# Hand gestures as a method of interacting with Virtual Reality

**by**

G. Shaw

Submitted in partial fulfilment of the
requirements for the degree of
Bachelor of Science (Honours)
of Rhodes University

November 1998

## Abstract

This thesis describes a simple efficient feature based gesture recognition system using Polhemus trackers. The intention is to stay away from neural nets, and to use very simple recognition methods. The features used are chosen for their ease of calculation. The gestures that are used to test the system are a set of six that can control a paint-box application. Two recognition methods were investigated, a solid cutoff system that proves to be very rigid and unforgiving of mistakes and an error minimisation method that is very successful. The values used by these recognition methods are taken from multiple tests of the system.

The results show that on average 81% of the gestures are recognised correctly, with a few that are mis-classified due to similarities between gestures. Some solutions to this are outlined in the discussion of the results.

This thesis describes a viable solution to gesture recognition and shows that it is possible to have a gesture recognition system that is both fast and accurate.

# Contents

## Chapter 1 : Introduction



## Chapter 2 : Background information

## Chapter 3 : System



## Chapter 4 : Design



## Chapter 5 : Development

# Chapter 6 : Results



# Chapter 7 : Conclusion

# Chapter 1 : Introduction

Hand gestures are one of the most natural means of conveying information, after speech. Gestures are often used to add meaning to the spoken word, by emphasising a point or indicating the subject of the spoken sentence, amongst others. In Virtual Reality, information about the user's intentions must be conveyed to the computer. This must be carried out in as simple a way as possible. The user should not at any point be confused as to the control of the system.

## 1.1 Statement of problem

The system described here uses hand gestures to convey commands to the computer. Gesture based command systems are normally complicated and the code is difficult to modify and maintain. The gestures are also often hard coded into the system making a system difficult to adapt for a different application to the one for which it was designed. This system takes a feature based approach at gesture recognition and attempts to simplify gesture recognition and to make both gestures and features replaceable. The system is also designed to be real-time, which limits the complexity of the recognition algorithms. A side effect of this is that this system will provide some evidence on whether or not accurate gesture recognition can be done with very simple algorithms.

## 1.2 Interacting with Virtual Reality

The method of interacting with virtual reality is dependent on the application that is being run. For any application that requires direct manipulation of the environment, a glove is a good

controller, as the user can pick up, move and examine virtual objects as they would in the real world. If, however, the application requires the user to give other types of commands to the system some method must exist for the user to 'enter' the commands. One option is to have a VRMenu floating in the world that the user can interact with. The menu acts just like any normal object in the scene, except that some command would be executed when the menu is interacted with. The other problem with this type of design is that, unlike 2D desktops, no strongly defined metaphors exist within virtual reality. This would prevent wide scale acceptance of virtual reality applications when VR becomes available to the public. Whereas, in just about any windowing system on the current market, a menu can be clearly identified, as can a dialogue box, a scroll box, a mouse pointer, etc, in virtual reality there is no such similarity. One system may implement a menu as a box that floats in front of the user, another may represent it as a scroll that a user must 'unroll' before using. Other widgets may be equally varied between systems. The ideal development for VR would be a method of interacting with the system that requires few, if any metaphors to be understood by the user, a method that could be used for multiple different types of application. It is to this end that many Virtual Reality groups are looking at hand or body gestures as a method of interacting with the world**.**

## 1.3 The possibility of gesture-based control

Hand gestures are an often overlooked method of communication. Sign language is often considered the main, if only use of hand gestures in communication. Despite this perception, hand gestures are an important part of normal communication. Communication with a computer is normally limited to typing at a keyboard and moving a mouse. Recently voice-recognition technology has allowed users to speak to the computer with a fairly good chance of the computer understanding the command. Since the primary means of communication is

now (at least partially) understood by computers it is a logical extension to allow a computer to understand hand gestures.

Gestures have the disadvantages of being continuous, hence it can be non-trivial to determine where one gesture starts and ends, and difficult to identify. In their favour is the fact that gestures can be easy for users to learn, providing they are similar to gestures used in real life. Gestures can be used to completely control a system with little to no dependence of any other form of input. In a well designed system, in which the gestures are chosen carefully, the user may require little to no learning time. This is desirable in the computer industry at the moment where there is a drive toward ever greater productivity.

The development of the next great wave in computing is difficult to foresee but virtual reality is one of the next paradigms . Before this can happen some developments must occur. Desktop graphics must become able to handle virtual reality and the interface devices must become easy to use and understand. The success of this system will prove the effectiveness of gesture recognition as an interface method.

# Chapter 2: Background Information

Gesture based input systems are not a new idea. A primitive gesture based input experiment occurred in 1979 (as mentioned in [1]). Gesture based input devices have become popular research areas recently because of virtual reality. The idea of being able to control an entire world with one wave of a hand has a certain melodramatic feel about it.

This chapter will examine some interface devices used for computers in general, and comment on the applicability of the devices for virtual reality. Also some work that has already been done on gesture-based inputs will be examined.

## 2.1 Definitions

**Feature**: A property of a gesture that can be easily identified and assigned a value that can be used to help identify the gesture

**Gesture**: The movement or position of the hand in space. A movement that conveys information or a command to a listening device.

## 2.2 Existing control mechanisms

One problem that exists with virtual reality is that the peripherals which have been used to control normal computer applications for years have become inappropriate. Virtual reality is meant to be an application that conveys to a user the illusion of being part of an artificial environment [2]. Because of this desire to simulate a real environment, standard computer

interfaces are no longer the most appropriate devices to control the computer. While some devices can be adapted to control some applications, it is more appropriate to create new devices to control a new computing environment. To discover what is required of any new device, the limitations of existing devices should be examined.

**The keyboard**

There are two disadvantages to using a keyboard in a virtual environment. The first is that, typically, the user cannot see the real world due to a VR headset, and the second is that using a keyboard requires (in most cases) that the user can see it. Thus, without creating a virtual keyboard which precisely matches the real one in location and appearance, the keyboard is of little use in controlling a virtual environment.

**The Mouse**

The mouse has different problems to the keyboard when it comes to interacting with a virtual environment. The mouse does not require the user to be able to see it in order to use it, however, the mouse is a two dimensional control device designed to move a pointer across a flat desktop. While movement in three dimensions is possible and at least one 3D mouse exists, a 3D mouse suffers from the problems that the widgets so common on desktops are not standard across virtual reality systems and so the point-and-click use of the mouse may not be sufficient to control a virtual reality application.

**The Glove**

The glove is a natural choice for a virtual reality controller as we use our hands to interact with the real world every day. The glove is especially good for applications that require the

user to manipulate objects in the virtual world. The only problem here is the lack of tactile feedback. It can be quite disconcerting to see an object in one's hand but not to be able to feel it. The applications that gloves excel at is indicating an object within the virtual world by means of pointing. When a user points at an object, the system casts a ray from the pointing finger and selects the first object that it intersects. When it comes to giving other types of commands to the system, a system of hand gestures is usually designed, each gesture representing a command to the system. The gestures are made up from one or more of the following: hand shape, hand position relative to the body, hand movement over a short period. The disadvantage of this is that the gestures can be complicated if there are to be many commands.

**Video camera**

The video camera is a common method of providing a wire-free control system for virtual reality. The camera is positioned so that it captures the hand position, facial expression, body posture, etc. The image is then fed through edge enhancers and other tools to extract a portion of the picture. The gestures or expressions are then analysed. The drawback with video systems is that the user cannot turn around, unless there are multiple cameras linked together, which can prove expensive. The edge enhancing algorithms can also be time consuming, which combined with limited data rate, limits the use of cameras in applications that need to be strongly real-time (tolerating very little lag).

**Position trackers**

Position trackers are small devices than calculate their position and orientation relative to a fixed origin. They are generally electromagnetic in nature, using the nature of near-field electromagnetic waves to calculate spacial position. Alone, trackers are of limited use in

controlling a virtual world, but coupled with software, a couple of trackers can become a controller for an application. Controlling methods vary from activating menus by 'colliding' with them to natural gestures specifying what the world is to do.

## 2.3 Background

There appears to be very little work that is or has been done using position trackers to implement a gesture-based interface. The main background to my work is a paper by Dean Rubine [3] in which he described a single-stroke gesture recognisor used in a toolkit called GRANDMA. Also significant is a project, called JOVE, whose aim is to recognise infantry arm signals using three Polhemus trackers.

### 2.3.1 GRANDMA

The gestures used in the GRANDMA system are single-stroke, 2D hand movements entered with a mouse or stylus. The gestures specify the operation to be performed, the operand and possible additional parameters (eg for a movement command, the new location can be specified). While the GRANDMA toolkit and applications thereof are of limited interest, the statistical single-stroke gesture recogniser that Rubine describes is very interesting and relevant. The single-stroke requirement of the system solves the segmentation problem (described in section 2.5) of gesture interfaces. To implement the recognition, Rubine defined 13 features of a curve. These features are designed to change in proportion to the gesture (a small change in the gesture produces a small change in the value of a feature). The features are computable in real time as new points are added and are generic enough to be applied to any curve, yet still provide useful information about the curve.

The classification algorithm Rubine implements is an algorithm known as a linear discriminator. For each gesture, the features have an associated weight. Once all the features are calculated, the sum of the weighted features is calculated. The gesture chosen is the class with the greatest sum. The training of the system determines the weights of the various features. The problem with this is that the classifier will always identify a gesture. Sometimes it is desirable for the system to reject the gesture if it is too far from the norm, rather than just classifying the input. Rubine suggests a rejection algorithm, but notes that it has a tendency to reject good gestures.

To test the system, Rubine created four sets, each containing between 10 and 13 classes. These sets were taught to the grandma system using 15 examples of each class. Fifty examples of each class were then presented to the system to be classified. The recognition rate varies between 98% and 100%

### 2.3.2 JOVE

The implementation of this is called the Jove project[4] and involved the recognition of arm and hand position and movement making up infantry command signals. Seventeen gestures were chosen from a standard army set. Six of the gestures are static, depending on the body position at one time only, the other eleven are dynamic with the position of the body changing over time. The purpose of the system was to instruct troopers in a virtual battle field, and as a training tool. The gestures were recorded using three Polhemus trackers, one on each wrist and one on the back.

Two approaches are used for recognition: template matching and trajectory matching

**Template matching**

The template matching is done by setting up regions of space so that if the trackers are within a region, a gesture is recognised. For dynamic gestures, the trackers have to move from one region to another within a fixed amount of time.

**Trajectory matching**

This is done by viewing the dynamic gestures as curves in space that have values of torsion, curvature, starting and ending points. These describe the curve independent of the position and orientation of the subject.

The Jove project proves that neural nets or fuzzy logic are not required for gesture recognition by achieving results similar to those returned by neural net systems. The accuracy of the system at recognising gestures is listed at an average of 82%. The static gestures are recognised with an accuracy of 93% and the dynamic gestures with an accuracy of 78%.

## 2.4 Other gesture-based systems

### 2.4.1 The glove and American sign language

One implementation of a gesture based American Sign Language(ASL) recogniser was done by Dr. Gregory B. Newby in his paper "Gesture Recognition Using Statistical Similarity"[5]

Newby notes that there are three components to an ASL sign: the location relative to the body, the finger position, and the movement. All three of these must be considered when implementing an ASL recogniser. The DataGlove is used to return the data for the recognition

and returns two values for each finger. Two methods of recognition are listed: Fixed-parameter and sum-of-squares statistical recognition

**Fixed-Parameter recognition**

With fixed parameter recognition, lists of the maximum and minimum values of the fingers are set up in such a way as to differentiate the gestures. The incoming data from the dataglove is compared with these lists and the input is classified. The problem with this is that the parameters differ for different users. It is also not suitable for large lists of gestures.

**Sum-of-Squares statistical recognition**

A prototype gesture is setup for each gesture in the set. The sum-of-squares for an input is then calculated for each prototype in the set. The gesture with the lowest sum-of-squares is then selected as the 'correct' gesture. A tolerance is built in so that the system will discard any gesture that deviates by more than a certain value. The sum-of-squares can accommodate many more gestures than the fixed-parameter method can, however it is computationally intensive. As the number of gestures in the system increases, it may become impossible to do the computation in real time, a requirement of VR.

Newby notes that the system was barely suitable for ASL speakers who spell at about 10 letters per second. The accuracy of the system is acceptable as, of the 36 gestures in the system, only four were not recognised reliably.

### 2.4.2 Pen-based recognition

Pen-based systems have been available for the computer for a long time, but they have had

limited success due to poor handwriting recognition and poor integration into the system as a whole. They do have several advantages, pens are light, portable and can be used one handed while offering fine motor control. Gesture recognition is often avoided by pen-based application because of the difficulty of recognition.

The only research available on pen-based interfaces is a proposal by A. Chris Long, Jr. [6]. He is working on a system that improves gesture-based interaction for pen based interfaces. Unfortunately no results are available at the time of writing.

He intends to create a system that can simplify the training of a recogniser by generating gestures from information about them. The system will also allow users to define their own gestures and provide some idea of what gestures are easily remembered by users. The system will be feature based and will do recognition using weighted sums of feature values, similar to Rubine[3]. Additionally it will be able to warn the designer if the gesture set is not independent and will be able to identify the features that result in the lack of independence.

The project looks promising and contains several interesting ideas and it will be interesting to see how it develops.

## 2.5 Advantages and disadvantages of hand gesture recognition

Much as for any other input device there are advantages and disadvantages in using a position tracker to record gestures that convey the user's intent to the computer.
Thomas Baudel[1] noted several advantages and disadvantages which will be repeated here. I will add some advantages and disadvantages afterwards. Some of the disadvantages can be removed with more powerful software and/or hardware, others are intrinsic in gesture-based

systems.

The advantages of gesture input are that it provides:

**Natural interaction:** Gestures are a natural everyday method of communication and is relatively easy to learn to control a computer with gestures

**Powerful interaction:** The tracker can provide information on both the position and the motion of the hand. Thus, a single gesture can specify both a command and the object to be acted upon.

