A Flexible Gesture Interface*

Richard Watson and Paul O'Neill
Computer Vision Group,
Department of Computer Science, Trinity College, Dublin 2. (IRELAND)

email: Richard.Watson@cs.tcd.ie
Tel: +353-1-6081435, Fax: +353-1-6772204

Abstract

This paper describes an interface employing gestures to improve the utility of interaction between a user and an artificial world populated by graphical objects. The interface is based on a novel technique for recognising gestures. Gestures are represented by what are called *approximate splines*, sequences of critical points (local minima and maxima) of the motion of each degree of freedom of the hand and wrist. This scheme allows more flexibility in matching a gesture spatially and temporally and reduces the computation required, compared with a full spline curve fitting approach.

The recognition process remains logically independent from the client application and the producers of its input data. The communications/messaging protocol has been designed and demonstrated to allow the client, gesture interface and data gathering processes to run on a heterogeneous distributed system. Further independence is afforded by separating the virtual world *application specific* tokens from the *physical gestures* they are mapped onto, allowing for arbitrary application support for gestures.

Keywords: Gesture Interface, Gesture Classification, Virtual World Commands.

1 Motivation

A simultaneous improvement in the enabling technologies and the need for gesture based input to complex applications were the motivating factors for the recent interest in gesture interfaces. This interest and relevant research is documented in a previous paper [1]. The present paper concentrates on the design of a gesture interface developed as a response to that work.

The Gesture Interface which has been developed as part of the GLAD-IN-ART project, provides this increase in bandwidth in communicating with applications. The present system can recognise static gestures, posture-based dynamic gestures, pose-based dynamic gestures, a "virtual control panel" involving posture and pose and simple pose-based trajectory analysis of postures. Training the gesture set is accomplished through the interactive presentation of a small number of samples of each gesture. The usefulness of such indirect manipulation as a mode of interaction is demonstrated by the use of the Gesture Interface to control virtual world actions such as navigation, panning and zooming of viewpoint and graphical object manipulation.

2 Approaches to Recognising Gestures

2.1 First Steps

One of the earliest posture recognition efforts was by Grimes [2], whose Digital Data Entry Glove was designed specifically for recognising the signed alphabet. Carefully placed sensors registered fingertip contact, flexion of specific joints, and hand attitude. Posture recognition was hard-coded into the control electronics. The advantage of this technique is rapid and robust posture recognition. The disadvantage is inflexibility, in that only those postures it was designed for can be recognised.

In 1980, Bolt used a prototype Polhemus 6D pose tracking system for sensing the direction of a point in his "Put-that-there" system [3], The sensor was attached to the user's wrist and pose information

*This research was funded by the Commission of the European Communities under the ESPRIT II Framework enabled the system to (at least) estimate where he was pointing at.

2.2 Template Matching

VPL's ground breaking work in 1987 [4] used a template maching maethod for recognising postures. VPL's software also provided hysteresis values for each sensor value to widen the range of the match once a posture has been recognised, helping the user to hold a posture after recognition. Lipscomb [5] also used a template matching based method for recognising used for recognising stroke, i.e. 2D, gestures. This was a variant of the usual technique where multiple templates were maintained for each gesture, corresponding to increasingly coarse resolutions of sensor values. To approach *gesture* recognition using a template matching scheme, gestures would have to be recognised as sequences of postures. In this technique, trajectories of the degrees of freedom are not modelled, hence spatial scaling would be impossible, and temporal scaling although possible, would be somewhat inflexible.

2.3 Neural Networks

Neural networks have provided several somewhat successful systems, notably Fels' [6]. Fels' work concentrated on building a gesture-to-speech interface, using a VPL DataGlove connected to a DECtalk speech synthesiser via a series of back-propagation model neural networks. Brooks [7] also reports use of a neural net, (in this case a Kohonen model) to control a mobile robot by interpreting DataGlove motion. In ways, this surpasses Fels' work, particularly by incorporating dynamic gestures into the system's vocabulary. In [8], Beale and Edwards use a multilayer perceptron model to classify input into one of five postures, taken from the American Sign Language.

