CHARADE: Remote Control of Objects using Free-Hand Gestures

Thomas Baudel and Michel Beaudouin-Lafon

L.R.I. - CNRS URA 410
Bâtiment 490, Université de Paris-Sud
91405 Orsay Cedex - FRANCE
33+ 1 69 41 69 10
thomas@lri.fr, mbl@lri.fr

ABSTRACT

This paper presents an application that uses hand gesture input to control a computer while giving a presentation. In order to develop a prototype of this application, we have defined an interaction model, a notation for gestures, and a set of guidelines to design gestural command sets. This works aims to define interaction styles that work in computerized reality environments. In our application, gestures are used for interacting with the computer as well as for communicating with other people or operating other devices.

KEYWORDS

Hand gesture input, Interaction model, Computerized reality, Remote controlled user interfaces, Computer-aided presentations.

INTRODUCTION

> *The machine was rather difficult to operate. For years, radios had been operated by means of pressing buttons and turning dials; then, as the technology became more sophisticated, the controls were made touch sensitive … now all you had to do was wave your hand in the general direction of the components and hope. It saved a lot of muscular expenditure of course, but meant you had to stay infuriatingly still if you wanted to keep listening to the same programme.* D. Adams, 1979 [1]

Using free-hand gestures as an input media is not a new idea. In 1979, the "put that there" experiment [3] already used primitive gestural input. Three main directions have been investigated so far:

• *Virtual Reality Systems*, in which the user directly manipulates the objects in the application, presented as embodied physical objects [8]. Most work in this area merely presents hand gesture recognition in the specific context of the application [2].

• *Multi-Modal Interfaces*, in which the user issues commands by using natural forms of human-to-human communication: speech, gesture and gaze (see for instance [4], [14]).

• *Recognition of Gestural Languages*, in which the user issues commands with gestures. Deaf sign language recognition constitutes the main stream of those attempts (see for instance [10]). Other approaches consist of recognizing specific gestural commands. For instance, Sturman [13] presents a system that recognizes gestures for orienting construction cranes; Morita et al. [9] show how to interpret the gestures of a human conductor to lead a synthesized orchestra.

The application presented in this paper fits in the last category and is also of interest to designers of multi-modal interfaces who wish to use gestural input. It allows a speaker giving a presentation to control a computer display by means of hand gestures. This application is an example of a computerized reality environment [15]: the display presented to the audience is an active surface that reacts to the speaker's gestures; yet the speaker can use gestures for communicating with the audience and operating other devices. In order to explore the possibilities of this style of interaction, we have developed:

• a model for identifying and recognizing gestures;
• a notation for recording gestures; and
• a Macintosh prototype.

We have also conducted user tests to evaluate the effectiveness of the software and the acceptance by users.

The paper is structured as follows: we first describe the advantages and drawbacks of hand gesture input, and the prototype application that we have developed; we then present the underlying interaction model, and a notation for gestural command sets. We then describe the implementation of the application and the results of the usability tests. Finally we outline other potential applications and the directions for future work.

HAND GESTURE INPUT

Using free-hand gesture input has several expected advantages:

• *Natural interaction*: Gestures are a natural form of communication and provide an easy-to-learn method of interacting with computers.

• *Terse and powerful interaction*: Devices that capture the precise position and movements of the hand provide the opportunity for a higher power of expression. A single gesture can be used to define both a command to be executed and its parameters (e.g. objects, scope).

• *Direct interaction*: From a cognitive aspect, the hand becomes *the* input device, theoretically eliminating the need for intermediate transducers. The user can interact with the surrounding machinery by simple designation and adequate gestures. It is also possible to emulate other devices (e.g. a keyboard using finger alphabets).

Hand gesture input also has drawbacks. Some are intrinsic to gestural communication:

• *Fatigue*: Gestural communication involves more muscles than keyboard interaction or speech: the wrist, fingers, hand and arm all contribute to the expression of commands. Gestural commands must therefore be concise and fast to issue in order to minimize effort. In particular, the design of gestural commands must avoid gestures that require a high precision over a long period of time.

