User Interface Aspects of a Human-Hand Simulation System [*]

Beifang Yi   Frederick C. Harris, Jr.   Sergiu M. Dascalu   Ali Erol
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV 89557
{b_yi, fredh, dascalus, aerol}@cs.unr.edu

ABSTRACT

This paper describes the user interface design for a human-hand simulation system, a virtual environment that produces ground truth data (life-like human hand gestures and animations) for computer vision experiments on hand pose estimation and tracking. The system allows users to save time in data generation and easily create any hand gestures. We have designed and implemented this user interface with the consideration of usability goals and software engineering issues.

Keywords: GUI, User Interface Design, Virtual Reality, Software Engineering, HCI

1. INTRODUCTION

In some applications requiring human-computer interaction (HCI), traditional controlling and navigating devices, such as keyboard, mouse, and joystick, become cumbersome and unsuitable for the communication between man and machine. One of the most promising natural HCI media is the use human hand [8, 13, 14]. In order to take advantage of this new input modality, emphasis has been on vision-based studies, hand modeling and hand gesture analysis and recognition [1, 3, 5, 6, 12].

Because computer users increasingly demand an ease-of-use environment for complicated systems, it is critical to design and implement an effective user interface. Most of the vision-based hand projects have no, or poorly-designed, user interfaces (UIs). This paper introduces some of our work in this area. A 3D, life-like hand constructed in a virtual environment is the base of a graphical user interface (GUI), which we call Virtual Hand. Virtual Hand simulates the human hand by adjusting such hand parameters as hand joint rotations and flexions. This software produces ground truth data for the hand for use in computer vision experiments. The UI design and implementation for this simulation system followed accepted standards in UI design and software engineering.

The organization of this paper is as follows: Section 2 briefly discusses the related work and the construction of the Virtual Vand, which is a prerequisite for the UI design. Section 3 describes the UI design issues and then gives UI software specifications such as requirements analysis and use case and scenarios. Section 4 presents UI design aspects with some prototypes and results. Section 5 gives conclusions.

[*]This work has been partially supported by NASA under grant NCC5-583

2. RELATED WORK

The Virtual Hand is a 3D, life-like hand together with the environment in which the hand is placed. It can be used in computer vision research on the human hand in the areas of hand gesture analysis, recognition and tracking. There are two different applications of Virtual Hand. One application is the production of ground truth data of the hand. Computer vision research on the hand needs many accurate images of a normal hand taken from different viewing directions with various camera parameters under certain (fixed) environmental conditions. It is very difficult, if not impossible, to ask a person to hold his hand in a fixed hand gesture for several minutes in order to move the camera to predefined locations in 3D space or to keep the illumination uniform in every direction. But in a virtual environment, with a life-like hand, we can define accurately any hand gestures, camera position, viewing geometry, and illumination configuration, and thus create hand images based on these parameters.

Another application of Virtual Hand is visualization. Once hand gestures are recognized and tracked, we need to display a virtual hand in a 3D virtual environment. Also, we can generate hand gesture animations by defining a sequence of hand gestures.

Hand modeling is based on results from hand biomechanic studies and techniques in computer graphics. We can think of the human hand as a machine with joints [2]. Thus hand motions involve movements at joints. The structure of the hand can thus be visualized as in Figure 1. From this figure, we see that each of the fingers (index, middle, ring, and pinky) and the thumb have three joints with the joint names and the number of DOFs (degrees of freedom) marked beside the joints. The human hand has 27 DOFs.

One DOF corresponds to one movement of a joint, and there are three possible movements at a joint: bending (flexion), side-to-side (abduction), and rotation (twist). These three movements can be described as rotations around certain axes. Detailed description and measurement of hand joint motions are given in [4] and [7]. By giving different rotation angles for hand joints, we can model various hand gestures. Two such hand gesture snapshots are shown in Figure 2. More information about hand modeling and hand gesture design can be found at `http://www.cs.unr.edu/~b_yi/vHand`.