2.4 Statistical Classification

Rubine [9] created not only a gesture recognition system, but GRANDMA[1], an object-oriented toolkit for building gesture-based applications based on a statistical pattern matching approach. The gestures considered in his work consist of the two dimensional path of a single point over time that may be input with a single pointer, such as a mouse, stylus or touch pad. Sturman [10] extended Rubine's system to deal with multi-path gestures using a VPL DataGlove. Significantly, the feature analysis was extended to three-dimensions and modified to permit continual analysis and recognition without explicit start and end points.

[1]Gesture Recognisers Automated in a Novel Direct Manipulation Architecture

2.5 Discontinuity Matching

The approach taken in this project is a novel one. Figure 1 shows how an MCP[2] joint changes as the hand posture changes. The features extracted in this system, are critical points of the motion of a degree of freedom or *discontinuities.* A discontinuity is a peak, trough, or either the start or end of a plateau. This first derivative representation is reminiscent of a spline approximation to a curve, but without the intermediate knots, (control points). The advantage this approach has over the classical template matching approach is that the relevant features are motion oriented, necessary for a system that performs gesture recognition. Also, since the discontinuities need not be matched at precisely the same points in time or space, the system is robust to scaling.

The other advantage of representing merely critical points is that the system is less susceptible to input noise. Analysing the input data from the proprioceptive glove and the pose calculation module, discontinuity extraction can be performed by analysing the angular velocity of a degree of freedom. Hand jitter is modelled simply by high frequency motion, thus the critical points are extracted using a low-pass filter.

The classification stage is a template matching process where sequences of discontinuities for each degree of freedom are compared against those extracted. The interface module maintains a set of gesture templates, composed of sequences of discontinuities for sequences of degrees of freedom. The templates may be viewed as the axes of a multi-dimensional gesture space; thus the aim of the classifier is to firstly calculate the axis to which a given set of observed motion discontinuities is closest, and then to decide whether this is *close enough* given a set of distance metrics.

The first process of mapping a set of observed discontinuities to a gesture subspace i.e. matching sequences of discontinuities, can be formulated as a finite state acceptor, shown here as the 5-tuple, M :

$$M = < Q, I, \delta, q_0, F >$$

M accepts an instance of the correct discontinuity

[2]Metacorpophlangeal Joints, or Knuckles

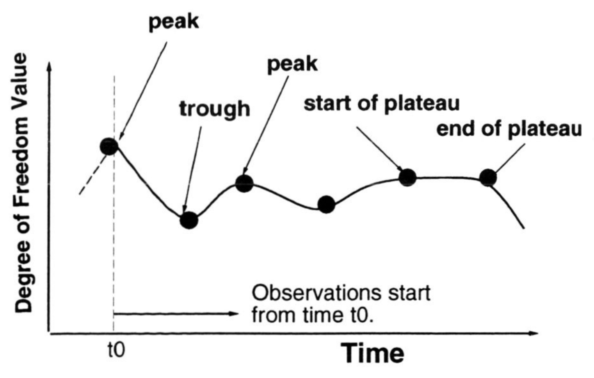


Figure 1: Time-space pattern of a MCP joint in performing a gesture

pattern, for a degree of freedom, j and a gesture class, c, where the state set, Q, is the set of partial pattern matches, the input alphabet, I, is the set of discontinuity types, the transition function, δ, is determined by the temporal sequence of discontinuities trained for this template, the initial state, q_0 is the first discontinuity in the sequence, and $F \subseteq Q$, acceptable halting states, is the final discontinuity. An example discontinuity pattern and its representation in this formulation is shown in Figure 2. A further