Unfortunately, there are also disadvantages to a gesture based input system:

**Fatigue**: Since the use of gestures to interface to a computer can require the user to hold their hand out in front of them, the muscles can become tired very quickly. To reduce this fatigue the gestures must be designed to be as short as possible. The system must also not require high precision movements for long periods of time.

**Unknown gestures:** The gesture set must be made known to the user. The user must also be given feedback as to whether a gesture was correct or not.

**Segmentation** : Gestures by their very design are continuous. The system must find some way of splitting a continuous hand movement into discrete commands. This method of segmentation must also be acceptable to the user.

**Discomfort** : Some users find the head mounted display and attached wires uncomfortable and a distraction to the VR world being presented.

I will add another advantage to the above list.

**Flexibility**: The gesture input is very flexible, restricted mainly by the software that deals with the input, rather than the concept as a whole. With additional software the gesture system could emulate a keyboard, a mouse, a joystick or as exotic a controller as the batons that are used to direct planes and helicopters.

Two disadvantages that I have noticed.

**Difference between users**: Everyone has a different way of doing things. This translates to a difference in the gestures between different people. Two methods of dealing with this are to adapt the user to the computer or adapt the program to be more tolerant of different users. The second method is more desirable but the first is often simpler

**Complex recognition methods:** Since gestures generally differ from one user to another, and even the same user is hard-pressed to perfectly replicate a gesture, the software required to classify gestures into one of a set of categories is generally complex. The problem increases when the number of gestures in the system increases. With this complexity comes a loss of speed. VR systems are of necessity real-time so this poses a problem. More powerful hardware can solve the problem, but if VR is ever to be used by the average person, powerful hardware must not be a requirement for the system to work.

To combat the problem of unknown gestures, the gestures should represent the action to be performed as much as possible. As an example consider a gesture that is to delete an object. Two possible gestures that are fairly intuitive are to cross the object out or to tear it up. The segmentation problem can be overcome by designing extra hardware that the user can use to indicate what the system must analyse and what it must ignore. In the case of the system being described I designed a switch that the user can use to indicate gestures. The other approaches to segmentation are to divide space into command areas and non command areas. The system then only considers movement within the command areas or else to specify start and end positions for the user to assume at the beginning and end of gestures. This is an approach used by glove applications

## 2.6 Recognition methods used

I decided to concentrate less on developing computationally intensive recognisors and more on carefully selecting gestures and discovering features that effectively differentiate between the gestures. As a result the two recognisors are very simple. The two recognisors are derived from the two listed in the paper by Newby [5].

## Summary

The interface devices for virtual reality, despite being plentiful have shortcomings. No interface device can be easily used to interact with every virtual reality application. Because of this, there is still work to be done in the field of virtual reality interface devices.

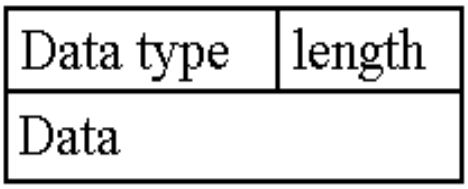
Gesture recognition is an interaction method curently under investigation by many groups because of the promise of a simple method of interacting with any application. The background information on gesture recognition is varied and quite plentiful (see [1],[3],[4],[5]). From the research done and the information that exists, there are clearly problems that have not yet been overcome. There is also little work, if any, done on 3D gesture recognition using position trackers. This is the area I am investigating, as well as trying to remove some of the disadvantages of gesture recognition.

# Chapter 3: System

The operation of CoRgi system depends on three things: getting data from input devices, processing that data and, finally, drawing the virtual world to a display device. The drawing of the virtual world is not part of this project, however the first two issues were dealt with during the development of the gesture recognition system.

## 3.1 CoRgi Peripherals

The CoRgi peripherals are all attached to a single machine whose sole purpose is to collect data from all the peripherals and place the data onto the network. Each peripheral has an associated data type and the data that is put onto the network consists of the data type followed by the length of the data, then the data.




### 3.1.1 Stick

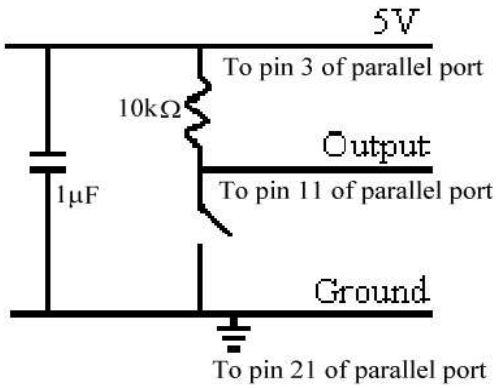
The stick device was created as an alternative to the VRGlove. The stick consists of four switches placed under four of the fingers. Each switch returns 1 or 0, 1 if the button is pressed, 0 if the button is not.

### 3.1.2 Switch

The switch was designed to provide a way for a user, using the gesture-based interface, to

indicate when the movements of the tracker were tracing out a gesture and when the movements were of no interest to the recognisor.

The switch is designed to avoid 'bouncing' which occurs when an electronic switch is either opened or closed. Electrical contact can be made and lost several times before the switch settles down to a stable state. This bouncing is undesirable since it can result in the recognisor receiving false information about the start and end of a gesture. To eliminate this problem, a capacitor was built into the switch to give a smooth variation between ON and OFF




Circuit Diagram - Switch

### 3.1.3 Polhemus Trackers

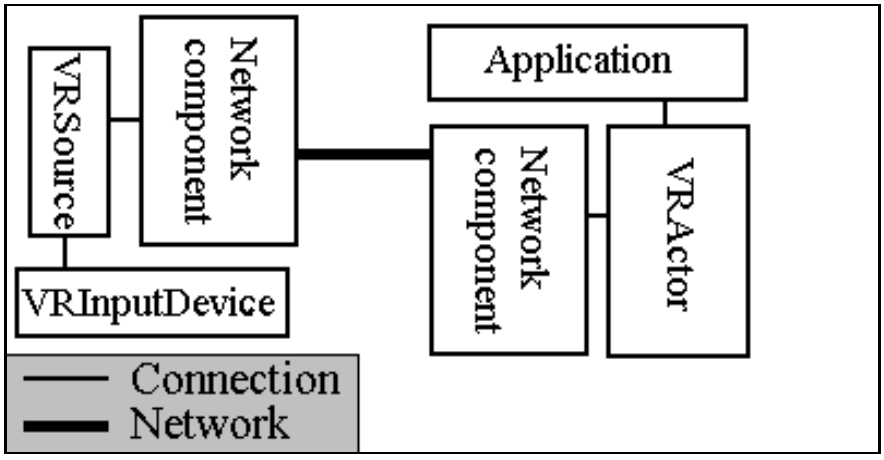
The Polhemus trackers are electromagnetic sensors. Each tracker has an associated transmitter which acts as the origin of the coordinate system. The receiver detects the electromagnetic field emitted by the transmitter and sends the sensed signals to the computer, where a mathematical algorithm computes the receiver's position and orientation relative to the transmitter.

The Polhemus trackers have a range of about 70 cm and are accurate to 1 cm (according to the manual, better accuracy has been achieved). Each transmitter has two receivers associated with it. This means that if a pair of trackers, with the same transmitter, are used, the coordinates that the two receivers return are guaranteed to return the same values for a fixed location. This makes it possible to use one tracker from a pair for controlling the viewpoint and the second for returning the position of a user's hand, without the need for setting the origins of the two coordinate systems.

## 3.2 CoRgi component architecture.

The CoRgi architecture is object-orientated and component based, where a class has a specific function and connections to other classes. All an application has to do is set up the components required, set up the connections then call the components thread-routines in an endless loop. The components are roughly divided into input, output, actor and environment components.

The entire system slots together as shown in the figure below.



### 3.2.1 VRComponent

The VRComponent is the base class for actors, input and output devices and the sink. The VRComponent contains a function (void)ThreadRoutine(void). This is the function that the application calls for each component in an infinite loop. The ThreadRoutine can neither take nor return data, so if the application needs to give data to any of the components, the component must set up a link to that component in order to get that data. The threadroutine is the only procedure called by the application once it has entered the infinite loop

### 3.2.2 Visual Representation

The visual representation components are the place where the appearance of an object is defined. The control of the output windows is controlled by the vroutputdevice as well as the vrsink. It is the visual representation however that specifies how the sink is to draw the object.

### 3.2.3 VRActor

The VRActor class is the base class for the participants within the virtual world. Classes derived from VRActor include ants, a head tracker and seaweed. The actor class provides mechanisms for setting and reading the various attributes of the actor objects, including position and orientation within the virtual world.

### 3.2.4 VREnvironment

The VREnvironment class is concerned with controlling the background VR world. This includes functions for getting, setting and removing object hierarchies, so that the position of an object is partially determined by the position of another, for getting collisions between objects as well as adding and removing objects from the world.

### 3.2.5 VRInputDevice

The VRInputDevice class is the base class for all the devices attached to the system. All the input devices are attached to a single machine which acts as an input server, facilitated by the distributed nature of the CoRgi system. The server gets data from the devices using the GetData function in the derived VRInputDevice. The data is then packaged and put onto the network for collection by the machine running the application.

### 3.2.6 VRApplication

Due to the modular nature of CoRgi, the applications are very simple. All the work is done by actors, input and output devices and the sink. The application usually involves setting up the components desired, connecting the ones that need to communicate, and then entering an endless loop in which the threadroutines of all components are run one after another.

An application normally has the following structure

```
#include <corgivr.h>

//Declare and initialise all variables
VRSink TheSink;
VREnvironment vrenv;
VRFormInputDevice forminput;
VRSource vrsource (&forminput);
objectID obj2=createThing();
VisualRepresentation z("object_files/sphere");
VRUserActor vruser (obj2);

//Setup connections between components that need to communicate
//with one another
```

```
Connection userinput (vrsource, vruser);
//Setup any initial conditions for the world
SetAbsolutePosition (obj2,Point3D(0,0,-10));
SetPhysicalRepresentation (&z,obj2)
//Pass any necessary data to the components
forminput.OriginIs(Point3D(0,0,-10));
while (1)
 RunComponents();
```

**Summary**

The CoRgi system provides a robust and easily extendable environment for virtual reality applications. The distributed nature of the system allows components on different machines to communicate. The development of an application is simplified by the presence of components that can be used without any adaption. It also allows different people to concentrate on different aspects of the system. The CoRgi peripherals allow interaction with applications, without the necessity of writing drivers and other low level details. All in all the CoRgi architecture allows a developer to concentrate on the code unique to their application and not have to repeat work done by others.

# Chapter 4: Design

The aim of the system is to provide a robust gesture recognition system for controlling a virtual reality application. The application planned to test this is a 3D paint box in which the user can create and manipulate three-dimensional objects using only gestures.

With this in mind, one of the first design issues is to select gestures, that should be as simple to recognise as possible, while retaining sufficient complexity to provide a good test of the system. With a small set of independent gestures, the recognition routines for identifying the gestures can be implemented without resorting to sophisticated neural networks or fuzzy logic theory.

The second design issue is to select features that can be calculated quickly, and whose values differ across the gestures sufficiently to differentiate them.

The third design issue is to choose a method of distinguishing gestures based on the values of the features. Depending on the level of independence of the gestures, the recognition method may be as simple as a case statement, or as sophisticated as a neural network classifier.

The design of a system can be broken down into the design of the problem solver and the design of the interface to the user. These are examined separately.

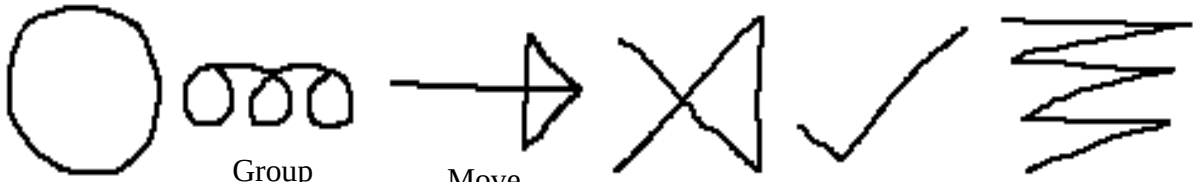
## 4.1 Interface

The interface section of the design deals with how the system is presented to potential users. The only portion of the design that deals with the user is the selection of gestures. The gestures are chosen to control a 3D paintbox application. While the application has not been developed, the theory is that any application designed within the CoRgi architecture that can accept commands from another actor could be controlled with the gesture interface.

### 4.1.1 Gestures

The gestures chosen represent six activities that occur within a graphics application. It is assumed that there is another method of creating the objects, and that gestures are used to manipulate them. If some form of VRMenu could be implemented, then gestures could be used to control that as well. The start, end and centre points of the gestures can be given to the application, enabling it to determine which objects are affected.

The six gestures chosen are as follows. The figures are captioned with the command that the gestures represent.



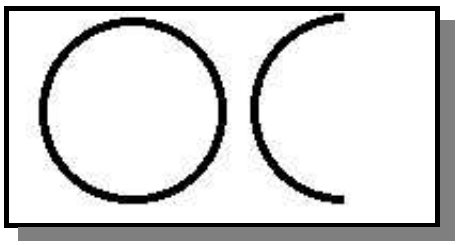
Select Group Move Delete OK (deselect) Change colour

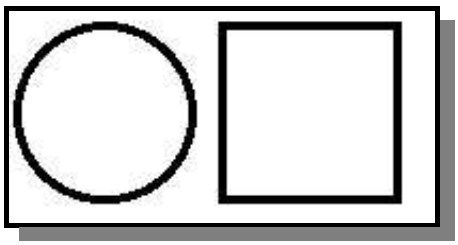
These gestures were chosen because they are more or less independent. The 'more-or-less' is due to the fact that if the user is not careful, the gesture 'select' can be identified as the gesture 'delete'. 'Move' may also be misidentified as 'delete' These misidentifications, however, are rare and require the user to be careless when making the gestures.