The large number of DOFs in the hand and various situations in the virtual environment make it awkward to adjust hand joint parameters for a correct hand gesture by using

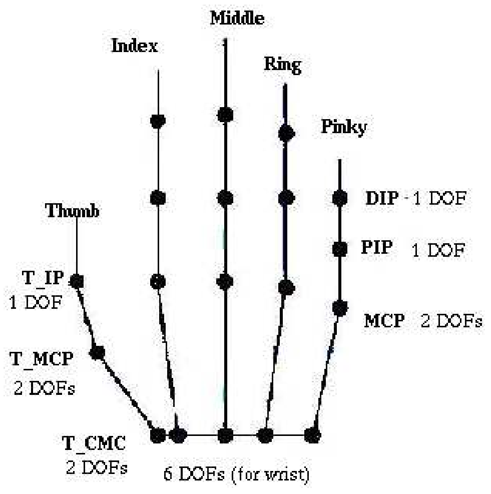


Figure 1: The stick representation of the hand structure with joints (adapted from [14]).

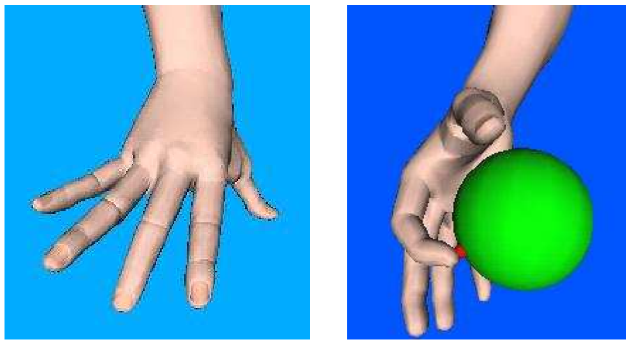


Figure 2: Virtual hand (environment) snapshots.

traditional keyboard input or a predefined parameter text file. Therefore there is a need to create a GUI for such a task.

3. USE INTERFACE DESIGN ISSUES AND SPECIFICATIONS

In this section, we describe the design issues, requirements, use cases, and scenarios for the UI system. The specifications follow the formats proposed in [9] and [10].

User Interface Design Issues

The intended users of the simulation system include not only computer vision researchers who do experiments on human hand gesture analysis and recognition and hand tracking, but also those people who are interested in hand animation and want to produce a sequence of hand gesture images. As a general principle, the user interface should be designed by assuming that the program model (how the program runs) should conform to the user model (the way users think the program should behave) [11].

Therefore, our UI design should consider the following issues:

- The design should reflect the features of the hand model and emphasize the convenient manipulation of the hand and its joints.
- Because the graphical library used for hand modeling is multi-platformed, the software package should be in multi-platformed.
- Display of the virtual hand will be from eight fixed cameras at the corners of a virtual cube in which the hand resides in the center. There should also be a movable camera so that we can look at the hand from any viewpoint.
- There should be interfaces for producing natural hand gestures by controlling the hand joints, for testing the hand's (inverse) kinematics, for calibrating the virtual hand with a real hand in the real world, and for recording the hand gesture image sequences (animation).
- There should be interfaces for displaying and recording hand (rendering) parameters such as the hand poses, hand location, joint angles, and OpenGL matrices for hand rendering.
- There should be interface for adjusting the cameras' parameters, setting the environment (for example, background) hand materials.

As for the usability, the program should be easy to learn. Once the user starts up the program, he/she should know how to control hand joints, adjust/choose cameras, display the effect, and record the images immediately. The user should not have to remember anything to use the system. There should be "what-is" (the question mark) tips to help user for relatively complicated operations. The menu (commands) design should follow the styles in such popular softwares as Microsoft Office and Linux's OpenOffice.

Making hand gestures is a natural behavior, so the system should provide a convenient operational process to generate hand gestures. The GUI items should give some kind of affordance and good mapping. Different control operations should be clearly visible and the effect and actions (feedback) should be accurately displayed in the displaying windows.

Functional and Non-Functional Requirements

The functional requirements for our UI system are:

1. A hand model has been constructed which allows for rotations, translations, scaling, and rotational angle adjustments at hand joints. The virtual hand is at the center of a virtual box when the system starts up.
2. A big displaying window and several small ones are needed for displaying the hand from different view points. These windows should be combined with other GUI control widgets into a single interface.
3. There are eight cameras fixed at the corners of the box and one non-fixed camera can be set up by the user. There are GUI widgets for choosing and controlling these cameras.
4. The system has GUI widgets to control hand joint rotation angles for all the DOFs.
5. There is an image (sequence) recording mechanism in which users can set up the image size, speed and duration (in animation), file names, and image format. Also, the corresponding hand and OpenGL parameters with the images are recorded in text files.
6. The system provides interfaces for calibration of the hand and testing hand (inverse) kinematics.
7. The system allows the user to choose background colors, set illumination conditions, and specify materials for rendering the virtual hand. For example, sometimes

different parts of the hand need represented with different colors.