• *Non self-revealing*: The set of gestures that a system recognizes must be known to the user. Hence, gestural commands should be simple, natural, and consistent. Appropriate feedback is also of prime importance.

Other drawbacks are due to limitations in the current technology and recognition techniques:

• *Lack of comfort*: Current hand gesture input devices require wearing a glove and being linked to the computer, reducing autonomy. Using video cameras and vision techniques to capture gestures [7] will eventually overcome this problem.

• *"Immersion Syndrome"* Most systems capture every motion of the user's hand. As a consequence, every gesture can be interpreted by the system, whether or not it was intended, and the user can be cut off from the possibility of communicating simultaneously with other devices or persons. To remedy this problem, the system must have well-defined means to detect the *intention* of the gesture. It should be noted that this problem does not occur in virtual reality systems, since they promote the notion of immersion: the user is visually and acoustically surrounded by a synthesized world, hence his or her gestures can be addressed only to the system.

• *Segmentation of hand gestures*: Gestures are by nature continuous. A system that interprets gestures to translate them into a sequence of commands must have a way of segmenting the continuous stream of captured motion into discrete "lexical" entities. This process is somewhat artificial and necessarily approximate. This is why most systems recognize steady positions instead of dynamic gestures.

In order to reduce the intrinsic drawbacks of hand gesture input and to overcome its current limitations, we examined the characteristics of the structure of gestural communication. This led us to an interaction model that overcomes the immersion syndrome and segmentation problems. The interaction model was developed during the design of a prototype application. This application aims to demonstrate the usability of hand gesture input applications in the real world.

THE APPLICATION

Our application uses hand gestures to control computer-aided presentations. Using a computer for a presentation is more and more common. Indeed it has several advantages over slides and overheads. The order of the presentation is not fixed, the production process is simpler, and last minute changes can be made easily. It is also possible to enhance the presentation with audio, video, animation, and interactive programs. Finally, it is easier to refer to the presentation when answering a question.

However, people rarely take advantage of all these features because operating the system is more difficult than using slides or overheads. The speaker has to use multiple devices (e.g. keyboard, mouse, VCR remote control) with unfamiliar controls. These devices are hard to see in the dark, and operating them disrupts the course of the presentation.

We propose to solve this problem by using hand gestures to control the system. Our current prototype allows browsing in a hypertext system (namely HyperCard™ on Apple Macintosh™), using the following hardware (photo 1 & 2):

Photos 1 & 2 : Application Hardware

• An overhead projector and LCD display project the display of an Apple Macintosh on a vertical screen. We call the projection of the display on the screen the *active zone*.

• A VPL DataGlove™ [16] is connected to the serial port of the Macintosh. The DataGlove uses fiber optic loops to measure the bendings of each finger, and a Polhemus™ tracker to determine the position and orientation of the hand in 3D space. The fixed part of the Polhemus tracker is set to the top-left corner of the screen. This defines the following coordinate system (figure 1): X and Y correspond to the traditional coordinate system of a graphics screen (Y increasing downwards); Z is the distance to the screen.

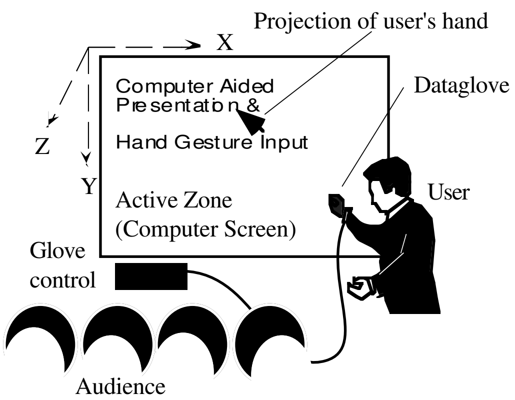

Figure 1 - Setting of the application.