### 4.1.2 The importance of independent gestures

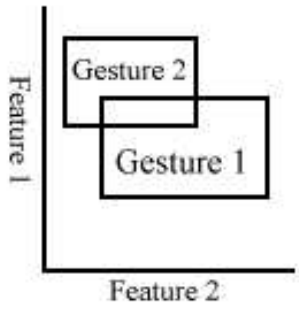
Independence : The condition of not being controlled or determined by anything else [7].
Independence, when it applies to a set of gestures, is an indication that the features that are calculated for that set of gestures return widely separate values for two gestures. Whether a set of gestures is independent or not depends on the features that are calculated for that set of gestures. As an example, consider a set of gestures where the only feature calculated is displacement, being the distance between the first and last point drawn. With this feature, a circle

and a square would not be independent, whereas a circle and a semicircle would be. If however the feature been calculated was the maximum change in the slope of the line over a short distance, the circle and square would now be independent due to the sharp corners of the square, but the circle and semicircle now would not be independent.

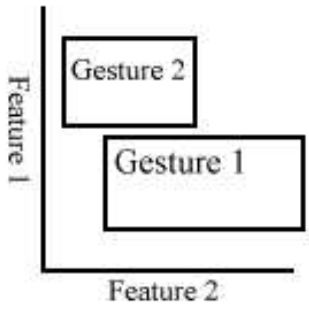




The easiest way to determine if a set of gestures are independent or not is to plot a graph with a feature value on each axis (Multiple graphs if the system has more than three features) and to see if the set of possible values for one feature overlaps the set for another.




Not independent




Independent

If the gesture set and features have been chosen so that the gesture set is independent, then the recognition routines may be dramatically simplified, possibly even to the degree that recognition can be done with a case statement. Another important property of an independent gesture set is that of certainty. If the feature value sets of two gestures overlap, then it is impossible to be totally certain which is correct, whereas with an independent gesture set, only a very inaccurate gesture would be misidentified

## 4.2 System

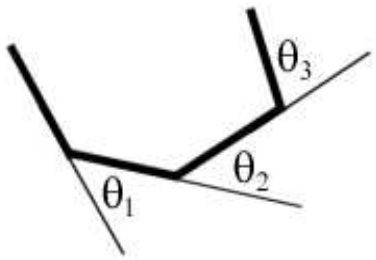
The design of the system primarily involved deciding upon features to be used within the system and how to distinguish gestures based on the values of the features.

### 4.2.1 Features

The three features decided upon are the curvature, the absolute curvature and the total length of the curve divided by the distance between start and end points. These three features were decided upon because they are not dependent on the size of the gesture or on the orientation in space

Curvature

I have defined the curvature to be the absolute value of the sum of the exterior angles of the shape. In this way a straight line will have a curvature of zero, while any closed figure will have a curvature of 360°.



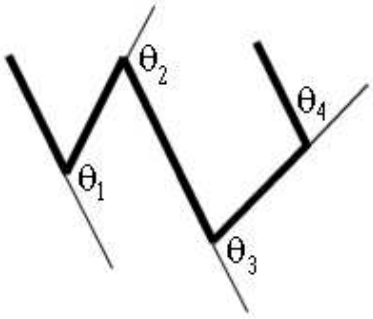

Curvature = $| \Sigma \theta |$

Curvature

Absolute Curvature

I have defined the absolute curvature of a gesture to be the sum of the absolute values of the exterior angles of the gesture. For some shapes this will be the same as the curvature, (eg figure

above) while in others (figure below) the two values will differ. In the figure below the angle $\theta_2$ is a different sign to all the others.

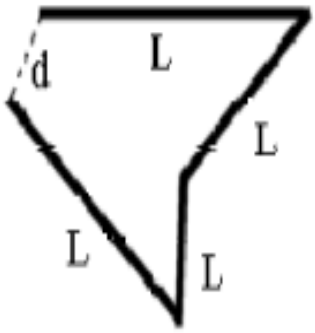



Absolute Curvature = $\Sigma \mid \theta \mid$

Absolute Curvature

Length divided by Displacement

This feature is simply the total length of all the segments of the curve divided by the distance between the start and end points. This provides a measure of how linear a gesture is. The ratio of the two values is requires as both length and displacement are dependent on the size of the gesture, but the ration of the two is not. If the shape is closed, this feature may turn out to be dependent of size, but it does not matter as the test for a closed object is simply Length/Displacement $\rightarrow \infty$




Length / Displacement = ( $\Sigma$ L ) / d

Length/Displacement

### 4.2.2 Recognition

Once the features have been decided upon, the recognition algorithm must be designed. In designing a recognition algorithm it is assumed that the feature calculator has done its job correctly and returned values that can distinguish gestures. The recognizor must then, on the basis of those values, decide which of the available categories the input belongs in. I decided to put more work into the feature calculators and in ensuring that the gestures I am using are independent and thereby simplify the classifiers. I implemented two classifiers, the second being more forgiving than the first.

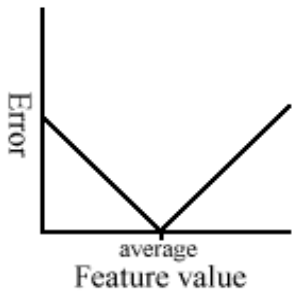
Solid cutoff

This is the simplest classification method I could think of. Each gesture has a defined minimum and maximum value for every feature. If the feature values calculated for the input fall within the minimum and maximum for every feature for a certain category then the input is of that category. If the input does not fit into any of the categories, then the input is listed as undefined.

The maxima and minima were defined by repeatedly calculating the feature values for the different gestures. Once a large enough sample had been accumulated, the maxima and minima were defined as values that very few of the samples fell outside of.

Error minimisation

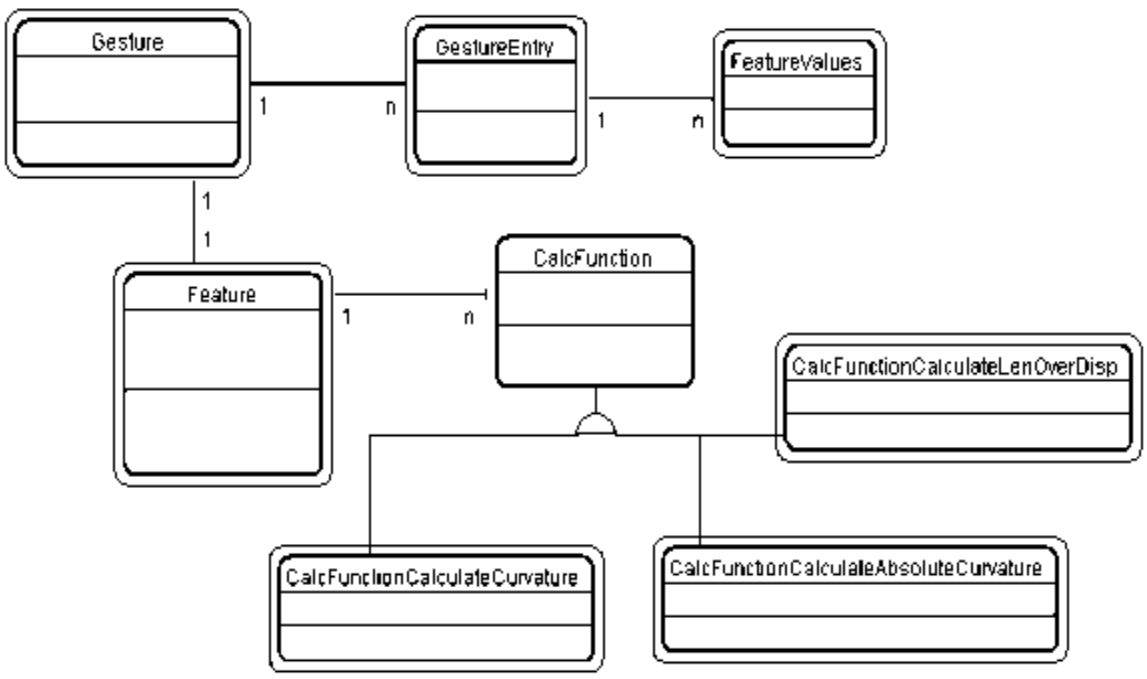
For error minimisation, each feature had a defined average value for a specific gesture. When an input is tested, the feature values for that input are calculated and an error computed for each gesture in the set. The error is defined as the difference between the input's value for the feature and the average value for that feature divided by the average value. The division is to ensure that features with large averages contribute the same to the error as features with small averages.



The problem with this method is if the gestures are clustered in 'feature space' then this may home in on the incorrect gesture. This inaccuracy may be removed by coupling the two recognition methods. While the first method often returns 'unknown' it very seldom returns the incorrect gesture as the gesture set I have chosen is independent. The second method is much more likely to return the correct value, but it can be confused and the situation where the first method returns the correct gesture and the second returns the incorrect gesture may occur. If the system gave the first method greater priority, assuming that it returned a gesture at all, then the accuracy of the system would be increased.

## 4.3 Extensibility

One of the aims of the system is to ensure ease of modification and extension. The system should be easy to adapt to applications other than the one for which it was designed. To this end the components of the system that contain information about the gestures and the features are replaceable.



Object structure of the gesture and feature representations

The features are represented as a name, which is purely cosmetic, and a function to calculate the value of that feature given a list of points. The gestures are represented as a name, also purely cosmetic, a list of maximum, minimum and average values for each feature in the system.

A major issue was how to ensure that each feature in the system is calculated when the system does not know at compile-time how many features are within the system. This is necessary so extra features can be added to the system if (or when) the current features become insufficient for classifying the gestures, if new gestures are added. The solution to this problem is to add the features to a dynamic list which contains a reference to the calculator routine for that feature

## Summary

The design of a system proceeds along two paths, the design of the core system and the design of the interface that users will be presented with. The core system issues are those of calculation and recognition. The features must be selected carefully. They must be fast to calculate and must return useful data. The length of the curve, while satisfying the first criteria, is of little use if the gestures can be drawn big or small.

Another core system issue is that of recognition methods. The afore stated aims of speed and simplicity limit the recognition methods. This limitation is not crippling as long as care is taken with designing the bounds that the two recognition methods (solid cutoff and error minimisation) use. The solid cutoff has defined maxima and minima within which the feature values are allowed to lie. The feature values must lie between the maxima and minima of at most one gesture. The error minimisation finds the gesture with the lowest error between hoped for values and the actual values.

The interface design is primarily concerned with finding gestures to control a specific application. The gestures should be classifiable on the basis of the feature values returned by the core system. The gestures must be independent, single-stroke and easy to remember. Six gestures form the set selected. These six are designed to control a paint-box application.

The gestures must be replaceable, so that other applications can also be supported. To this end, the features must also be replaceable so that gestures very different from the present ones can be added. To this end gestures within the system can be replaced by over-riding a single class, which specifies all the details about the gestures. Similarly the features can be added to be defining new calculation routines and adding them into the system.

The end result of the design is a robust, fast system that is easily adaptable to the needs of an application.

# Chapter 5: Development

The development, like the system design, has two major components, development of the core system and development of the interface to the user. Issues that cropped up during the development of the system are described here, as well as the solutions used.

## 5.1 Interface

The interface section deals with issues that came up during development of the user interface, as well as adjustments that had to be made to the system to facilitate users.

### 5.1.1 Acceptance of points

The first problem concerns the acceptance of points by the system. Initially, it was decided that every point that the Polhemus tracker returned would be added to the list of points making up the gesture. This, however, proved to be a bad decision as the trackers are prone to small vibrations about a point. These vibrations become more evident in the presence of metal, or as the trackers move away from the transmitter. These small vibrations are fed into the gesture recognition system. The vibrations are almost unnoticeable on screen and when the sum of the angles between the segments is calculated the additional angles caused by the vibrations mostly cancel out. It is, however, when the sum of absolute values of angles is calculated that the vibrations become very noticeable. An example of how noticeable the vibrations can be: a straight line should have a sum of exterior angles approximately equal to $0^\circ$ but with small vibrations, a line that looks straight might return a value over $1000^\circ$. To combat this problem, the tracker points are checked before adding them to the list of points making up the gesture. A minimum length is defined for points. If the point that has just been received is less than the minimum distance

from the previous point, it is discarded. Finding an exact value for the minimum distance is a trial-and-error process. The value must be large enough to cut out random vibrations, whether from the tracker or from a user's shaking hand, but small enough that the user does not lose sections from the gesture.

### 5.1.2 Visual feedback

It is a standard VR theory that if a user receives visual feedback on what they are doing, they will more accurate than without visual feedback. To this end the system is designed to give the user accurate information about what they are doing. This includes an object that marks the position of the tracker within the virtual world, much as a mouse pointer indicates the mouse position on a desktop. This is quite effective as the user can always tell where a gesture will begin, important with gestures whose start and end points have special meaning. The second component to the visual feedback is an indication of what the user is drawing. This is achieved by displaying on the screen the points as they are received from the tracker, each pair of points joined by a line segment. This shows the user exactly what they have already done, theoretically enabling the user to complete gestures correctly.

A tracker also follows the current position of the head of a user wearing a head-mounted display. This ensures that the user can always find the position of the tracker in the virtual world as it mirrors the position of the tracker in the real world. Without this, finding the position in the world that the tracker has can be time consuming and frustrating.

### 5.1.3 Ease of use

The system must be easy to use, both for novice and experienced users. Experienced users should be able to see a similarity between the tracker and a mouse, enough similarity that experience with a mouse should translate to a familiarity with the tracker. For novice users, the system

should be intuitive enough so that they can grasp the concepts quickly.

The object showing the current position of the tracker helps the user keep track of their location in the virtual world. In addition to this, a tracker monitors the position of the user's head and changes the camera view to match

## 5.2 System

Since the main aim of the system is to accurately recognise gestures, using various features to do the classification, the implementation of the feature calculator is of prime importance. The second issue that has to be addressed during the development of the system is how the system will tolerate bad gestures. The system must be robust and discriminating.