And the non-functional requirements can be seen as:

1. The end product must be able to run on a variety of platforms such as Windows, Linux, and Unix.
2. The users of the system include both well-trained computer scientists and computer novices.
3. The output image sequences and their corresponding parameter text files should be named intuitive names, so that users can easily find the files they want.

Use Case Modeling

The main use case diagram for the hand simulation system is shown in Figure 3.

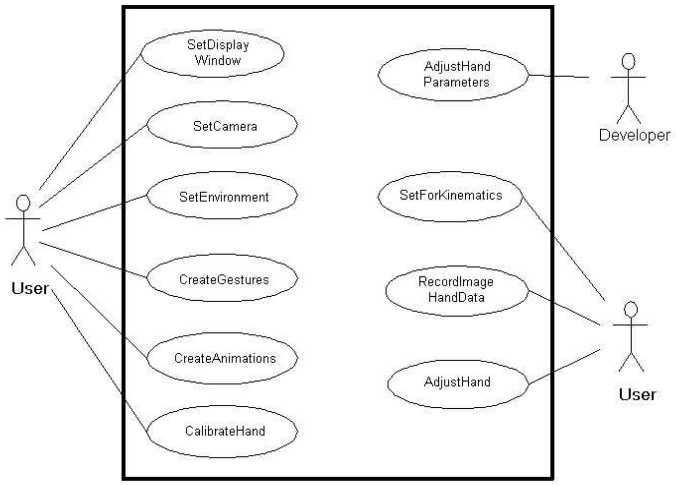
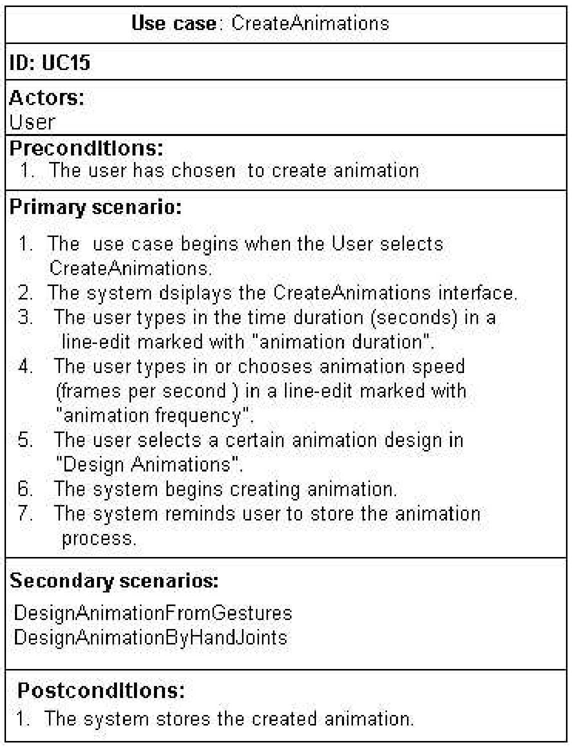


Figure 3: The system use case diagram.

One of the use cases in the diagram, *CreateGestures*, might involve the following:

1. The user chooses *createGestures* tab and goes into an interface.
2. The user selects one of the four fingers or the thumb.
3. The user selects one of the four finger/thumb joints.
4. The user selects one of the movements: flexion (bending), abduction (side-side), and rotation (twist).
5. The user types in the rotational angles or uses a slide-bar to modify the angles.
6. The hand will update according to the user's inputs.
7. The user repeats the above steps until he produces the desired gesture.
8. If the user chooses to store the gesture in the gesture database:
   - The system asks for gesture name.
   - The user types in the name.
   - The system asks for the gesture type.
   - The user types in a new type or chooses one from the list of current gesture types.
   - The system stores the gesture in its database.
9. A new gesture is displayed.

