Editors: Jim X. Chen, jchen@cs.gmu.edu
R. Bowen Loftin, bloftin@odu.edu

REAL-TIME NATURAL HAND GESTURES

By Beifang Yi, Frederick C. Harris, Jr., Ling Wang, and Yusong Yan

IN HUMAN–COMPUTER INTERACTION (HCI) APPLICATIONS, TRADITIONAL DEVICES, SUCH AS KEYBOARD AND MOUSE, CAN BECOME CUMBERSOME AND UNSUITABLE. RESEARCHERS CONSIDER THE HUMAN HAND TO BE ONE OF THE MOST PROMISING NATURAL HCI media, specifically, using hand gestures to input computer commands.[1] Vision-based HCI hand-gesture analysis and recognition studies require large numbers of a variety of hand gestures as input and a virtual hand as output to display the results. Creating a virtual hand with natural hand gestures would improve human hand HCI research by providing hand ground truth data for hand analysis systems and generating visual output for hand simulation systems.

The human hand is highly articulated (with many joints) and organic (with considerable deformation during hand motion). Thus, hand modeling is an important part of overall human body simulation. Sophisticated algorithms and implementations in hand modeling and animation already exist—for example, modeling with underlying anatomical structure,[2] data-driven algorithms in hand animation,[3] and example-based deformable modeling using medical images.[4] As an alternative, our virtual hand considers human hand movement constraints, has less deformation, and allows realistic real-time animation. Our modeling and rendering algorithms are also more efficient; the system emphasizes the interactions between the hand model and its users; and a convenient GUI makes implementing all the finger movement operations easy and effective for new users.

Here, we'll introduce our real-time human hand visualization system, describe how to construct a lifelike hand model, and discuss a built-in hand gesture and animation data structure accessible through an easy-to-use GUI. Using the results provided in this article, we are designing human gestures in the development of a sign language interfacing system.

From Real to Virtual

Virtual hand construction must fully consider the human hand's biomechanical features. To that end, we reviewed medical studies and applied the research results to construct our virtual hand.

The Human Hand

Although the hand is a complex biological organ, we can also think of it as a mechanical machine and apply mechanical principles to study it. In this context, the hand involves three elements: muscles serve as the motor for providing driving force, tendons, bones, and joints transmit the motor's driving force, and skin and pulp tissues apply the force.[5]

We analyzed the various movements of the hand's different parts and categorized its biomechanical motions. This gave us guidance for computer modeling the human hand. While medical researchers and biomechanical scientists use sophisticated methods and measurements to describe and quantify human hand motions,[6,7] computer scientists and engineering professionals use a more abstract hand model, principally to allow interaction and fast rendering in real time.[8]

We can look at hand motions as complex combinations of the movements (rotations around different axes) of different bones at various joints. Fingers have distal interphalangeal (DIP), proximal interphalangeal (PIP), and metacarpophalangeal (MCP) joints, whereas thumbs have interphalangeal (IP), metacarpophalangeal (MCP), and carpometacarpal (CMC) joints. Figure 1 illustrates the hand's joints and their degrees of freedom (DOF).

There is only one motion at each finger DIP and PIP joints and at the thumb IP. We call this *flexion* bending, although sometimes the term flexion includes both bending and extending motions. The thumb MCP and CMC joints and the finger MCP joint also have side-to-side movement called *abduction–adduction*. Abduction–adduction is a finger's motion away from–to the middle finger. Sometimes we call both movements simply as abduction.

While the wrist bones have the most complicated movements, the palm bones' movements tend to converge to a point in the wrist bones. Thus, for simplicity, we use only one point to

 Copublished by the IEEE CS and the AIP 1521-9615/05/$20.00 © 2005 IEEE

represent the wrist joint and its six movements: one for bending (flexion), one for side-to-side movement (abduction–adduction), one for rotation (*supination–pronation*), and three for displacement in 3D space.