### 5.2.1 The feature calculator

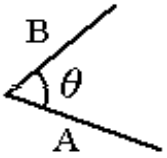
The feature calculator has to return similar values for a specific gesture, no matter in what orientation in space the gesture has been drawn.

The first implementation of the feature calculator, the angle between two line segments was calculated directly. The understanding of this calculation requires some background in vector mathematics. Simply put, if A and B are two vectors in 3D space then

$$|A \times B| = |A||B|\sin\theta$$

and

$$A \cdot B = |A||B|\cos\theta$$




where $\theta$ is the angle between the two vectors,

as shown in the diagram above. Hence

$$\tan\theta = |A \times B| / (A \cdot B) \quad \text{and} \quad \theta = \tan^{-1}( |A \times B| / (A \cdot B) )$$

The angle theta returned by this is a value between -180° and 180°

It was later noticed that the results of this calculation are unpredictable as the gesture is not required to be in one plane. The property of not being planar results in the deviation from the average plane of the gesture adding to the curvature of the gesture. This is highly undesirable as this deviation cannot be predicted and it is difficult to form a gesture in a single plane without any reference (eg a wall)

To combat this, the system calculates the average plane or the gesture, transforms that plane to x-y and then projects all the points of the gesture onto the x-y plane, resulting in a 2D representation of the gesture. Once this has been done, the angles can be calculated accurately without any concern about random deviations from the plane.

The plane that the gesture lies in could be calculated using a least squares fit. Least squares however requires a large amount of calculation and can easily slow a system down. The method I chose gives a worse fit to a plane than least squares but is much easier to compute. The accuracy loss in not significant as all the system requires is a plane that the gesture partially lies upon in order to convert the gesture to a 2D representation. The method selected to find the plane simply selects from the list of points making up the gesture the three points furthest from each other and sets two of the vectors joining those points as the basis vectors for the plane. The points are not directly projected onto this plane for two reasons, first the projection may be complex and second if the two basis vectors are not at right angles, the geometry of the gesture may be distorted by the projection. The gesture is first transformed so that the plane it falls in is the x-y plane, then the points are projected onto the x-y plane.

Once the points have been transformed to the x-y plane, the angle between each pair of line segments can be calculated accurately using the formula listed above.

### 5.2.2 System Tolerance

A requirement for an effective gesture recognition system is that it must be able to determine when the user has, possibly inadvertently, entered a shape that bares no similarity to any of the gestures in the system, as well as correctly identifying gestures. There are two ways of doing this. Either the recogniser can have a tolerance where gestures that fall too far from the norm are rejected, or the application that uses the gesture recogniser can provide an easy means of undoing something if the system recognised incorrectly. The two recognition algorithms that I implemented deal with this differently.

The solid cutoff recognition has no problem with eliminating gestures with extreme values as its operation depends on checking whether the feature value is within fixed bounds. If it is outside the bounds then the gesture is rejected. The fixed bounds for the gestures do not overlap, so in general the gesture is either correctly identified, or rejected. A few cases where an incorrect gesture has been recognised have been seen, but they are definitely the exception.

The second recognition method, however, has no definite cutoffs. The recognisor works by finding the gesture with the smallest error in feature values when compared to the input. It is not concerned whether or not the error is an acceptable value. I have elected to follow the second method of dealing with errors, to accept them and provide an easy method of undoing in the application. A cutoff can be provided for the error value, but it must be carefully chosen to ensure that correct gestures are not thrown away[3]. Whether the error cutoff is implemented or not depends on the type of application. The cutoff reduces the incorrectly identified gestures, but may not classify gestures that otherwise would be correct. If a low error is desirable then the cutoff should be implemented, if however the application needs a high accuracy and the effects of incorrect gestures an be undone easily then the cutoff should be ignored.

The rejection can be refined as follows: the first recognition method is much less likely to return

a value than the second, but when it does return a value it can be assumed that the gesture returned is in fact the one the user was intending. If the two recognition methods both return a value, and the values are different, than the first can be trusted more than the second. This provides a small defence against incorrect recognition, however the probability that the two recognition algorithms will return different values is small, though not zero.

## 5.3 Extensibility

Since the gestures were designed specifically for a graphics application, and graphics is not the only type of application in existence, there must be some way to replace the gestures with different ones for different applications. The method I decided upon is to have classes that represent the features and the gestures, classes that can be overridden by anyone wishing to add to or replace the gestures in the system. The functions that calculate the feature values can also be added to or replaced as they all derive from a base *Calculator* class.

The *Gesture* class and *GestureEntry* class contain all the information necessary to describe a gesture totally. The gesture class contains the recognition algorithms, the number of gestures in the system as well as pointers to the feature class and the gesture entry class. The *GestureEntry* class contains the maximum, minimum and average values for each gesture. A separate *GestureEntry* class is setup for each gesture in the system. Both the *Gesture* class and the *GestureEntry* classes are over rideable without breaking the system, providing an easy method for replacing or adding gestures to the system.

The *Gesture* class has the following structure:

```
class Gesture
 {
  public:
   GestureEntry * AGesture;
```

```
   Feature FeatureCalculator;
   int NumberOfGestures;
   int Recognise1(float * values);
   int Recognise2(float * values);
   void Output(float * values, int Length, int Answer1, int Answer2);
   Gesture();
   ~Gesture();
  int WhatHaveWeHere (linkedlist<Point3D> &LotsOfPoints, Vector3D TheNormal);
 };
```

In the above listing,

- *AGesture* is a reference to an array of *GestureEntry* classes
- *FeatureCalculator* is a reference to the *Feature* class.
- *Output* is a debugging routine that writes feature values and the gestures that have been recognised to a file. *Recognise1* and *Recognise2* are the two recognition routines.
- *WhatHaveWeHere* is the routine that is responsible for taking the list, passing it to the feature calculators, passing the feature values to the recognisers and returning the gesture number to the calling procedure. It is the only routine in this class that should be called from outside of it.

The *GestureEntry* class is defined as follows

```
class GestureEntry {
  public:
    char * GestureName;
    FeatureValues * TheFeatures;
    GestureEntry();
    ~GestureEntry();
    void SetName (char * Name);
    void SetFeatures (int NoOfFeatures);
};
```

- *GestureName* is simply the name of the gesture (cosmetic).

- *The Features* is a link to a list of the minimum, average and maximum values for each feature.
- The constructor of this class is where the maximum, minimum and average values are of the features are set for each gesture.

The features are contained within a *Feature* class. This class is a linked list that contains pointers to the calculation functions. This class also contains functions for adding features to the system.

```
class Feature
 {
  private:
   struct FeatureNode {
    CalcFunction * TheFunction;
    char * FeatureName;
   };
   CalcFunctionCalculateCurvature * firstfunction;
   CalcFunctionCalculateAbsoluteCurvature * secondfunction;
   CalcFunctionCalculateLenOverDisp * thirdfunction;
   linkedlist<FeatureNode *> * FeatureList;
  public:
   Feature();
   ~Feature();
   float * Calculate(linkedlist<Point3D> &TheList, Vector3D TheNormal);
   void WriteToXFigFile(linkedlist<Point3D> &TheList);
   void AddFeature(char * Name,CalcFunction * AFunction);
   int NumberOfFeatures;
};
```

- *firstfunction*, *secondfunction* and *thirdfunction* are the three defined feature calculators.
- The function *Calculate* runs through the list *FeatureList* and calls each function in that list.
- *WriteToXFigFile* is another debugging outine that allows the gesture to be drawn to a file so that a user can see exactly what it looks like.
- *AddFeature* is used to add extra features and feature calculators to the system.

The feature calculation functions are all derived from a base calculator class. The base class has no function except to provide a framework for actual calculation functions.

To add a feature to the system, requires a calculation class to be defined that derives from the base class. The new calculation class must have the following structure:

```
class CalcFunctionCalculateCurvature : public CalcFunction
  {
    virtual float CalcValues (linkedlist<Point3D> &TheList, Vector3D
                              PlaneNormal);
  };
```

where *CalcValues* is the function that does the feature calculation work. Once the new class has been defined, it needs adding to the list of features maintained by the class *Feature* with the following function..

```
void AddFeature(char * Name,CalcFunction * Afunction);
```

where *Name* is the name of the feature (purely cosmetic at this point) and *Afunction* is a pointer to the new feature calculation class.

Adding a gesture requires a developer to increase the defined *NumberOfGestures* (defined within class gesture) and to set the maximum, minimum and average values for each feature within the system for that gesture. The setting of values is done with the *SetFeatures* function within the *GestureEntry* class.

This functionality ensures that both the gestures and the features can be extended and replaced without rewriting the system, but the maximum, minimum and average values for the features still need entering for each gesture. This means that the addition of gestures and features is not as simple as might have been desired, but at this point in time there is no automated calculation of the cutoff and average values.

**Summary**

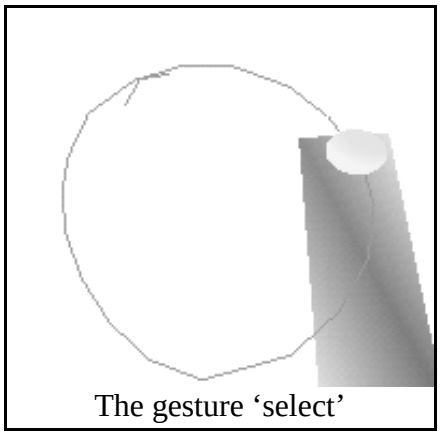
During the development of this system, several issues cropped up. These issues included deciding upon the meaning of angles between segments in three dimensional space, deciding on the desired tolerance of the system, the effect of small vibrations of the tracker on the results the system returns and the desire of the user to see the effects of the system.

The problem of angles in 3D which is intrinsic to a three dimensional gesture recognition system is solved by calculating a plane that the gesture lies within and projecting the points to 2D simplifying the problem by one dimension. The tolerance of the system is dependent what the user is willing to accept, a higher accuracy or a no-class reply rather than an incorrect classification. The small vibrations are caused by a combination of noise in the environment and the steadyness (or lack thereof) of a user's hand. The vibrations can never be eliminated before reaching the system so there must be some way of dealing with them. I chose to ignore small movements of the tracker, smoothing out the received data. The visual feedback is achieved by indicating the position of the tracker in the world as well as leaving a trail in the virtual world of where the tracker has been.

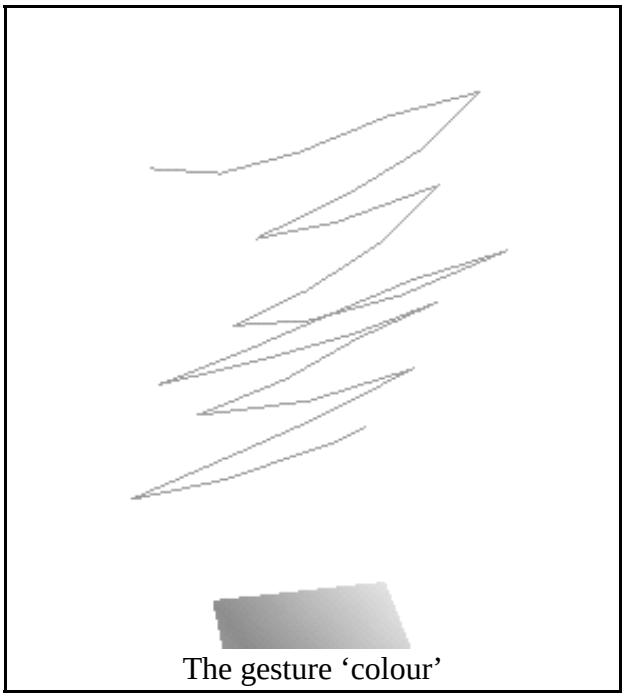
# Chapter 6 : Results

The results gained from this system can be divided into two sections: First how well the system itself worked, whether or not the system could keep up with the users and the general effectiveness of the system. Second how well the system responded to input from users, what percentage accuracy the system has and any relationships between the characteristics of the users and the accuracy of the system.

First two images that show the system in action:



The gesture ‘select’



The gesture ‘colour’

## 6.1 Interface

Several volunteers were selected to test the system to see how good the recognition rate is for people who have never used this type of interaction device before. The volunteers were told that the size and orientation of the gestures has no bearing on the system and neither did the speed of drawing. This is because all the features that the system calculates are independent of position

in space and size of gesture. Despite this most users preferred to draw the gestures in a horizontal line from left to right. Most users also used as little space as possible to draw the gesture and to draw it quite fast.

The tests proceed as follows:

i. A volunteer fills in a short questionnaire designed to determine the degree of artistic ability and computer competency, as well as which hand is dominant.

ii. The user is introduced to the basics of the system and shown the gestures, then instructed to draw each of the gestures. The user then puts on the head-mounted display, so that the gestures can be seen as they are drawn. Readings of the feature values are taken as well as the correctness of the recognition algorithms (whether or not the two algorithms have identified the correct gesture)

iii. The user is given a few minutes to practice the gestures, to get the feel of the system and to try and correct misidentified gestures

iv. The user is then asked to draw the gesture again one by one, first with the dominant hand, then with the non-dominant hand, then without the head-mounted display, to test how well the gestures can be drawn if the user cannot see them.

