An Approach to Natural Gesture in Virtual Environments

ALAN WEXELBLAT
MIT Media Laboratory

This article presents research—an experiment and the resulting prototype—on a method for treating gestural input so that it can be used for multimodal applications, such as interacting with virtual environments. This method involves the capture and use of natural, empty-hand gestures that are made during conventional descriptive utterances. Users are allowed to gesture in a normal continuous manner, rather than being restricted to a small set of discrete gestural commands as in most other systems. The gestures are captured and analyzed into a higher-level description. This description can be used by an application-specific interpreter to understand the gestural input in its proper context. Having a gesture analyzer of this sort enables natural gesture input to any appropriate application.

Categories and Subject Descriptors: H.1.2 [**Information Systems**]: User/Machine Systems—*human information processing*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*artificial realities*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*evaluation/methodology*; *input devices and strategies*; *interaction styles*

General Terms: Design, Experimentation, Human Factors, Performance

Additional Key Words and Phrases: Gesture, input methods, multimodal, natural interaction

1. INTRODUCTION

Most existing virtual environment interfaces permit users to make a limited number of hand gestures for input, usually by providing users with hand-worn input devices such as cybergloves. However, the allowed gestures in these systems are quite limited. They form a command interface which severely restricts the input available to users. The command set commonly seen is small, often unnatural, and usually restrictive, leading one to wonder what benefit the user gains by using this mode and learning the new gesture command language. Surprisingly little work has been done on the use of

This research was supported by Thompson-CSF and by the Advanced Research Projects Agency (ARPA) under Rome Laboratory contract F30602-92-C-0141. The opinions and conclusions in this article are solely those of the author and should not be attributed to any agency or other person mentioned herein.

Author's address: MIT Media Laboratory, 20 Ames Street, Cambridge, MA 02139; email: wex@media.mit.edu; URL: http://wex.www.media.mit.edu/people/wex/

natural hand gestures at the interface. This form of gesture, sometimes called *gesticulation*, is the ordinary form that people use in conversations and especially when giving descriptions.

Natural gesture interfaces can be applied to virtual environments and to nonimmersive systems such as artificial realities [Krueger 1991; Maes 1993]. The Advanced Human Interface Group (AHIG) at the MIT Media Lab works on multimodal natural dialog, an aspect of which is a style of interface we call *directive* (see Thorisson et al. [1992] and Koons et al. [1993]). In a directive interface, the user makes a multimodal presentation describing the task to be done, and the computer executes the task. Directive-style interfaces are also commonly seen in the work on learning interfaces (for example, Bos [1992]), which observe the user's repetitive actions and offer to do the action sequence for the user. In addition, the ideas of computer agents which do the users' bidding also depend on humans being able to give direction in describing tasks for agents to perform.

This article presents experimental observations and an implementation of a continuous-gesture recognizer as part of larger-scale research into the analysis and use of natural gesture as an input modality for computer systems. AHIG's overriding goal in this research is to enable more-natural interaction, where "natural" means that the computer system adapts to the abilities and limitations of the human being rather than the other way around. Within that context, the goal was to understand and encapsulate gestural interaction in such a way that gesticulation can be treated as a data type—like graphics or speech—and incorporated into any computerized environment where it is appropriate.

1.1 Background

In general, people are quite fluent at making multimodal presentations involving speech, gaze, sketching, and gesture. We do this in everyday conversation, moving fluidly among the modes as needed. Primarily we communicate by speech, but gesture is an important supplement in making ourselves understood. The gestures used in these presentations are rapid, relatively continuous, and highly free-form. For the purposes of this research we distinguish empty-hand gestures from the strokes made with a pen or pointer, and we focus on empty-handed gesticulation.

Gesticulation is not a language per se. In this respect it differs from sign languages such as ASL (American Sign Language). ASL is generative—new signs can be created, used, and adopted into the vocabulary; there are dialects of ASL used in different regions of America. ASL and other sign languages are also combinatorial—two existing signs can be merged to form a new sign. Neither of these properties applies to gesticulation. Gesticulation is also highly idiosyncratic, unlike ASL which has a proper standard form for its gestures. We do not teach people to gesture the way we teach a sign language, though certain cultural conventions may be imposed; for example, strictures are given to children who make inappropriate gestures such as

pointing at strangers in public places. All these factors differentiate natural gesticulation from gesture languages.

In the experiments (described in Section 2.1) people used a wide variety of gestures. When two people talked about the same thing they did not necessarily use the same gestures. Gestures also varied over time when made by one person; for example, subjects would mention cars frequently, but mention of a car was not invariably accompanied by a gesture, let alone accompanied by the same gesture every time. This variability makes input of natural gestures a particular challenge.

Unfortunately, in current virtual environments (for example, see Codella et al. [1992]), this challenge is ignored. Users are prohibited from making their normal gestures; in order to interact with objects in the virtual world, users must make one of a select set of memorized gesture commands. These commands are usually specific hand motions or configurations. The system then matches these inputs to prestored or prelearned templates and activates commands in response to this input. This one-to-one mapping of input to command reduces gesture to only the expressive power of a function-key pad. Making a hand gesture to "fly forward" is equivalent to pressing F1 to "fly forward." There is little gain in functionality by using gestures this way, and there is an added cognitive load on the user who must memorize the gestural commands.

Gesticulation, though difficult, is a rich, expressive medium, used by people all over the world to convey ideas and information that cannot easily be carried in other modes. Indeed, many people find it hard or impossible to speak without using their hands. People who cannot speak a common language often manage to communicate via gesture. Being able to use this kind of gesture for input in virtual environments would be rewarding: users would not be required to learn so much, and their sense of immersion would be enhanced.

For example, imagine that a potential house buyer is touring a realistic virtual building. He approaches a door and wishes to open it. In a conventional implementation there might be a "door-opening gesture"—say, pushing the palm of the hand forward. But if the user forgets this gesture and instead reaches for what appears to be the doorknob and tries to "turn" it—say, by making a circle of thumb and forefinger, cupping the palm upward, and rotating—he would fail. Because he has not remembered the correct command, his experience of interacting with a virtual door is destroyed.[1]

By contrast, a virtual world which accepted general gesticulative input would analyze the grasping/turning gesture and apply that to the door model, thereby realizing that the user was attempting to turn the "doorknob" —which might be only a texture map—and react appropriately. How this can be done is explained in the body of this article.

[1]Brenda Laurel reports similar efforts to make gestural input more "natural" by taking advantage of people's preconceptions of how objects in the world behave ordinarily [Frenkel 1994]. Her solution is still limited, but it shows that work is beginning in this direction.