Even though it's a highly articulated organic structure, the human hand can't generate just any random arbitrary gesture. It is *constrained*: there are limitations on the natural movements of hand parts at their rotation joints:[8,9]

- A finger has a dependency between the flexions of the DIP and PIP joints. Specifically, when you bend your finger at the PIP joint for θ degrees, your finger tip will automatically bend for $2/3\theta$ at its DIP joint.
- The middle finger's MCP joint displays little adduction.
- There are various other constraints among the fingers. For example, when you bend your ring finger at the MCP joint, your middle and little fingers will bend at their MCP joints to different degrees.

These medical biomechanical studies gave us a theoretical base for modeling the human hand and creating hand movement animation sequences.

Constructing a Virtual Hand

We implemented our hand model in Open Inventor (a toolkit for developing interactive graphics; www.sgi.com/products/software/inventor) on an SGI workstation and then transformed it to Coin3d Inventor (www.coin3d.org) on a Linux platform. To improve realism, we included a portion of the forearm to clearly illustrate wrist movements and because the forearm's movement contributes to the hand's displacement in 3D space.

We identified 16 hand parts: three for the thumb, three for each finger, and one for the palm. We determined the hand joints (finger DIP, PIP, and MCP, thumb IP, MCP, and CMC, and wrist) by first identifying the boundary plane of two adjacent hand parts and then calculating the center of the plane. Then, we attached a coordinate system to each of the 16 joints. The coordinate systems serve as the object (local) coordinate systems for hand parts: one hand part belongs to one coordinate system. Each coordinate system attaches to another coordinate system just as its corresponding hand part attaches to another hand part. For example, the index finger has three parts in its corresponding coordinate systems, with origins at its DIP, PIP, and MCP joints. The coordinate system at the DIP joint attaches to the one at the PIP joint, which, in turn, attaches to the one at the MCP joint.

During hand motions, we maintain anatomical features by keeping constant the distance between the bones of two adjacent coordinate systems. We set the origin of a hand part's coordinate system at the hand joint. This approach is similar to the method that medical and biomechanical professionals use for measuring human hand motions.[6]

There are flexions at all hand joints and there are additional abductions at finger MCP joints, at thumb MCP and CMC joints, and at the wrist joint. For added realism, we also added other wrist joint movements: rotation caused by forearm rotation and 3D translation caused by shoulder motion.

The hand parts are primarily rigid except for the components around the boundaries of two adjacent hand parts. When a hand part rotates at a joint, the mesh points that are close to the hand joint on both adjacent hand parts produce the most deformations. We distribute deformation weights according to the point closeness and position. Figure 2 shows the completed virtual hand.

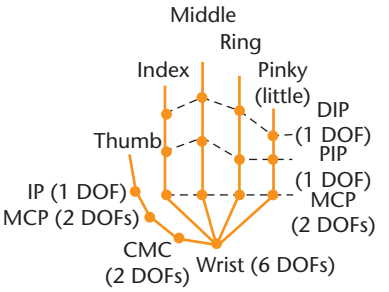

Figure 1. Simplified representation of the hand structure. The lines represent the hand's bones and the dots its joints. Hand motions at joints are marked with degrees of freedom, for example, at the middle finger's, metacarpophalangeal joint, there are two different movements possible.

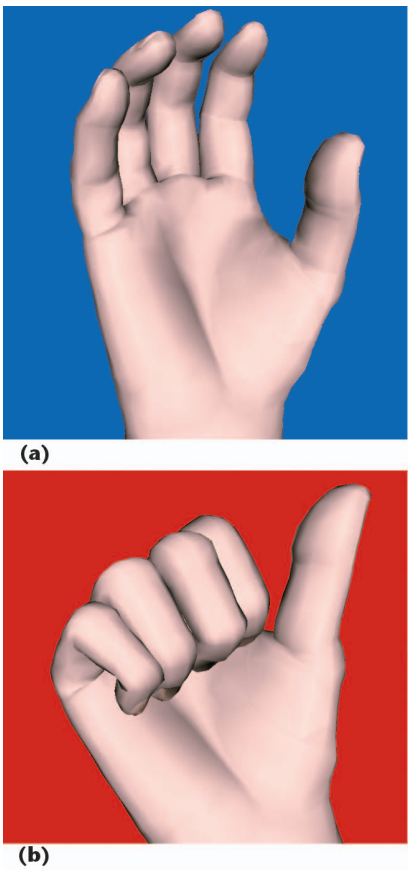

Figure 2. The virtual hand. (a) Open and slightly curved hand and (b) fist with thumb up. These two hand configurations represent the wide range of finger and thumb motions needed to accurately represent a human hand.