Two use cases with UML scenarios with primary and secondary scenarios are presented. Figures 4 and 5 show the use case *CreateAnimations*' primary and secondary scenarios. Figures 6 and 7 show the use case *RecordImagehandData*'s primary and secondary scenarios.



Use case: CreateAnimations

ID: UC15

Actors:
User

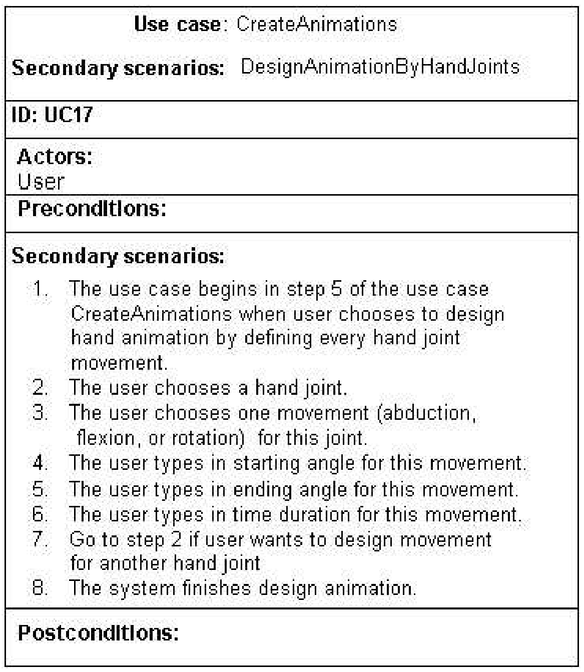
Preconditions:
1. The user has chosen to create animation

Primary scenario:
1. The use case begins when the User selects CreateAnimations.
2. The system dsiplays the CreateAnimations interface.
3. The user types in the time duration (seconds) in a line-edit marked with "animation duration".
4. The user types in or chooses animation speed (frames per second) in a line-edit marked with "animation frequency".
5. The user selects a certain animation design in "Design Animations".
6. The system begins creating animation.
7. The system reminds user to store the animation process.

Secondary scenarios:
DesignAnimationFromGestures
DesignAnimationByHandJoints

Postconditions:
1. The system stores the created animation.

Figure 4: UML Scenario: use case *CreateAnimations*' primary scenario.



Use case: CreateAnimations

Secondary scenarios: DesignAnimationByHandJoints

ID: UC17

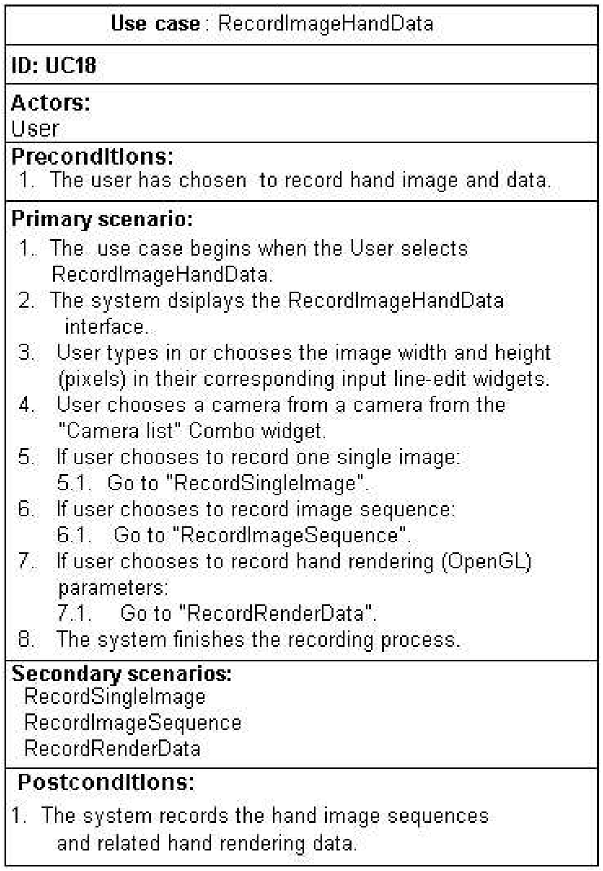
Actors:
User

Preconditions:

Secondary scenarios:
1. The use case begins in step 5 of the use case CreateAnimations when user chooses to design hand animation by defining every hand joint movement.
2. The user chooses a hand joint.
3. The user chooses one movement (abduction, flexion, or rotation) for this joint.
4. The user types in starting angle for this movement.
5. The user types in ending angle for this movement.
6. The user types in time duration for this movement.
7. Go to step 2 if user wants to design movement for another hand joint
8. The system finishes design animation.

