RAL-92-019

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-92-019

Dynamic Gesture Machine

M Bordegoni

March 1992

Dynamic Gesture Machine

Monica Bordegoni

February 1992

Dr. Monica Bordegoni's parent institution: IMU/CNR Via Ampere 56, Milano - Italy
email: mb@imucad.mi.cnr.it
RAL contact: Dr. Victoria Burrill email: vab@uk.ac.rl.inf

Dynamic Gesture Machine

TABLE OF CONTENTS

1. Introduction

This report describes work performed during a 6 month ERCIM (European Research Consortium for Informatics and Mathematics) fellowship in the Informatics Department at Rutherford Appleton Laboratory. This research has been carried out as part of an on-going study at RAL into the use of gesture in a computer interface.

Gestures represent one of the most intuitive way humans can use to communicate with each other, so it is natural to extend the use of gestures to the communication between humans and computers. In particular, in Virtual Reality and other highly complex 3D graphical manipulation applications, the use of gesture is becoming more important since it provides the user with a very intuitive interface that cannot really be supported by traditional 2D input devices, such as mice.

The research described here aims at providing a system which allows the wearer of a DataGlove to demonstrate a series of hand gestures, and then use those gestures as input to another application. The device used for this purpose is a VPL DataGlove 2 connected to a Sun 3/60 workstation running the MIT X Window System. This report describes the architecture of the system and the structure and implementation details of the algorithm used for the gesture recognition. During the course of experiments with the system, various factors and issues have emerged concerning human gesture and the feedback of the system.

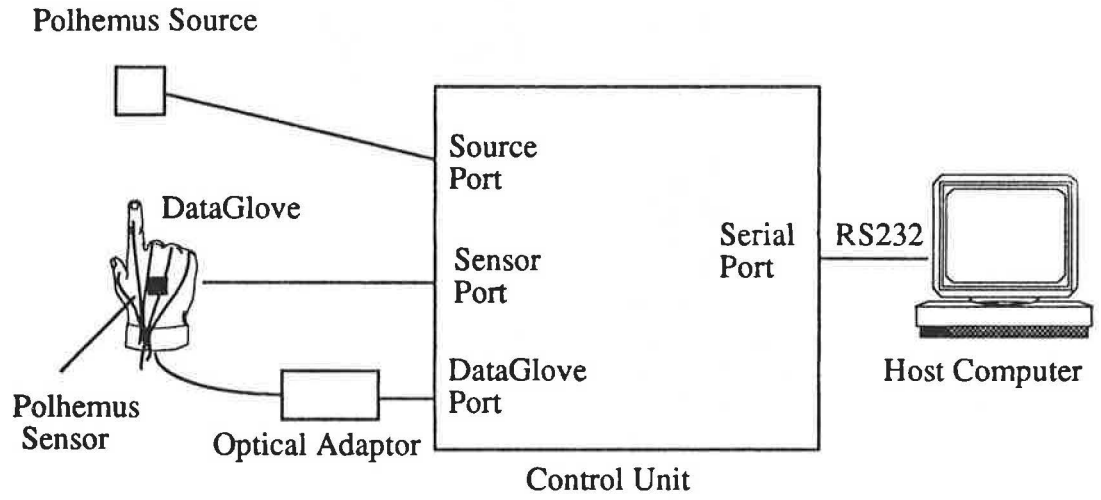
The research has two main aims: firstly, to identify the various issues relating to the problems of gesture recognition and secondly, the visualisation of gesture feedback to aid the user. Both these aims encompass complex issues, so this research does not claim to have solved the problem but it does go some way towards understanding it.

2. The VPL DataGlove

2.1 Description

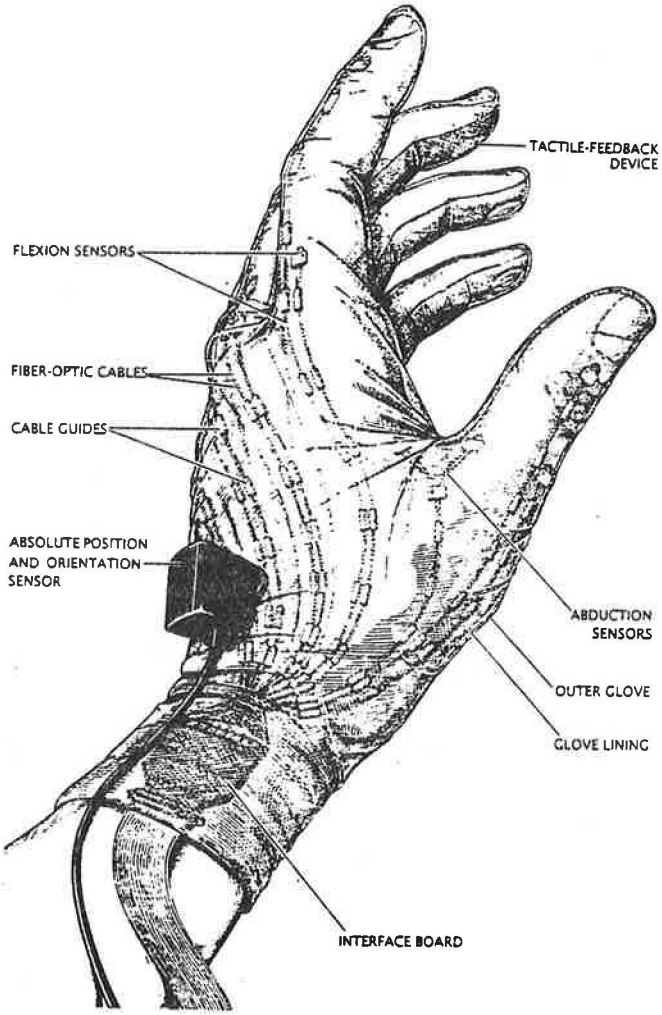
The VPL DataGlove 2 [1] is an instrumented glove which can be used to provide real time information about the wearer's hand movements. It consists of a Data Glove and a Control Unit, connected to a host computer. The Control Unit converts analog signals coming from the DataGlove into a suitable format for a host computer and sends this data to the host via an RS232 serial port (Figure 1).

FIGURE 1. DataGlove System



The DataGlove consists of 10 optical fibre cables located on the back of the hand, two per finger, such that one measures the flexion of the knuckle joints, the other the mid-way joints (Figure 2). When the finger bends, the light passing through the optical fibre cable is attenuated according to the amount of flexion. The light levels are converted into meaningful values and sent to the host computer.

FIGURE 2. Data Glove



A Polhemus sensor system (Polhemus 3Space Isotrak) [2] measures 6 degrees of freedom: position of the hand in 3D space plus pitch, yaw and roll.

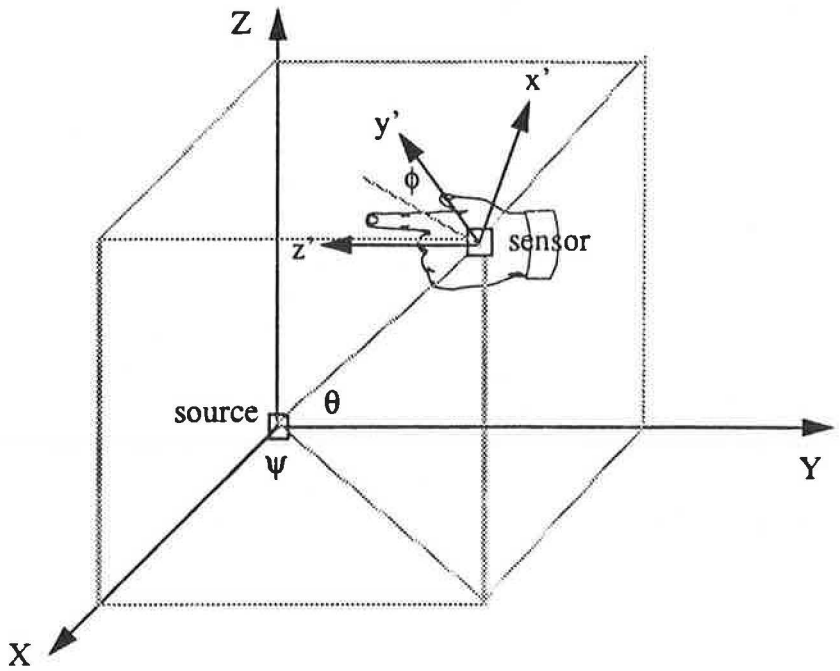
The Polhemus system consists of:

- a *Source*, which is placed in a fixed position and emits an electromagnetic field,

- a *Sensor,* attached to the back of the glove, which detects its position and orientation relative to the source and transmits this data to the Control Unit.

The X, Y, Z axes of the source provide the default reference frame from which all sensor readings are measured (Figure 3). To obtain consistent data, the source should not be moved during sensor measurements.

FIGURE 3. Polhemus Reference System



2.2 Technical Issues

During development of the project, several technical issues became apparent. These are discussed below.