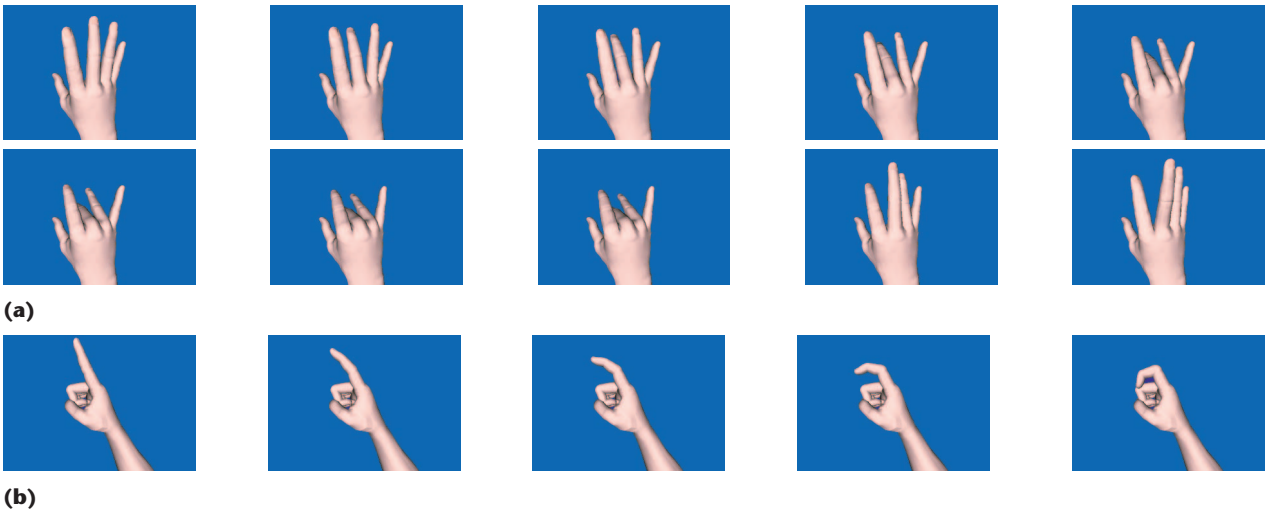

Figure 3. Application of hand constraints at finger joints. (a) Bending and extending the middle finger at the metacarpophalangeal joint and (b) bending the index finger at the distal interphalangeal joint.

Applying Constraints

In motion and under constraints, the human hand generates what we call *natural* hand gestures. As mentioned earlier, one example of a natural gesture is when you bend your middle finger at its MCP joint, your index and ring fingers automatically follow the middle finger's movement; your little finger also follows but to a lesser degree. Although people display differences in hand constraint magnitudes, all natural movements observe the same pattern: related fingers rotate the same way, only less so.

We based our model's hand constraints on the values other medical researchers have obtained.[6,7] We embedded the static hand constraints in the model, to set up motion (rotation) ranges for joint movements. Specifically, there are limitations on flexions at different hand joints, and abductions at each finger MCP joint, at the thumb MCP and CMC joints, and at the wrist. There also are limits on the hand rotation at the wrist.

We then applied the *intrafinger* dynamic constraints (how a joint in one finger affects other joints in the same finger) by making the flexion angle at the DIP joint two thirds of that at the PIP joint, and vice versa. During development, we had to make some constraint corrections because we hadn't strictly applied the medical studies' hand constraints to our hand model because the data we initially used to model our hand didn't match that of the medical studies' hand model. We also implemented the *interfinger* (how one finger moving at its MCP joint affects the other fingers) dynamic constraints at finger MCP joints.