Postconditions:

Figure 5: UML Scenario: use case *CreateAnimations*' secondary scenario.



Use case : RecordImageHandData

ID: UC18

Actors:
User

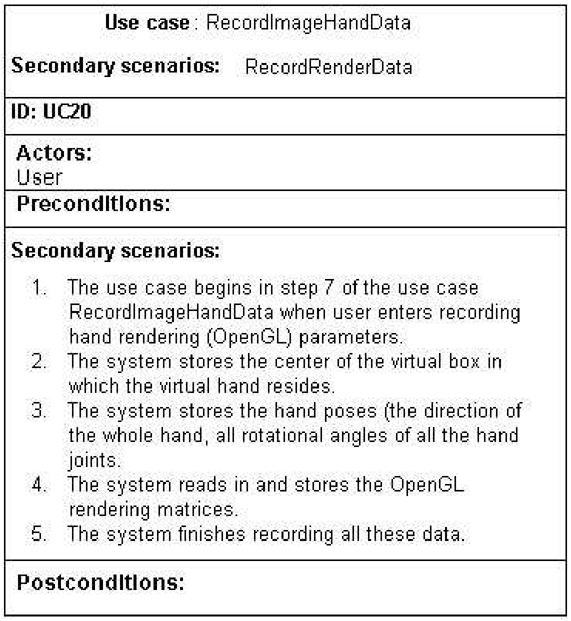
Preconditions:
1. The user has chosen to record hand image and data.

Primary scenario:
1. The use case begins when the User selects RecordImageHandData.
2. The system dsiplays the RecordImageHandData interface.
3. User types in or chooses the image width and height (pixels) in their corresponding input line-edit widgets.
4. User chooses a camera from a camera from the "Camera list" Combo widget.
5. If user chooses to record one single image:
   5.1. Go to "RecordSingleImage".
6. If user chooses to record image sequence:
   6.1. Go to "RecordImageSequence".
7. If user chooses to record hand rendering (OpenGL) parameters:
   7.1. Go to "RecordRenderData".
8. The system finishes the recording process.

Secondary scenarios:
RecordSingleImage
RecordImageSequence
RecordRenderData

Postconditions:
1. The system records the hand image sequences and related hand rendering data.

Figure 6: UML Scenario: use case *RecordImageHand-Data*' primary scenario.



Use case : RecordImageHandData

Secondary scenarios: RecordRenderData

ID: UC20

Actors:
User

Preconditions:

Secondary scenarios:
1. The use case begins in step 7 of the use case RecordImageHandData when user enters recording hand rendering (OpenGL) parameters.
2. The system stores the center of the virtual box in which the virtual hand resides.
3. The system stores the hand poses (the direction of the whole hand, all rotational angles of all the hand joints.
4. The system reads in and stores the OpenGL rendering matrices.
5. The system finishes recording all these data.

Postconditions:

Figure 7: UML Scenario: use case *RecordImageHand-Data*' secondary scenario.

4. DESIGN ASPECTS

An Overview

The basic concept in the UI design for a human-hand simulation environment is the interaction between the user and the virtual environment. Figure 8 illustrates the design architecture.

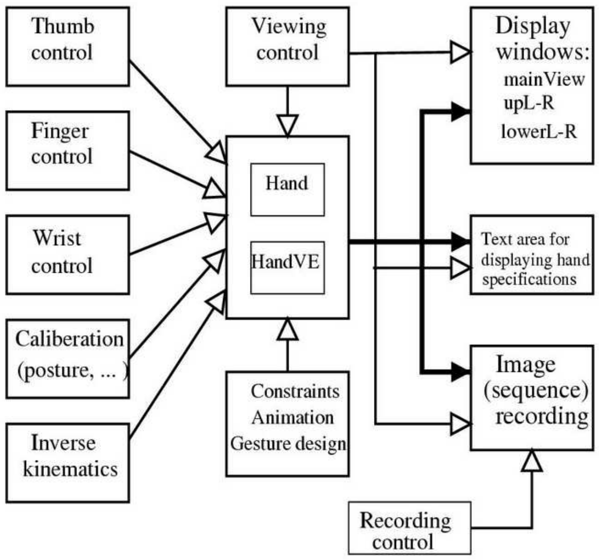


Figure 8: Virtual hand user interface design architecture.