2.2.1 Calibration and Gesture Table

The *DataGlove Test & Calibration Software Manual* for the Apple Macintosh [3] states that the host computer connected to the Glove System can download both Calibration and Gesture Tables to the Control Unit. In practice, several problems were encountered.

Calibration Table

If the Calibration Table has been downloaded, the Control Unit sends back to the host computer the values which correspond to the angles (calculated from 0 to 115 degrees) of the finger joints. Otherwise, it sends back raw values. A summary description of how this calibrating activity is performed by the Control Unit is given in the Macintosh Operation Manual. The relevant command (LoadTable) is described in the *DataGlove Model 2 Operation Manual* [1]. Unfortunately, its description is so succinct that no information can be elicited concerning the format of data to be downloaded. In the event, the system was implemented without downloading the Calibration Table, as it is sufficient to check raw values of finger joints instead of

calibrated values to recognise hand postures. A few months later, some code was received from Sidney Fels at the University of Toronto, Canada, who had succeeded in establishing, by trial and error, the format of the command.

Gesture Table

According to the manual, it should also be possible to download Gesture Tables by specifying static hand postures (bending values of the finger joints) and sending them to the Control Unit. When one of these postures is later performed by a user, it is recognised by the Control Unit and a posture identification code is sent to the host computer. Unfortunately, no mention of the appropriate command was found in the manual and the additional information given by VPL was insufficient to understand which format of data to download.

The manual seems to suggest that only 10 postures can be downloaded; in the event, this would have been an insufficient number to support later work.

2.2.2 Control Unit

Two problems were identified concerning the Control Unit. Firstly, the use of the DataGlove System for a period of about 30 - 40 minutes caused the Polhemus Source to become very hot and fears were expressed that this might damage its components. For this reason, the Control Unit had to be switched-off at regular intervals to cool.

Secondly, the use of the Repeat30 or Repeat60 commands (which request DataGlove records at 30 or 60 records/second) sometimes caused problems such that the host failed to initialise the Control Unit. In this case even switching off or resetting the Control Unit had no effect and the host computer had to be rebooted.

3. Background Work

3.1 State of the Art

User interfaces of virtual reality applications aim at giving users the impression of directly interacting with their application, rather than via a computer. While great interest has been shown in the uses of computers to visualise output, less effort seems to have been spent studying input devices. Traditional 2D input devices, such as mice and tablets, are no longer adequate for these applications, and more powerful and expressive devices are required. One of the most interesting approaches is the use of hand-tracking devices, particularly to support gesture.

Gesture represents one of the most intuitive ways humans can communicate with each other. When two people meet (who do not speak the same language), they often use gestures to make themselves understood. In addition, gestures are often used as common actions in everyday life. Therefore, it is natural to extend this use of gesture to make more intuitive the communication between humans and computers (who do, after all, "speak" very different languages!).

Gestures are very powerful in that they allow humans to combine many parameters in parallel. Different gestures can be expressed by combining different flexion values of the fingers, the orientation of the hand and the sequence of positions of the hand in space. As in the case of icons, gestures can have different meanings and interpretations according to the users' cultural background. It could happen that a gesture which has meaning for one person of one culture, has no meaning (or even worse, an impolite meaning) to someone from another culture. For example, a two-fingered "V" sign in dorsal presentation is extremely impolite in British society, but would be interpreted as "two" in Italian.

It has been shown [4] that although gesture can be intuitive, it must be incorporated judiciously into an interface. For example, datagloves are suitable for applications which involve the manipulation of 3D objects, but for simpler 2D applications the number of degrees of freedom provided by such devices can be too great, and some simplification of the data is required according to context.

The concept of direct manipulation is very powerful when applied to traditional interfaces, but results in some unexpected problems when applied to virtual reality. For example, stretching out a hand to pick up a (virtual) object is direct and intuitive; so too is using one finger of the hand as a virtual screwdriver to tighten a screw using a series of repeated in, rotate, out, reposition, in, rotate, out, reposition movements. (Direct manipulation with real world realism.) But using the hand to dig up a section of road would not be possible in real life since humans are not strong enough. (The direct manipulation metaphor remains, but real world realism does not.) Taking this one step further, the finger of the hand could also be presented as a virtual screwdriver tool from a virtual toolbox where the user makes the gesture once then indicates "repeat that until some condition is met". (Both the direct manipulation metaphor and real world realism are broken.)

Another intriguing application field is telepresence and telerobotics where the operator uses virtual reality to navigate a robot through a hazardous environment. The system records input events such as gestures with virtual reality meanings, and generates the corresponding action or movement of the robot within the real world.

Finally, the use of more than one input device in parallel could increase the power of the system [5]. For example, the interface to an application could be enhanced by using both speech and gestures. A 3D object could be rotated by saying "rotate" and pointing at the object. It would also be interesting to define systems which allow users to use two hands to perform actions in parallel. For example, the user could move an object using a DataGlove on one hand and simultaneously cycle through a series of colour changes by clicking a mouse button using the other hand.

Some research has been done to develop systems for gesture recognition [6] [7]. In particular, Neural Networks have been used to recognise a finger alphabet and Recurrent Neural Networks to recognise words and gestures [7].

3.2 Previous Work at RAL

Most of the work done at RAL on advanced input devices was within the EWS (EuroWorkStation) project, an Esprit II Project funded by the EEC [4]. A taxonomy and evaluation of input devices was performed, and three different input devices (mouse, Spaceball and DataGlove) were analysed.

An experiment was done to compare the use of the mouse and DataGlove as manipulation devices in 3D applications. Several users performed the same manipulation operations on 3D objects using both a mouse and a DataGlove. The experiment showed that the Data Glove is useful if the task involves the direct manipulation of an object in 3D space, but for simple tasks, it has too many degrees of freedom and became too difficult to use effectively.

The power of the glove consists of not only providing 6 degrees of freedom, but also allowing the definition of gestures so as to enrich the application interface. A taxonomy of gestures has been provided [8]. A static gesture is defined as one in which the fingers of the hand are held stationary; conversely, a dynamic gesture is defined as one in which the whole hand is moved. Four combinations of finger and hand movements can be identified:

- static fingers and static hand, for example thumbs up;
- static fingers and dynamic hand, for example waving goodbye;
- dynamic fingers and static hand, for example from a fist to an opened hand;
- dynamic fingers and dynamic hand, for example catch and throw a ball.

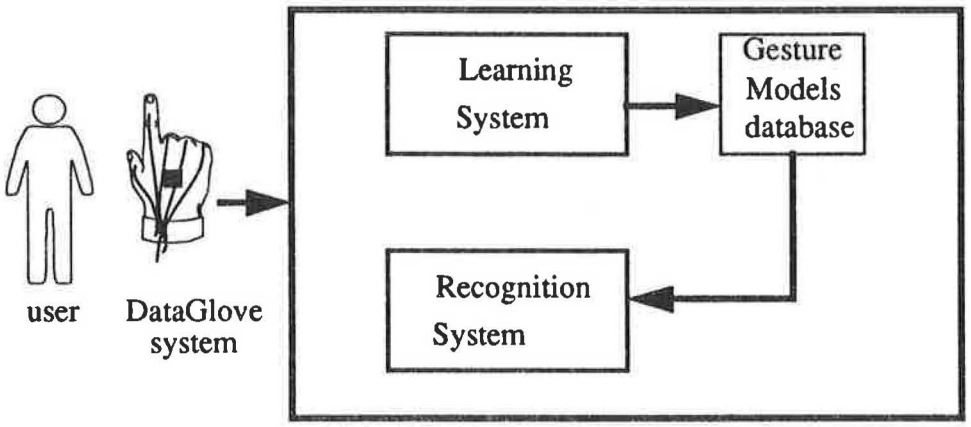
Another study at RAL investigated the use of the Data Glove in a popular environment: the desktop of the Open Windows window manager [9]. A Gesture Machine was implemented using Squeak [10], a user interface implementation

language developed at AT&T Bell Laboratories to handle multiple input streams concurrently. A system based on this Gesture Machine was implemented and static gestures allowed the user to manipulate windows on the screen. For example, the user could open and close a window by pointing at it or its icon as appropriate, or could move windows around the screen using a flat hand wiping motion. The user interface of the system was implemented using the NeWS toolkit.

4. Architecture of the System

The system described in this report comprises two modules. A Learning Module, to learn and store dynamic gestures, and a Recognition Module, to recognise the taught gestures and provide input to another application. The first module stores gesture models in a database and the second recognises the gestures loaded from the database. The system structure is shown in Figure 4.

FIGURE 4. General System Structure



This chapter describes the architecture of the two systems and the solutions adopted for their implementation. The system has been designed so as to be integrated under the X Window System, as it represents a *de facto* windowing system. The terminology used for the description of the system is presented in the following section.

4.1 Definition of Terms

4.1.1 Pose

A *pose* is a static configuration of the hand. It is characterised as follows:

- bending values of the 10 joints (2 per finger). These values measure the bending of the inner and outer joint of each finger. If the calibration of the glove has not been done, the Control Unit returns raw values, otherwise it returns calibrated values (degree values between 0 and 115).
- three Euler angle values (pitch, yaw, roll) describing the orientation of the hand. These values actually describe the orientation of the Polhemus Sensor placed on the back of the hand relative to the Source.
- three values describing the position of the hand. The Glove system returns the X, Y, Z coordinates of the Polhemus Sensor relative to the Source.

