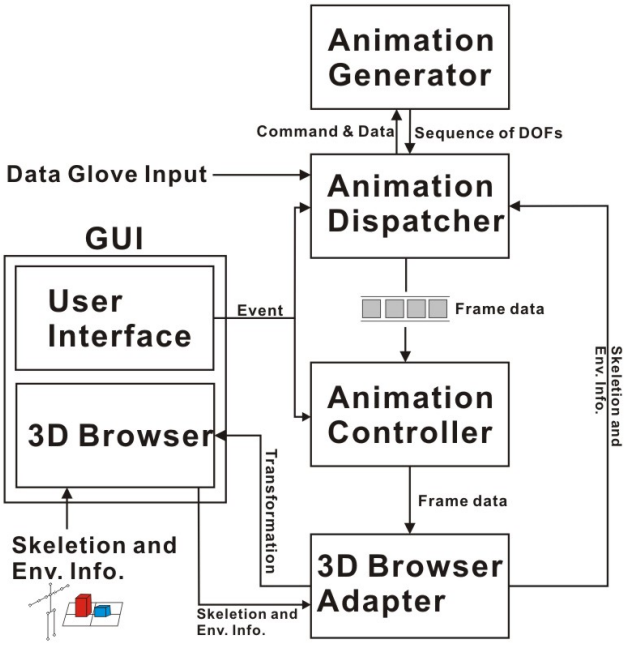


Figure 1. IMPuppet system architecture

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 the relevant command and data to the appropriate animation generators. An animation generator is both a local planner in charge of generating animation motions with the predefine procedural and a motion editor utilizing and modifying the motion capture data.

The detail structure of animation generator and animation dispatcher is shown as 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 the user interface then 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. For 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.

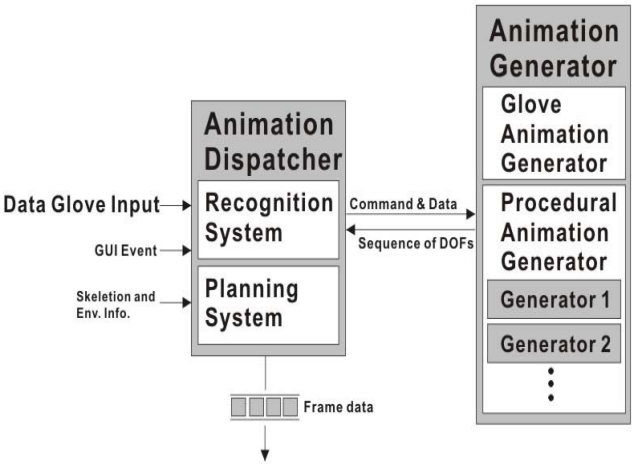


Figure 2. Detail structure of animation generator (right) and animation dispatcher (left)

III. SMARTGLOVE

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 will synthesize 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, virtual reality system or computer for further analysis and application through wired or wireless communication.

A. 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 each joint's position is described using 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 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 freedoms (DOFs) located in the hand skeleton model [13][12]. As shown in Fig.5, five finger rays can be divided into three different groups in terms of the kinematic structure. Each of the four fingers has four DOFs. The Distal Interphalangeal (DIP) joint and the Proximal Interphalangeal (PIP) joint both have one DOF and the remaining two DOFs are located at the Metacarpophalangeal (MCP) joint. Different from the four fingers, the thumb has five DOFs. There are two DOFs at the Trapeziometacarpal (TM) joint (also referred as Carpometacarpal (CMC) joint), and two DOFs 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 above mentioned 21 DOFs. The abduction/adduction motions only occur at each finger's MCP joint as well as thumb's MCP and TM joints. Another two internal DOFs 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 DOFs 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.