2. REQUIREMENTS FOR NATURAL-GESTURE INPUT

In order to enable computers to take gesticulative input, we must first understand human gesture better. Most of the existing research on gesticulation has been done by cognitive scientists or neurologists who are often most interested in gesticulative dysfunctions and understanding how gestures express people's thoughts (for example, McNeill [1993] or Rime and Schiaratura [1991]). In contrast, we are working to understand not why people perform (or fail to perform) certain gestures but rather how we could make gesticulation understandable to computers as an input mode.

The literature on gestures and our experimental evidence show that humans can make an enormous variety of gestures. Rather than building an interface which would handle all possible gestures, I wanted to understand the sorts of gestures people would make while giving descriptions. To that end, I conducted an experiment in which subjects were videotaped giving descriptions. Two research questions needed to be answered:

(1) What sorts of gestures do people make in giving descriptions?
(2) How do we characterize these gestures so that a computer could understand them?

2.1 The Experiment

The experiment involved setting up a situation in which subjects would be giving narrative descriptions of scenes they had viewed. I chose two popular movies and edited out ten scenes of 30–45 seconds duration from each movie. Subjects viewed one movie's scenes or the other, selected randomly.

Subjects were seated in a room with a large-screen television, videocamera, and tape player. The subject's chair was positioned away from tables and other objects in the room. This left his or her hands free for gesturing, if they so chose, although we avoided mentioning gestures in our instructions to subjects so that we would not bias their behavior.

Subjects viewed ten scenes from either *The Terminator* or *Casablanca*. Each scene was 20–45 seconds in length. After viewing all scenes, subjects were asked to describe a specific scene (referred to by number) in such a way that it could be distinguished from the other scenes. In order to facilitate recall and eliminate presentation-order effects, subjects rewatched the target scene immediately before giving their description, which was videotaped. This rewatch, describe, and tape procedure was done twice for each subject. Subjects were told to be as complete and accurate as possible in their descriptions. During the description, experimenters gave paralinguistic feedback (nods) but did not attempt to correct errors or elicit clarification from the subjects.

Subjects were paid US $5 for their participation, which took less than 30 minutes in all cases. Thirty-five subjects participated, 18 viewing *The Terminator* clips and 17 viewing *Casablanca* clips.

The scenes shown were selected to focus on four major characters from each film, and action sequences were duplicated wherever possible. For

example, in *The Terminator* clips, subjects viewed four "chase" scenes; in the *Casablanca* clips, subjects saw four "drinking" scenes. The overlap of characters and actions was deliberately maximized to encourage subjects to use other differentiating information. That is, by having Rick (the protagonist) appear in the majority of the *Casablanca* scenes, subjects could not simply say "Oh, this is the one scene with Rick in it." This worked surprisingly well. Thirty-one subjects (89%) gave what would be considered "complete" descriptions in that they narrated all the action of the clip. Four subjects found specific information which sufficed to distinguish their target clips and gave only that information. These subjects were among the eight (23%) who made no gestures during one or both of their descriptions.

2.1.1 *Experiment Output.* The videotaped descriptions were transcribed for text and then annotated with gestural information using a style similar to McNeill's. An example of a transcribed scene is shown in Figure 1. In this transcript, the subject's text is in plain font and the annotations in italics. Words which coincided with the stroke phase of a gesture are in brackets. The hash mark (#) is used to identify a place where a gesture stroke occurred between words. Ellipses (. . .) are used to mark places of verbal hesitation and paralinguistic utterances. Numbers at the beginning of lines are for reference purposes.

Gestures are first identified as to category, using McNeill's [1993] distinction between character-viewpoint (C-VPT) and object-viewpoint (O-VPT) iconic gestures. A character-viewpoint iconic is one which is made as though the speaker were the character performing the action. An object-viewpoint iconic is one which is made as though the speaker's hands/body was the object being talked about. Then the gesture is described, using salient featural components. In cases of iconics, the intended object of representation is identified; in the case of deictics, the thing pointed to is identified.

It is worth noting that the classification scheme used by McNeill [1993] is problematic. In particular, it uses both morphological (what the hand is actually doing) and functional (what purpose the gesture serves) characteristics for classification. However, since my interest was in identifying approximately how many and what sorts of gestures people would make, this well-established classification scheme was used, rather than invent a new special-purpose set of annotations.

Table I shows the summary statistics from this experiment. Initially, all gestures were counted, giving the figures in the first column. The beat (nonsemantic, rhythmic strokes used to keep timing) gestures were subtracted, and the figures for all other gestures are shown in the second column. In both analysis cases, there is a large range of variability. Informally, it seemed that subjects could be divided into "high gesture" and "low gesture" types. The former (28% of subjects) made 20 or more nonbeat gestures in both scene descriptions. The latter (72% of subjects) made fewer than 20 nonbeat gestures in one or both of their scene descriptions.

2.1.2 *Analysis and Insights.* Several hypotheses were proposed in an attempt to generalize about the gestural information. Table II shows the

Subject 4:
1) It starts with [one of the characters]
Deictic: hand, palm up, finger points to Rees location in gesture space.

2) [throwing off his jacket] and pulling out a shotgun.
C-VPT iconic: hand, fingers curled, moves from center of body to over right shoulder (representation of pulling open trenchcoat).

3) And [they're...] it [seems like] a nightclub because there are many women
O-VPT iconic: hand, palm up, waves back and forth (representation of bodies dancing). Gesture repeated.

4) dancing behind him. It then cuts to the [Terminator guy]
Deictic: hand, palm up, fingers point to Terminator location in gesture space.

5) [cocking] his gun and starting to bring it up.
C-VPT iconic: hand moves back and up (representation of moving slide back to cock hammer).

6) And then [...the other character] -- I don't remember his name --
Deictic: hand up, palm up, finger points to Rees location in gesture space.

7) [starts shoving people] out of his way.
C-VPT iconic: hand, palm up, waves back and forth (representation of shoving people).

8) It then cuts [to] the Terminator guy
Deictic: points to Terminator location in gesture space.

9) -- [he had] a laser sight on his gun...
C-VPT iconic: looks down at palm of hand as if at object in hand (representation of pistol in hand).

10) It [cuts to] [#] what's her name sort of staring at him
O-VPT iconic, then beats: two hands held up, palms facing (representation of Sarah's head from close-up shot); beats during attempt to remember name.

11) with the [little red dot] on her head from the sight. It cuts again, I think, to
C-VPT iconic: index finger touches center of forehead (representation of laser light circle).

12) the Terminator and he's about to fire, then [Rees] fires
Deictic: points to Rees location.