Both orientation and position values refer to the Data Glove Source, which represents the absolute reference system.

The actual position and orientation of the hand is not necessarily relevant when recognising a static pose.

4.1.2 Dynamic Gesture

Dynamic gestures are characterised by a sequence of poses performed over a particular trajectory. Dynamic gestures consider posture as well as movement of the hand.

Three types of gestures can be identified:

Gesture: a sequence of poses performed over a period of time.

Time-dependent Gesture: the same physical series of poses performed over differing time periods. The sequence of poses may be semantically different or semantically identical according to the context/application. For example, a slow side-swipe of the hand may indicate that an object is to be pushed to one side of the view; a faster side-swipe may indicate that it is to be pushed out of the view altogether.

Gesture Family: a sequence of poses can occur as a gesture in its own right, or as a part of a longer, more complex gesture. A particularly interesting case is when two or more gestures start with the same sequence of poses but end differently. Even if the shorter gesture A is recognised, the system must wait and see if the user is really performing gesture A, or if it is the first part of gesture B. Only if gesture B is not recognised can the system undoubtedly say that gesture A was intended.

4.2 Learning System

The Learning System comprises a module which allows users to define and store gestures. As the Glove - User's hand couple varies from user to user, each user has to teach the system their own gestures. Users can create their own directory in which their Calibration Table (CT) and Gesture Models (GM) are stored.

The stored data is then used by the Recognition System.

The Learning System consists of a module for the acquisition of the gestures and a module to collapse the acquired data.

4.2.1 Data Acquisition

Raw glove data values depend very much upon the physical characteristics of the individual wearer's hand. The system must learn the required gestures for that person before it can recognise them.

The system must:

- Ask the user the name of the gesture.
- Ask the user to make the initial pose of that gesture (repeat 3 times).
- Ask the user to make the final pose of the gesture (repeat 3 times).

- Calculate the average values of initial and final pose data.
- Ask the user to perform the full gesture. The system starts to learn the gesture when the initial pose is recognised.
- Store data related to the sequence of poses which define the gesture. This includes raw values of inner and outer finger joints, orientation values (pitch, yaw and roll) and spatial position relative to the position of the starting pose.
- Stop learning a gesture when its final pose is recognised.
- Collapse the acquired data.
- Store the data in the gesture model file.

The initial and the final poses are important as they delimit the gesture. The system starts to learn the gesture when the initial pose is recognised and stops when the final pose is recognised. For this reason, the acquisition of these poses has to be accurate, and the system asks the user to repeat them 3 times. (The choice of 3 repetitions was fairly arbitrary and was based on the original VPL demonstrations using a Macintosh. Three seemed a sufficient number of trials to obtain valid average values.)

Each acquired pose is stored using the C-structure:

```
struct Pose {
    char thumb1     [2]; /* minimum and maximum inner joint bending value */
    char thumb2     [2]; /* minimum and maximum outer joint bending value */
    char index1     [2];            "               "
    char index2     [2];
    char middle1    [2];            "               "
    char middle2    [2];
    char ring1      [2];            "               "
    char ring2      [2];
    char little1    [2];            "               "
    char little2    [2];
    int orientation [3]; /* orientation values */
    float position  [3]; /* position values */
    }
```

Joint Angles

For each finger joint the system calculates a minimum and maximum value. These values are calculated by adding and subtracting a fixed factor from the acquired data. To be recognised, the value for that bending joint has to be between those two extremes. If the fixed factor is too high, the recognition of the gesture could be inaccurate.

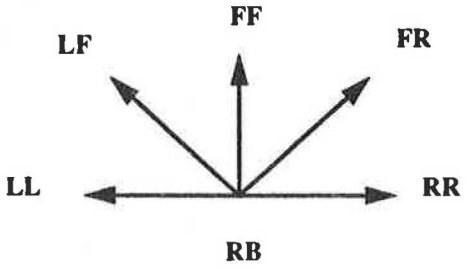
Orientation Values

The Glove System returns Euler angles (azimuth, elevation and roll) which describe the orientation of the hand. The Orientation values depend on the position of the Source of the Glove System. A table containing a correspondence between the glove orientation values and some symbols has been defined (Figure 5). The table has to be updated if the position of the Polhemus Source changes.

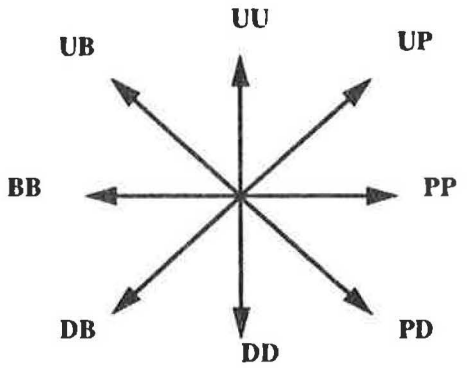
The names of the symbols are completely arbitrary, but as an example, the orientation codes describing the yaw, pitch and roll values of a flat hand are "FF PP NN" respectively.

FIGURE 5. Orientation

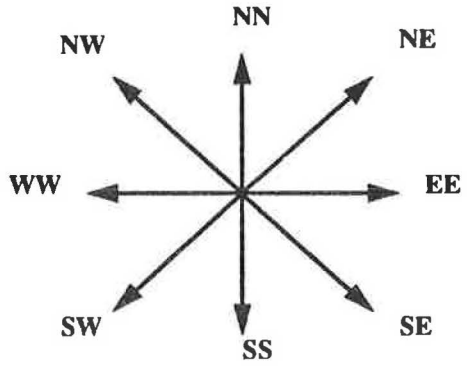
YAW (Euler Azimuth)



PITCH (Euler Elevation)



ROLL (Euler Roll)

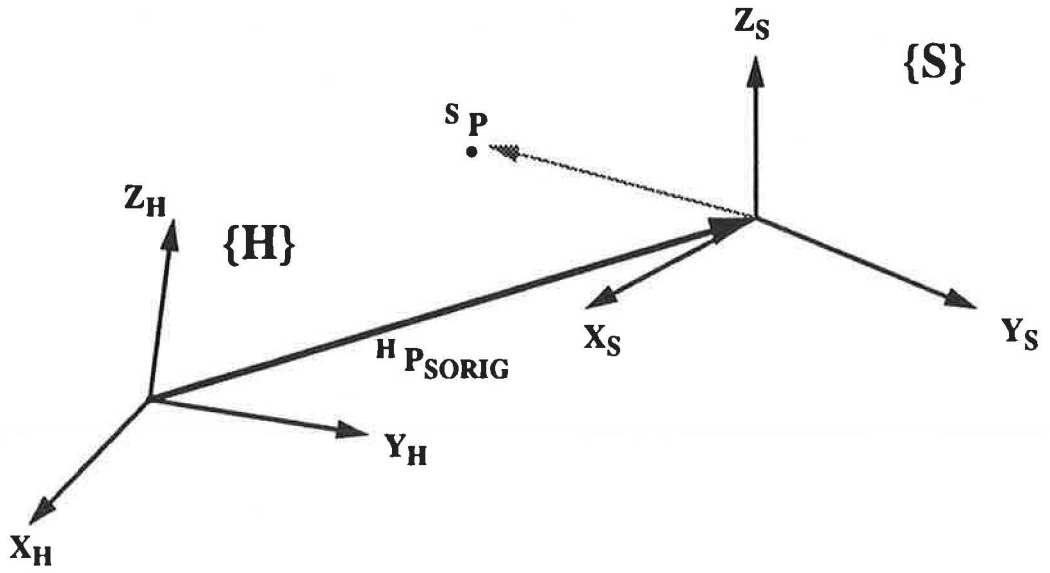


Position Values

The position of the hand at a certain point is measured relative to the position that the hand assumes at the beginning of the gesture. Each position is calculated by multiplying the read values by a rotation matrix and a translation vector. The rotation matrix uses the Z-Y-X Euler angles [11].

Let {S} be the Source frame and {H} the Hand frame (Figure 6).

FIGURE 6. Source and Hand frames



$^S P$ is the vector of the position of the hand relative to {S}.

$^h P_{sorig}$ is the vector containing the position values of the hand read when the original starting pose of the gesture is performed.

$^h_s R$ is the rotation matrix which operates on the Euler angles.

The transformation equation is:

$$^h P = {}^h_s R \, {}^S P + {}^h P_{sorig}$$

where $^h P$ is the position of the hand relative to the{H} frame.

4.2.2 Data Collapsing

In a typical gesture lasting a couple of seconds, up to *60 * (10 fingers + 3 XYZ + 3 ypr) = 960* data values (sampling at 30 Hz) can be captured. In order to try and simplify the recognition algorithm and to improve system performance, the number of poses read during the learning phase was reduced by joining two or more similar poses. The algorithm, called the *Collapsing Algorithm*, checks if two or more poses have a similar bending range and the same orientation. If so, it stores the minimum and the maximum average values (calculated over the collapsed bending values) for each finger joint, the orientation value, the average position of the hand, and the number of collapsed poses.