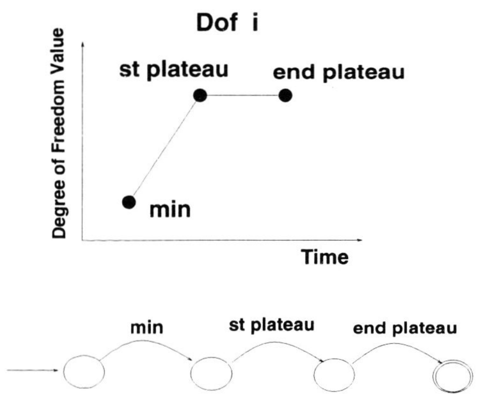


Figure 2: Template Discontinuity pattern for a single degree of freedom and a labelled digraph corresponding to its FSA.

stage calculates whether the gesture is acceptable according to several fit metrics. The computation of these metrics is desctibed in a further paper [11].

3 The Gesture Interface Module

The high-level design of the Gesture Interface Module (GIM) is shown in (Figure 3). The GIM consists

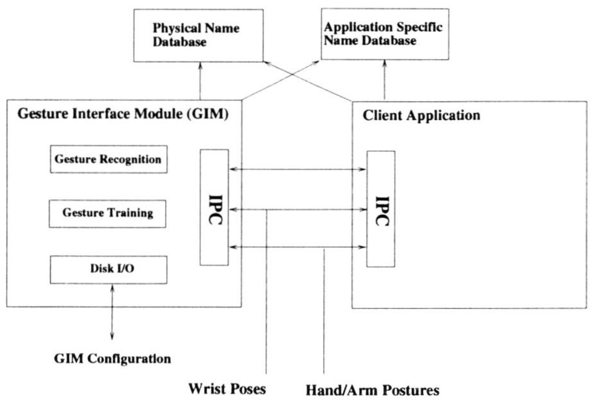


Figure 3: The Gesture Interface Module.

of a gesture recognition module, a gesture training module and an I/O module which handles communication with the client application (in this case the ART subsystem).

3.1 Inputs

The Gesture Interface Module receives three streams of input:

- A stream of time-stamped homogeneous transformations describing the pose (position and orientation) of the wrist with respect to the Control Space Base Frame. This input stream is generated by the GESTURE (POSE) subsystem.
- A stream of time-stamped values describing the posture of the hand and arm. Each value gives the magnitude of a particular degree of freedom of hand/arm posture. This input stream is generated by the GLAD-IN subsystem (ie. the instrumented glove and exoskeleton)
- A stream of commands and responses to I/O requests. This input stream is generated by the client application. The client application controls the high-level behaviour of the Gesture Interface Module and provides it with I/O services. These I/O services are used for communication with the user (during gesture training, for example).

3.2 Outputs

The Gesture Interface Module produces two output streams:

- Each time the Gesture Interface Module recognises a physical gesture it sends at least two distinct gesture notifications to the client application. These notifications are described in Section 5.
- It also responds to commands received from the client application and generates I/O service requests.

4 Gesture Training Module

The purpose of the Gesture Training Module is to semi-automatically compute a representation for each physical gesture. The details of the Gesture Training Module are described in [11].

5 Gesture Notifications

The Gesture Interface Module notifies the application of the start and end of each gesture. It may also notify the application with update information concerning the point direction of the forefinger and the position of the tip of the forefinger[3]. This section explains the mechanism through which gesture notifications are issued by the Gesture Interface Module and the complementary mechanism through which these gestures are accepted by the application (in this case the ART subsystem).

5.1 Sender

Gesture notifications are sent by the Gesture Interface Module when a physical gesture is recognised. Notifications are only sent when a gesture has been *positively* identified. The two mandatory notifications are for the start and end of the gesture.

The end notification may contain physical gesture parameters such as temporal and spatial scale. The Gesture Interface Module queries the application-specific database (Fig. 3) to determine whether this information is required. Similarly it can determine whether the client application should be regularly updated over the course of a static gesture about the evolving forefinger tip position and point direction. This update information is supplied via the third type of gesture notification: the update notification.

[3]This information is calculated by the Gesture Interface using a kinematic model of the hand/wrist supplied at startup by the client.