### 6.1.1 Recognition accuracy

The results of the tests done are very encouraging. While this system does not contain as many gestures as other implementations, or as many features as others, the accuracy speaks highly of applying the axiom 'Simplify, simplify, simplify' to gesture recognition systems.
The accuracies of the two recognition algorithms is given in the following table.

| | Solid cutoff | Error minimisation |
|---|---|---|
| First attempt | 33% | 79% |
| After practice, dominant hand | 41% | 83% |
| After practice, non-dominant hand | 40% | 81% |
| No head mounted display | 40% | 81% |

While not splendid, these accuracies are fairly similar to those achieved by other gesture recognition systems. The Jove project [4] has five static gestures, for which there is no match in this system, and six dynamic gestures. The accuracy with which the six dynamic gestures in the Jove project are recognised was listed at 78%. Though these results pale in comparison to neural-net recognisors, which can achieve an accuracy of 92% with over 200 gestures[4], or to Rubine[3] with an accuracy of over 97%, the fact that the entire system is less complex than the neural net system proves that a simple solution can achieve results comparable to more sophisticated solutions. The accuracy of Rubine's toolkit is due to the large number of features in the system (over four times the number I have) and the fact that Rubine's system is 2D and hence does not have to face issues of non-planar gestures.

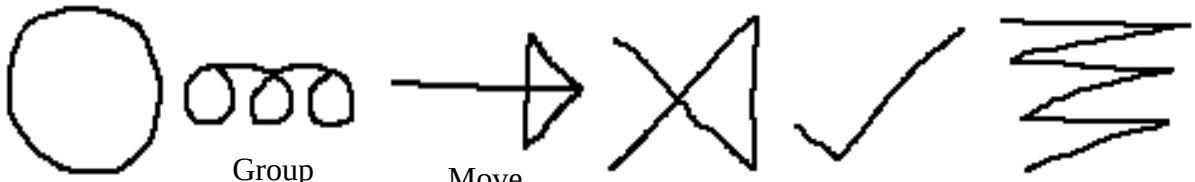
### 6.1.2 Variation of feature values

As can be expected, the values for the features of each gesture vary between users. I will also examine the relationship between accuracy rates and computer knowledge, artistic ability and speed of drawing (determined by the number of points drawn).
For the computer knowledge, the user selected a category, for the artistic ability I divided the users into groups based on the quality of their reproduction of two small images.

The graphs on the following pages show the feature values returned by each tester. Each gesture has three graphs. The feature values are taken from the gestures done with the dominant hand, after practice. The dark lines on the graphs, in order from the bottom, are the fixed minimum,

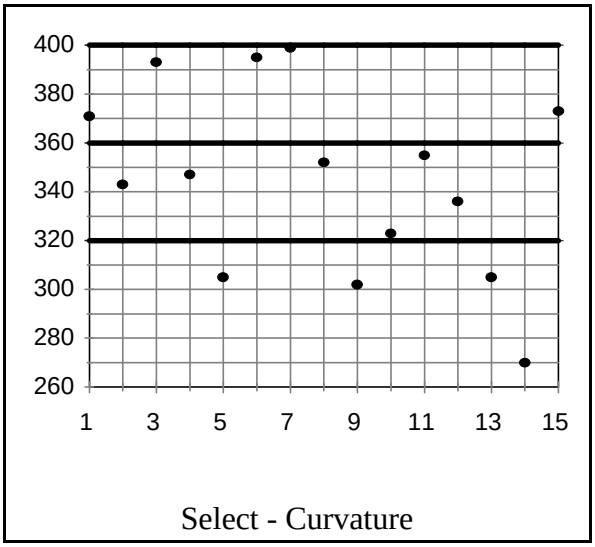
average and maximum values used by the recognition algorithms.

For reference the gestures are repeated here.

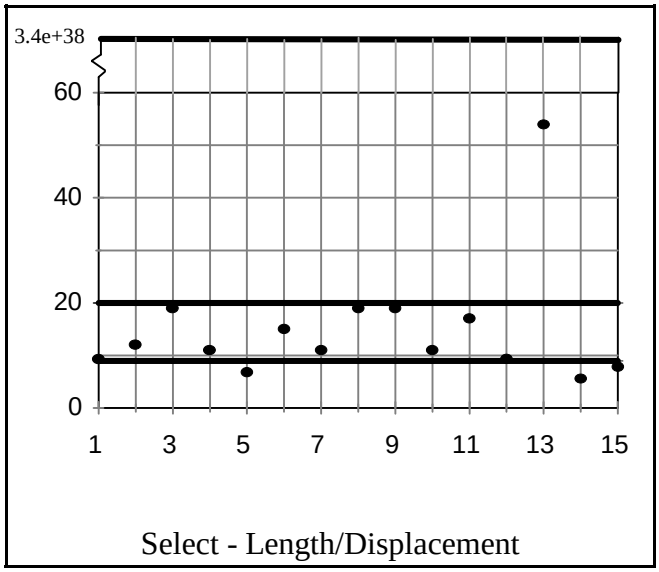


Select Group Move Delete OK (deselect) Change colour

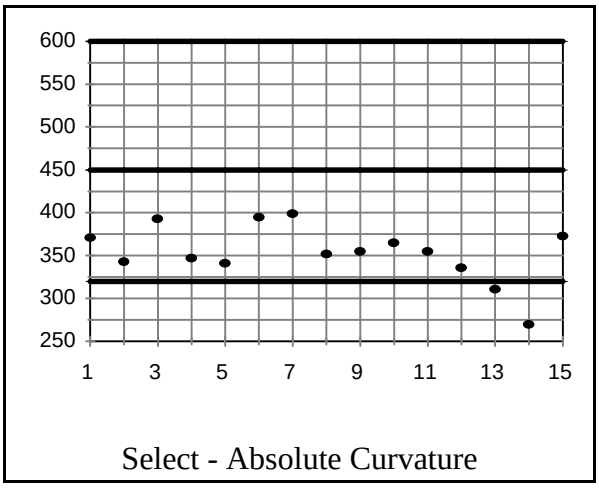
The raw data taken from the tests is listed in the appendix.