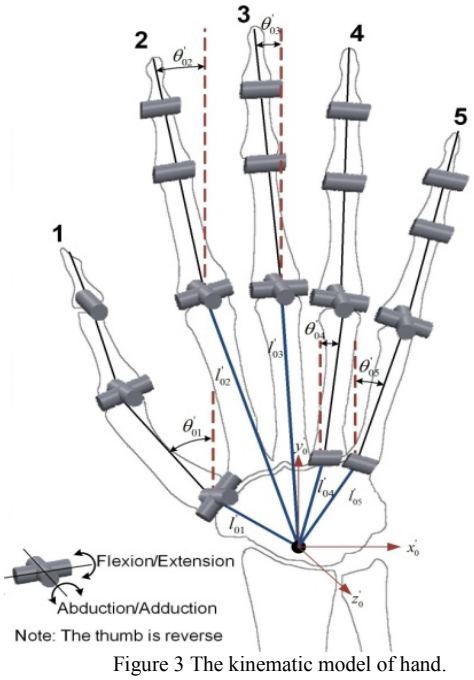


Figure 3 The kinematic model of hand.

B. 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 DOFs. The basic working principle of SmartGlove uses a variation of OLE principle by placing multiple OLEs 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 OLEs 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 bend angles as φ1, φ2 and φ3 respectively. Three OLEs are to be placed and fixed at positions (A), (B) and (C). Assume the displacement readings obtained by these three OLEs are D1, D2 and D3. Due to the accumulated displacement at the distal joints, we have

$$D_1 = L_1 = \frac{2\pi R_1 \varphi_1}{360} \tag{4}$$

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

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

Because of the natural arches of 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, it is able to use the multi-point sensing method to capture the finger's movement.

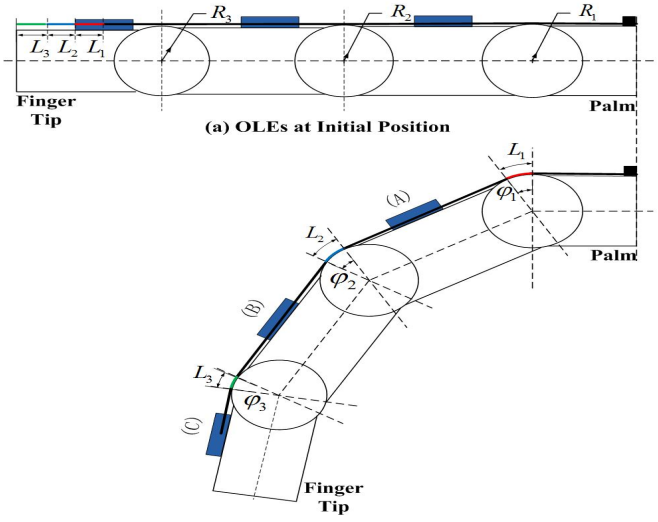


Figure 4 Multi-point sensing principle.

C. *Prototype*

The hardware of SmartGlove comprises two main components, the OLE module and the microcontroller.

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 the strip sliding 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 OLEs via SPI protocol and sending out all the motion data to PC via USB/Bluetooth.

For ease of the replacement or maintenance of the sensors, the OLEs are mounted onto the glove using Velcro and the microcontroller connects OLEs by ribbon wires. Thus, the glove can be separated from the OLEs 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.