In our hand model, middle and ring fingers have abduction–adduction. The middle finger also can have very slight side-to-side movement. The ring finger follows the little finger when it moves far away from the ring in abduction. For flexions at finger MCP joints, we considered three issues:

- When an active finger bends at its MCP joint, it influences the MCP flexions of other fingers (the passive fingers). The influence depends on the active finger's flexion angle value and on the proximity of the passive finger to the active one. For example, when bending at the MCP joint, the index finger has the strongest influence on the middle finger, moderate on the ring finger, and the least on the little finger; the influence becomes stronger as the index finger bends from small angles to larger angles.
- We define the *influence factor* as an offset by which the passive finger should follow the active finger. There is a maximum difference in flexion angles between the active finger and the passive one. When the flexion difference between the active finger and the passive finger is greater than the maximum value, the passive flexion angle adjusts by the offset so that the difference is equal to the maximum value. When the difference is within the maximum value, the passive finger will also follow the active finger movement by a different offset.
- We estimated all the influence factors among all four fingers through experiments on our hand model and on real human hands.

Figure 3 shows some examples of the application hand constraints at different finger joints.

Creating Gestures

To facilitate users creating hand gestures, we emphasized usability and user experience goals;[10] the system should behave the way users think it should. Specifically, we considered

- building fixed and movable virtual cameras;
- creating several display windows for

virtual hands rendered from various cameras;
- using interface metaphors and affordances for controlling the virtual hand;
- constructing a built-in database for storing hand gestures and animation sequences; and
- combining all these operations in easy-to-use GUIs.

Figure 4a shows an interface that controls thumb joint movement by adjusting joint rotation angles. Figure 4b shows the control panel from which users can create and edit hand animation sequences by inserting and deleting hand gestures stored in gesture database. There are also corresponding control panels for finger movements, controlling camera parameters, and recording rendered hand images.

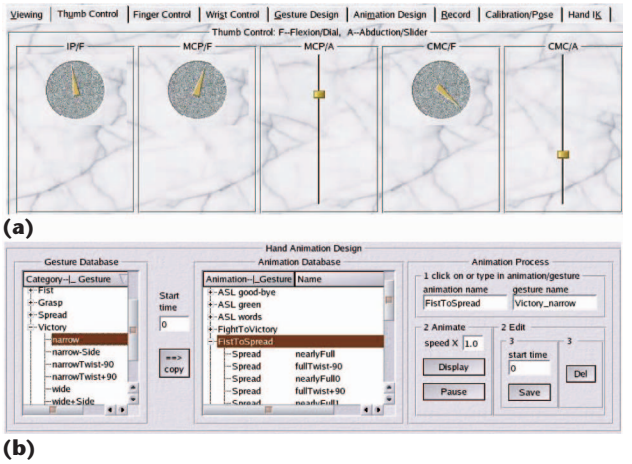

Figure 4. GUIs for generating hand gesture. (a) Thumb joint angle control panel and (b) creating and editing hand gestures.

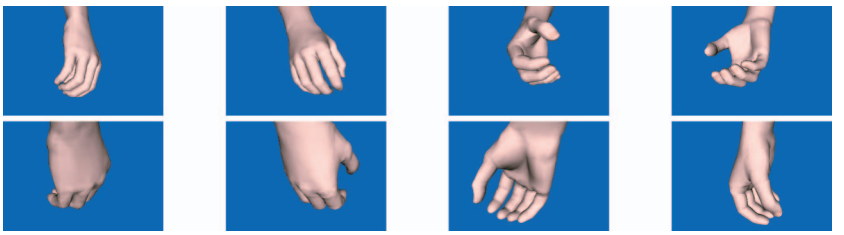

Figure 5. A virtual hand rendered by six different virtual cameras. Up to nine cameras (eight fixed cameras and a movable one) can simultaneously render a hand configuration.

Virtual Hand Rendering

The virtual hand resides in a virtual rectangular box with eight virtual cameras at its corners and a movable camera with its initial position at the center of the front side. All the cameras face the hand. Users control the virtual environment via a Viewing control panel; operations in this panel include setting the background color, coordinating display windows with virtual cameras, and adjusting camera parameters.