Figures 7 and 8 shows an example of how a set of acquired poses has collapsed into a single pose. The figures also show the structure of the files storing respectively the total set of poses (grip.gm) and the collapsed ones (grip.clp) of the acquired gesture.

Figure 7 shows a sequence of poses compounding part of the gesture *grip*. The initial pose, pose 1, cannot be joined with other poses and is copied directly into the collapsed poses file. The orientation of poses from pose 2 into pose 6 is the same and finger flexion values are similar, so these poses are collapsed in pose 2 of the grip.clp file. Pose 7 is not collapsed with the previous poses as its orientation is different.

FIGURE 7. Extract from the File Containing the Acquired Poses of the Gesture *grip*

```
GESTURE: grip

POSE: 1
        INNER THUMB:    62   93
        OUTER THUMB:    69   100
        INNER INDEX:    110   140
        OUTER INDEX:    33   63
        INNER MIDDLE:   34   64
        OUTER MIDDLE:   31   61
        INNER RING:     86   146
        OUTER RING:     7   69
        INNER LITTLE:   96   157
        OUTER LITTLE:   37   98
        ORIENTATION: FF   UP   NN
        POSITION:       77.921165       -5.564973       -49.608387

POSE: 2
        INNER THUMB:    60   90
        OUTER THUMB:    67   97
        INNER INDEX:    100   130
        OUTER INDEX:    31   61
        INNER MIDDLE:   34   64
        OUTER MIDDLE:   30   60
        INNER RING:     113   143
        OUTER RING:     19   49
        INNER LITTLE:   113   143
        OUTER LITTLE:   47   77
        ORIENTATION: FF   UP   NN
        POSITION:       77.752167       -5.925180       -51.259869

POSE: 3
        INNER THUMB:    59   89
        OUTER THUMB:    67   97
        INNER INDEX:    100   130
        OUTER INDEX:    31   61
        INNER MIDDLE:   34   64
        OUTER MIDDLE:   30   60
        INNER RING:     113   143
        OUTER RING:     19   49
        INNER LITTLE:   113   143
        OUTER LITTLE:   46   76
        ORIENTATION: FF   UP   NN
        POSITION:       76.367393       -4.996798       -51.440376
```

```
POSE: 4
                INNER THUMB:    59   89
                OUTER THUMB:    67   97
                INNER INDEX:    100   130
                OUTER INDEX:    31   61
                INNER MIDDLE:   34 64
                OUTER MIDDLE:   30   60
                INNER RING:     113   143
                OUTER RING:     19   49
                INNER LITTLE:   113   143
                OUTER LITTLE:   47   77
                ORIENTATION: FF   UP   NN
                POSITION:       76.066010       -4.879420       -49.916515
POSE: 5
                INNER THUMB:    59   89
                OUTER THUMB:    67   97
                INNER INDEX:    99   129
                OUTER INDEX:    31   61
                INNER MIDDLE:   34   64
                OUTER MIDDLE:   30   60
                INNER RING:     113   143
                OUTER RING:     19   49
                INNER LITTLE:   112   142
                OUTER LITTLE:   47   77
                ORIENTATION: FF   UP   NN
                POSITION:       75.935356       -5.052757       -50.509060
POSE: 6
                 INNER THUMB:   59   89
                OUTER THUMB:    66   96
                INNER INDEX:    99   129
                OUTER INDEX:    31   61
                INNER MIDDLE:   34   64
                OUTER MIDDLE:   30   60
                INNER RING:     113   143
                OUTER RING:     19   49
                INNER LITTLE:   112   142
                OUTER LITTLE:   46   76
                ORIENTATION: FF   UP   NN
                POSITION:       77.021179       -2.906705       -49.534473
POSE: 7
                INNER THUMB: 59  89
                OUTER THUMB: 66  96
                INNER INDEX: 99  129
                OUTER INDEX: 31  61
                INNER MIDDLE: 35  65
                OUTER MIDDLE: 30  60
                INNER RING:     112  142
                OUTER RING:     19  49
                INNER LITTLE:   111 141
                OUTER LITTLE:   46  76
                ORIENTATION:    FF   UP   NE
                POSITION:       76.584793       -2.374920       -47.473469
```

FIGURE 8. Extract from the File Containing the Collapsed Poses of the Gesture *grip*

```
GESTURE: grip
POSE: 1
                Collapsed Poses: 1
                INNER THUMB: 62   93
                OUTER THUMB:    69   100
                INNER INDEX:    110   140
                OUTER INDEX:    33   63
                INNER MIDDLE:   34   64
                OUTER MIDDLE:   31   61
                INNER RING:     86   146
                OUTER RING:     7   69
```

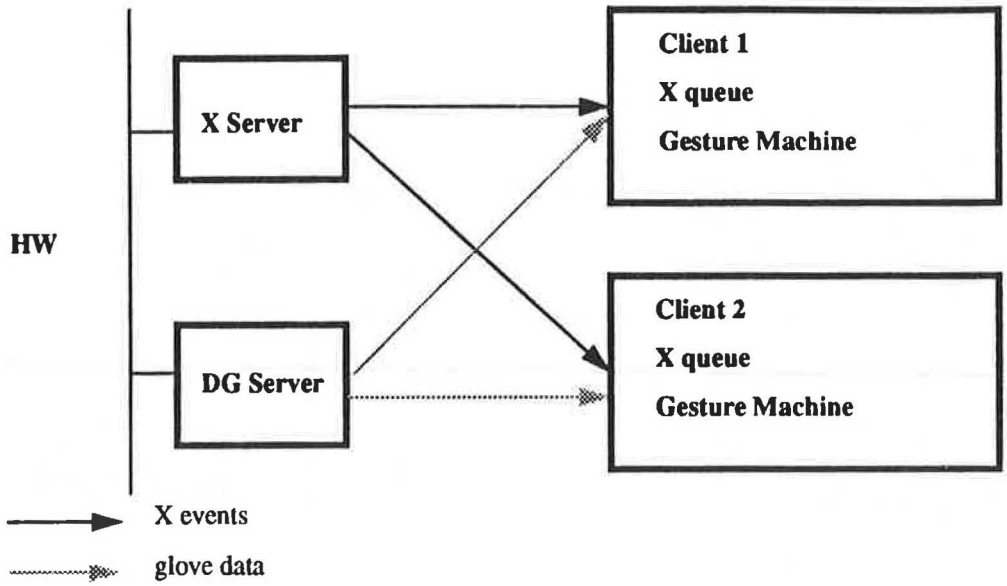
```
        INNER LITTLE:     96   157
        OUTER LITTLE:     37   98
        ORIENTATION: FF   UP   NN
        POSITION: 77.921165   -5.564973      -49.608387

POSE: 2
        Collapsed Poses: 5
        INNER THUMB:      59   90
        OUTER THUMB:      66   97
        INNER INDEX:      99   130
        OUTER INDEX:      31   61
        INNER MIDDLE:     34   64
        OUTER MIDDLE:     30   60
        INNER RING:       113  143
        OUTER RING:       19   49
        INNER LITTLE:     112  143
        OUTER LITTLE:     46   77
        ORIENTATION: FF   UP   NN
        POSITION: 76.635155   -4.009093      -50.052834
```

4.3 Recognition System

Because it is intended that the Recognition System should be integrated into the MIT X System, the architecture of the Recognition System is of the standard client-server model (Figure 9). Keyboard and mouse events are sent to the X Server, while DataGlove events are sent to the DataGlove Server (DG Server). Each client application has its own X queue and Gesture Machine. The client asks the X Server to inform the client about particular kinds of events, so it writes events into the X queue only if they have been requested by the application.

FIGURE 9. Client-Server Recognition System Architecture



The proposed Architecture of the Recognition module is shown in Figure 10. The main components of the system are:

- X Server (extended version)
- Data Glove Server
- Data Base
- Client Application

Their descriptions follow.

FIGURE 10. Architecture of the Recognition Module

4.3.1 X Server

The X Server allows the communication of input device values (keyboard and mouse) to the client application through an X queue. An extended version of the X Server [12] should also allow the communication of raw glove and gesture events coming from the glove (Figure 11).