In order to use the system, the user wears the DataGlove. When the projection of his or her hand along its pointing direction intersects the active zone, a cursor appears on the screen and follows the hand. The speaker can issue commands by pointing at the active zone and performing gestures. By means of 16 gestural commands, the user can freely navigate in a stack, highlight parts of the screen, etc. For instance, moving the hand from left to right goes to the next slide, while pointing with the index and circling an area highlights part of the screen.

Using gestures to navigate in the system enables the user to suit the action to the word: the gestural commands fit quite naturally the course of the presentation, and most gestures are actually performed at the limit of consciousness. This sense of control lets the user feel free to orient the presentation according to his or her feelings rather than follow the ordered set of slides. The user can still perform any action in the real world, since the gestures are interpreted only when the hand points to the screen. The user can even show the screen and point at it, since only gestures known to the system are interpreted as commands.

INTERACTION MODEL

The design of this application led us to define an interaction model, i.e. a set of rules and guidelines that an interactive application should follow to provide a consistent interaction scheme within a particular context. In our case, we wanted to use hand gesture input in a real world context, avoiding the immersion syndrome and the fatigue generated by arm motion, and solving the segmentation problem. Some of the rules and suggestions we propose are implied by the media and the context of use, while others are more application specific.

We first describe the *rules* we have adopted. They define the general structure of the human-computer dialog and thus can be considered as axioms of the model. We then describe a notation for gestures applicable to our model. Finally, we present *guidelines* for designing gestural command sets, based on our design experience and tests of gestural interfaces.

Structure of the Model

Our model is based on the notions of active zone and gestural commands. The *active zone* is a 2D area, typically the projection of a computer screen on a wall. Gestures are interpreted only when the user designates this area, i.e. when the projection of the hand enters the active zone.

Each *gestural command* is described by a start position, a dynamic phase and an end position. The user issues a command by pointing to the active zone, using one of the start positions and moving his or her hand (and arm) according to the dynamic part. The user can end the command either by leaving the active zone or by using an end position. The start and end positions do not require the hand to be steady, allowing fast and smooth input of commands.

The recognition of a command involves three steps: detection of the intention to address a command to the system, segmentation of gestures (recognition of start and end positions), and classification (recognition of a gesture in the command set). As soon as a command is recognized, it is issued. Gestures that are not recognized are simply ignored.

• *Detection of the intention*. Gestures are interpreted only when the projection of the hand is in the active zone. This allows the user to move and perform gestures in the real world. It also makes it possible to use several active zones to address several different systems.
• *Segmentation of gesture.* Start and end positions are defined by the wrist orientation and finger positions. These dimensions are quantized in order to make positions both easier to recognize by the system and more predictable by the user. We use seven orientations of the wrist, four bendings for each finger, and two for the thumb. This gives theoretically 3584 positions, among which at least 300 can be obtained with some effort and between 30 to 80 are actually usable (depending on the user's skill and training).
• *Classification*. The different gestures are classified according to their start position and dynamic phase. The dynamic phase uses the path of the projection of the hand, the rotation of the wrist, the movements of the fingers, and the variation of distance between the hand to the active zone (allowing for push-like gestures). For example, our application uses the same start position to navigate to the next

and previous pages. The main direction of the gesture (right-to-left or left-to-right) indicates whether to navigate to the next or previous page. Moreover, opening the hand once or twice during the movement allows to skip one or two pages.

In order to increase the usability of the system, we imposed two constraints on the gestural command set. The first constraint requires that all start positions differ from all end positions. This enables users to issue commands smoothly, without being forced to hold their hands steady or stop between commands. This also makes it possible to issue multiple commands with a single movement. The second constraint requires that gestural commands do not differ solely by their end positions. This gives users the choice of terminating a command either by using an end position or by leaving the active zone. In practice, except for gestures with a steady dynamic phase, most users choose the latter.