To display the virtual hand outputs rendered from different virtual cameras, we defined five display windows positioned above the GUI panels. The GUI widgets include buttons, slider bars, and dials on the viewing panel to set camera parameters (such as viewing point and focus) and background colors, to connect display windows with cameras, and to select render styles (such as wireframe and colored hand). Figure 5 displays a hand posture rendered from different cameras.

Figure 4a's control panel for the thumb's motion has three dials and two slider bars for the thumb flexions at its three joints and abductions at its lower two joints. For wrist movement, three dials and three slider bars control bending, side-to-side rotation, twisting, and displacement in 3D space. Users can adjust finger joint movement in the same way on each control panel.

Generating Gestures and Animation

We designed and implemented a data structure in the simulation system for storing, retrieving, and editing hand gestures, and constructed and stored some basic hand gestures in the database to help users create particular hand gestures. A Hand Gesture Design GUI panel operates with the gesture database.

To create a particular hand gesture, users can load a hand gesture that best approximates the desired one from the gesture database. They then use the thumb, finger, and wrist control panels to fine-tune hand joint motions. For viewing, users can adjust the movable camera, connect it to the largest display window, and then connect some other cameras to the remaining display windows.

Users can select a GUI menu item to automatically apply hand constraints. Turning on the hand constraints function speeds up the construction process by linking joints affected during the gesture's construction. After constructing a gesture, users can store it in a particular gesture group (or category), and then name the gesture and the group

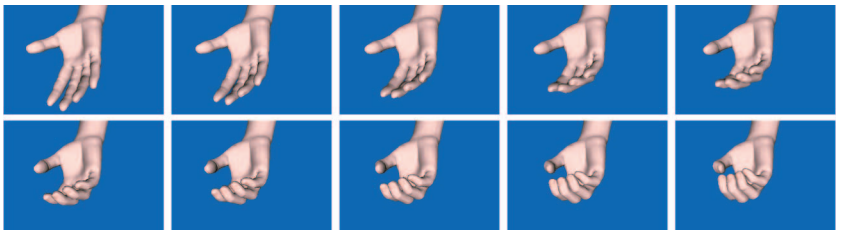

Figure 6. Hand animation (grasping) sequence. Rendering clips of a hand animation from a spread to a cupped hand.

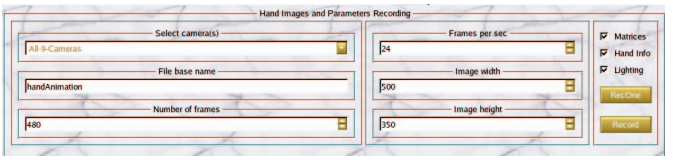

Figure 7. Control panel for recording hand images and rendering parameters, such as joint rotation angles and coordinates). Users can select cameras and define image size and rendering speed.

and edit, save, or delete their entries. Figure 6 shows part of a hand animation sequence (grasping).

The default animation speed is 20 frames per second (fps). While rendering an animation sequence, the visualization system automatically generates new hand configurations based on two adjacent gestures in the animation sequence; new hand configurations are interpolated, and the system renders the newly calculated gestures according to the rendering speed. Users can change the rendering speed by typing in a multiplication factor.

Users can also edit an animation process. To insert a gesture, they choose one from the gesture database, select where it will reside, give the starting time. To change the starting time for a gesture in an animation sequence, they choose that gesture, modify the starting time displayed in the input area under the Edit frame.

Recording Rendering Processes

The system records virtual hand gestures and animations in image files and saves corresponding hand configurations and rendering parameters (such as OpenGL rendering matrices) in text files. Figure 7 shows the recording control panel.

The recording process begins by choosing the cameras (the rendered image sources). The eight fixed cameras and one movable camera can record images from nine different viewing points for a single hand rendering configuration. To get started, users choose an image size and specify a filename.

Recording animation sequences require users to specify the animation recording parameters: total number of frames and the fps. An animation sequence renders images according to the parameters in its database. Because writing images to a hard disk takes time, when the system decides to record a frame (with the result calculated from the animation recording parameters), it stops the rendering mechanism's inner timing clock to write the rendered images for the current hand configuration. After finishing the writing process for the current configuration, it restarts the clock and continues the rendering and recording process.