Assumption The assumption for the UI design is that a hand model has been constructed: a C++ class named *Hand* is created with a cross-platform graphical library, *Coin3d*. We also assume that *Hand* has provided all APIs for operations on the virtual hand such as:

- Displace the hand to a location.
- Rotate the hand to a direction.
- Assert various kinds of movements at any hand joints in the ranges of natural hand movements.
- There are (OpenGL) rendering parameters (matrices) corresponding to the hand (joint) movements. And these parameters can be read out.
- The virtual hand is rendered and displayed with a pre-defined parameters.

Also, we will use the GUI toolkit *Qt* for UI design and a software package *SoQt* to connect UI packages with the rendered images of the hand model.

Conceptual models The system will be mainly based on an instructing model and manipulating and navigating models.

- Instructing model: the user gives instructions directly not with command line environment but by manipulating widgets such as buttons, slide-bars, combo boxes, dials, checkboxes, and radio buttons.
- Manipulating and navigating model: direct manipulation will be used for selecting and operating on any part of the virtual hand in the displaying window. Other operations in this conceptual model include setting the

virtual environment and choosing and designing hand gestures.

Interface metaphors and affordance This system provides direct denotations for the operations. When the system is started up and the interface is displayed, many GUI widgets such as those above described are available to control and adjust the virtual hand. Intuitive names for these widgets indicate their functtion.

Key elements and characteristics HCI design principles for this simulation system will follow the guidelines proposed in [9]. UI design rules and programming experiences given in [11] are applied as much as possible in the design process. The key rules and guidelines include:

- Keep the program model in line with the user model (the user' mental understanding of what the program will do for them).
- Consistency issues. For example, menu commands should follow the common way.
- Interface layout.
- Users don't have to read operation instructions to operate the virtual hand, and no operation process has to be remembered. For some real complicated operations, we used *Qt*'s *Tooltips* and *"What's This" Windows* as a guidance for the operation. The widgets are laid out in such a way that users can use a mouse to point to and move them easily.

Results

We present two high-fidelity prototypes as results. They are nearly the same as the control panels in our user interface for the system. More results, including demos, can be found at `http://www.cs.unr.edu/~b_yi/vHand`.

Use case *CreateGestures* prototype Once the user chooses *CreateGestures* tab and selects one of the four fingers or the thumb, the interface appears as in Figure 9.

Use case *RecordImageHandData* prototype Figure 10 shows the use case *RecordImageHandData* prototype (control panel). The user can use this panel to record hand gestures in image format and their corresponding OpenGL parameters. Also, the hand animation process can be recorded at different speeds. The controlling parameters include speed choice, image size, camera selection, and rendering styles.

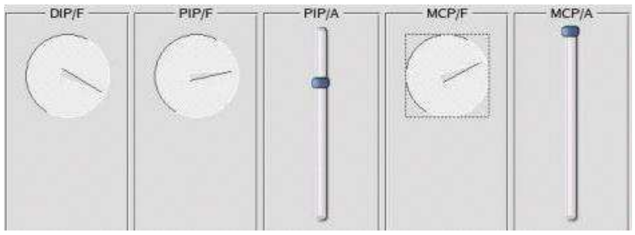
Use case *SetCamera* prototype Use case *SetCamera*'s prototype (controlling panel) is shown in the lower part of Figure 11. With this panel, the user can select a camera and its corresponding display window, adjust the camera's parameters, and move the camera in the virtual environment.

Layout A screenshot of Virtual Hand showing the interface layout is presented in Figure 11.