13) and hits [him] with the shotgun. The [Terminator] guy stumbles back and Rees
Deictic: two pointings to Terminator location.

14) fires [three or four more times] with
O-VPT iconic: hand, palm up, sweeps away from body (representation of Terminator falling (see below)).

15) the Terminator guy [falling back].
O-VPT iconic: same gesture.

16) Then it shows people [stampeding] out of the nightclub.
O-VPT iconic: hand curled into cylinder sweeps across (left to right).

17) They show [one with] [the woman] on the ground covering her head with
???, deictic: two hands, one curled resting in palm of other held out; then points to lower center (where Sarah was lying on ground).

Fig. 1. Sample transcription.

Table I. Summary Statistics

35 Subjects	All Gestures	Nonbeat Gestures
min	0.0	0.0
max	58.0	49.0
mean	15.9	11.2
median	10.5	6.5
stdev	16.1	12.2

Table II. Correlation Statistics

Factor	Correlation with Total	Correlation with Nonbeat
Type	0.011	0.103
Language	0.220	0.385
Gender	0.578	0.447

correlations derived between the total number of gestures and nonbeat gestures and the three possible explanatory factors:

Type of Scene: *Action Versus Nonaction.* Action scenes were defined as chase scenes or fight scenes or scenes where characters entered and left the frame more than twice. The presumption here is that increased movement of objects and actors in the scene would encourage subjects to make more gestures showing these motions. If this supposition is correct, the correlation with nonbeat gestures should be significantly stronger.

Native Language: *The Subject's Native Language Might Have an Effect.* It is commonly stated in gesture literature that people speaking in their nonnative language produce more gestures, presumably because they experience difficulty formulating precise sentences in an unfamiliar language, so more of their communication goes through the gesture channel. If this supposition is correct, nonnative English speakers should make significantly more gestures than native speakers. Subjects' non-English native languages were Korean (2 people), Mandarin Chinese (1 person), Hindi (3 people), and Italian (1 person).

Subject's Gender: *Popular Culture Images of Females as Being More Shy and Less Willing to Express Themselves Openly than Males.* If this hypothesis is correct, female subjects should make significantly fewer gestures than males.

As Table II shows, none of these correlations holds up. The number of gestures is almost completely uncorrelated with the type of scene. Gender and native language were also not reliable predictors of gesture frequency.

Although the preexperiment hypotheses were not borne out, performing the transcription and analysis led to several important post facto insights.

It quickly became clear that in general we could not predict *what* users would gesture about. An initial assumption on our part had been that subjects' gestures would concentrate on action descriptions, showing the

placements and movements of objects in the scenes. This turned out to be completely false. Notably, one subject used only a single gesture in describing an entire chase scene, and that gesture clarified the placement of lights on top of one of the vehicles involved.

We noted a number of gestures repeated approximately the same way by several subjects. In addition, we noted that even with the sound turned off for the videotape playback, we could usually tell when the subjects made a significant gesture. This suggested that there were things in common between subjects that were not being seen at a full-gesture analysis level. Generally, gesture command languages operate only at a whole-gesture level, usually by matching the user's gesture to a prestored template. Seeing these commonalities in our subjects' gestures indicated that attempting to do gesture recognition solely by template matching would lead quickly to a proliferation of templates and would miss essential commonalities.

At the same time, we found a number of problems with gestural classification systems such as those used by McNeill [1993]. In particular, to write down classifications of the gestures required using knowledge about the scene being described and about subjects' intentions. This information would not be known by a computer system attempting to understand the same gesture. In transcribing the gestures, I was forced to make descriptions based on key elements or *features* of the gestures. These features, such as hand configuration and relative motion of the hands, formed a descriptive language that worked well across subjects and across gestures within subjects.

An important insight from this was that while the human body was capable of making an enormous range of gestures, in fact people tended to make only a relatively narrow range of gestures in normal interaction. Equally important, these gestures can be decomposed into feature-based descriptions. These descriptions serve as an *intermediate language* for gestural input, one which provides a higher-level description than usually seen from raw input devices but which still lies below the level of assigning meaning.

By analogy, think of what a speech recognizer does: it takes sound waves from the air and produces sentences according to a grammar. Given rules of English grammar, a recognizer might capture a sentence like "Green ideas slept furiously"—which is syntactically correct. It is up to a person or application hearing the input to determine if a recognized utterance is meaningful.

This fit in with the general AHIG approach of combining bottom-up and top-down methods; we decomposed gesture recognition into two discrete but related stages: *analysis*, where features are detected, and *interpretation*, where meaning is assigned to the input. The analyzer would be equivalent to the speech recognizer and as such could be attached to any application or virtual environment which wanted to capture gesture input. The interpreter would be specified differently for different applications because of the size of the problem—just as the range of understood grammars differs for different natural language applications today—but could also be more general. The

implementation of a full multimodal interpreter is part of the doctoral degree research undertaken by my colleague David Koons.

In addition, by separating analysis and interpretation, we enable applications to take gestural input in context. Gestures can "mean" different things in different situations or in conjunction with different speech inputs. For example, a gesture made by moving the hand forward with the index finger extended might be a pointing gesture made to select an object from a group. In another context, it might be a command to move a robot forward. In both cases, the gesture analyzer produces the same intermediate representation; the interpreter combines this with knowledge of the user's environment and the capabilities of the virtual objects to produce an action.

The rest of this article gives an overview of our implementation of such a gesture analyzer.

3. PROTOTYPE IMPLEMENTATION

We implemented a feature-based gesture analyzer along the theoretical lines described above. This analyzer sits in a data path from user to system, as shown in Figure 2. It receives input from a body model and communicates with an interpreter.

The body model coordinates input from a number of sensor devices: Virtual Technologies' Cybergloves, Ascension Flock of Birds' position sensors, and an ISCAN eye-tracking camera. The body model is responsible for sampling the sensors at appropriate data rates and converting sensor data (absolute positions) into body-relative information (joint angles) before passing the information on to the analyzer. This allows the analyzer to be device independent; new generations of sensors can be put in by modifying only the lowest level of the body model. In addition, the body model filters the data. It samples the gloves and cubes at 100Hz and uses a Gaussian filter to produce 20Hz samples with data spikes removed. The ISCAN is also sampled at 60Hz, and a filter is applied to remove blinks and saccades (the involuntary motions made by our eyes).

In order to make the computer clothing as unobtrusive as possible, the wiring for the gloves and position sensors is sewn into the lining of a lightweight windbreaker, as shown in Figure 3. This jacket is very comfortable and can be worn for reasonable periods of time without discomfort. The gloves have fingertip openings, which make typing and fine-grained object manipulation possible.