Notation for Gestural Commands

When designing gestural command sets, the need for a notation to describe the available commands becomes obvious: designers must be able to document commands for application users. We examined deaf sign language notations as a possible model, however these are usually incomplete and difficult to understand. Since our interaction model precisely delimits the types of gestures that can be recognized, we devised a simple and complete icon-based notation that should be easily understood by the end user. The application that we have developed to create and edit command sets automatically generates the graphical description of the commands according to this notation.

Figure 2 shows an example of the notation. We assume here that the *right* hand is used for issuing commands. A gestural command is represented by a set of 3 icons. The first icon describes the start position, the second describes the dynamic phase of the gesture, and the last icon shows the end position. Start and end position icons show the orientation of the wrist and the position of the fingers. The dynamic phase icon shows the trajectory of the projection of the hand. Additional marks describe finger and wrist motions that are not implicitly defined by differences between start and end position icons: V-shapes indicate one or two finger bendings, lines parallel to the trajectory indicate variations in the distance to the active zone (for "button press"-like gestures), and short segments indicate wrist rotations.

Coordinates of the active zone can be sent to the application upon recognition of a command by specifying their location in the dynamic phase icon. These locations are indicated by circles along the trajectory. Most often, these locations are at the start and end of the trajectory.

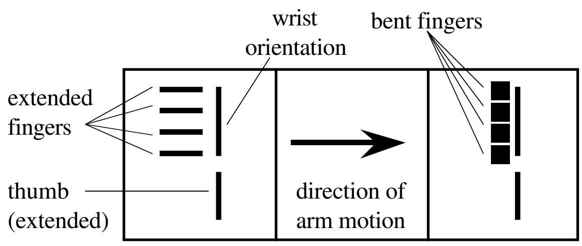

Figure 2 - "Next Chapter" gesture (for the right hand). When pointing at the active zone, this command is issued by orienting the palm to the right (thumb down), all fingers straight, and moving from left to right. The gesture can be completed by bending the fingers or moving the arm to the right until the projection of the hand leaves the active zone.

Figure 3 shows the complete command set of our prototype application. Some commands illustrate the use of the marks described above. For example, the "Go Chapter" dynamic phase icon contains a circle at the start of the trajectory. This means that the position of the cursor when the gesture is started is sent to the application. The application uses this location to determine which chapter to go to (the chapters are represented as icons on the screen). As another example, V-shapes in the dynamic phase icons of the "Next Page x2" and "Next Page x3" commands indicate one or two bendings of all four fingers during the arm motion. Finally, the dot in the dynamic phase icon of the "Start/Stop Auto-Play" indicates that the only motion is the wrist rotation between the start and en positions.