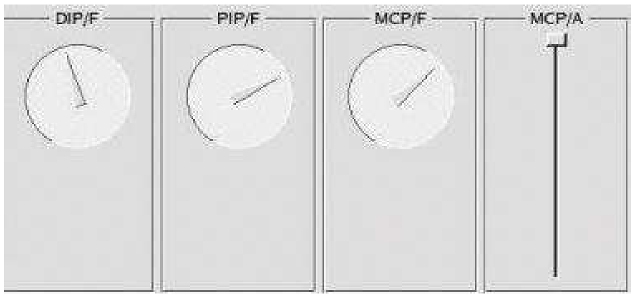
5. CONCLUSIONS

We have designed and implemented a user interface for a human-hand simulation environment, with emphases on usability goals and UI issues. Hand modeling and its application in computer vision research is not new, but construction of a good user interface for the purpose of effectiveness, efficiency, and ease of use is relatively new. In this paper we have presented our work on this task.

We have already combined this system with the output from several modules related to hand tracking and computer vision. The goal is to set up a virtual glove box to train scientists preparing to go to the International Space Station. Results are encouraging and show the flexibility of our interface and design.

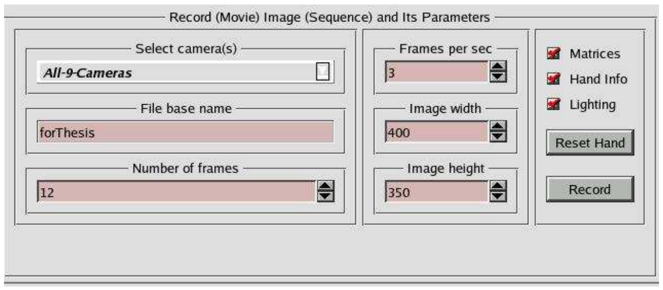


(a) Selection of thumb.



(b) Selection of finger.

Figure 9: Use case *CreateGestures* panels.



Record (Movie) Image (Sequence) and Its Parameters

Select camera(s): All-9-Cameras
File base name: forThesis
Number of frames: 12
Frames per sec: 3
Image width: 400
Image height: 350
✓ Matrices
✓ Hand Info
✓ Lighting

Figure 10: Use case *RecordImageHandData*'s panel.

REFERENCES

[1] Inductive learning in hand pose recognition [online, cited July 3, 2003]. Available from World Wide Web: `http://www.computer.org/proceedings/fg/7713/77130078abs.htm`.

[2] Paul W. Brand. *Clinical Mechanics of the Hand*. The C. V. Mosby Company, 1985.

[3] Lars Bretzner and et al. A prototype system for computer vision based human computer interaction. Available from World Wide Web: `http://www.nada.kth.se/cvap/abstracts/cvap251.html` [cited July 3, 2003].