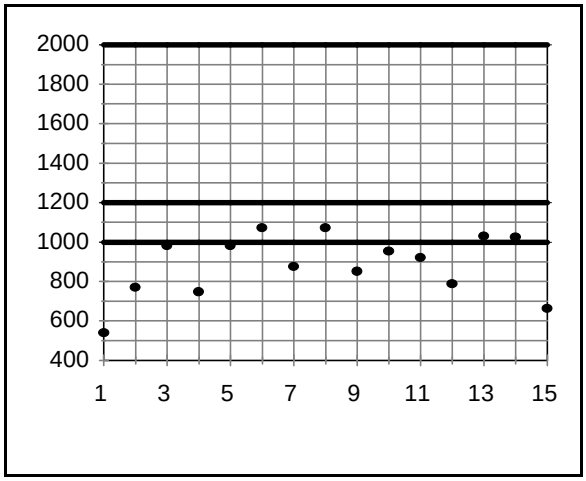

Select - Curvature



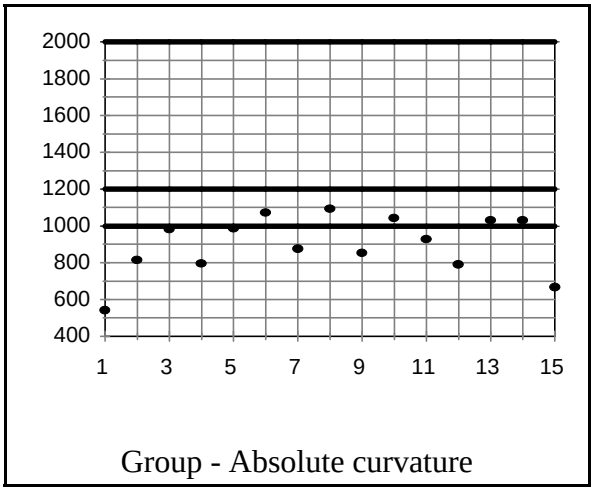

Select - Length/Displacement



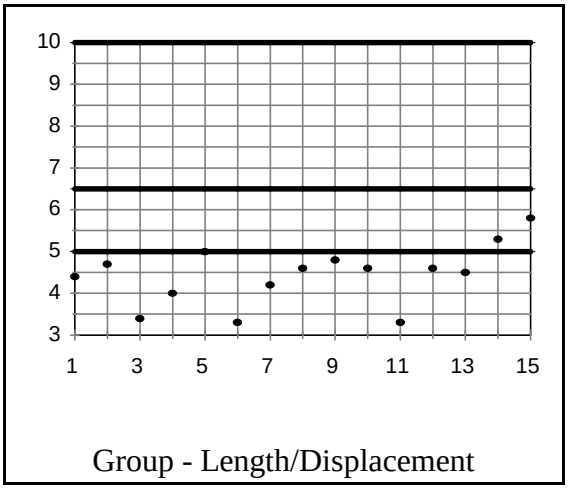

Select - Absolute Curvature



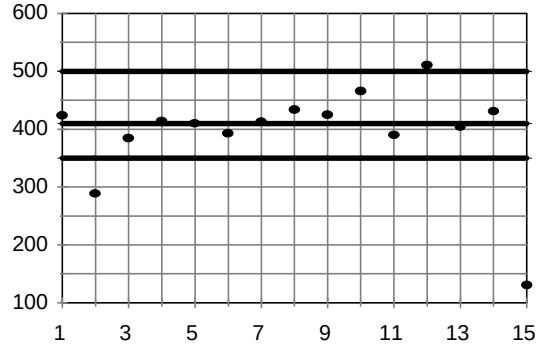

Group - Curvature



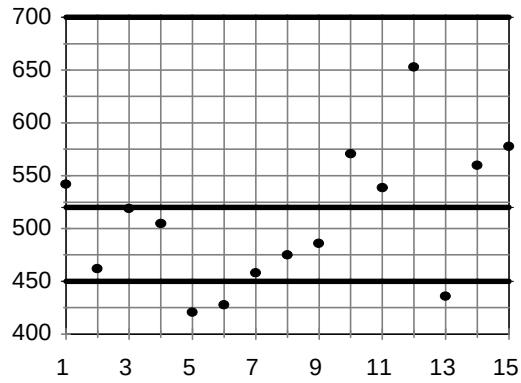

Group - Absolute curvature



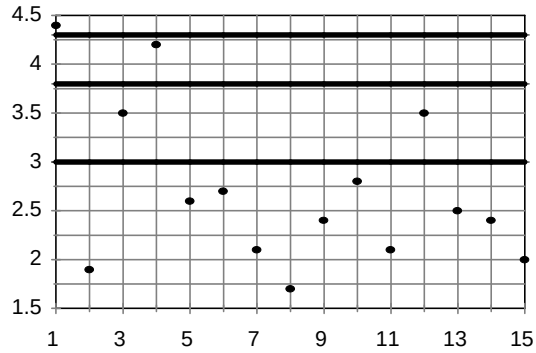

Group - Length/Displacement



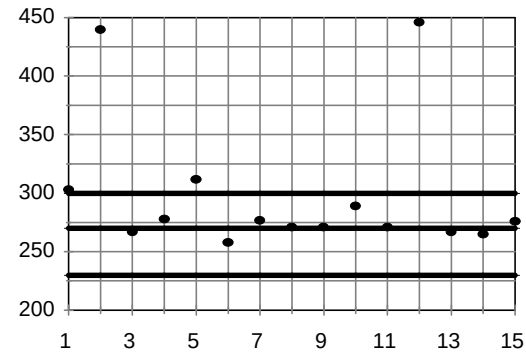

Move - Curvature



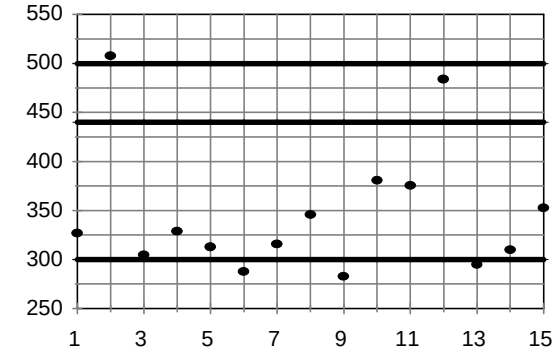

Move - Absolute Curvature



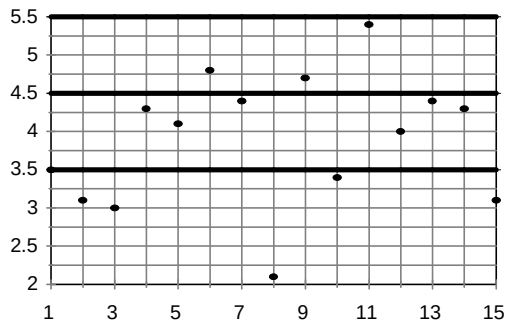

Move - Length/Displacement



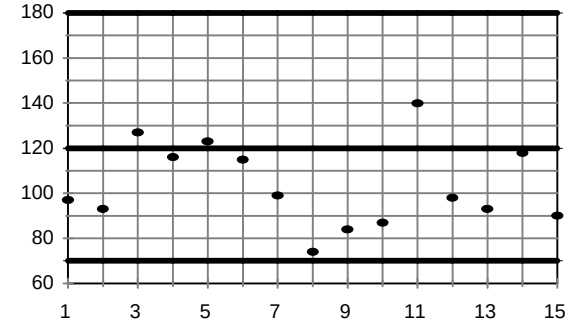

Delete - Curvature



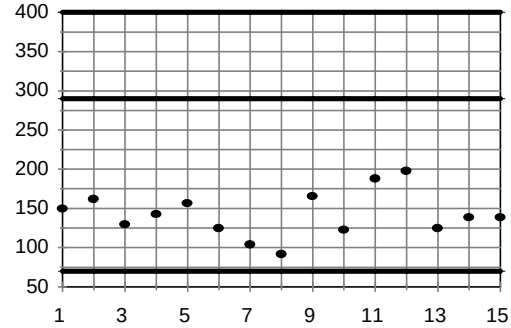

Delete - Absolute Curvature



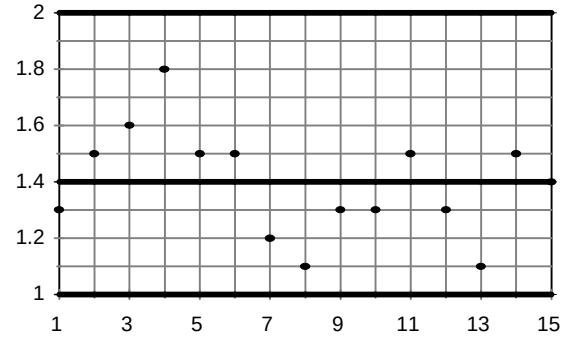

Delete - Length/Displacement



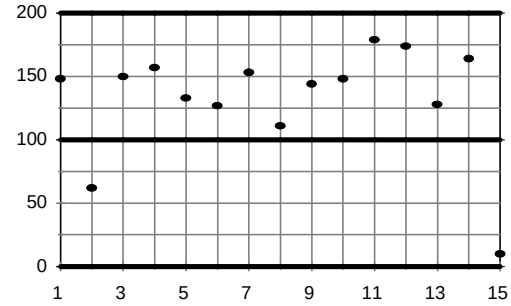

OK - Curvature



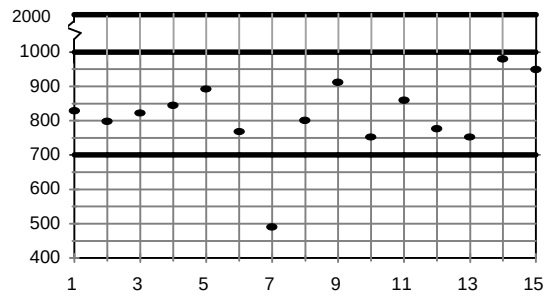

OK - Absolute Curvature



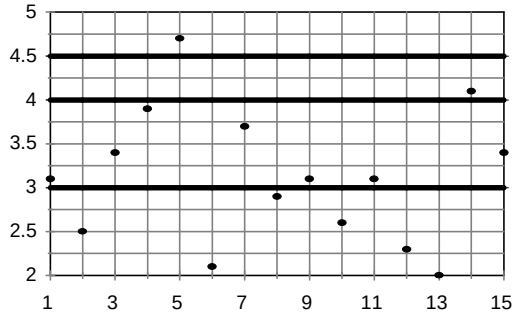

OK - Length/Displacement




Colour - Curvature




Colour - Absolute Curvature




Colour - Length/Displacement

From examination of the graphs the variation of feature values calculated for different users can be seen. The main lesson that can be learnt from these is that using a single user to calculate minimum and maximum values is a bad idea. While setting the average for a feature to a mathematically calculated value sounds like a good idea, there is a good chance that at least a few of the users of the system will form the gesture in such a way as to return a value vastly different from the theoretical value. A good example of this is for gesture 2 (group). This gesture is formed from three circles linked at the top. Mathematically, the sum of the curvature should be $360^\circ$ for each circle, coming to a total of 1080. In reality, the average value returned for the curvature of 'group' is $885^\circ$. Whether this variation is due to an error in the system or simply differences between users is unknown at this time. A possible system reason for this deviation is that the plane fitting algorithm may have failed, hence distorting the projected points.

When the gestures are examined individually it is observed that some were recognised very accurately while others were mistaken relatively often. The values stated here are for the second recognition algorithm. The differences in recognition rate between gestures can be because of a bad choice of gesture or a poor average value.

| **Gesture** | **Accuracy** |
|---|---|
| Select | 52% |
| Group | 72% |
| Move | 74% |
| Delete | 95% |
| OK | 97% |
| Colour | 90% |

The low accuracy for 'select' can be explained by its similarity to 'delete', a similarity not noticed when the gestures were decided upon. This similarity can be seen if the circle is drawn and overlooped. 'Group' and 'move' have poor values chosen for average values. This can be seen from the graphs shown previously. The average for all

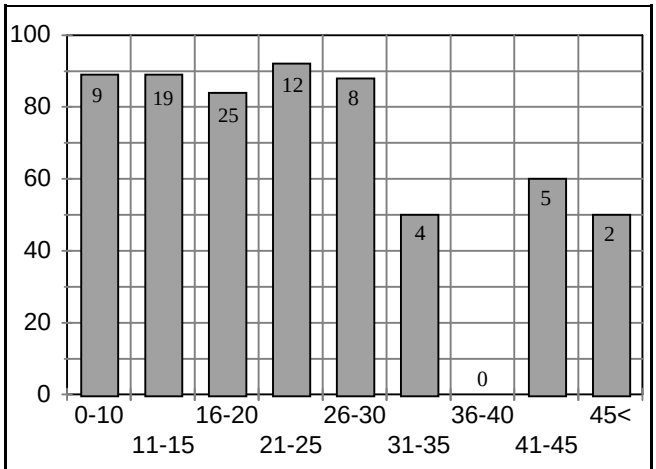
Similarity between delete and an overlooped select

three features for ‘group’ is above most of the points. For ‘move’ the average value o f length/displacement is far too high.

With an adjustment of the maxima and minima, which as can be seen from the graphs above are far from optimal, I calculate that the solid cutoff recognition method can achieve an accuracy as high as 81%, a great improvement over the present accuracy of 39%. This adjustment of the boundaries must be done carefully to ensure that the gestures remain independent. Although if is difficult to calculate, I estimate that with a similar adjustment of the average feature values, the error minimisation method can achieve recognition rates up to 95% or higher.

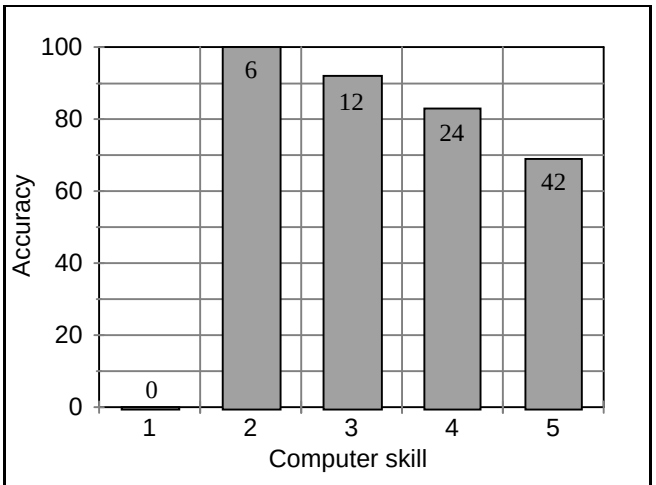
### 6.1.3 Relationship between accuracy and various factors

The relationship between the speed of drawing and accuracy is interesting. My personal feeling, gained from repeatedly testing the system, is that a slower gesture has a greater chance of being correct. This however is not the picture shown by the results. This could be due to the network lag mentioned above. The numbers in each bar show the number of instances of a gesture that fall into a category. The speed is measures by a count of the number of points that comprise a gesture. The speed that the points are received from the Polhemus is assumed to be constant. While the accuracy at slower speeds (more points) is lower, there are two few examples of gestures to know if that is the fact, or if this is an anomaly. This may also not be totally representative of the speed as the number of points drawn is also dependent on the size of the gesture. The graph does show that users tended more to the small gestures, drawn fast than large gestures drawn slower.

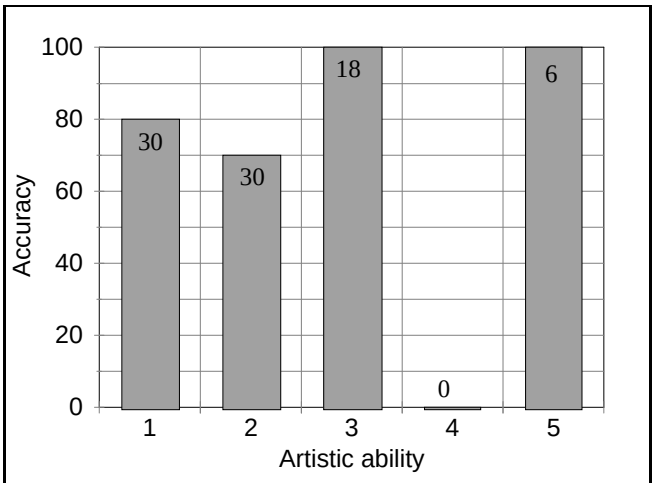



Relationship between speed of drawing and accuracy.

The graphs showing the relationship between computer skill and accuracy, is also interesting. In this graph, the values across the x-axis are levels of computer knowledge from 1='No experience with computers' to 5='Computer Expert'. The numbers on the bars are again the number of examples of gestures in that category. This seems to indicate that the less a person has been exposed to traditional computer interfaces, the better they will be able to adapt to new interface types. This is especially encouraging if one considers the time taken to train users for new computer systems today. These results may also indicate that some functionality expected by people experience with computers is not present. More work is required to tell why this trend exists.



Relationship between computer skill and accuracy



Relationship between artistic ability and accuracy

The relationship between artistic ability and accuracy is not as clear as the above relationships are. This does show that some artistic ability would assist in the use of the system, but it does not appear to be crucial. Possibly additional training could remove this discrepancy.

## 6.2 System

The system responds well to real-time data and handles it well. Some network lag is present that interferes with this real-time aspect slightly. The user sees points appearing on the screen where the tracker had been a moment before. This network lag is beyond the scope of system and very

little can be done about it. Other than that the system keeps up well. There is no slow-down as gestures grow bigger and the short delay as the system calculated the feature values and classified the input is not noticed by the users. Hence I conclude that this small delay does not affect the performance of the system appreciably.

**Summary**

The system performed better under tests than was initially expected achieving results comparable with other gesture-recognition systems. With an average accuracy of 83% this system is capable of controlling an application with an error rate that is sufficiently low to satisfy most users. As shown by the feature values gained from testers, the system is not working as well as it might. With an adjustment of the boundary and average values of the features, I believe an accuracy approaching 95% is possible. The relationships between the gesture recognition rate and speed of drawing, artistic ability and computer knowledge are interesting and may be of importance to those designing the next generation of computer interface.

# Chapter 7 : Conclusion

The solution presented here is a different slant on a problem being worked on by many individuals. The difference here is that I am working on a 3D gesture recognition system using only position sensors. While work has been done on gesture recognition in 2D systems as well as gesture recognition using DataGloves, little to no work has been done or is being done on 3D gesture recognition using position trackers.

This system accurately recognises 3D gestures. The gestures are independent of size, and orientation. This ensures that this system can be used in a virtual reality environment allowing the user to move freely within the environment and still being able to control it.

## 7.1 Interface

This system is designed as part of the CoRgi system, a component based virtual reality system. The component-based nature of the system makes it easy to slot together components to create applications. This enabled me to concentrate on the gesture based interface without having to take time to work out details of drawing to the screen or talking to the network.

The design issues for this system are the selection of gestures and features that can be used to differentiate between these gestures, as well as methods of classifying the feature values into classes, one class for each gesture in the system. The gestures are chosen to be independent, the features are easily and quickly calculated.

Six gestures are recognised by the system at present, gestures chosen to control a paint box application. These gestures are planar, single stroke and independent. Due to this independence the gestures can be classified using three simple features, the curvature, the absolute curvature and the length of the curve divided by the displacement.

## 7.2 System

The core system is concerned with the calculation of feature values and the classification of the input. The system has to be able to work in real time, so lengthy calculations are to be avoided. The work concentrates more on developing features that can be used to distinguish the gestures than applying a brute-force classification algorithm to the gestures in order to classify them. This method has resulted in a system that can receive points at over 100/s and not fall behind what the user is doing.

The recognisors are simple by design, the first is a solid-cutoff system. Bounds are set by the system. If the feature value is within those bounds it is acceptable, otherwise not. The second is an error-minimisation system. The gesture that has the smallest total error between the actual feature values and ideal feature values must the gesture that the user intended. The simplicity of the recognisors, along with the fact that the feature values are only calculated once the gesture is completed ensures that the system is as close to real-time as is possible.

Both the gestures and the features are designed to be added to or replaced without major difficulty. This extensibility of the system allows a developer to adapt the system to control different applications without rewriting major parts of the system.

The recognisors developed are successful with an average accuracy of 81% across all the gestures. The accuracy with which the system recognises a user's first attempt was 79 % indicating that this system can be used effectively even by people with no prior experience in gesture based systems, or even with virtual reality. Once the user had some experience, the accuracy rose to 83%. This did not require more than a few minutes practice so conceivably more experience can result in even greater accuracy than that.

## 7.2 Future Work

Future work includes increasing the number of features that the system knows how to calculate and possibly automating the adding of gestures to the system. The recognisors could also be enhanced further. The bounds for the solid-cutoff algorithm need more study, possibly including a method for a user to give a few test gestures to the system before beginning and have the bounds adjusted for that user. For the error minimisation algorithm, the calculation of error needs enhancing to reduce the number of misidentified gestures. Some form of ignoring incorrect gestures is needed, possibly ignoring gestures with an error above a certain value. The system can also be extended to allow gestures using both hands, allowing more variation on gestures

# References


[1] Baudul, T. CHARADE: Remote Control of Objects using Free-Hand Gestures. Found at http://www.lri.fr/~thomas/Docs/cacm.html , July 1998

[2] The Hutchinson Dictionary of Science. Helicon Publishing, Oxford, 1993. pg 627

[3] Rubine, R. Specifying Gestures by Example. SIGGraph 1991. Pp 329-337

[4] Searles, D., Smith, J., Baratoff, G., Bohmueller,B. Recognition of Signals for Combat Formations and Battle Drills. http://www.hitl.washington.edu/projects/knowledge_base/virtual-worlds/JOVE/Articles/dsgbjsbb.txt

[5] Newby, G.B. Gesture Recognition Using Statistical Similarity. Presented at "*Virtual Reality and Persons with Disabilities*" California State University 1993

[6] Long, A.C. Dissertation Proposal: Improving Gestures and Interaction Techniques for Pen-Based User Interfaces. Found at http://www.cs.berkeley.edu/~allanl/research/proposal/ , Aug 1998.