The clothing holds the four Bird sensors in place on the backs of the user's forearms, at the top of the spine, and on the head (on the ISCAN itself). Given these four known points in space, we can create a model of the user's body—at least a model of the upper body, including the head, torso, and arms. This model uses a common constraint satisfaction technique, called *inverse kinematics*, to estimate the positions and angles of the wearer's shoulders, elbows, wrists, and neck joints (see Badler and Webber [1993] for a complete description and application of inverse kinematics).

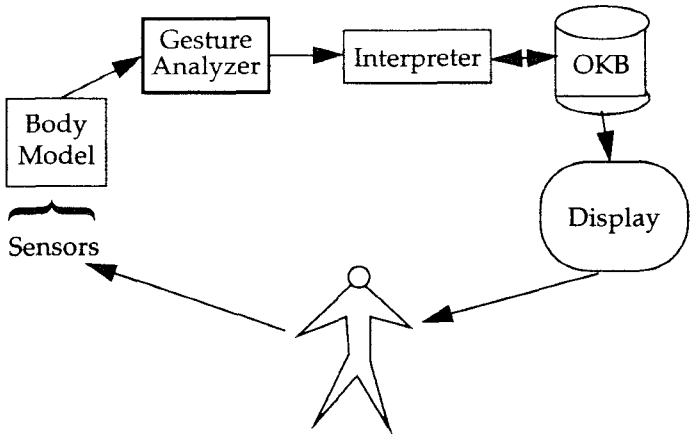

Fig. 2. Data flows in the AHIG system.

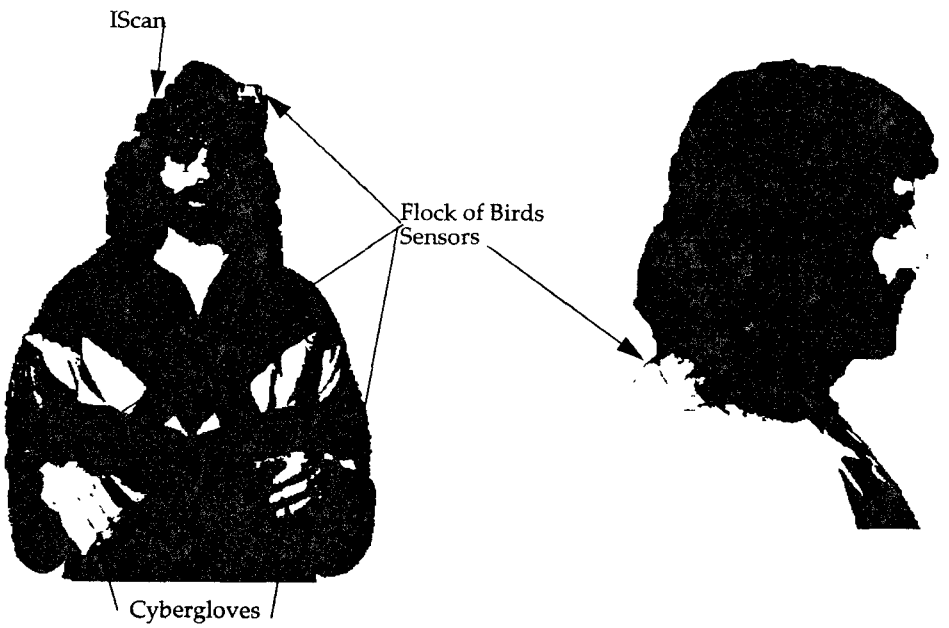

Fig. 3. The author wearing the AHIG "Data Suit."

In order to make the data suit and the body model person independent, the system startup procedure requires the wearer to assume a series of poses. These poses are used to measure the normal and extended positions of the joints and to get an estimate of the person's armspan. Once the wearer has completed this configuration (which takes less than two minutes), the body model is calibrated and can perform continuously for that person. Of course, if a different person puts the jacket on, the configuration must be repeated.

In our demonstration system, the body model knows the fixed point at the top of the spine and the moving point on the back of the forearm. The position of the shoulder is computed during the configuration that the user does at system startup and is then treated as a fixed point. The body model computes the position of the elbow and the shoulder and elbow angles.

Shoulder angles are:

—*elevation*: the degree to which the shoulder is raised above straight down;
—*abduction*: the angle of the raised shoulder away from the center of the body; and
—*twist*: the turn of the shoulder inward/outward from the belly.[2]

Elbow angles are:

—*closure*: how much the elbow moves the wrist toward the shoulder; and
—*roll*: the rotation of the elbow which produces a roll of the hand.[3]

The angles produced by the body model for the elbow and shoulder joints are used by the analyzer to decompose complex movements of the hand into recognizable segments for analysis.

The body model is important to the operation of our gesture analyzer. It provides a device-independent, rate-controllable interface to data about the user's body. In addition, the use of joint angles enables the body model and the gesture analyzer to run on separate machines over the network. They communicate by TCP/IP sockets at a data rate of approximately 4Mb/second, well within the capability of a standard Ethernet.

The Interpreter module accepts inputs from the gesture analyzer and a speech recognizer (BBN's HARK continuous-speech recognition product). It assigns semantic value to the collected input information to produce transformations in the system. In our demo, the application is a simple room layout system which allows users to create and move furniture and similar objects in a simulated living room. The interpreter uses an object-part knowledge base (OKB in the diagram) to model the virtual room being constructed and to focus the collected user inputs into transformations of the system objects. The results are displayed to the user via a large-screen display. Because the display is done via a large screen rather than with an immersive environment such as a head-mounted display, the visual feedback is the same as on a traditional computer screen.

We chose this mode of feedback because our research sponsors are most interested in applications which can be simultaneously viewed by many users. A projective display gives true stereoscopic views for only one person at a time. In addition, our research is focused on issues of interaction rather

[2] Imagine stabbing yourself in the belly with a knife held in your closed fist.
[3] This roll actually occurs by the two long bones of the forearm (radius and ulna) crossing over each other, but it is treated as a property of the elbow joint because (1) the wrist joint does not move in this angular rotation and (2) all changes are assigned to the nearest moving joint.

than of immersion. However, the gesture analyzer could easily be applied to an immersive virtual environment as well.

The user gives commands such as "Put a chair here" and makes a gesture indicating the location of the chair to be created. Commands are spoken into a headset microphone worn by the user and are interpreted by the HARK continuous-speech recognizer. The system draws a simple graphic representation of a chair in a position in the room relative to the user. That is, if the user gestures toward his or her right, the chair is created on the right side of the screen. The user can then give commands such as "Move the chair like this" and demonstrate an arbitrary path with his or her hand. The gesture analyzer captures the path information (as explained below) and passes this to the multimodal interpreter. The interpreter uses this information along with the data from the speech recognizer to create a transformation function which moves the chair in the virtual environment.