The system can render a virtual hand using different lighting models and with different-colored fingers. The system can save on disk the rendered images and their rendering parameters such as OpenGL projection and modelview matrices. It also retains hand configurations such as hand orientation, position, and hand joint angles, which are available through the recording panel.

Our current hand model can't represent different hand sizes (for example, thin or thick), so we plan to calibrate it to different representative hand shapes. Also, we should consider more sophisticated deformation at the thumb joints and the application of finger constraints at finger PIP joints to provide more natural hand movements when generating between-finger gestures.

We are extending our hand gesture model to create an HCI system that includes the whole body, which will provide a platform to create virtual human gestures. As a current application, our system is suitable for sign-language studies, but apart from linguistic implementations, a sign is a gesture sequence—an animation of virtual body. CiSE

References

1. D.J. Sturman, *Whole-Hand Input*, doctoral dissertation, MIT Media Laboratory, MIT, 1992.
2. I. Albrecht, J. Haber, and H.-P. Seidel, "Construction and Animation of Anatomically Based Human Hand Models," *Proc. Eurographics Symp. Computer Animation* (SIGGRAPH 2003) ACM Press, 2003, pp. 89–109.
3. G. ElKoura and K. Singh, "Handrix: Animating the Human Hand," *Proc. Eurographics Symp.Computer Animation* (SIGGRAPH 2003), ACM Press, 2003, pp. 110–119.
4. T. Kurihara and M. Miyata, "Modeling Ddeformable Human Hands from Medical Images," *Proc. Eurographics Symp. Computer Animation* (SIGGRAPH 2004), ACM Press, 2004, pp. 355–363.
5. P.W. Brand, *Clinical Mechanics of the Hand*, C.V. Mosby Co., 1985.
6. E.Y.S. Chao et al., *Biomechanics of the Hand: A Basic Research Study*, World Scientific Publishing, 1989.
7. Am. Academy Orthopedic Surgeons, *Joint Motion: Method of Measuring and Recording*,

Churchill Livingstone, 1988.

8. J. Lin, Y. Wu, and T.S. Huang, "Modeling the Constraints of Human Hand Motion," *Proc. Workshop on Human Motion (HUMO'00),* IEEE Computer Society, 2000, pp. 121–126.
9. J. Lee and T.L. Kunii, "Model-Based Analysis of Hand Posture," *IEEE Computer Graphics & Applications*, vol. 15, no. 5, 1995, pp. 77–86.
10. J. Preece, Y. Rogers, and H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*, John Wiley & Sons, 2002.

Beifang Yi is a PhD student in the Department of Computer Science and Engineering at the University of Nevada, Reno. His research interests are in graphics, human–computer interactions, user interface design, and computer vision and image processing. Contact him at b_yi@cse.unr.edu; www.cse.unr.edu/~b_yi.

Frederick C. Harris Jr. is an associate professor in the Department of Computer Science and Engineering at the University of Nevada, Reno. He also has a joint appointment with Desert Research Institute helping establish the Immersive Visualization Lab with several multiwalled visualization systems doing research and visualization in a variety of scientific disciplines including research in desert terrain, forest fires, and atmospheric modeling. He has a PhD in computer science from Clemson University and is a member of the ACM, the IEEE Computer Society, and ISCA. Contact him at fredh@cse.unr.edu; www.cse.unr.edu/~fredh.

Ling Wang is a professor in the College of Computer Science at Sichuan Normal University, China. Her research interests include graphics, digital image processing, wavelet analysis and algorithm analysis. Wang has a PhD in applied mathematics from Xidian University, Xi'an, Shaanxi, China. Contact her at lwang@sicnu.edu.cn.

Yusong Yan is a professor and vice president of Sichuan Normal University in Chengdu, China. His research interests are in applying information technology and virtual reality to transportation related applications. Yan has a BS, an MS, and a PhD in railway transportation and communication from Southwest Jiaotong University, China. Contact him at yanyusong@263.net.