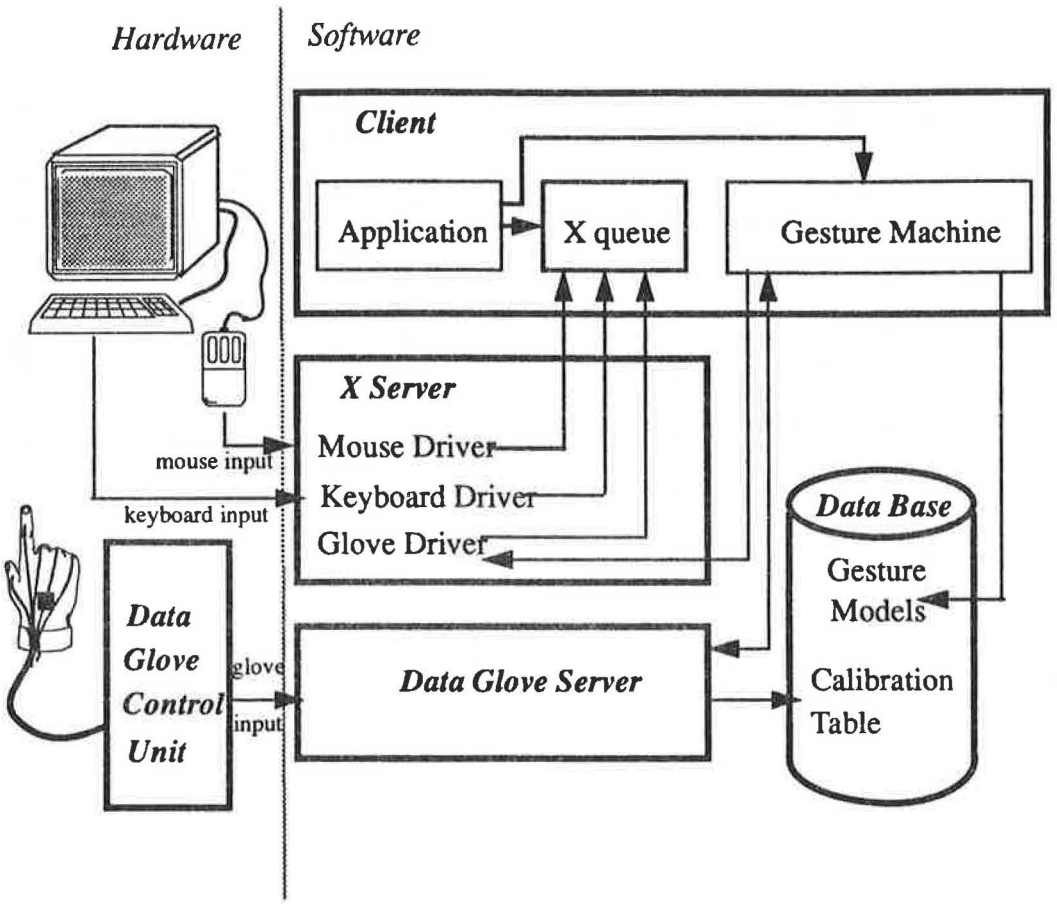
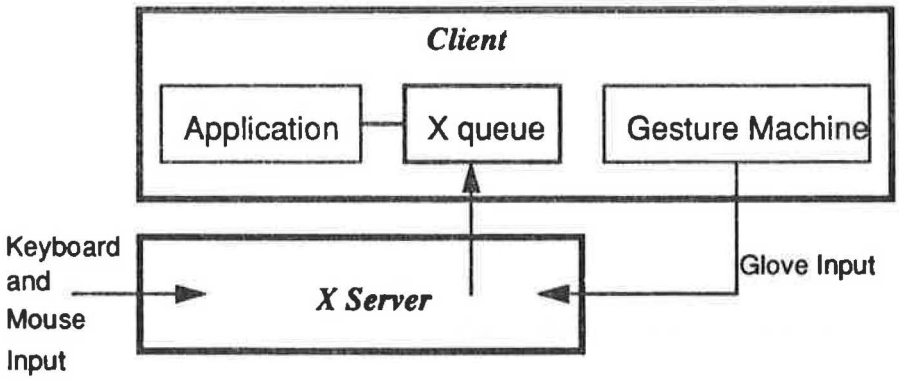


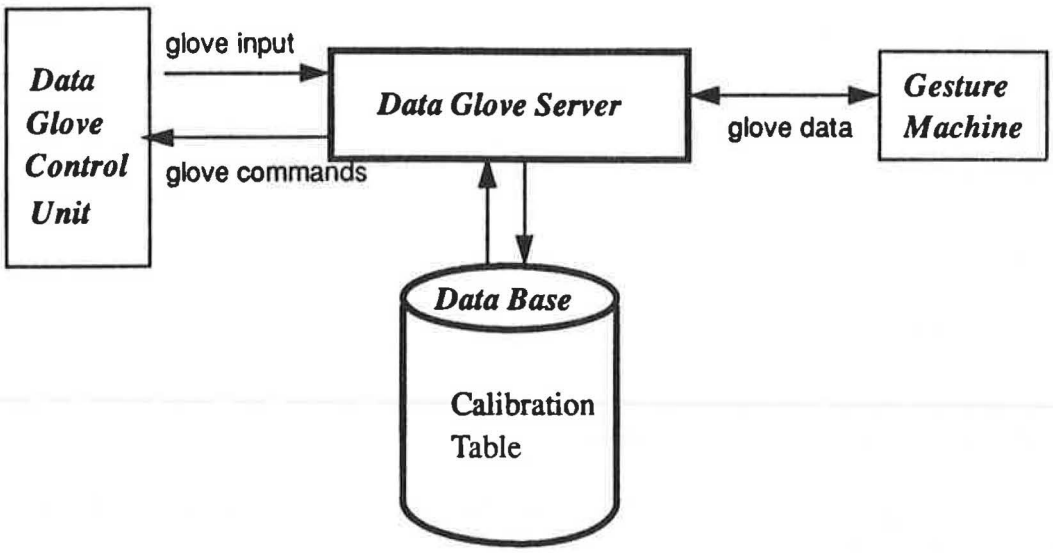
FIGURE 11. X Server extension



4.3.2 Data Glove Server

The structure of the Data Glove Server is shown in Figure 12.

FIGURE 12. Data Glove Server structure



The available sample frequencies of the Control Unit are 30 or 60 Hz. This means that data records are transferred between the Control Unit and the host computer 30 or 60 times a second. This speed is rather slow compared to that of a Unix workstation (the host computer), so that a reading loop can consume the sampled data as soon as they appear. However, if the host is busy performing other activities (such as providing graphics), then the input data just read cannot be guaranteed valid. In this case, the Server should sample the value, examine both the input data and the buffer and then send only the last valid sample to the Gesture Machine.

The functionality of the Data Glove Server is:

- Receive system configuration parameters from the Gesture Machines. Each Gesture Machine sends its user's name, which corresponds to the directory containing the Calibration Table for the user who is currently using the DataGlove system.

  Although more than one client-Gesture Machine may be active at any one time, there will only be one user wearing the glove controlled by the DG Server. Thus, one Calibration Table is loaded from the DataBase, and then downloaded to the Control Unit.
- Load the Calibration Table for a particular user from the DataBase.
- Download the Calibration Table to the Control Unit. The DG Server constructs the table of values to calibrate the glove for a particular user (the values have been loaded during the previous phase).
-
-

- Receive raw input values from the Control Unit. These consist of:
  - 10 finger joints, calibrated according to the Calibration Table
  - position relative to the Source Reference System (SRS)
  - Euler angles for orientation
- Send data received from the Control Unit to the Gesture Machine. Those are raw values.

4.3.3 DataBase

The system stores the acquired data in a DataBase. There are two kinds of data: Calibration Tables and Gesture Models.

This data is typical for each user and is loaded at run time. As the implemented system works with raw glove values (not calibrated) only the structure of the file containing the description of the Gesture Machine has been defined. It contains information about the number of collapsed poses, the finger joint flexion values, the orientation and the position values. An example of the format of the gesture file has been shown above in Figure 8.

[Ideally, in future the X Resource DB should manage the stored data [13]].

4.3.4 Client and Gesture Machine

A client consists of the application code, an X queue to receive events from the input devices (mouse, keyboard and glove) and a Gesture Machine. The Gesture Machine is described in detail in the following sections. The client characteristics which influence the Gesture Machine, the architecture of the Gesture Machine and the recognition algorithm are also described.

4.3.4.1 Client Characteristics

Different kinds of application may require different characteristics from the gestures they use. Such characteristics include:

Time-dependent gesture: in some applications, two gestures characterised by the same physical series of poses performed over differing time periods can have the same or different semantic meanings. An example of the latter case is when it is important to recognise if a gesture is emphasised (and hence is likely to be slower and more deliberate than normal), or to recognise if a gesture has physically deteriorated in accuracy (due to frequent use by the user). Moreover, the application can set the overall duration of time-dependent gestures.

Importance of the middle poses: in some applications, checking the correctness of all the middle poses, (i.e., all except the first and last), may be of no interest. In other cases, the entire sequence of poses has to be checked. An example of the first case is when a gesture is used to simulate the arm of a robot picking up an object and moving it to another place. The application has to know the initial pose (picking up the object) and the final pose (releasing the object) of the entire gesture, but does not need to know about the sequence of poses in the middle.

Ring and little finger accuracy: the movement of the ring and little fingers often follows the movement of the index and the middle ones. Since these movements are dependent and imprecise, it may be better to ignore them entirely. In some applications (such as a remote sensing application controlling the arm of a robot) the position of these two fingers could be ignored if the robot was equipped with only three "fingers".