3.1 Analyzer Architecture

The analyzer's internal architecture is shown abstractly in Figure 4. The data flow from the top of the picture toward the bottom, as represented by the lines. The gesture analyzer prototype was implemented in under 10,000 lines of C++ and runs on a DECstation 5000/100 under UNIX. Each layer is implemented as a separate class to allow easy overriding and extension of the model.

3.1.1 *Segmenters.* At the top level of the analyzer architecture are segmenters, which are responsible for receiving the synchronous 20Hz data from the body model and dividing it into periods of linear change and steady state as shown below.

In this example the datum has an arbitrary value at the time the system begins. The precise number associated with this value is not important; rather, the change from one observation to the next is important. In this case the value is increasing. The segmenter notes the new value and continues to observe, as long as the number increases monotonically. After some time, the value stops changing and stays steady. This change of state is noted, and a data segment is produced indicating that the value increased from its initial value at t0 to its final value at t1. Similarly, at t2 the value begins to change again, and this is noted by the production of another segment. The change at t3 from decreasing to increasing and the one at t4 from increasing to decreasing are also noted as segment boundaries.

Each data value—position dimension or joint angle—produced by the body model is watched by one segmenter. Segmenters embody knowledge about what kind of data they are segmenting and use this to detect only significant changes. For example, although there are segmenters for the joints of the first finger as well as for the joints of the pinkie, the pinkie joint segmenters are set to look for a larger change before they report a significant event. That, of course, is because our index fingers are generally more expressive than our pinkies. We count movements of the index finger as more important and pay

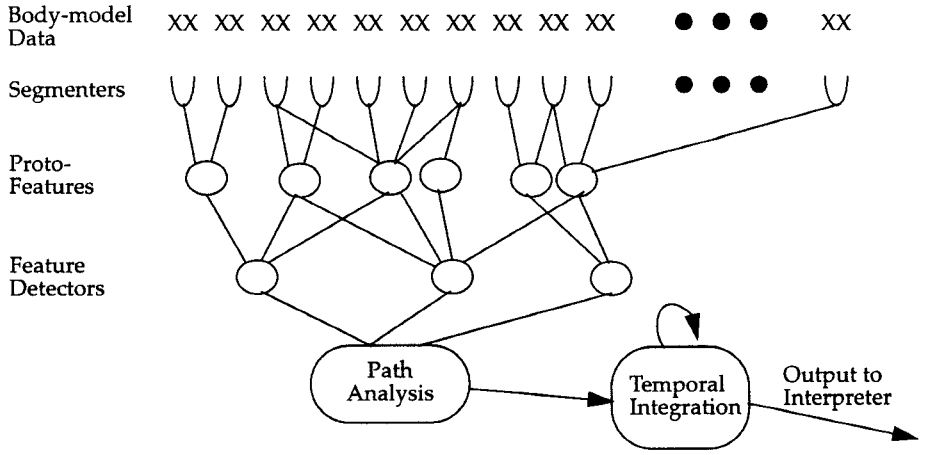

Fig. 4. Internal architecture of the gesture analyzer.

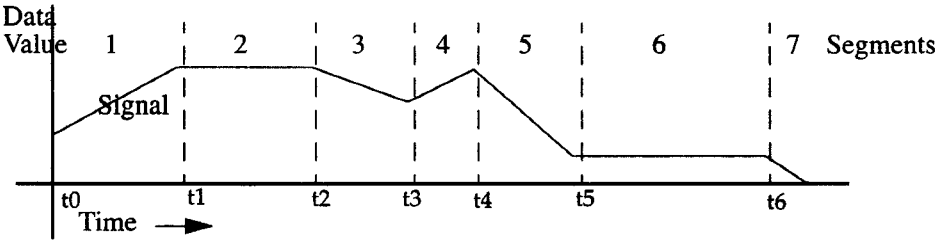

Fig. 5. Conceptual view of data segmentation.

more attention to it as humans observing gestures; therefore, the gesture analyzer accepts smaller changes in this finger as significant.

3.1.2 *Feature Detectors.* Beyond the segmenters are protofeature detectors. These detectors work by monitoring one or more segmenters and receiving asychronous reports of segments appearing in the data. The protofeature detectors look for simple things like extension of a finger. In turn, they feed their data to higher-level feature detectors, which combine these reports into more-intelligent features. For example, extension of all four fingers and lack of curvature in the palm can be seen to be a flat-hand configuration. The features implemented in the prototype system are summarized in Table III.

Table III indicates that the prototype analyzer recognizes a feature called "curvature" (measuring the degree of curl in the palm) and that there are two such detectors in the system, one for each hand. In total, the prototype contains 30 features and protofeatures. Spread detects the degree to which the fingers are apart from one another; this protofeature takes input from three segmenters, one for each of the interfinger joints. Thumb position

Table III. Features and Protofeatures Implemented

Type	Number
Extended	10–1 per finger
Contracted	10–1 per finger
Curvature	2–1 per palm
Spread	2–1 per hand
ThumbPos	2–1 per thumb
Move	2–1 per hand
Orientation	2–1 per hand

tracks the thumb's location relative to the palm of the hand and its closeness to the index finger.

The feature detectors also coordinate knowledge about disparate parts of the body. For example, information about the left and right hands is compared to see if the user is making a coordinated presentation—for example, using two hands to show the movement of a table—or separate presentations —for example, showing the placement of a chair next to the table.

The analyzer is designed to be both flexible and extensible. New features can be added as needed to extend the system into areas such as head and torso gestures and to integrate eye-tracking information—detecting eye-specific features such as fixations and smooth pursuits.

Each feature is initialized with a set of *needs*—those segmenter data that must be present in order for the feature to "fire." When features are initially coded, a certain number of data segments can be specified to be present, and a stack of specific types of data can also be specified. This information is initialized at startup time and remains fixed throughout analyzer operation. Needs are always a strict superset of the segments in which the feature expresses interest. For example, in the implemented system, the hand curvature feature receives input from nine segmenters (two for each of the four fingers and one for the angle of palm closure). However, not all of these are needed for the curvature feature to be read out. The implemented system will report curvature if the palm closure datum is present and if any four of the eight finger joints are known. This was an implementation choice for this prototype and could be changed for another gesture analyzer by rewriting a few lines of code in the startup module.