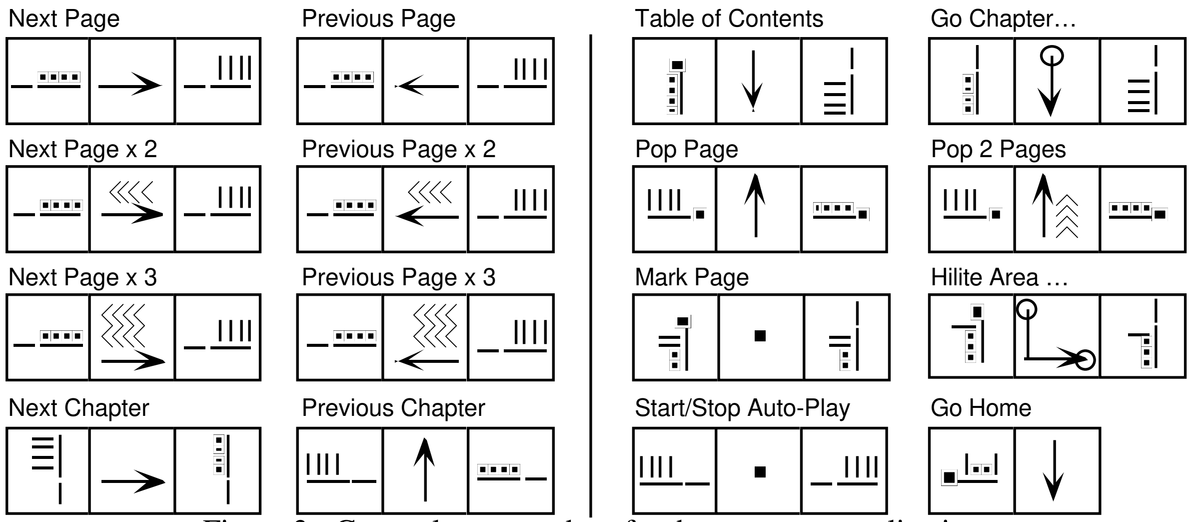

Figure 3 - Gestural command set for the prototype application.

Guidelines for Defining Gestural Command Sets

The description we have given so far delimits the set of commands that can be issued. However it does not help in defining a set of commands that provides natural interaction. The following guidelines were determined empirically after trying different command sets for our application. They express trade-offs and design aspects of the gestural command sets we designed. Some guidelines concern the form the gestural command sets should take; others identify general characteristics desirable for applications that use hand gesture input.

Form of Gestural Commands

• *Use Hand Tension* . Start positions should be *tense*, i.e. they should correspond to a non-usual (but not un-natural) position of the fingers and wrist. For instance they should require full extension of finger joints, clenching the fist or orienting the palm up. This tension of the muscles corresponds to the variations of intonation one may observe between ordinary conversation and imperative orders in oral expression. Tense positions make the user's intention of issuing a command more explicit or, as explained by Buxton [5], "it determines the ebb and flow of tension in a dialogue". This allows the user to move and perform gestures in the real world, since only specific and clearly intentional (to a more or less important extent) gestures will be interpreted by the system. The tension required for issuing a command can nevertheless be extremely short and therefore should not generate fatigue.
Conversely, end positions should correspond to a relaxed position of the hand. This already happens naturally when the user lowers his or her arm and leaves the active zone: the arm's muscles come to a rested position that corresponds to the completion of command.

• *Provide Fast, Incremental, Reversible Actions*: Similarities exist between the principles of direct manipulation [12] and the remote manipulation paradigm of our interaction model. Gestures must be fast to execute and must not require too much precision in order to avoid fatigue. In particular, an aspect of prime importance when designing a gestural command set is the resolution of each dimension as captured by the input device. If the position of the hand cannot be determined with less than 1 cm of precision, precise tasks cannot be performed. For instance, the application should not rely on drawing fine details or manipulating objects smaller than a few centimeters.

• *Provide Undo Facilities*: Despite our effort to enable efficient detection of intention, recognition of a gesture can be wrong and commands can be issued involuntarily. The command set must therefore provide an undo command or symmetric commands that let the user easily cancel any unintended action. Appropriate feedback (see below) also improves the user's confidence in the system.

• *Favor Ease of Learning*: The choice of appropriate gestural commands results from a compromise between the selection of natural gestures, which will be immediately assimilated by the user, and the power of expression, in which more complex gestural expression gives the user more efficient control over the application. Of course, the notion of "natural" gesture depends heavily on the tasks to be performed: are common gestural signs easily applicable to meaningful commands?

In order to improve the usability of the system, we assign the most natural gestures, those that involve the least effort and differ the least from the rest position, to the most common commands. The users are then able to start with a small set of commands, increasing their vocabulary and proficiency with application experience. Also, the command set should be consistent and avoid confusable commands. Since these guidelines also depend on the application, we suggest iteration and user testing during the design process of the command set.

Structure of Interaction

• *Ensure Appropriate Feedback:* Good user feedback is mandatory because gestural commands are not self-revealing. First, syntactic feedback, such as the shape of the cursor, should inform the user of the state of the recognition system. Second, semantic feedback should relate the command to the gesture used to issue it. For example, consider the command that goes to the next page. It is issued by moving the hand from left to right. Hence, semantic feedback that wipes the next page over the current one, from left to right, is appropriate. Finally, feedback about the history of actions, or of the last command issued, allows the user to gain confidence in the system and to repair easily unwanted commands.