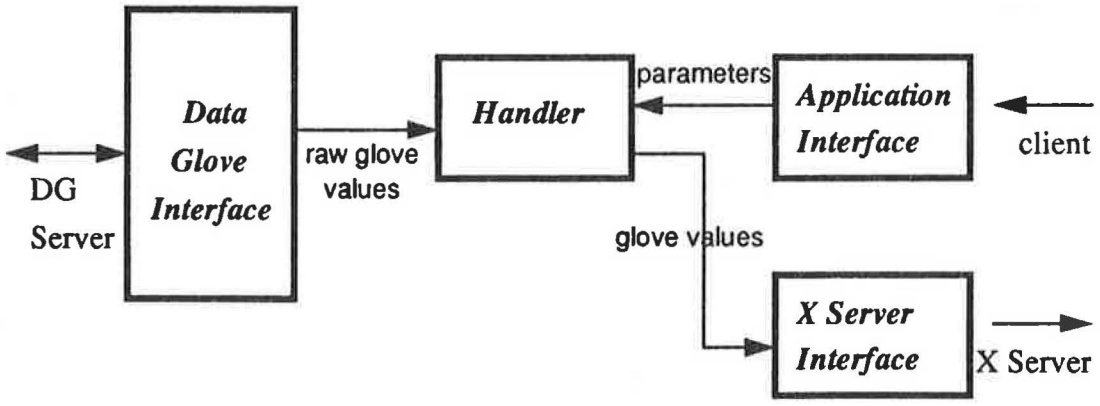
Trajectory: sometimes, it is sufficient to detect just the orientation and trajectory of the hand, finger bendings are irrelevant. An example of this is the use of a scalpel in a surgical operation. The surgeon can cut into a piece of skin giving the cut different shapes and directions each time, but his basic grip on the scalpel remains the same.

Confidence factor: the application might use a wide vocabulary of gestures in which many gestures are similar to each other. This implies that this parameter, indicating the percentage of recognised poses over the total number of poses, has to be high.

4.3.4.2 Gesture Machine Interface

The architecture of the Gesture Machine is shown in Figure 13, and a brief description of the functionality of each component follows.

FIGURE 13. Gesture Machine architecture



X Server Interface

- Calculates the values to be sent to the X server by manipulating the raw glove values. In particular, it calculates the coordinate transformations and whether the position of the hand can be mapped within the screen and mapped within a window.
- Converts raw glove data values to glove events.
- Fills the X queue structure with this data and sends it to the X Server.

Application Interface

- Receives the application parameters from the client application. The client can call application functions to set the parameters.
- Receives the structure containing the names of the gestures to be loaded.
- Receives the user-name used by the DG interface to set up the appropriate Calibration Table.

DataGlove Interface

- Sends to the DG Server the user-name to download the appropriate Calibration Table.
- Receives raw glove values from the Data Glove Server and sends them to the Handler

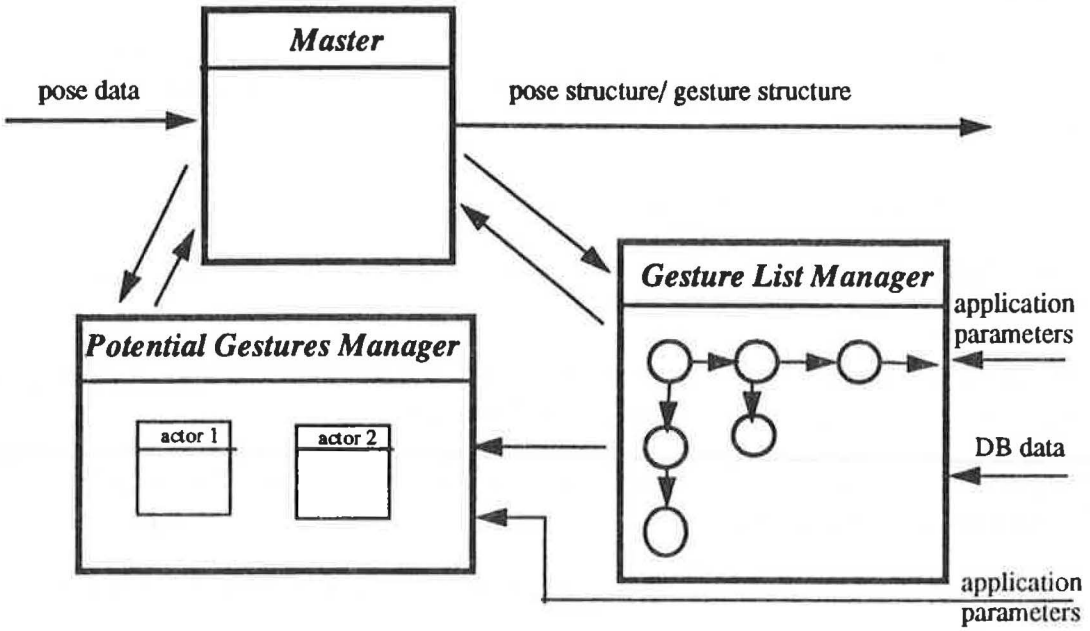
Handler

The handler manages the data coming from the other modules and implements the Gesture Recognition Algorithm. Due to the nature of the problem, it seems that a good solution for the implementation of the handler is to adopt an object-oriented approach.

4.3.4.3 Recognition Algorithm

The structure of the Handler, which implements the Recognition Algorithm, is shown in Figure 14.

FIGURE 14. Handler Structure



The objects which comprise the Handler are:

Master

1. receives DataGlove data from the DataGlove interface
2. sends a message to the Gesture List Manager containing pose data if no gesture is currently being recognised
3. sends a message to the Potential Gesture Manager containing pose data if a gesture is currently being recognised
4. sends DataGlove data to the X Server interface

Gesture List Manager (GLM)

1. loads gesture models from the DataBase
2. manages a list of gestures
3. checks if a pose is the initial pose of one or more of the gestures
4. sends a message to the Master containing information about active gestures
5. sends a message to the Potential Gesture Manager containing the list of gestures to activate

Potential Gestures Manager (PGM)

1. creates an *Actor* for each active gesture
2. executes a gesture recognition algorithm for each Actor
3. sends messages to the Master containing the list of Actors still alive

Algorithm Parameters

The application may set up different parameters to indicate the characteristics of the gestures used in the application (as described earlier), and to set up various parameters of the algorithm. These are as follow:

Parameters to define Gesture Characteristics:

Extensibility: the maximum extension to the duration of a gesture; used in the recognition of time-dependent gestures.

Middle Poses: whether just the first and last poses should be considered, or the middle poses as well. (This parameter is not yet implemented.)

Ring and Little Finger Accuracy: this specifies whether or not these two fingers should be considered by the algorithm.

TrajectoryError: the permitted deviation of the actual trajectory from the model, expressed in centimetres.

Confidence Factor: the percentage of poses that must be matched so that the performed gesture is recognised with greater or lesser accuracy.

Internal Algorithm Parameters:

Feedback: the application can graphically allow the user to visualise the evolution of the gesture as it is recognised.

Pose To Check: the number of (collapsed sets of) poses ahead of the current one which the algorithm may look at if the current pose is unrecognised. This value is typically set to the next 2 (collapsed sets of) poses.

Consecutive Mismatches: the number of mismatched poses of an entire gesture (as a percentage calculated over the total number of poses of the model) over which the gesture can not be recognised any more.

Input Errors: sometimes the data sent by the Control Unit are wrong. Sometimes, due to a communication problem between the Control Unit and the host computer, the Leading Byte values (which controls the communication flow) returned is not the expected one; another reading has to be done. In this case the system may lose data and totally disrupt the recognition procedure. The algorithm skips a number of model poses directly proportional to the amount of error data received. The parameter Input Errors sets the maximum number of consecutive errors accepted before the recognition of the gesture gives up.

Flexion Range: range of values within which the finger bending values must fall. (It is used in the Learning module.)

Algorithm Description

Terminology used in the algorithm description is defined as follows: an *input pose* is a single pose performed by the user when describing a gesture; a *model pose* is a single pose of a gesture model defined by the user during the Learning phase. An *Actor* is the object associated with each active gesture, and a *pointer* is the object which addresses the current pose of the gesture list.

The algorithm loads the gesture models from the Data Base and constructs a data structure. Each model is described by a sequence of collapsed poses.