3.1.3 *Path Analysis and Temporal Integration.* It is useful to think of the analysis module as a set of special-purpose recognizers, each of which is looking for changes in the data stream that it considers significant. For example, the direction of movement of the hand may change from downward to sideways, or the hands may start a roll. When such a significant change is seen (based on the changing signals from the segmenters) the recognizer which detected the change calls for a new data structure, called a *frame*, to be generated. All the recognizers contribute their accumulated data into the new frame which is then output. All gestures result in sequences of one or more frames, with the number of frames, and their specific contents, determined as a result of how the user made the gesture.

The results of all the feature detectors are therefore effectively "summed" by the path analysis module, which produces a description of the gesture, based on the current best-available description. This description came to be called a frame because it is like a keyframe in an animation of the hands. Frames combine static and dynamic information about the gesture. Frames are computed dynamically, based on the presence of interesting features. That is, if the hands are not doing anything, no frames are created; when frames are created, they last as long as the relevant change they report lasts. This represents a significant simplification of gestural information. In effect, the path analyzer attempts to discard all possible irrelevant information.

However, because the software operates continuously as the user gestures —rather than waiting for a complete gesture and then beginning analysis—it is possible for the path analyzer to become mired in the details of the moment and miss the higher picture. For this reason, there is also a history module which keeps a record of the frames which have already been computed as part of the current motion. This history module performs a process called temporal integration, which is shown conceptually in Figure 6. In this example, the user makes a four-part gesture, as shown by the three divisions in the "gesture" data line. The analog signal of the moving hands and arms are sampled (data), which introduces a large number of divisions into the stream. The samples are passed to the segmenters which remove most of the divisions, leaving only the boundaries which correspond to directional changes (as discussed in Section 3.1.1). Feature detection removes still more and the combination of features into frames still more.

It is important to note that there is no notion in the current prototype of "waiting" for the gesture to be "done" before analysis is attempted. Instead, the system operates continuously, producing frames as soon as sufficient featural information is available.

The most-important function of temporal integration is to "paste together" segments of motion. Because of the way the human body is constructed, different joints operate in sequence as well as in parallel to produce smooth motions. Since the data representation is read initially from the joint angles, this results in artificial divisions being put into what people perceive as continuous motions. These divisions are picked up by the feature detectors.

By ignoring the specific joints and concentrating solely on temporal information, this final layer of the analyzer is able to remove these artifacts and present a coherent picture of motions made by the user's hands. For example, assume that the user is making a semicircular arc. Figures 7–9 show how this process operates.

In the initial movement (first picture) several joint angle changes will be made in order to accomplish the circular motion. The path analyzer recognizes these as arc fragments, each with specific start and end locations, where each arc fragment corresponds to the movement made in a particular joint angle. In the final picture, the temporal integration module notes that the start of arc segment 2 corresponds precisely to the end of arc segment 1 and merges the two into a single arc which starts at the beginning and ends at the currently known best ending point. As the following frames (3 and 4)

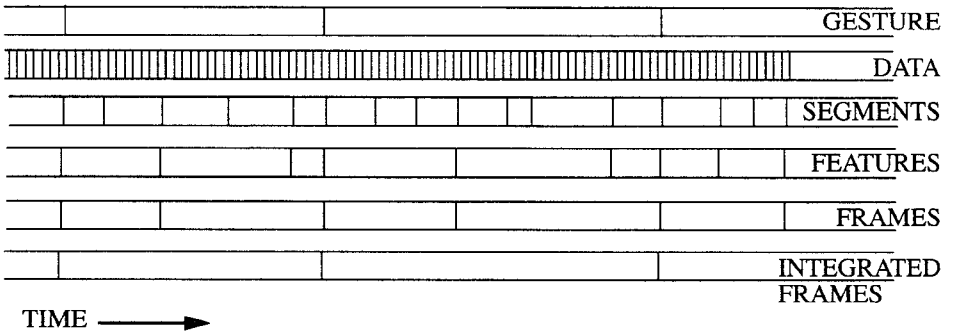

Fig. 6. The process of temporal integration.

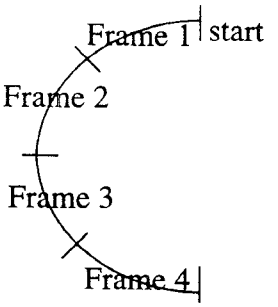

Fig. 7. User's motion with joint-induced divisions.

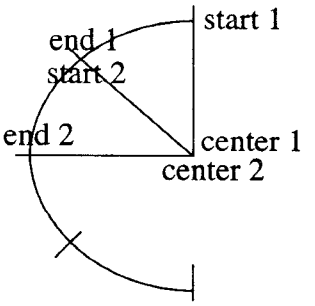

Fig. 8. Frame information before temporal integration.

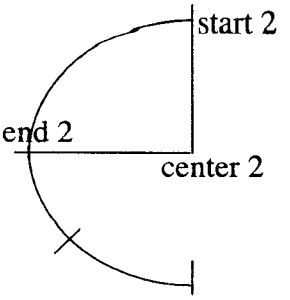

Fig. 9. Frame information after temporal integration.

are produced (Figure 7), they too will be integrated until eventually the output frame will contain the complete arc motion.

The frames output by the analyzer contain symbolic and numeric information (in roughly the same representation as a Lisp alist). For example, a frame might contain the information that at time t1 the user's right hand curled into a fist shape with the index finger extended; at time t2 the hand in that shape moved forward from coordinates (X2, Y2, Z2) stopping at time t3 at coordinates (X3, Y3, Z3).

The analyzer attempts to err on the side of producing too much information, rather than too little. For example, given the forward-move frames described above, it is possible that the user is simply commanding a robot to move forward. In this case, the precise geometry of the motion is unimportant; all that matters is that a "go-forward" command be relayed to the robot. However, in another instance, the user could be making the same gesture to select a particular object. In this case, the forward motion is relatively unimportant; what matters is the geometric information, which can be used by the interpreter to pick out the object desired by the user.

This over-information is a necessary consequence of the separation between analysis and interpretation. Since the analyzer cannot know what the user "means" by any given gesture, all the potentially relevant features of the gesture must be captured and passed on.

Temporal integration is the final step in the analysis process. In an ideal implementation, we would end up with precisely the correct number of frames to represent the movement made by the user—four, in this example. In practice, of course, we can only approach this limit. In operation, the prototype produces 10–25% more frames than desirable, due to the imprecision of the data-gathering devices and in the gesture made by the user. That is, when we think we have made a perfectly straight line gesture, in fact we often make a line with a slight curve. The analyzer sees this curve and reports it as such.