• *Use Hand Gestures for Appropriate Tasks*: Navigational tasks can easily be associated to gestural commands. For instance, the hand should move upward for a "move up" command. Widely-used iconic gestures (e.g. stop, go back) should be associated with the corresponding command. Drawing or editing tasks also have several significant natural gestures associated to them (select, draw a circle, draw a rectangle, remove this, move this here, etc.).
Abstract tasks (e.g. change font, save) are much harder to "gesturize" and require non-symbolic gestures. Using deaf sign language vocabulary could be considered as an alternative and would have the advantage of benefiting an important community of people with disabilities. Another solution would be to use indirect selection gestures, in a way similar to menus in direct manipulation interfaces. However, the best solution probably is to use speech input to complement gestural commands. This would keep the directness and naturalness of the interaction scheme.

IMPLEMENTATION

The algorithm that parses samples received from the DataGlove is implemented as a driver on a Macintosh IIx. When a gesture is recognized, an event is sent by the driver to the active application (in this case HyperCard), containing information on the issued gesture (name, start and end position, etc.).

Each sample is compared to the set of possible hand positions, using a tree search (fig. 4). Hand positions are grouped, with separate branches for wrist orientation, thumb and each finger. Start and end positions are stored in separate trees, with a maximum of six lookups for any sample received.

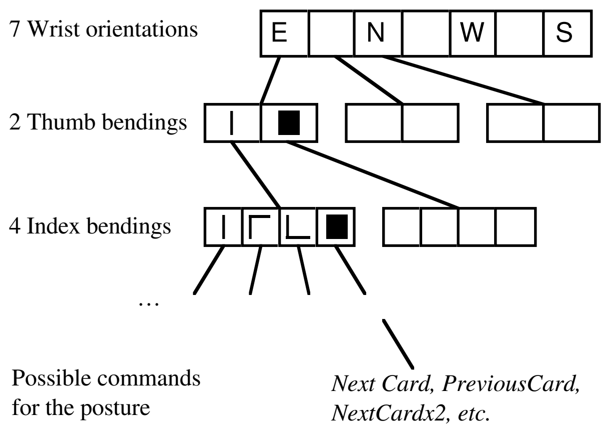

Figure 4 - Tree for recognizing start positions.

We use an extended version of the algorithm defined by Rubine [11] to analyze the dynamic phase of gestures. This algorithm was designed to extract features from 2D gestures, such as the total angle traversed, the total length of the path followed by the hand, etc. Mean values for each gestural command and each feature are determined by training the system when the application is designed. When a command is issued, the features characterizing the gesture are compared to the mean values for each possible command, determining which gestural command was meant by the user. In order to use this algorithm with full-hand gestures, we extended it by adding features for each finger bending, wrist orientation and distance from the active zone. An average of 10 training examples for each gestural command has proved sufficient to provide user-independent recognition.

The DataGlove is sampled at 60 Hz. Processing of each DataGlove sample is in constant time, and no significant overhead of the driver has been observed. The driver uses 22 Kbytes of code, and a typical command set uses 40 Kbytes of memory. We have developed a separate application to create and edit command sets interactively with the DataGlove. This application also generates the description of the command set according to our notation. Hence, it can be used by end users to create, customize and document command sets.

USABILITY TESTING

We conducted two usability tests of this application. First, we assessed the learning time of the command set and the recognition rate of the algorithm. Ten users were presented the application and the graphical notation for the gestures. After trying each gesture once, they performed a series of 50 gestures. Each gesture was prompted by the system, and the recognition rate was computed as the proportion of gestures recognized correctly. Two subjects had a recognition rate around 50%: their hands were too small for the DataGlove we used. The other subjects had recognition rates of 72% to 84% . A trained subject regularly obtains 90% to 98%.