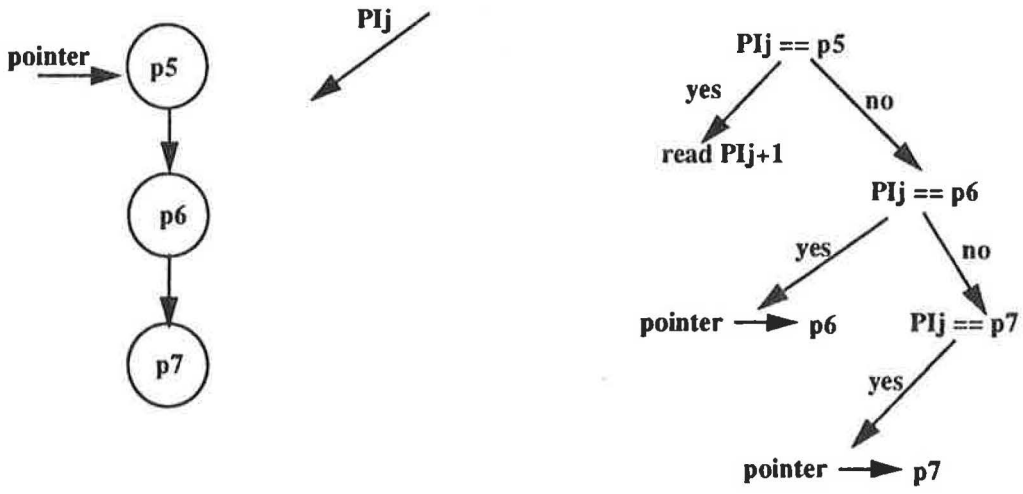
Then, the algorithm starts reading the user's input poses. For each of them, it checks if the input pose matches the starting pose of one or more of the gesture models. If a match occurs, the corresponding gesture is called an *active gesture* (a gesture that might be recognised). An *Actor* object is created for each active gesture. An *Actor* keeps the history of the gesture, and a pointer stores the current expected pose.

The algorithm now checks if the following input pose matches the current pose of the model. If it does, it reads the following input pose. If it does not, it checks if one of the following "n" poses is matched, where "n" is determined by the *PoseToCheck* parameter. If one of these matches, the *pointer* is moved to that model pose (Figure 15).

The algorithm stores the number of match and mismatch events for each pose.

In the case in which a pose is matched, the *pointer* is not moved along the gesture list. The algorithm checks if more incoming input poses match the current pose.

FIGURE 15. Example of the Execution of the Algorithm



PoseToCheck = 2

where:

PIj: input pose
p5, p6, p7: model poses

The algorithm waits for the same pose for a time equal to the number of collapsed-poses plus the Extensibility parameter. This means that the performed gesture could be faster or slower than the one performed during the Learning activity.

If no input poses match the current pose, this latest is marked as an *unrecognised pose* and the Pointer is moved to the following pose of the model.

When the Algorithm Gives up

The algorithm gives up recognising a gesture when:

- the number of consecutive mismatched poses is higher than *Consecutive Mismatch* parameter value.
- the number of wrong input data is higher than *Input Data Error* parameter value.

How a Gesture is Recognised

The algorithm stops when it reaches the final pose of one or more active *Actors* (say, "Rec-Actor").

It calculates, for each "Rec-Actor":

- Confidence value: the number of matched collapsed-poses over the total number of collapsed-poses. If the value is higher than *Confidence* parameter, the gesture is recognised ("Rec-Actor"), otherwise it is unrecognised ("Unrec-Actor").
- Recognised poses: the number of *input poses* over the number of model poses. If this value is not approximately 1 and the *Flexibility* parameter was set to off, then the gesture is unrecognised.
- Trajectory (if the *TrajectoryError* parameter is on): the algorithm checks if the trajectory of the performed gesture is similar to that of the gesture model. If it is, the gesture is recognised, otherwise it is unrecognised.
- The algorithm asks each "Rec-Actor" for its successful lifetime duration. From all these values, it chooses the highest one, which identifies the recognised gesture. The algorithm returns information about the recognised gesture name, flexibility percentage and confidence level percentage.

5. Results

Before this project began, it was apparent that the problems of reliable gesture capture and recognition were quite considerable. Only as the project progressed could the various problems and issues be identified and various solutions suggested. Due to the short duration of the project, it was not possible to explore all of these issues in detail so instead a more general approach was taken and a feedback mechanism provided whereby future exploration and data could be studied in more detail.

The work progressed in three main stages. The first stage was essentially to develop the algorithm and the Learning and Recognition modules. Obviously, the majority of gestures taught at this stage were unrecognised. It soon became apparent however that a better method of gesture visualisation was required, so the second stage of development was to design and implement a graphical feedback mechanism. This displayed both the expected and actual values of a gesture and could be used to study (in conjunction with raw data values) the successes and failures of the Recognition system in more detail. This feedback mechanism is described in more detail in the next section. The third phase of development looked at ways in which the parameters of the Recognition algorithm could be refined with greater or lesser success. A description of this process is also given below.

5.1 Graphical Feedback

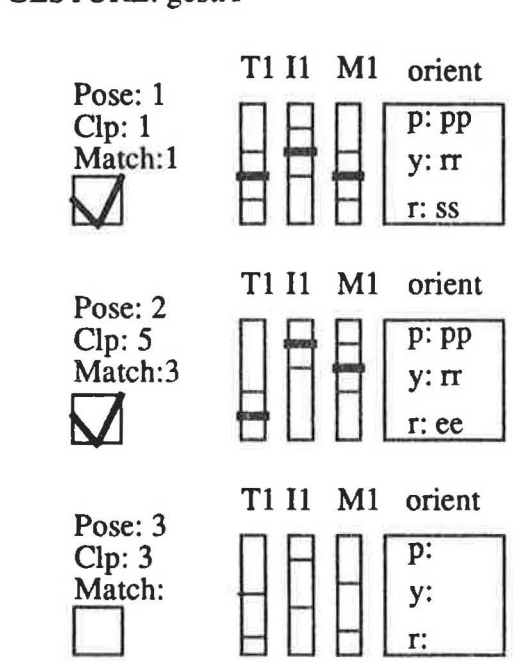
If the user requires feedback, the appropriate parameter in the Recognition System (Parameter.Feedback) should be set to "ON". The feedback system is displayed as a series of sets of slide bars, an orientation panel and a success/failure indicator (see Figure 16). To save space, values for just one joint of the Thumb, Index and Middle fingers are given.

When the Recognition system is running, but before the start of any recognised gestures, the system dynamically displays the actual values of the finger bends as a thick horizontal bar in each of the finger slide bars. Codes for the actual orientation are displayed in the "orient" panel. When the first pose of one or more gestures is recognised, the range of expected values for each set of collapsed poses over the whole gesture are indicated as a pair of lines on each finger slide bar. The expected orientation values are also displayed alongside the actual values. The user can now clearly see whether or not their fingers and/or hand are within the required limits of the pose.

As each set of collapsed poses is successfully recognised or deliberately skipped by the Recognition process, the success/failure indicator box is ticked or left blank as appropriate. When the entire gesture is recognised, an appropriate message is displayed at the bottom of the window giving the name of the recognised gesture and the confidence level achieved. If the recognition process is abandoned for some reason, a message to that effect is given instead. In both cases, the display remains on the screen so that the data can be analysed.

This feedback system is based on that used by VPL on the Macintosh. Time permitting, it would be interesting to explore these issues further and provide instead a more graphical visualisation of the feedback, perhaps by drawing iconic versions of the expected poses of the hand and superimposing the current, actual pose. Some additional thought would have to be put into the representation of the hand's position in 3D space.

FIGURE 16. Graphical Feedback of the on-going Algorithm



5.2 Refinement of the Algorithm's Parameters

As the experimentation and results capturing phase commenced, the glove hardware became highly unstable; even standard test programs gave erratic results not only during the operation of the glove but also in initialising the hardware. Therefore, the results described below can give only a partial set of results, and consequently tentative conclusions. It is intended that more exhaustive tests be performed when the hardware is stabilised again.

Reproducibility of gestures

It has already been noted that humans find it extremely difficult to reproduce exactly a series of hand movements. As an example, a simple gesture (fist to flat hand) was performed twice in succession by an experimenter deliberately concentrating on making the two instances of the gesture as similar as possible.

Global parameters to the Recognition System (Figure 17) and extracts of the two data files produced are given below (Figure 18). Note that since all the fingers go from one (clenched) extreme to the other (extended), finger bendings can be reproduced fairly precisely. (The aberrant value of the outer index finger is probably a hardware glitch.) Had the gesture involved any part-way finger bends then more differences would have occurred. Note also that although positions within the same gesture remain relatively similar with respect to the starting position, they differ markedly between the two instances of the gesture, as does the pitch orientation of the hand. This when the experimenter was deliberately trying to recreate the same gesture.