It is possible that a gesture is incorrectly identified (confused with another gesture or with normal hand motion). If this gesture is mapped onto an important (difficult to reverse) action in the virtual world then the results could be disastrous. A simple mechanism has been provided whereby a physical gesture must be repeated n times consecutively before it is notified to the client application. The value of n is individually configurable for each application-specific gesture and in most cases is set to a value of 1. A more sophisticated future solution to this problem lies in system interaction with the user to validate his intentions.

It is possible that overlapping gesture notifications will be produced by the Gesture Interface Module. This depends upon the gesture set used to train the Module. The Gesture Interface Module can be configured to react in three different ways when this happens.

1. Send the gesture notifications in any case. This is the default mode of operation.
2. When an overlapping notification is generated send it, but first of all force a conclusion to the existing gesture by sending an end notification. This is in most cases a counter-intuitive mechanism since it results in a gesture being notified as finished even though, *physically*, it is still continuing.
3. Discard any overlapping gesture notifications.

The latter two handling mechanisms are useful in the case where the client application's notification handling mechanism is not sufficiently sophisticated to deal with concurrent notifications. This is not the case with the ART subsystem.

5.2 Receiver

The portion of the Gesture Interface residing in the client application is responsible for acting on received gesture notifications. The following mechanism is typical: Each application-specific gesture type has three associated handler functions: one each for *start* notifications, *update* notifications and *end* notifications. Each function modifies particular fields in a global handler table. This table is examined regularly and its state is used to drive (*virtual world*) actions.

For example, while the NAVIGATE gesture is performed a particular variable in the handler table is set. The state of this variable is used to drive incremental updates of the View Frame pose and Control Space Base Frame pose.

5.3 Gesture Types

The gesture notification handler is flexible enough to allow several different handling mechanisms to be used.

Immediate Action The occurrence of this type of gesture causes a particular (*virtual world*) action to be performed exactly once. This type of gesture is useful in the creation and deletion of graphical objects and in the handling of mouse button clicks (when the hand is emulating a pointing device).

Repeated Action (Modeless) Between the start and end notification of the gesture a (*virtual world*) action will be performed repeatedly. The number of times the action is performed depends upon the number of times that the client application processes the handler table. This gesture type is useful for incremental actions such as panning, zooming, pointing, navigating etc.

Repeated Action (Modal) The overall effect is that between the start and end notification of the gesture a (*virtual world*) action will be performed repeatedly. When the start notification is received a once-off setup action will be performed in the virtual world. When the end notification is received a once-off take-down action will be performed in the virtual world.

This type of gesture is useful for controlling the grasp procedure on graphical objects.

- The fixed pose relationship between the hand and the grasped object is maintained by repeated (*virtual world*) actions.
- The fixed pose relationship between the hand and the graphical object is asserted at the start of the gesture (through the setup function).
- The fixed pose relationship between the hand and the graphical object is deasserted at the end of the gesture (through the takedown function).

Dynamic physical gestures are not suited to the repeated action (modal or modeless) mechanism. This is because the start notification for a dynamic gesture is not issued until the gesture is completed.

Cooperating Gestures This type of gesture is useful for controlling rotation of the user's view frame (in a Virtual World). The rotation is commenced by the ROTATE gesture (which as one of its parameters specifies the axis of rotation). The rotation is terminated by the *stop* gesture.

Meta-Gestures Gestures can be mapped to *meta-actions* in the client application. An example of this type of gesture is quitting from the client application.

6 Gesture Mapping

In preceding sections the software design of the Gesture Interface Module has been discussed in detail. It is now appropriate to consider how gestural interaction may be used profitably by a particular client application: the ART subsystem of the GLAD-IN-ART system in this example case.

6.1 Grasping an Object.

The core idea of the GLAD-IN-ART system is that of direct manipulation where the interaction between the user's hand and each virtual object is modelled to a high level of detail. However, in a simplified version of the system, a grasp gesture may be used for object manipulation. Under this approach, the user grasps a virtual object by clenching his fist beside (or inside) the object. The pose (position and orientation) of a grasped object is continuously updated (by the ART subsystem) in order to maintain a *fixed* relationship between the object pose and the pose of the palm of the user's hand. The object is released when the user unclenches his fist.