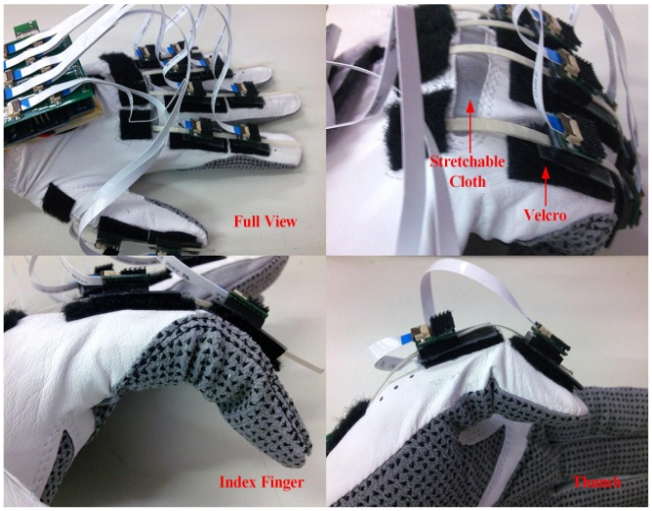


Figure 5. SmartGlove prototype

IV. PUPPET MODEL

A. 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 at the neck, shoulder, thigh, 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. 6, the additional links and joints are attached to the clothes at the front and back of the legs such that we can move the clothes when the legs are lifted.

B. Control of Puppet

SmartGlove can directly control the puppet hand and head movement. As shown in in Fig. 7, the position of point *O* is the center of puppet. Currently SmartGlove can not capture the abduction/adduction (a/a) motions of the thumb and the index finger. The angles of two joints are set with θ_1 and θ_2, respectively. θ_1 represents the angle of the joint between the thumb and the index finger; θ_2 represents the angle of the joint between the index finger and the middle finger. The two angles are set as the reference in the animation system. Except these two joint, SmartGlove can capture the motion of other four joints, t_1, t_2, m1 and m_2. The angle of joint m_3 is assumed to be equals to the angle of m_2. The end point position of the two fingers can be computed based on the above joint angle and the length between the above joint. By using inverse kinematic, the rotation matrix of the puppet's shoulder and elbow on two puppet's hands can be computed. Thus, the positions of the puppet's hands and the user's two fingers can be matched with each other. It is realistic to move puppet's hands as the motion of the puppet's hands reflects the physical motion of the user's fingers. Similarly, SmartGlove can detect the flexion/extension (f/e) of the index finger to control the puppet head.

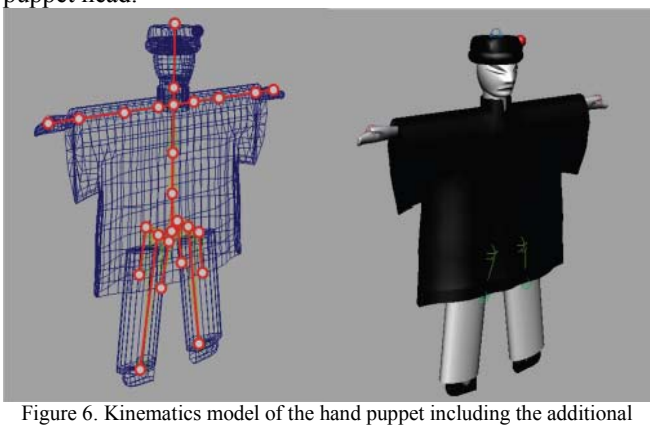


Figure 6. Kinematics model of the hand puppet including the additional linkages for clothes

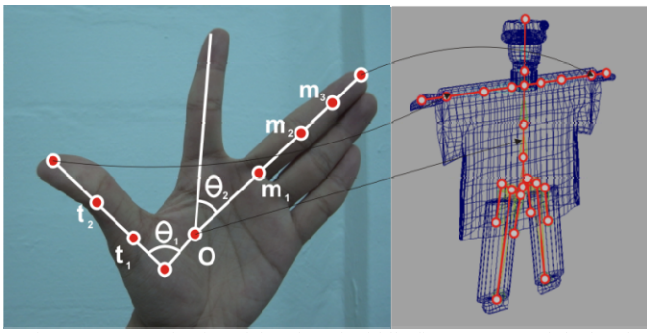


Figure 7. Use end point of thumb and middle finger to control the hand position of the puppet

V. MOTION RECONGITION

Recognizing motion such as walking and jumping by using the traditional method is a quite complexity question because of too many DOFs of (human) body. Thus, a dimension reduction method is necessary to reduce the number of the DOFs. In the present study, we find that every meaningful motion of puppet is the combination of the six motion features shown in Fig. 8. Each motion feature is represented by an alphabet, for example. 'a' stands for the motion of moving right hand in, and 'b' stands for the motion of moving right out. Through this method, each motion can be represented by a string (e.g. acbd or abde). The similarity value which indicates the similarity between the input motion and the motion in the database can be computed by comparing the two strings which represents the input and stored motions. If the similarity value is larger than the defined threshold, the current input motion can be regarded as the one previously stored in the database. Then the stored motion can drive the corresponding puppet animation without the try for generating the new puppet animation.