FIGURE 17. Global Algorithm Parameters

Ring Accuracy = OFF
Little Accuracy = OFF
Confidence = 0.70
Trajectory Error = 10.0
Number of Consecutive Mismatches = 0.20
Extensibility = 0.30
Input Errors = 10

FIGURE 18. File Data

```
GESTURE: gestA-1
POSE: 1
        INNER THUMB:  17   48
        OUTER THUMB: 24   64
        INNER INDEX:   10   44
        OUTER INDEX:  -3   32
        INNER MIDDLE: 54   85
        OUTER MIDDLE: -2   28
        INNER RING:    -2   62
        OUTER RING:    -21   42
        INNER LITTLE:  -7   54
        OUTER LITTLE:  -14   47
        ORIENTATION:  FF  PP  NN
        POSITION: 90.946426  77.199387  58.279907
```

```
POSE: 2
                Collapsed Poses: 2
                INNER THUMB:  15   47
                OUTER THUMB: 23   56
                INNER INDEX:   10   40
                OUTER INDEX:  9   39
                INNER MIDDLE:  53   84
                OUTER MIDDLE: 5 28
                INNER RING:    12   44
                OUTER RING:    5   27
                INNER LITTLE:  9   39
                OUTER LITTLE:  1   31
                ORIENTATION:  FF   PP NN
                POSITION: 91.001877   77.563049   57.885597

GESTURE: gestA-2
POSE: 1
                INNER THUMB:  17   48
                OUTER THUMB: 26   60
                INNER INDEX:   8   41
                OUTER INDEX:  -2   31
                INNER MIDDLE:  58   89
                OUTER MIDDLE: -2   29
                INNER RING:    2   62
                OUTER RING:    -18   46
                INNER LITTLE:  -5   56
                OUTER LITTLE:  -14   48
                ORIENTATION:  FF UP  NN
                POSITION: 119.009743   33.657909 -38.894096

POSE: 2
                Collapsed Poses: 1
                INNER THUMB:  17   47
                OUTER THUMB: 23   53
                INNER INDEX:   10   40
                OUTER INDEX:  1   31
                INNER MIDDLE:  57   87
                OUTER MIDDLE: 5 28
                INNER RING:    15   45
                OUTER RING:    5   27
                INNER LITTLE: 10   40
                OUTER LITTLE: 2   32
                ORIENTATION: FF   PP NN
                POSITION: 119.950989   35.371265   -39.335354
```

Gestures involving orientation only

The Learning System was taught two gestures involving a change in orientation only: the first (the "gate" gesture) was to move a vertical hand with straight fingers though 90-degrees, as though illustrating the opening of a gate; the second, (the "ball" gesture) was as though the experimenter was clenching a ball then moving the hand through 180-degrees from the downwards position to the upwards position.

The gate gesture was performed five times, and successfully recognised each time. The ball gesture was also performed five times, but recognised only twice. The results are given below:

Gate Gesture

1. 20% slower than taught, 100% confidence of recognition
2. 17% slower " "
3. 17% faster " "
4. 14% slower " "
5. Same speed " "

Ball Gesture

1. Failed
2. 52% slower than taught, 70% confidence of recognition
3. Failed
4. Failed
5. 42% slower than taught, "

It would be interesting to explore the reasons why the gate gesture was performed with differing speeds.

Note that the apparent reduced confidence on the ball gesture is due to the fact that this gesture was longer than the gate gesture. The results for the ball gesture appear to be highly unreliable. It is thought that this is because the hardware was failing. Subsequent test with the gate gesture for a second time also produced highly unstable results. No conclusions can be drawn, other than that the hardware is failing.

Gestures involving finger bends only

The Learning System was taught two gestures involving a change in positions only: the first (the "trigger" gesture) kept the hand in a vertical position with the thumb pointing upwards and the Index and Middle fingers going from straight to bent, as though pulling an imaginary trigger; the second (the "thumb up" gesture) was a fist, but moving the thumb only from the clenched to vertical position. The trigger gesture was performed and successfully recognised three times (see the data below), the thumb up gesture was performed seven times, but on each occasion was either unrecognised or resulted in an orientation error.

Trigger Gesture

1. 83% slower than taught, 100% confidence of recognition
2. 83% slower " "
3. 33% faster " "

Previous experience seems to point to the fact that the algorithm tends to fail more often when recognising gestures in which the finger bending values change rapidly. Sometimes this appears to be due to the inaccuracy of the algorithm, at other times to the lack of valid data received from the Control Unit. The latter is perhaps a problem with the hardware and/or RS232 driver performance. The same problems sometimes occurred when the algorithm was checking values for the Ring and Little fingers. Unfortunately it was not possible to analyse this further.

Final note

At this point, due to the lack of time and apparent overwhelming problems with the glove hardware it was reluctantly decided to discontinue experimentation.

5.3 A New Gesture Window Manager

Following refinement of the Learning and Recognition systems within a test harness, a small but "real" application was built. The application chosen was a gesture-based interface to a window manager (as had been implemented previously under a different windowing system). Several gestures were set up and successfully used to iconise/open windows, and to move them around on the screen.

While this application does not further the interface to the window manager, it does provide "proof of concept" that such interfaces could be used to great effect within the X environment.

It would be useful to use the Recognition Algorithm in the acquisition phase for testing the accuracy of the taught gesture. After the user has performed a gesture, the system could ask them to repeat it several more times. The application of the Recognition Algorithm to the acquired data allows the system to check if the gesture has been performed accurately and to adapt the data acquired in the first gesture performance according to the average values calculated in the following acquisitions.

7. Acknowledgements

I would like to thank Dr. Victoria Burrill, Dr. Ken Robinson, Damian Mac Randal and Ian Wilkinson for the helpful and constructive ideas they gave me during my research work.

In particular, many thanks to Toria (Dr. Victoria Burrill) who helped me in experimenting with the implemented algorithm, writing Paragraph 5 of this Report and translating my Italian English into her much better Berkshire English.

8. References

[1] *DataGlove Model 2- Operation Manual*, VPL Research Inc., August 25, 1989.

[2] *3 Space user's manual*, Polhemus - A Kaiser Aerospace & Electronics Company, May 22, 1987.

[3] *DataGlove Test & Calibration Software For the Apple Macintosh - Operation Manual*, VPL Research Inc., March 6, 1989.

[4] Prime, M., *Human Factors Assessment of Input Devices for EWS*, RAL Report 91-033, 1991.

[5] D. Weimer, S.K. Ganapathy, *A synthetic Visual Environment with hand gesturing and voice input*, CHI'89 Proceedings, 1989, pp 235-240.

[6] Takahashi, T., Kishino, F., *Hand gesture coding based on experiments using a hand gesture interface device*, SIGCHI Bulletin April 1991, vol. 23, n. 2, pp. 67-73.

6. Conclusions and Future Work

The report has described the structure and implementation of a system for dynamic gesture recognition developed at the Informatics Department of Rutherford Appleton Laboratory. The system has been developed using a VPL DataGlove 2 connected to a Sun workstation running MIT X Window System. This work is not exhaustive but represents a first attempt to give a partial solution and some insights into the non-trivial problem of hand gesture recognition.

The designed architecture of the system goes part-way towards integration under the X Window System so that gestures recognised by the system can be treated as X events from the application. Due to lack of time, the required X extensions have not yet been done. The gesture recognition algorithm has been implemented and tested, and some graphical feedback of gestures has been provided. Finally, some comments on the behaviour of the algorithm have been noted and commented upon.

This work should make a good starting point for continued work in this area. Some ideas for future work are given below.

The algorithm requires a formal evaluation in order to identify its strengths and weaknesses. A series of tests should be performed by a variety of users under different conditions and different algorithm parameter values. Whenever the algorithm does not work, these tests should help identify if this is due to the algorithm itself, due to the glove hardware or if it is related to the inability of humans to reproduce movements with robot-like accuracy!

It would be very interesting to develop a method of feedback of the state of the hand and of the state of the recognition algorithm so that the user could immediately perceive what is going on. This might be a 3D representation of the hand, but this requires the use of sophisticated visualisation techniques which display the movement of the hand in real-time. A solution to visualise both the expected gesture and current gesture would be desirable.

Some applications should be developed to demonstrate the performance and the various application fields of the system. The application should require 3D gesture input to take advantage of the full capability of the glove. It would also be interesting to explore the use of the glove in creating "meta gestures". Such gestures could be created in two ways. Firstly, a "shorthand" gesture could be used to represent a frequently used, but more complex, set of gestures. Secondly, a "macro" gesture could be used to indicate that the system should perform <this gesture> repeatedly until <this condition> is met. (This is the virtual screwdriver concept mentioned earlier.)

The theoretical overall architecture of the Recognition System should be implemented, integrating the system under X and extending the X Server. Moreover, due to the structure of the Recognition Algorithm, a good solution would seem to be to implement it using an object-oriented programming language (for example, C++) instead of the C language adopted for the current implementation.

[7] Murakami, K., Taguchi, H., *Gesture recognition using Recurrent Neural Networks*, ACM 1991, pp. 237-242.

[8] Prime, M., *Hand tracking devices in complex multi-media environments*, to be published.

[9] Wilkinson, I., Prime, M., Goswell, C., *A louder squeak: enhancing the desktop with gesture*, to be published.

[10] Cardelli, L., Pike, R., *Squeak: a Language for Communicating with Mice*, SIGGRAPH '85, Computer Graphics 19, 3, July 22-26 1985, ACM, pp. 199-204.

[11] Craig,J.J., *Introduction to Robotics - Mechanics and Control*, Addison-Wesley publishing company.

[12] *Definition of the Porting Layer for the X 11 Sample Server*, MIT.

[13] *Xlib - C Language X Interface, MIT X Consortium Standard*, manual.