6.2 A Point & Click Device.

The hand may be used as a pointing device. The forefinger is used to specify a 3D vector. The abduction angle of the thumb may be compared to a threshold, thus allowing motion of the thumb to map onto a "button click". The use of the hand as a pointing device will enable, in the future, the operation of virtual buttons, sliders and pop-up menus. It will also allow the selection and de-selection of virtual and physical objects for subsequent (menu-selected) operations.

6.3 Navigation of the Virtual World.

The user may change the position of his point of view in the virtual world using the navigate gesture. The forefinger of the hand is used to specify a 3D vector (as in the point gesture). The user's point of view, and the Control Space Base Frame (and therefore the representation of his hand), is translated along this vector at a pre-determined rate[4]. (Figure 4). Translation stops when the navigate gesture is broken. The navigate gesture is distinguished from the point gesture by the position of the index finger (which, in the case of the navigate gesture, is aligned with the forefinger).

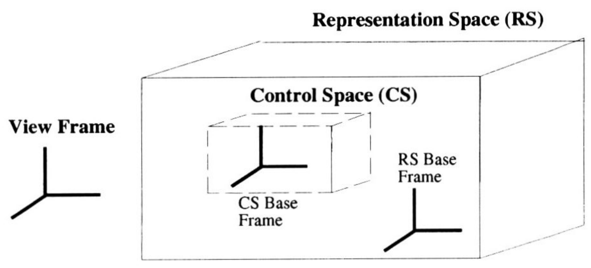


Figure 4: The user's hand pose is tracked within the Control Space. However, the size of the Control Space may be small compared to the desired size of the Virtual World (or Representation Space). One solution to this problem is to scale all hand-motions performed in the Control Space. A preferred solution is to allow the user to *navigate* around the Representation Space using gestures. The navigate gesture specifies a translation which is incrementally applied to the viewing frame and the Control Space base frame. Therefore, the positions of these frames change with respect to the Representation Space base frame.

6.4 Controlling the View Frame Orientation.

The view frame orientation may be modified by incrementally rotating it around a user-specified axis. The user specifies this axis by drawing a circle in 3-space. The boundary of the circle is drawn by the tip of the forefinger of the hand. The drawn circle lies in a particular plane. The normal vector to this plane, when translated to the origin of the view frame, is the desired axis of rotation.

This approach, which may seem complicated at first sight, will allow the user to specify the desired rotation in a very intuitive fashion (Fig 5).

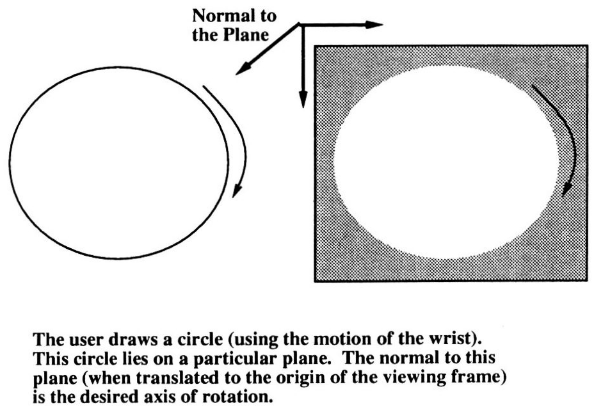

Normal to the Plane
The user draws a circle (using the motion of the wrist). This circle lies on a particular plane. The normal to this plane (when translated to the origin of the viewing frame) is the desired axis of rotation.

Figure 5: The view frame orientation may be modified by incrementally rotating it around a user-specified axis. The desired axis of rotation may be specified (indirectly) by a circular drawing motion of the forefinger.

The view frame is rotated around the specified axis (at a pre-determined pace) until the user performs the *stop* gesture.