To recognize each feature, the rotation value on each joint is inputted to a finite state machine. As the rotation value on each joint change, it triggers the state change of the state machine. When the state machine reaches its final state, the feature is recognized and record.

There are two types of motions to be recognized by the present recognition system. The first type of motion can be recognized after the motion input for some time. This type of motion, such as walking, can not be recognized until a complete picture of the motion is perceived. The second type of motion can be recognized after only parts of motion are inputted. Jumping is one example. Once a fast movement from nodding down to nodding up is detected, jumping is followed.

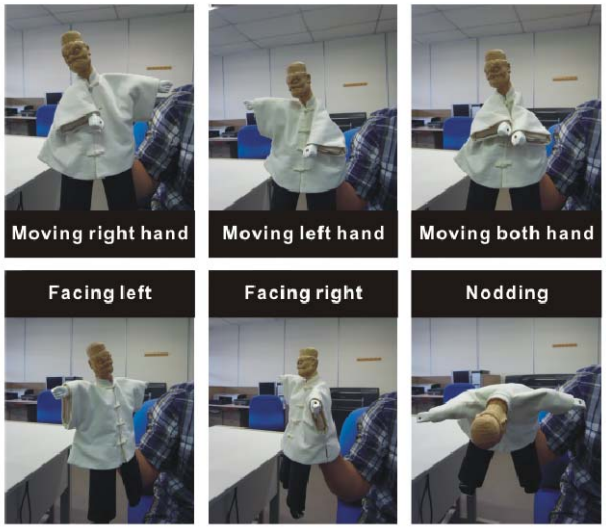


Figure 8. Six motion features for motion recognition

VI. 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 is computed by appropriate interpolation functions satisfying the temporal or spatial constraints. 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 plausible motions.

In procedural animation, different motions require different animation generators. The process of an animation procedure is carried out by following 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. Take walking for example, the motion parameters include step length, body swing, and some timing information. Second, the motion is decomposed into several phases by defining key frames to separate these phases according to the above motion parameters. Last, the procedure for interpolation between the key frames is defined. 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.

Here the walking is analyzed in order to illustrate how the motions are generated in a procedural manner. Walking can be performed in various ways. Advised by the puppet master, a normal walking is chosen as shown in Fig. 9. In this walking motion, the adjustable parameters include step length, step height and body swing angle. Four key frames are defined according to the given motion parameters.