[7] The Oxford Paperback Dictionary. Oxford University Press, Oxford, 1990. pg 411

# Appendix

No Practice, Dominant hand

| User | Gesture | Length of list | Curvature | AbsCurvature | Len/Disp | Recognise1 | Recognise2 |
|---|---|---|---|---|---|---|---|
| 1 | Select | 14 | 341 | 341 | 20 | Yes | Yes |
| | Group | 32 | 893 | 904 | 5.6 | No | Yes |
| | Move | 14 | 287 | 343 | 2.1 | No | No |
| | Delete | 10 | 294 | 331 | 3.9 | Yes | Yes |
| | OK | 9 | 84 | 130 | 1.4 | Yes | Yes |
| | Colour | 20 | 113 | 736 | 2.5 | No | Yes |
| 2 | Select | 17 | 399 | 399 | 7.4 | No | No |
| | Group | 31 | 663 | 782 | 3.3 | No | No |
| | Move | 19 | 515 | 584 | 2.2 | No | Yes |
| | Delete | 14 | 299 | 341 | 4.2 | Yes | Yes |
| | OK | 9 | 100 | 159 | 1.7 | Yes | Yes |
| | Colour | 25 | 13 | 787 | 3.2 | Yes | Yes |
| 3 | Select | 23 | 606 | 606 | 11 | No | No |
| | Group | 49 | 1146 | 1170 | 4.5 | No | Yes |
| | Move | 18 | 446 | 593 | 2.7 | No | Yes |
| | Delete | 19 | 214 | 341 | 2.8 | No | No |
| | OK | 11 | 102 | 146 | 1.4 | Yes | Yes |
| | Colour | 45 | 147 | 1104 | 6.2 | No | Yes |
| 4 | Select | 19 | 364 | 364 | 14 | Yes | Yes |
| | Group | 26 | 1002 | 1002 | 3.1 | No | Yes |
| | Move | 13 | 21 | 403 | 2.1 | No | No |
| | Delete | 14 | 305 | 328 | 4 | No | Yes |
| | OK | 9 | 133 | 161 | 146 | No | No |
| | Colour | 19 | 132 | 806 | 2.8 | No | Yes |
| 5 | Select | 21 | 341 | 371 | 14 | Yes | Yes |
| | Group | 41 | 1087 | 1101 | 4.6 | No | Yes |
| | Move | 11 | 367 | 396 | 2.1 | No | Yes |
| | Delete | 18 | 309 | 387 | 3.5 | No | Yes |
| | OK | 11 | 108 | 154 | 1.4 | Yes | Yes |
| | Colour | 12 | 150 | 718 | 2.6 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Select | 14 | 306 | 306 | 25 | No | Yes |
| | Group | 40 | 878 | 903 | 5.1 | No | Yes |
| | Move | 14 | 467 | 523 | 2.6 | No | Yes |
| | Delete | 12 | 276 | 333 | 2.9 | No | Yes |
| | OK | 9 | 87 | 168 | 1.3 | Yes | Yes |
| | Colour | 17 | 153 | 750 | 2.3 | No | Yes |
| 7 | Select | 19 | 276 | 379 | 13 | Yes | Yes |
| | Group | 28 | 235 | 635 | 3.8 | No | No |
| | Move | 13 | 354 | 476 | 2.2 | No | Yes |
| | Delete | 13 | 229 | 352 | 5.1 | No | Yes |
| | OK | 8 | 100 | 141 | 1.4 | Yes | Yes |
| | Colour | 25 | 168 | 1150 | 3.6 | Yes | Yes |
| 8 | Select | 15 | 323 | 323 | 6.5 | No | No |
| | Group | 43 | 480 | 516 | 3.4 | No | No |
| | Move | 21 | 562 | 601 | 3 | No | Yes |
| | Delete | 13 | 262 | 323 | 2.6 | No | Yes |
| | OK | 11 | 113 | 156 | 1.3 | Yes | Yes |
| | Colour | 32 | 106 | 872 | 3.9 | Yes | Yes |
| 9 | Select | User had prior experience with the system | | | | | |
| | Group | | | | | | |
| | Move | | | | | | |
| | Delete | | | | | | |
| | OK | | | | | | |
| | Colour | | | | | | |
| 10 | Select | 17 | 383 | 383 | 7 | No | No |
| | Group | 39 | 1167 | 1167 | 4.4 | No | Yes |
| | Move | 14 | 134 | 547 | 2.2 | No | No |
| | Delete | 13 | 265 | 293 | 3.7 | No | Yes |
| | OK | 9 | 101 | 141 | 1.3 | Yes | Yes |
| | Colour | 17 | 81 | 757 | 21 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | Select | 23 | 346 | 357 | 33 | Yes | Yes |
| | Group | 34 | 591 | 635 | 9.4 | No | No |
| | Move | 18 | 272 | 330 | 3.6 | Yes | Yes |
| | Delete | 18 | 272 | 330 | 3.6 | Yes | Yes |
| | OK | 10 | 65 | 151 | 1.3 | No | Yes |
| | Colour | 27 | 636 | 725 | 3.2 | No | No |
| 12 | Select | 9 | 335 | 335 | 16 | Yes | Yes |
| | Group | 35 | 1025 | 1165 | 3.9 | No | Yes |
| | Move | 21 | 274 | 636 | 2.7 | No | No |
| | Delete | 14 | 259 | 314 | 5.3 | Yes | Yes |
| | OK | 9 | 115 | 169 | 1.3 | Yes | Yes |
| | Colour | 29 | 112 | 1055 | 2.1 | No | Yes |
| 13 | Select | 27 | 377 | 400 | 17 | Yes | Yes |
| | Group | 33 | 1104 | 1119 | 3.4 | No | Yes |
| | Move | 20 | 371 | 694 | 2.5 | No | Yes |
| | Delete | 15 | 257 | 318 | 4.9 | Yes | Yes |
| | OK | 11 | 138 | 178 | 1.7 | Yes | Yes |
| | Colour | 20 | 168 | 826 | 4.1 | Yes | Yes |
| 14 | Select | 12 | 532 | 332 | 5.9 | No | No |
| | Group | 34 | 1091 | 1103 | 4.2 | No | Yes |
| | Move | 12 | 433 | 517 | 202 | No | Yes |
| | Delete | 14 | 316 | 358 | 2.7 | No | Yes |
| | OK | 9 | 120 | 135 | 1.4 | Yes | Yes |
| | Colour | 23 | 155 | 854 | 3 | Yes | Yes |
| 15 | Select | 20 | 333 | 421 | 9.2 | No | No |
| | Group | 29 | 977 | 977 | 3.9 | No | Yes |
| | Move | 24 | 118 | 737 | 2.3 | No | No |
| | Delete | 18 | 254 | 332 | 4.2 | Yes | Yes |
| | OK | 12 | 151 | 152 | 1.4 | Yes | Yes |
| | Colour | 27 | 110 | 896 | 4.4 | Yes | Yes |

With practise, dominant hand

| User | Gesture | Length of list | Curvature | AbsCurvature | Len/Disp | Recognise1 | Recognise2 |
|---|---|---|---|---|---|---|---|
| 1 | Select | 9 | 270 | 27070 | 5.6 | No | No |
| | Group | 31 | 664 | 667 | 5.8 | No | No |
| | Move | 13 | 131 | 158 | 2.0 | No | No |
| | Delete | 14 | 303 | 327 | 3.5 | No | Yes |
| | OK | 10 | 93 | 162 | 1.5 | Yes | Yes |
| | Colour | 20 | 150 | 822 | 3.4 | Yes | Yes |
| 2 | Select | 17 | 371 | 371 | 9.3 | Yes | No |
| | Group | 42 | 770 | 815 | 4.7 | No | No |
| | Move | 30 | 289 | 462 | 2.9 | No | No |
| | Delete | 17 | 267 | 305 | 3 | No | Yes |
| | OK | 10 | 116 | 143 | 1.8 | Yes | Yes |
| | Colour | 26 | 133 | 892 | 4.7 | No | Yes |
| 3 | Select | 21 | 399 | 399 | 11 | Yes | Yes |
| | Group | 54 | 1071 | 1103 | 4.6 | No | Yes |
| | Move | 20 | 434 | 475 | 1.7 | No | Yes |
| | Delete | 1.9 | 271 | 283 | 4.7 | No | Yes |
| | OK | 12 | 87 | 123 | 1.3 | Yes | Yes |
| | Colour | 33 | 179 | 860 | 3.1 | Yes | Yes |
| 4 | Select | 12 | 301 | 301 | 30 | No | Yes |
| | Group | 24 | 982 | 982 | 3.4 | No | Yes |
| | Move | 14 | 385 | 519 | 3.5 | Yes | Yes |
| | Delete | 14 | 278 | 329 | 4.3 | Yes | Yes |
| | OK | 8 | 123 | 157 | 1.5 | Yes | Yes |
| | Colour | 18 | 127 | 768 | 2.1 | No | Yes |
| 5 | Select | 20 | 305 | 311 | 54 | No | Yes |
| | Group | 41 | 1025 | 1031 | 5.3 | Yes | Yes |
| | Move | 19 | 431 | 560 | 2.4 | No | Yes |
| | Delete | 20 | 276 | 353 | 3.1 | No | Yes |
| | OK | 11 | 97 | 150 | 1.3 | Yes | Yes |
| | Colour | 22 | 62 | 798 | 2.5 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Select | 18 | 352 | 352 | 19 | Yes | Yes |
| | Group | 26 | 852 | 852 | 4.8 | No | Yes |
| | Move | 15 | 425 | 486 | 2.4 | No | Yes |
| | Delete | 18 | 289 | 381 | 3.4 | No | Yes |
| | OK | 12 | 140 | 188 | 1.5 | Yes | Yes |
| | Colour | 22 | 174 | 777 | 2.3 | No | Yes |
| 7 | Select | 18 | 355 | 355 | 17 | Yes | Yes |
| | Group | 41 | 788 | 791 | 4.6 | No | No |
| | Move | 21 | 511 | 653 | 3.5 | No | Yes |
| | Delete | 15 | 267 | 295 | 4.4 | No | Yes |
| | OK | 9 | 118 | 139 | 1.5 | Yes | Yes |
| | Colour | 23 | 10 | 949 | 3.4 | Yes | Yes |
| 8 | Select | 15 | 336 | 336 | 9.4 | Yes | No |
| | Group | 34 | 1030 | 1030 | 4.5 | No | Yes |
| | Move | 16 | 405 | 436 | 2.5 | No | Yes |
| | Delete | 14 | 265 | 310 | 4.3 | Yes | Yes |
| | OK | 8 | 90 | 139 | 1.4 | Yes | Yes |
| | Colour | 26 | 148 | 829 | 3.1 | Yes | Yes |
| 9 | Select | 29 | 393 | 393 | 19 | Yes | Yes |
| | Group | 61 | 749 | 796 | 4 | No | No |
| | Move | 30 | 414 | 505 | 4.2 | Yes | Yes |
| | Delete | 19 | 312 | 313 | 4.1 | No | Yes |
| | OK | 12 | 115 | 125 | 1.5 | Yes | Yes |
| | Colour | 21 | 153 | 490 | 3.7 | No | No |
| 10 | Select | 17 | 347 | 347 | 11 | Yes | Yes |
| | Group | 30 | 982 | 988 | 5 | No | Yes |
| | Move | 14 | 410 | 421 | 2.6 | No | Yes |
| | Delete | 13 | 258 | 288 | 4.8 | No | Yes |
| | OK | 6 | 99 | 104 | 1.2 | Yes | Yes |
| | Colour | 16 | 111 | 801 | 2.9 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | Select | 16 | 305 | 341 | 6. | No | No |
| | Group | 29 | 1071 | 1071 | 2.2 | No | Yes |
| | Move | 17 | 393 | 428 | 2.7 | No | Yes |
| | Delete | 15 | 277 | 316 | 4.4 | Yes | Yes |
| | OK | 6 | 74 | 92 | 1.1 | Yes | Yes |
| | Colour | 17 | 144 | 912 | 3.1 | Yes | Yes |
| 12 | Select | 23 | 302 | 355 | 19 | No | Yes |
| | Group | 440 | 954 | 1042 | 4.9 | No | Yes |
| | Move | 24 | 466 | 571 | 2.8 | No | Yes |
| | Delete | 19 | 271 | 376 | 5.4 | Yes | Yes |
| | OK | 11 | 98 | 193 | 1.3 | Yes | Yes |
| | Colour | 23 | 128 | 752 | 2.0 | No | Yes |
| 13 | Select | 18 | 395 | 395 | 15 | Yes | Yes |
| | Group | 45 | 876 | 892 | 4.2 | No | Yes |
| | Move | 17 | 413 | 458 | 2.1 | No | Yes |
| | Delete | 21 | 271 | 346 | 4.6 | Yes | Yes |
| | OK | 14 | 84 | 166 | 1.3 | Yes | Yes |
| | Colour | 15 | 148 | 752 | 2.6 | Yes | Yes |
| 14 | Select | 16 | 373 | 373 | 7.8 | No | No |
| | Group | 33 | 540 | 542 | 4.4 | No | No |
| | Move | 18 | 424 | 542 | 3.4 | Yes | Yes |
| | Delete | 20 | 440 | 508 | 3.1 | No | No |
| | OK | 10 | 127 | 130 | 1.6 | Yes | Yes |
| | Colour | 22 | 157 | 345 | 3.9 | Yes | Yes |
| 15 | Select | 28 | 323 | 365 | 11 | Yes | No |
| | Group | 32 | 920 | 928 | 3.3 | No | Yes |
| | Move | 25 | 390 | 539 | 2.1 | No | Yes |
| | Delete | 31 | 460 | 484 | 4.0 | No | Yes |
| | OK | 10 | 93 | 125 | 1.1 | Yes | Yes |
| | Colour | 22 | 164 | 980 | 4 | Yes | Yes |