6.5 The Creation & Deletion of Objects.

The user may wish to interactively create and delete virtual objects. Gestures will allow him to do this in a very convenient and natural fashion. Because there is no menu-based interface at present, each graphical object type (eg. cube or sphere) will be *created* by *its own* associated physical gesture. There is, however, a single *delete* gesture. When it is issued, the graphical object closest to the hand is deleted from the virtual world.

6.6 Finger-Spelling.

The Gesture Interface Module can be used to recognise the gestural symbols of the Irish Sign Language finger-spelling alphabet. These symbols can be used in the following ways: the alphabet symbols can be

[4]It should be noted that this means that the relationship between the Control Space Base Frame and the Virtual World Base Frame is not fixed. This is appropriate due to the small size of the Control Space with respect to the desired size of the Virtual World.

mapped directly onto virtual world actions (*Implemented*) or the alphabet symbols can be used to input text.

6.7 Quitting from the System.

At a given point in his work, the user may wish to quit from the GLAD-IN-ART system. This may be conveyed by a single gesture.[5]

6.8 Reset.

This command resets all the system coordinate frames to their initial poses. It can be used to undo the effect of zooms, rotates etc.

6.9 Toggle Recognition Mode.

This gesture is used to toggle on whether or not ART should act on received gesture notifications. Note that if the start notification for a given gesture instance is acted upon then the end notification for that gesture will also be acted upon *regardless of the state of the toggle switch.*

6.10 Viewpoint Manipulation.

Gestures may also be used to directly invoke certain actions in the graphical representation (ART Graphic Presentation Library). The available functions are:

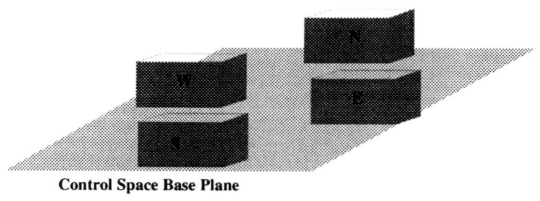
- Zoom in, Zoom out.
- Pan left, Pan right, Pan up, Pan down.
- Advance point of view.
- Rotate point of view meridian, Rotate point of view parallel.

The functions are invoked incrementally at regular intervals throughout the duration of their associated *static gesture.*

[5]A facility of the Gesture Interface Module is that for commands with actions that are occasionally performed or difficult to reverse, like quitting, it may be stipulated that a number of consecutive repetitions of the physical gesture be performed. A database of *hardness* scores for each command is maintained, essentially the number of times the gesture must be repeated before a notification is sent to the client. This is useful in avoiding actions resulting from non-deliberate action or a mis-recognition from the interface.

6.11 Viewpoint Control Panel.

Static gestures may be defined to have a particular posture *and* a particular pose. More accurately, the pose of the gesture may be defined to lie within a particular (*usually small*) sub-volume of the Control Space. A Control Panel consisting of four such sub-volumes (within the Control Space) can be imagined. The semantics attached to a particular static posture are modified by the sub-volume which the wrist is lying in. The semantics used in this situation can be related to viewpoint manipulation (Fig. 6). Note the similarity between this idea and that of a virtual keyboard.



A Static gesture may be defined by both its posture and its position.

Posture	Position	Virtual World Action
(1)	N	ZOOM_IN
(1)	S	ZOOM OUT
(1)	E	ROTATE VIEW MERIDIAN.
(1)	W	ROTATE VIEW PARALLEL
(2)	N	PAN UP
(2)	S	PAN DOWN
(2)	E	PAN RIGHT
(2)	W	PAN LEFT
(3)	N	ADVANCE VIEW
(3)	S	RESET VIEW

Figure 6: Through the use of the pose and posture of a static gesture it is possible to create a virtual control panel for viewpoint manipulation.