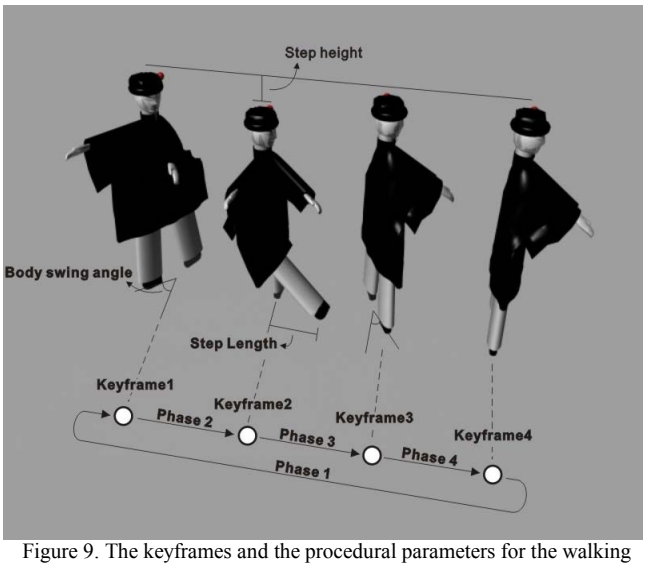


Figure 9. The keyframes and the procedural parameters for the walking

VII. ANIMATION RESULTS

As mentioned in section II, the system is constructed based on an IMHAP experimental testbed for procedural animation. The 3D browser is JMonkey. The development language is Java. the geometry of the virtual characters was modeled with the Alias Maya package. In following Fig. 10, three fingers, thumb, index finger and middle finger, control the puppet's upper body, including the head, left arm and right arm. There three controls lead to three motions recognized before by using the real puppet (see Fig. 8). Thus, each finger motion is recognized by the system and the three corresponding motions are implemented by the procedural animation.

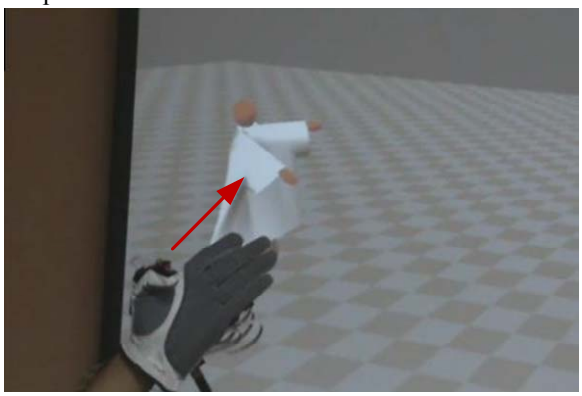
Fig. 11 shows the result of procedural animation by taking into account the finger motion input. The user attaches the thumb and the middle finger together and the animation system recognizes the activation of motion ("moving puppet hand together"). After that, the animation system continues generating the walking motion by moving the puppet's legs with the designed motion parameters of each key frame. The user then abducts the thumb in order to generate the animation "waving right hand animation". As seen from Fig. 11, the animation of waving the right hand and the animation of walking motion are performed simultaneously.

VIII. CONCLUSION

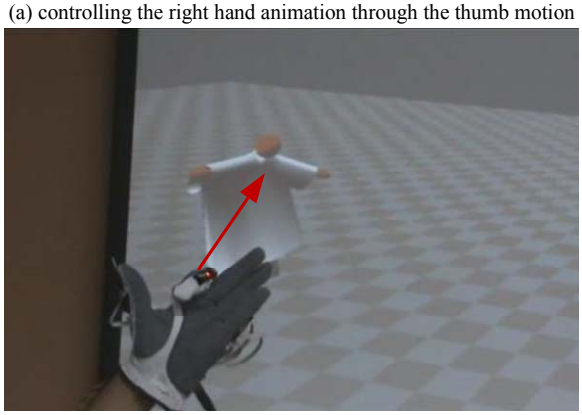
The present study proposes an animation system to combine the procedural animation technique and the input of the motion data from the user's hand. The input of the motion data from SmartGlove can help the user to real time generate and control the puppet animation. One efficient way is to generate the designed procedural animation by recognizing the motion input based on the stored motion database. When the motion is not recognized, the motion input can 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.

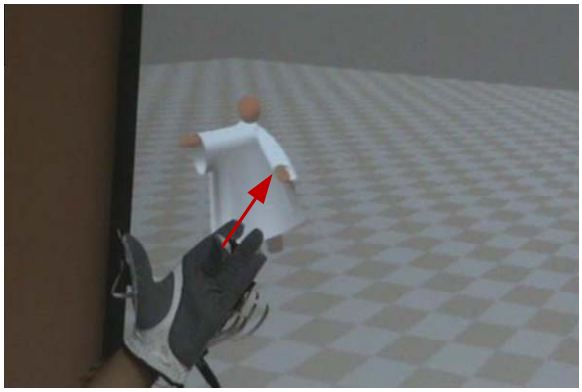
The future work is twofold. The immediate work is to build a glove puppet animation with personality. This can be done by automatically adjusting the parameters of the procedural generate motion, such as setting up the dynamic timing of animation to fit the motion feature data. Another work is to help the puppet learner's to practice during their training. The puppet learner can try the SmartGlove to start their practice virtually (without the puppet wore on hand), get the animation feedback and then try the real puppet after virtual practice.