With practice, non-dominant hand

| User | Gesture | Length of list | Curvature | AbsCurvature | Len/Disp | Recognise1 | Recognise2 |
|---|---|---|---|---|---|---|---|
| 1 | Select | 10 | 281 | 281 | 4.4 | No | No |
| | Group | 36 | 626 | 634 | 4.8 | No | No |
| | Move | 15 | 489 | 503 | 2.7 | No | Yes |
| | Delete | 15 | 274 | 314 | 3.5 | Yes | Yes |
| | OK | 11 | 143 | 162 | 1.7 | Yes | Yes |
| | Colour | 17 | 492 | 786 | 4.4 | No | No |
| 2 | Select | 20 | 352 | 353 | 21 | Yes | Yes |
| | Group | 45 | 893 | 948 | 6.8 | No | Yes |
| | Move | 24 | 414 | 532 | 2.3 | No | Yes |
| | Delete | 19 | 279 | 359 | 3.6 | Yes | Yes |
| | OK | 14 | 101 | 198 | 1.7 | Yes | Yes |
| | Colour | 14 | 131 | 941 | 3.5 | Yes | Yes |
| 3 | Select | 21 | 488 | 488 | 5 | No | No |
| | Group | 52 | 739 | 756 | 5 | No | No |
| | Move | 14 | 406 | 464 | 2.6 | No | Yes |
| | Delete | 24 | 268 | 351 | 3.6 | Yes | Yes |
| | OK | 12 | 114 | 144 | 1.5 | Yes | Yes |
| | Colour | 34 | 277 | 991 | 4.2 | No | Yes |
| 4 | Select | 15 | 343 | 343 | 12 | Yes | Yes |
| | Group | 26 | 978 | 978 | 3.7 | No | Yes |
| | Move | 18 | 467 | 517 | 2.7 | No | Yes |
| | Delete | 17 | 314 | 351 | 3.6 | No | Yes |
| | OK | 11 | 127 | 145 | 1.3 | Yes | Yes |
| | Colour | 29 | 113 | 857 | 2.9 | No | Yes |
| 5 | Select | 16 | 317 | 317 | 19 | No | Yes |
| | Group | 43 | 1033 | 1033 | 6.1 | Yes | Yes |
| | Move | 22 | 339 | 576 | 2.6 | No | Yes |
| | Delete | 18 | 277 | 329 | 3.8 | Yes | Yes |
| | OK | 12 | 115 | 145 | 1.5 | Yes | Yes |
| | Colour | 22 | 147 | 845 | 3.3 | Yes | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Select | 16 | 343 | 343 | 10 | Yes | No |
| | Group | 32 | 960 | 960 | 4.7 | No | Yes |
| | Move | 13 | 427 | 453 | 2.4 | No | Yes |
| | Delete | 13 | 276 | 335 | 4.1 | Yes | Yes |
| | OK | 10 | 119 | 210 | 1.4 | Yes | Yes |
| | Colour | 16 | 152 | 751 | 2.2 | No | Yes |
| 7 | Select | 17 | 386 | 386 | 5.8 | No | No |
| | Group | 42 | 981 | 983 | 5.4 | No | Yes |
| | Move | 18 | 459 | 494 | 3.4 | Yes | Yes |
| | Delete | 14 | 279 | 313 | 3.4 | No | Yes |
| | OK | 13 | 175 | 178 | 1.9 | Yes | Yes |
| | Colour | 21 | 144 | 893 | 5.1 | No | Yes |
| 8 | Select | 14 | 317 | 317 | 9.5 | No | No |
| | Group | 34 | 884 | 893 | 4.1 | No | Yes |
| | Move | 19 | 380 | 433 | 2.5 | No | Yes |
| | Delete | 14 | 246 | 293 | 4.8 | No | Yes |
| | OK | 13 | 123 | 151 | 1.5 | Yes | Yes |
| | Colour | 26 | 143 | 853 | 3.6 | Yes | Yes |
| 9 | Select | 21 | 469 | 469 | 32 | No | Yes |
| | Group | 34 | 655 | 655 | 3 | No | Yes |
| | Move | 20 | 321 | 359 | 2.4 | No | Yes |
| | Delete | 15 | 303 | 315 | 3.6 | No | Yes |
| | OK | 9 | 130 | 158 | 1.7 | Yes | Yes |
| | Colour | 18 | 21 | 669 | 4.4 | No | Yes |
| 10 | Select | 12 | 329 | 329 | 11 | Yes | No |
| | Group | 30 | 272 | 368 | 4 | No | No |
| | Move | 16 | 107 | 568 | 2.7 | No | No |
| | Delete | 11 | 251 | 332 | 4.9 | Yes | Yes |
| | OK | 7 | 75 | 92 | 1.2 | Yes | Yes |
| | Colour | 17 | 114 | 723 | 2.3 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | Select | 18 | 356 | 368 | 5.3 | No | No |
| | Group | 29 | 970 | 970 | 4.7 | No | Yes |
| | Move | 20 | 144 | 604 | 2.8 | No | No |
| | Delete | 15 | 249 | 325 | 5.3 | Yes | Yes |
| | OK | 9 | 91 | 135 | 1.3 | Yes | Yes |
| | Colour | 17 | 160 | 811 | 3.4 | Yes | Yes |
| 12 | Select | 25 | 337 | 423 | 22 | Yes | Yes |
| | Group | 39 | 812 | 820 | 4.7 | No | No |
| | Move | 24 | 443 | 532 | 3.0 | Yes | Yes |
| | Delete | 20 | 283 | 442 | 4.2 | Yes | Yes |
| | OK | 12 | 88 | 125 | 1.4 | Yes | Yes |
| | Colour | 25 | 135 | 800 | 2.4 | No | Yes |
| 13 | Select | 25 | 337 | 461 | 19 | Yes | Yes |
| | Group | 39 | 1019 | 1019 | 4.9 | No | Yes |
| | Move | 18 | 127 | 556 | 2.7 | No | No |
| | Delete | 17 | 266 | 292 | 4.1 | No | Yes |
| | OK | 12 | 130 | 155 | 1.5 | Yes | Yes |
| | Colour | 19 | 140 | 674 | 1.9 | No | Yes |
| 14 | Select | 12 | 147 | 190 | 4.6 | No | No |
| | Group | 26 | 883 | 892 | 4 | No | Yes |
| | Move | 13 | 447 | 456 | 2.4 | No | Yes |
| | Delete | 16 | 277 | 352 | 4.0 | Yes | Yes |
| | OK | 9 | 114 | 142 | 1.6 | Yes | Yes |
| | Colour | 22 | 161 | 889 | 4.1 | Yes | Yes |
| 15 | Select | 28 | 328 | 372 | 11 | Yes | No |
| | Group | 36 | 949 | 949 | 3.3 | No | Yes |
| | Move | 29 | 257 | 654 | 2.1 | No | No |
| | Delete | 31 | 188 | 386 | 3.5 | No | Yes |
| | OK | 15 | 81 | 176 | 1.2 | Yes | Yes |
| | Colour | 30 | 134 | 879 | 3.6 | Yes | Yes |

With practise, dominant hand, no head-mounted display

| User | Gesture | Length of list | Curvature | AbsCurvature | Len/Disp | Recognise1 | Recognise2 |
|---|---|---|---|---|---|---|---|
| 1 | Select | 24 | 365 | 365 | 13 | Yes | Yes |
| | Group | 37 | 975 | 975 | 6 | No | Yes |
| | Move | 24 | 375 | 476 | 2.7 | No | Yes |
| | Delete | 22 | 296 | 353 | 3.9 | Yes | Yes |
| | OK | 14 | 313 | 355 | 1.9 | No | No |
| | Colour | 23 | 156 | 542 | 3.2 | No | No |
| 2 | Select | 30 | 411 | 477 | 5.7 | No | No |
| | Group | 41 | 906 | 906 | 9.3 | No | Yes |
| | Move | 21 | 371 | 526 | 2.8 | No | Yes |
| | Delete | 21 | 303 | 333 | 3.3 | No | Yes |
| | OK | 15 | 121 | 170 | 2.6 | No | Yes |
| | Colour | 17 | 95 | 327 | 1.9 | No | No |
| 3 | Select | 14 | 345 | 245 | 14 | Yes | Yes |
| | Group | 29 | 963 | 963 | 5.5 | No | Yes |
| | Move | 17 | 460 | 496 | 2.7 | No | Yes |
| | Delete | 14 | 309 | 318 | 4.7 | No | Yes |
| | OK | 8 | 58 | 303 | 1.5 | No | Yes |
| | Colour | 27 | 142 | 890 | 3.4 | Yes | Yes |
| 4 | Select | 20 | 350 | 350 | 22 | Yes | Yes |
| | Group | 35 | 1008 | 1008 | 5.3 | No | No |
| | Move | 29 | 215 | 589 | 3.6 | No | No |
| | Delete | 15 | 309 | 314 | 3.9 | No | Yes |
| | OK | 10 | 115 | 120 | 1.5 | Yes | Yes |
| | Colour | 35 | 113 | 900 | 3.6 | Yes | Yes |
| 5 | Select | 24 | 359 | 354 | 3.6 | Yes | Yes |
| | Group | 48 | 984 | 986 | 6.1 | No | Yes |
| | Move | 24 | 473 | 469 | 3.1 | Yes | Yes |
| | Delete | 25 | 283 | 346 | 4 | Yes | Yes |
| | OK | 18 | 31 | 47 | 2.12 | No | Yes |
| | Colour | 32 | 127 | 806 | 3.4 | Yes | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | Select | 22 | 363 | 363 | 19 | Yes | Yes |
| | Group | 36 | 780 | 780 | 5.2 | No | No |
| | Move | 22 | 159 | 584 | 2.6 | No | No |
| | Delete | 14 | 263 | 313 | 3.9 | Yes | Yes |
| | OK | 12 | 129 | 178 | 1.7 | Yes | Yes |
| | Colour | 17 | 140 | 709 | 1.9 | No | Yes |
| 7 | Select | 18 | 360 | 360 | 8.5 | No | No |
| | Group | 36 | 993 | 993 | 6.9 | No | Yes |
| | Move | 19 | 387 | 475 | 3.3 | Yes | Yes |
| | Delete | 16 | 279 | 315 | 4.2 | Yes | Yes |
| | OK | 8 | 115 | 142 | 1.5 | Yes | Yes |
| | Colour | 24 | 215 | 870 | 3.9 | No | Yes |
| 8 | Select | 19 | 314 | 314 | 9.4 | No | No |
| | Group | 38 | 1006 | 1006 | 5.9 | Yes | Yes |
| | Move | 24 | 453 | 491 | 3.2 | Yes | Yes |
| | Delete | 22 | 286 | 340 | 5.1 | Yes | Yes |
| | OK | 12 | 136 | 198 | 1.5 | Yes | Yes |
| | Colour | 26 | 157 | 773 | 3 | No | Yes |
| 9 | Select | 14 | 352 | 352 | 33 | Yes | Yes |
| | Group | 32 | 860 | 860 | 4.4 | No | Yes |
| | Move | 16 | 406 | 456 | 2.9 | No | Yes |
| | Delete | 14 | 316 | 316 | 3.6 | No | Yes |
| | OK | 9 | 119 | 153 | 1.8 | Yes | Yes |
| | Colour | 17 | 162 | 504 | 4.5 | No | No |
| 10 | Select | 15 | 353 | 353 | 7.3 | No | No |
| | Group | 31 | 1027 | 1027 | 5.5 | Yes | Yes |
| | Move | 15 | 412 | 438 | 3.4 | No | Yes |
| | Delete | 14 | 269 | 294 | 4.8 | No | Yes |
| | OK | 8 | 105 | 140 | 1.3 | Yes | Yes |
| | Colour | 18 | 127 | 703 | 2.2 | No | Yes |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 11 | Select | 23 | 394 | 394 | 7.6 | No | No |
| | Group | 26 | 1021 | 1021 | 3.7 | No | Yes |
| | Move | 21 | 442 | 480 | 3.4 | Yes | Yes |
| | Delete | 15 | 285 | 303 | 4.1 | No | Yes |
| | OK | 10 | 139 | 183 | 1.4 | Yes | Yes |
| | Colour | 32 | 120 | 1006 | 5.5 | No | Yes |
| 12 | Select | 22 | 363 | 363 | 14 | Yes | Yes |
| | Group | 35 | 447 | 447 | 4.9 | No | No |
| | Move | 19 | 521 | 541 | 2.5 | No | Yes |
| | Delete | 17 | 291 | 304 | 5.1 | Yes | Yes |
| | OK | 8 | 101 | 139 | 1.3 | Yes | Yes |
| | Colour | 25 | 15 | 932 | 2.8 | No | Yes |
| 13 | Select | 40 | 402 | 402 | 9.1 | No | No |
| | Group | 38 | 1056 | 1056 | 6.2 | Yes | Yes |
| | Move | 27 | 510 | 576 | 3.6 | No | Yes |
| | Delete | 20 | 253 | 343 | 4.3 | Yes | Yes |
| | OK | 15 | 161 | 172 | 2.2 | No | Yes |
| | Colour | 30 | 14 | 818 | 3.6 | Yes | Yes |
| 14 | Select | 27 | 401 | 401 | 7.5 | No | No |
| | Group | 44 | 1054 | 1054 | 7 | Yes | Yes |
| | Move | 23 | 210 | 626 | 3.3 | No | No |
| | Delete | 18 | 293 | 302 | 3.9 | Yes | Yes |
| | OK | 16 | 22 | 360 | 1.8 | No | Yes |
| | Colour | 36 | 174 | 834 | 4.3 | Yes | Yes |
| 15 | Select | 24 | 356 | 356 | 35 | Yes | Yes |
| | Group | 26 | 987 | 987 | 3.8 | No | Yes |
| | Move | 22 | 171 | 551 | 2.2 | No | No |
| | Delete | 25 | 293 | 321 | 3.9 | Yes | Yes |
| | OK | 13 | 131 | 170 | 1.9 | Yes | Yes |
| | Colour | 24 | 138 | 796 | 2.7 | No | Yes |