4. RELATED WORK

Other approaches to open/empty-handed gestures of the sort we are interested in can be divided into two different broad categories. In one the hands are used as a form of three-dimensional mouse in that they provide a pointer/selector with depth and mobility in three-space. In the other approach, the hands are used to create a command language; the person makes gestures which then are matched to command verbs. This approach is often characterized by users making a gesture to fly around a virtual world, a technique which Myron Krueger disparages as *finger-flying* in his public talks. In both cases, virtual environments work has been the most-prolific area for gestural interfaces to computer systems.

4.1 Other AHIG Work

It is important to remember that the current system was implemented as part of an ongoing research program, directed by Richard Bolt, into gesture

and especially its uses as part of a multimodal system for natural interaction [Bolt 1984]. As mentioned above, key components of the system—the Body Model and the Interpreter—are the subject of intensive research by other members of the group.

In addition, several students within the AHIG group have done significant work on gesture analysis, using different approaches. For example, Herranz [Bolt and Herranz 1992] implemented a two-handed gesture-based system which recognizes basic object manipulations (rotation, translation, scaling) by detecting coordinated changes in spatial location and deducing primary vectors from the input of data gloves and Polhemus cubes.

Sparrell [1993] produced a gesture analyzer with goals similar to the system described here. His work, though, concentrated on detection of specific configurations ("primes") and required the user to stop and start gestures explicitly in order for the recognizer to work. Elimination of these additional pauses was one of the primary goals of the current work.

4.2 Hand as 3D Mouse

In cases where the hand is used as a mouse, the data glove serves as an input source for a real-time, animated computer graphics model of the hand. The virtual hand moves around in response to the user moving his or her hand and may intersect with objects or may project a selection ray from an extended finger. For example, the selection ray approach was used by Lewis et al. [1991] and their successors in the game "Rubber Rocks."

In the game, users watch a computer monitor which has a representation of their (gloved) hand and that of an opponent. The hands move around a graphically depicted cube in pursuit of a set of virtual objects—simple polygonal shapes with Newtonian physical properties. The objective is to "grab" an object by pointing a finger at it. The computer projects a ray from the extended finger; if the ray intersects the object it is captured. It can then be "thrown" at the opponent's hand representation to explode. You win by destroying your opponent more times than he or she destroys you.

A similar approach using the hands as 3D mice was used by Weimer and Ganapathy [1989; 1992]. They constructed a system using data gloves which allowed users to generate three-dimensional curves for CIM applications. The user's extended index finger position was processed by corner detection and decimation algorithms to produce a series of piecewise cubic Bezier curves. In their earliest application, they treated the hand representation as a direct mouse equivalent—the user selected from control panels and from menus of B-splines by moving the hand analog until it intersected the desired choice. Later applications added more-conventional keyboard/mouse interactions for function selection.

Generally speaking, using the hands as 3D mice is an extension of the direct-manipulation approach to interfaces; although my demonstration system does not use direct manipulation of this sort, the gesture analysis system is compatible with using the hands in this fashion. That is, one could imagine

writing an interpreter for a modeling application which took the analyzed gestural input and converted it into manipulations of CAD/CAM objects.

4.3 Gesture Command Languages

In the second approach to gestural interfaces, a hand configuration or specific motion is recognized and used as a direct command input.[4] Gestures form a command language used to give directions to applications. This approach fits especially well when using gestures from a sign language such as ASL, as was done by Murakami and Taguchi [1991] in their neural-network recognizer for Japanese (JSL) word gestures.

Their recognizer uses a set of neural networks to recognize 42 finger alphabet gestures, each of which corresponds to a specific hand posture. They were able, with extensive training, to get the recognizer up to an accuracy of 92.9% for these static configurations. The user in their system is required to give a signal before and after the gesture so that the recognizer knows when to operate. They note this as a problem but do not provide a solution. My implementation avoids this problem by operating continuously and performing path analysis and temporal integration as described above.

Murakami and Taguchi then expanded their system to recognize ten JSL word gestures, all of which involved free hand movement. This worked much more poorly than the static recognition. Their neural nets were able to differentiate between two JSL gestures but not to identify an arbitrary gesture reliably from a learned set. Murakami and Taguchi identify three general problems to be solved by gesture recognizers that are going to work with dynamic gesture information:

—How to process time-series data;
—How to encode input data to improve the recognition rate; and
—How to detect delimiters between sign language words.

These issues appear in some form in all gesture recognition systems. Even when recognizing natural gesticulation rather than sign language gestures, we need solutions to the time problem and to the delimiter problem. I do not propose a perfect solution for these problems; rather, I believe that my approach gives a reasonable solution for operation in real time.

An interesting approach was taken by Darrell and Pentland [1993]. In using a vision-based system to capture hand motions, they constructed a system which learned its gestures interactively. No templates were stored or used; rather the system observed and characterized changes in the hand image. These sets of changes were matched to the learned templates by time-warping the input. This allowed a significant amount of variability in how input was matched to learned patterns. Unfortunately, their approach was limited to a small number of stereotypical gestures (waving side to side with fingers extended and palm forward as is often done to indicate "hello" or

[4] In the "Rubber Rocks" game mentioned above, "capture" and "release" gestures also are recognized. They are made by extending and then retracting the index finger.

"good-bye"). They did not attempt to apply their approach to more-complex and variable gestures such as are used in narrative description.

A typical example of the VR approach to gestures is provided by Väänänen and Böhm [1993]. Their system, called Gesture-Driven Interactions in Virtual Environments (GIVEN), is a generalized virtual environment which serves as a testbed for their research group's work. Interaction with GIVEN is by means of a data glove. Their recognition software, which is—like Murakami and Taguchi's—based on a neural network, captures any of roughly 10–20 hand configurations and matches them to system commands:

> The program "understands" different fist and finger positions (postures) received from the DataGlove as commands such as "fly forward," "fly faster," "reverse," "grab," "release," and "go to the starting point" [Väänänen and Böhm 1993, p. 97].

New gesture commands can be generated interactively by the user; the neural net is put into a learning mode, and the user makes the desired hand configuration which is then matched to a system command.

GIVEN allows what Väänänen and Böhm call "static" and "dynamic" gestures. A static gesture is just a hand "posture" (or configuration in my terminology). A dynamic gesture is

> ...the movement of the fingers in addition to the position of the hand in a sequence of time steps [Väänänen and Böhm 1993, p. 101].

Väänänen and Böhm use a fixed-time window of five steps to determine if a dynamic gesture has occurred. The user must match finger and hand movements in each of the five time steps closely enough for the neural net to recognize the gesture. However, because their system memory is only five steps deep, the user must be careful not to make any movements after completing the dynamic gesture before the gesture is recognized.