We found two main types of errors: system errors and user errors. The system had dificulties identifying gestures that differ only in their dynamic phase, especially when finger bending is involved (such as "Pop Card" and "Pop Card x2"). This indicates that our adaptation of Rubine's algorithm should be tuned, although the lack of resolution of the DataGlove may also be responsible. User errors correspond to hesitations while issuing a command. This often occurs when the user is tense and has not had enough practice with the interaction model. This problem disappears with a little training, when gestures are issued more naturally.

The second usability test consisted of an "in vivo" use of the system. Two trained users made several presentations of the system to an audience, using the sample application. The purpose of this test was not to evaluate the recognition rate, but rather to determine whether the application was usable in a real setting. Most mistakes were noticed immediately and could thus be corrected in one or two gestures. In a few cases, the user did not immediately realize he had issued a command, or did not know which command had been issued, and it took somewhat longer to undo the effect of the command.

Overall, the error rate was surprisingly low, because the most usual commands are the most natural ones and are better recognized. As a result, the users found the interface easy to use and the small learning time was worth the improvements in the presentation.

LESSONS LEARNED

The recognition of gestures in our interaction model enables users to interact with the system naturally: commands are concise and can be input smoothly. We achieved this through careful analysis of the structure of gestural communication and an iterative, user-centered design of the interaction model.

A significant problem was due to the lack of precision of the hardware that we used. First, the samples from the DataGlove are not stable even when the device is immobile. Second, since we use the projection of the hand on the screen, any instability (whether due to the hardware or to the user's arm) is amplified. In practice, the best resolution is about 10 pixels, which makes precise designation tasks impossible. Although filtering would help, it would not solve the problem of arm movements. Hence, it does not seem that this problem is likely to be solved within the interaction model. Precise tasks generally require a physical contact with a fixed stand, whereas our model is a free-hand remote manipulation paradigm. These restrictions should be taken into account when designing an application, or when deciding whether to use the interaction model for a given application.

The main problem remains the use of a DataGlove: it links the user to the computer, it is uncomfortable, and it is unreliable. We did not address this problem since we know that it can be replaced by future devices, such as video cameras, when they become available.

When we started this work, we did not expect to be able to perform real-time recognition of gestures and run the application on the same machine. The interaction model enabled us to devise a very simple recognition technique without significant loss in power of expression. We even claim that such simplification enhances the model in that it makes it easier to learn and to use: using an active zone to address the computer and using tense positions to start gestural commands is similar to the use of gaze and pointing in human-to-human communication; quantizing dimensions makes the system more predictable.

OTHER APPLICATIONS

Our application was developed as a mockup to assess the usability of interfaces based on free-hand gesture input. The interaction model and the guidelines that we have defined apply as well to a number of other applications, outlined below:

• *Multi-User Interaction & Large Panel Displays*: Elrod et al. presented a system to interact with Large Control Panels [6]. Air traffic control, factories, stock exchange and security services all use control rooms in which the workers have to inspect large panels of controls and displays collectively. Our interaction model could improve the user interface of these rooms by allowing easy remote control of the displays by means of designation and gestural commands. Gestures are particularly useful here because designation works in a noisy environment.
• *Multi-Modal Interfaces*: Pure speech-based interfaces also face the "immersion syndrome": it is very difficult to distinguish vocal commands addressed to the system from utterances to the "real world". The segmentation of gestures provided by our model can be used to detect the intention of speech. Combining gestural commands with speech would improve both media: speech would complement gesture to express abstract notions, and gesture would complement speech to designate objects and input geometric information.
• *Home Control Units*: In the longer term, we foresee the remote control of home or office devices: a few cameras linked to a central controller would track the gestures and recognize the intent to use devices such as TV's, hi-fi's, answering machines, etc. This would avoid the proliferation of remote control units that are cumbersome to use and hard to find whenever they are needed.

CONCLUSION AND FUTURE WORK

People are highly skilled in using gestures to communicate; yet few applications let them use gestures to control objects in the real world. We have introduced a novel interaction style that allows users to control computerized objects using free-hand gestures. Unlike traditional "instrumental" interaction styles, e.g. those that use hand-held devices such as the mouse, this style lets users take advantage of their existing skills in the use of gestures, thus greatly reducing learning time. This technique can be implemented efficiently with off-the-shelf hardware and software.