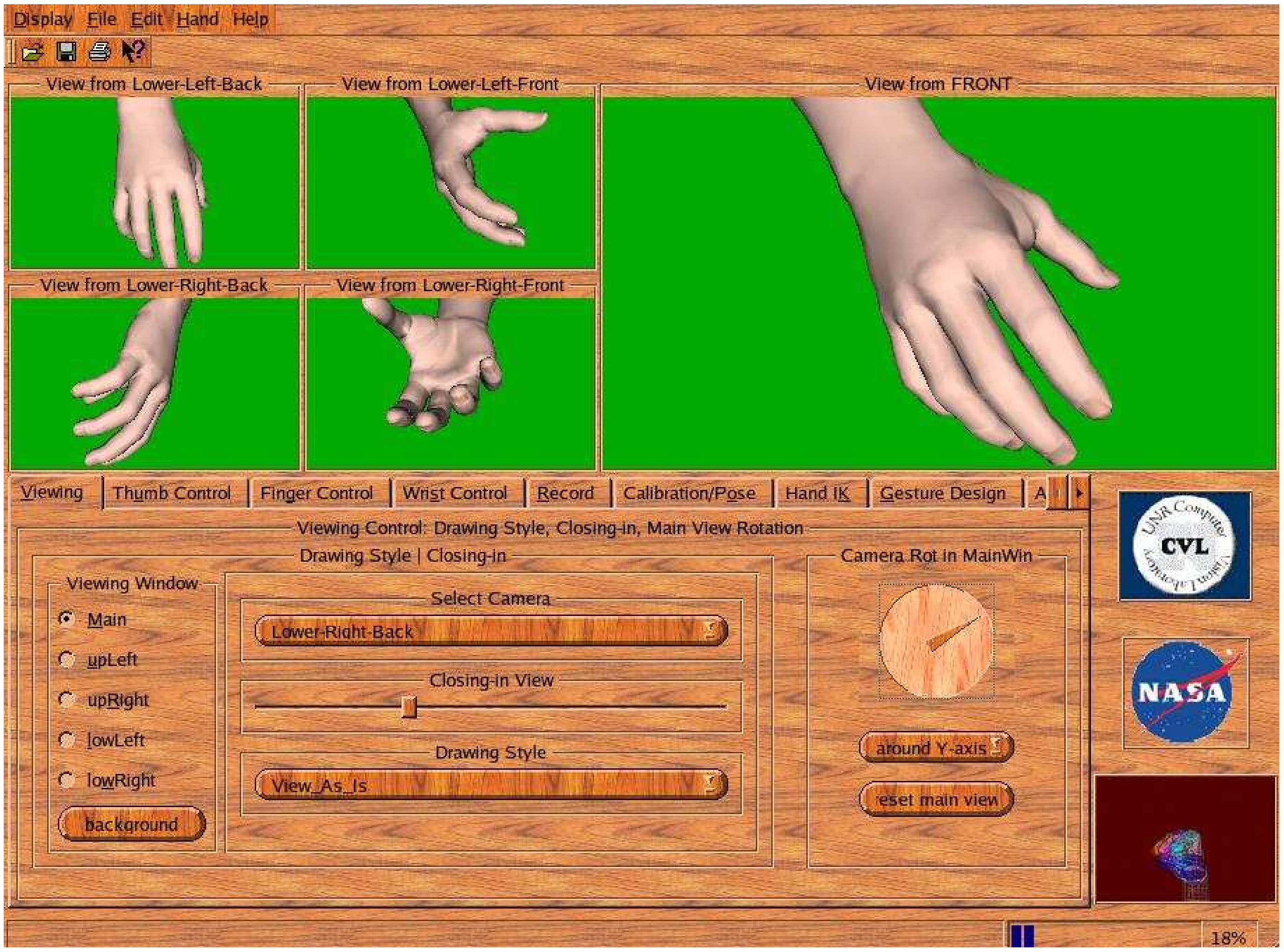


Figure 11: The simulation system user interface.

[4] Edmund Y. S. Chao, Kai-Nan An, William P. Cooney III, and Ronald L Linscheid. *Biomechanics of the Hand: A Basic Research Study.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1989.

[5] Hitoshi Hongo and et al. Focus of attention for face and hand gesture recognition using multiple cameras. Available from World Wide Web: `http://www.computer.org/proceedings/fg/0580/0580toc.htm` [cited July 3, 2003].

[6] Nebojsa Jojic and et al. Detection and estimation of pointing gestures in dense disparity maps. Available from World Wide Web: `http://www.computer.org/proceedings/fg/0580/0580toc.htm` [cited July 3, 2003].

[7] American Academy of Orthopaedic Surgeons. *Joint Motion: Method of Measuring and Recording.* Churchill Livingstone, New York, 1988.

[8] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. Available from World Wide Web: `http://citeseer.nj.nec.com/pavlovic97visual.html` [cited July 3, 2003].

[9] Jennifer Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design: beyond human-computer interaction.* John Wiley & Sons, Inc, 2002.

[10] Ian Sommerville. *Software Engineering.* Addison-Wesley, sixth edition, 2000.

[11] Joel Spolsky. *User Interface Design for Programmers.* Apress, 2001.

[12] Thad Starner, Joshua Weave, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. Available from World Wide Web: `http://vismod.www.media.mit.edu/cgi-bin/tr_pagemaker` [cited July 3, 2003]. PAMI; Submitted 4/26/96.

[13] David Joel Sturman. *Whole-hand Input.* PhD thesis, Massachusetts Institute of Technology, Febrary 1992. Available from World Wide Web: `http://xenia.media.mit.edu/~djs/thesis.ftp.html` [cited July, 2003].

[14] Ying Wu and Thomas S. Huang. Hand modeling, analysis, and recognition for vision-based human computer interaction. *IEEE Signal Processing Magazine,* 18(3):51–60, May 2001.