An advance along the lines of this basic approach was proposed by Quek [1994]. He suggests using optical flow tracking in a manner similar to Darrell and Pentland to segment a gesture stream and apply recognition techniques to it. His approach is similar to Väänänen and Böhm in that he uses the detected segments to match up to specific templates of features in a 1-1 manner. His approach handles a slightly larger set of gestures than theirs (30–50), but again it does not appear to generalize to full natural gesticulation.

5. CONCLUSION

Our analyzer was first implemented in January, 1994. It underwent several iterations over the succeeding months as we came to understand better what information should be captured in gestural analysis. The current implementation functions in our demonstration system are available to be shown to lab visitors as needed. No numerical analysis has been done of the accuracy of the system in operation. Quantitative results are difficult to obtain because the system does not produce a definitive result such as a recognized or not-recognized gesture. Rather, the computational representation of the ges-

ture is produced in all cases, and the interpreter then acts on the resulting frames. It is our sense from use that the system "gets it right" about three-fourths of the time for an unrestricted gesture set.

Readers may have noted that Figure 4 bears some resemblance to diagrams of the human visual system. This is not accidental. Our low-level visual processing operates in much the same way as the gesture analyzer: it receives a constant stream of input from the world and processes that input into key features (edges, texture gradients, motions). These features are then combined to give us high-level internal representations of complex visual scenes. Our vision system is "programmed" by evolution; the features—such as edges, texture gradients, and motions—it can detect and assimilate are predetermined, and the representation of those features in our brains is already set. By contrast, our efforts to find the key features of gestures and come up with appropriate computer representations are at best a first pass. We have some experimental and operational evidence to show that we are on the right track but as yet have no conclusive proof.

Even so, we feel we have made an important stride forward by demonstrating the use of a feature-based approach to capturing full natural gesture and by establishing the separation of analysis from interpretation. This demonstration opens the way for the use of gesticulation in virtual environments and shows how gesture can be treated as a data type, available for use in a wide variety of applications.

ACKNOWLEDGMENTS

Thanks to Alberto Castillo for help in performing the experiments, to Chris Schmandt for providing initial insights on the analogy between speech and gesture, and to David Koons for help in understanding the problems which impeded analysis and for his work in developing the interpreter module. Special thanks to Richard Bolt for his guidance and supervision.

REFERENCES

BADLER, P. AND WEBBER, B. L. 1993. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, New York.

BOS, E. 1992. Some virtues and limitations of action inferring interfaces. In *Proceedings of UIST '92* (Monterey, Calif., Nov. 15–18). ACM Press, New York, 79–88.

BOLT, R. A. 1984. *The Human Interface*. Van Nostrand Reinhold, New York.

BOLT, R. A. AND HERRANZ, E. J. 1992. Two-handed gesture with speech in multi-modal natural dialogue. In *Proceedings of UIST '92*. ACM Press, New York.

CODELLA, C., JALILI, R., KOVED, L., LEWIS, B., LING, D. T., LIPSCOMB, J. S., RABENHORST, D. A., WANG, C. P., NORTON, A., SWEENEY, P., AND TURK, G. 1992. Interactive simulation in a multi-person virtual world. In *Proceedings of CHI '92*. ACM Press, New York, 329–334.

DARRELL, T. AND PENTLAND, A. 1993. Space-time gestures. In the *IEEE Conference on Vision and Pattern Recognition* (June). IEEE, New York.

FRENKEL, K. A. 1994. A conversation with Brenda Laurel. *Interactions 1*, 1 (Jan.), 45–52.

KOONS, D. B., SPARRELL, C. J., AND THORISSON, K. R. 1993. Integrating simultaneous input from speech, gaze, and hand gestures. In *Intelligent Multi-Media Interfaces*, M. Maybury, Ed. AAAI Press, Menlo Park, Calif., 252–276.

KRUEGER, M. 1991. *Artificial Reality*. Addison-Wesley, Reading, Mass.

LEWIS, J. B., KOVED, L., AND LING, D. 1991. Dialog structures for virtual worlds. In *CHI'91 Proceedings*. ACM Press, New York.

MAES, P. 1993. ALIVE: An artificial interactive video environment. In *Visual Proceedings of the SIGGRAPH '93 Conference: ACM SIGGRAPH*. ACM Press, New York, 189–190.

MCNEILL, D. 1993. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, Chicago, Ill.

MURAKAMI, K. AND TAGUCHI, H. 1991. Gesture recognition using recurrent neural networks. In *CHI'91 Conference Proceedings*. ACM Press, New York.

QUEK, F. 1994. Toward a vision-based hand gesture interface. In *ACM Virtual Reality Systems and Technology (VRST'94) Proceedings*. ACM Press, New York.

RIME, B. AND SCHIARATURA, L. 1991. Gesture and speech. In *Fundamentals of Nonverbal Behavior*, R. Feldman and B. Rime Eds. Press Syndicate of the University of Cambridge, New York, 239–281.

SPARRELL, C. J. 1993. Coverbal iconic gestures in human-computer interaction. S. M. thesis, MIT Media Arts and Sciences Section, Cambridge, Mass.

THORISSON, K., KOONS, D., AND BOLT, R. 1992. Multi-modal natural dialogue. In *Proceedings of CHI '92*. ACM Press, New York, 653–654.

VÄÄNÄNEN, K. AND BÖHM, K. 1993. Gesture-driven interaction as a human factor in virtual environments—An approach with neural networks. In *Virtual Reality Systems*, M. A. Gigante and H. Jones, Eds. Academic Press, Ltd., London, U.K.

WEIMER, D. AND GANAPATHY, S. K. 1989. A synthetic visual environment with hand gesturing and voice input. In *CHI'89 Proceedings*. ACM Press, New York.

WEIMER, D. AND GANAPATHY, S. K. 1992. Interaction techniques using hand tracking and speech recognition. In *Multimedia Interface Design*, M. Blattner and R. B. Dannenberg, Eds. ACM Press, New York.

WEXELBLAT, A. 1994a. A feature-based approach to continuous-gesture analysis. S. M. thesis, MIT Program in Media Arts and Sciences, Cambridge, Mass. Available as ftp://media.mit.edu/wex/MS-Thesis.ps/gz.

WEXELBLAT, A. 1994b. Natural gesture in virtual environments. In *ACM Virtual Reality Systems and Technology (VRST'94) Proceedings*. ACM Press, New York.

Received November 1994; revised February 1995; accepted March 1995