We developed a sample application to demonstrate the effectiveness of this approach. This application lets users take full advantage of presentations created on a Macintosh computer. The speaker wears a DataGlove to control the application; he or she can use natural gestures to emphasize points in the talk and at the same time use gestures to control the presentation. The interaction model of the application is based on three key concepts:

- Creation of an active zone to distinguish gestures addressed to the system from other gestures.
- Recognition of dynamic gestures to ensure smooth command input.
- Use of hand tension at the start of gestural commands to structure the interaction.

We see two main directions for future work. First, we can improve the current implementation, by improving recognition and accuracy and by replacing the DataGlove with video cameras. Second, we can extend the range of applications that use this approach. This will provide greater insight into the design of gestural command sets and enable us to explore multi-modal interaction by integrating speech recognition.

ACKNOWLEDGMENTS

This work was conducted while the first author was at LIMSI. We thank J. Mariani, F. Néel, G. Sabah and D. Teil from LIMSI for making this research work possible. A. Braffort participated in the design and implementation of the prototype application.

REFERENCES

1. Adams, D. *The Hitch Hiker's Guide to the Galaxy.* Pan Books Ltd., London, 1979, Chapter 12.
2. Appino, P., Lewis, J., Koved, L., Ling, D., Rabenhorst, D. and Codella, C. *An Architecture for Virtual Worlds*, Presence, 1(1), 1991.
3. Bolt, R.*"Put-That-There": Voice and Gesture at the Graphics Interface*, Computer Graphics, 14(3), July 1980, pp 262-270, Proc. ACM SIGGRAPH, 1980.
4. Bolt, R. *The Human Interface*, Van Nostrand Reinhold, New York, 1984.
5. Buxton, W. There's More to Interaction than Meets the Eye: Some Issues in Manual Input. in Norman, D.A. and Draper, S.W. (Eds.), *User Centered System Design*, Lawrence Erlbaum Associates, Hillsdale, N.J., 1986, pp. 319-317.
6. Elrod, S., Bruce, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., Tang, J. and Welch, B. *Liveboard: A Large Interactive Display Supporting Group Meetings and Remote Collaboration,* CHI'92 Conference Proceedings, ACM Press, 1992, pp. 599-608.
7. Fukumoto, M., Mase, K. and Suenaga, Y. *"Finger-pointer": A Glove Free Interface,* CHI'92 Conference Proceedings, Poster and Short Talks booklet, page 62.
8. Krueger, M., *Artificial Reality (2nd ed.)*, Addison-Wesley, Reading, MA, 1990.
9. Morita, H., Hashimoto, S. and Ohteru, S. *A Computer Music System that Follows a Human Conductor*. IEEE Computer, July 1991, pp.44-53.
10. Murakami, K. and Taguchi, H. *Gesture Recognition Using Recurrent Neural Networks*, CHI'91 Conference Proceedings, ACM Press, 1991, pp. 237-242.
11. Rubine, D. *The Automatic Recognition of Gestures*, Ph.D. Thesis, Carnegie-Mellon University, 1991.
12. Shneidermann, B. *Direct Manipulation: A Step Beyond Programming Languages*, IEEE Computer, August 1983, pp. 57-69.
13. Sturman, D. *Whole-Hand Input*, Ph.D. thesis, Media Arts & Sciences, Massachusetts Institute of Technology, 1992.
14. Thorisson, K., Koons, D. and Bolt R. *Multi-Modal Natural Dialogue*, CHI'92 Conference Proceedings, ACM Press, 1992, pp. 653-654.
15. Weiser, M. *The Computer for the 21st Century*, Scientific American, September 1991.
16. Zimmerman, T. and Lanier, J. *A Hand Gesture Interface Device*. CHI'87 Conference Proceedings, ACM Press, 1987, pp. 235-240.