(a) controlling the right hand animation through the thumb motion



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



(c) controlling the right hand animation through the middle finger motion

Figure 10. Control of the puppet motion by three fingers, thumb (a), index finger (b) and the middle finger (c).

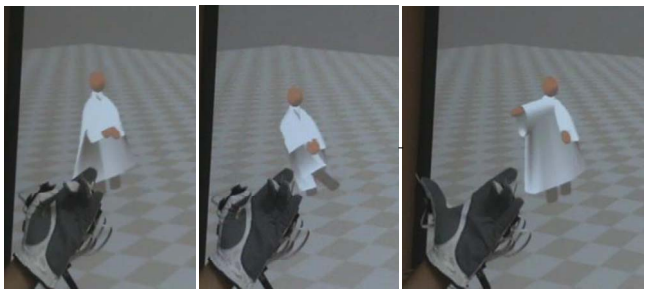


Figure 11. Puppet character waves the right hand while walking

ACKNOWLEDGEMENT

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

REFERENCES

[1] P.F. Chen, T. Y. Li, "Generating Humanoid Lower-Body Motions with Real-time Planning," Proc. of Computer Graphics Workshop, Taiwan, 2002

[2] D. Currell, "Puppets and Puppet Theatre". Wiltshire, UK: Crowood Press, 1999.

[3] Fifth Dimension Technologies, http://www.5dt.com, 2010.

[4] W. Goebl and C. Palmer, "Anticipatory Motion in Piano Performance", Journal of the Acoustical Society of America, Vol.120, Issue5, pp.3004, November 2006.

[5] jMonkey Engine, http://www.jmonkeyengine.com/, 2010.

[6] L. Kovar, M. Gleicher, F. Pighin, "Motion Graphs," In: Proc. of ACM SIGGRAPH, 2002.

[7] C.H. Liang, P.C. Tao, T.Y. Li, "IMHAP – An Experimental Platform for Humanoid Procedural Animation," Proc. of the third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Tainan, Taiwan, 2007.

[8] J. Lin, Y. Wu and T.S. Huang, "Modeling the Constraints of Human Hand Motion", Proceedings of the Workshop on Human Motion, IEEE, pp. 121-126, Los Alamitos, USA, 2000.

[9] K. Y. Lim, Y. K. Goh, W. Dong, K. D. Nguyen, I.-M. Chen, S. H. Yeo, H. B. L. Duh, and C. G. Kim, "A Wearable, Self-Calibrating, Wireless Sensor Network for Body Motion Processing", IEEE ICRA 2008, pp. 1017-1022, Pasadena, California, May 2008.

[10] K. Mitobe, T. Kaiga, T. Yukawa, T. Miura, H. Tamamoto, A. Rodger and N. Yoshimura, "Development of a motion capture system for a hand using a magnetic three-dimensional position sensor", ACM SIGGRAPH Research posters: motion capture and editing, Boston, USA, 2006.

[11] K. Perlin,"Real Time Responsive Animation with Personality," IEEE Transactions on Visualization and Computer Graphics, 1995.

[12] T. Rhee, U. Neumann and J.P. Lewis, "Human Hand Modeling from Surface anatomy", Proceedings of the 2006 symposium on interactive 3D graphics and games. pp. 27-34, Redwood City, California, USA, 2006.

[13] Y. Wu and T.S. Huang, "Human Hand Modelling, Analysis and Animation in the Context of HCI", Proceedings of the International Conference on Image Processing. Vol.3, pp. 6-10, 1999.