7 Conclusions

7.1 Results

Up to twelve **static gestures** can be recognised: these are all gestures from the Irish single-handed deaf alphabet. The following **posture-based dynamic gestures** can be recognised: "Come Here"; The initial posture of this gesture is a flat-hand. The forefinger is flexed and then extended again in one smooth motion. Variants of this gesture can also be recognised: the "Come Here 2" gesture recognises two *consecutive* flexion/extension motions of the forefinger, the "Come Here" gesture may be based upon the ring or middle finger instead of the index finger, the "Come Here Combined" gesture recognises a flexion/extension motion of the index finger *followed by* a flexion/extension motion of the middle finger. Recognition of this variant demonstrates that sequencing constraints can be enforced across degree-of-freedom boundaries. "Thumb Click". Thumb flexion and yaw is brought from its minimum value to its maximum value and then back to its minimum value in one smooth motion, while the other degrees of freedom maintain a static point gesture. This is an example of a *cooperating gesture* (Section 5.3): the static gesture for "point" must be active while the thumb-click is performed.

The following **pose-based dynamic gestures** can be recognised based upon their discontinuity patterns: Circle; the user traces a circle in space, with his wrist and X gesture; the user traces an X pattern in space with his wrist.

Pose-based motions of the hand, have been recognised through comparison of the start pose of a so-specified posture-based static gesture with its end pose. By comparing the wrist's end position with its start position a *naïve* notion of trajectory is calculated. If that trajectory is exclusively parallel to one of x, y or z axes, the gesture is a candidate for a punch gesture. Similarly if, comparing the start and end orientations of the wrist the wrist is calculated to have rotated 90^o along the major axis of the forearm, it may be deemed to have 'turned over'. These gestures are termed **Post-dynmaic static**.

7.2 Future Work

Future work will concentrate on development of a more flexible discontinuity pattern representation which allows variability to be expressed elegantly and orientation-invariant descriptions of pose-based gestures. At present the computational task of recognising gestures is $O(n)$, where n is the number of gesture classes (or templates). A method of constructing a tree (or hash table) of partial discontinuity sequence matches would (in theory) reduce this complexity to $O(\log n)$. The integration of the Gesture Interface with alternative graphical clients is another immediate need.

References

[1] Richard Watson. A Survey of Gesture Recognition Techniques. Technical Report TCD-CS-93-11, Department of Computer Science, Trinity College Dublin, July 1993. Available at ftp://ftp.cs.tcd.ie/pub/tcd/tech-reports/reports.93/TCD-CS-93-11.ps.Z.

[2] Gary J. Grimes. Digital data entry glove interface device. Technical Report US Patent 4,414,537, Bell Telephone Laboratories, November 1983.

[3] Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. *Computer Graphics*, 14, No. 3:262–270, July 1980.

[4] Thomas G. Zimmerman and Jaron Lanier. A hand gesture interface device. *ACM SIGCHI/GI*, pages 189–192, 1987.

[5] J.S. Lipscomb. A trainable gesture recogniser. *Pattern Recognition*, 1991.

[6] S. Sidney Fels and Geoffrey E. Hinton. Building adaptive interfaces with neural networks: The glove-talk pilot study. In *Human-Computer Interaction—INTERACT '90*, pages 683–688. IFIP, Elsevier Science Publishers B.V. (North-Holland), 1990.

[7] Martin Brooks. The dataglove as a man-machine interface for robotics. In *The Second IARP Workshop on Medical and Healthcare Robotics*, Newcastle upon Tyne, UK, September 5-7 1989.

[8] R Beale and A Edwards. Recognising postures and gestures using neural networks. In R. Beale and Finlay J., editors, *Neural Networks and Pattern Recognition in Human Computer Interaction.* E. Horwood, 1992.

[9] Dean Rubine. *The Automatic Recognition of Gestures.* PhD thesis, Carnegie Mellon University, December 1991.

[10] David J. Sturman. *Whole Hand Input.* PhD thesis, Massechusetts Institute of Technology, 1992.

[11] Richard Watson and Paul O'Neill. Gesture Recognition for Manipulation in Artificial Realities. In Y Anzai and G Ogawa, editors, *Proceedings of the 6th International Conference on Human-Computer Interaction, Pacifico Yokahama, Yokahama, Japan*, July 1995.