The University of New South Wales

Schools of Electrical Engineering
and
Computer Science & Engineering

GRASP:
Recognition of Australian Sign Language Using Instrumented Gloves

Waleed Kadous

Submitted in part for the degree of:
Bachelor of Computer Engineering

October 1995

Supervisor: Andrew Taylor

Abstract

Instrumented gloves – gloves equipped with sensors for detecting finger bend, hand position and orientation – were conceived to allow a more natural interface to computers. However, the extension of their use for recognising sign language, and in this case Auslan (Australian Sign Language), is possible. Several researchers have already explored these possibilities and have successfully achieved finger-spelling recognition with high levels of accuracy, but progress in the recognition of sign language as a whole has been limited.

Most, however, have focussed on the use of expensive hardware and/or complex learning techniques. An attempt is made in this thesis to explore the possibilities using cheap hardware (a PowerGlove worth approximately $100) and computationally simple approaches.

95 discrete Auslan signs were selected, and five different users provided instances of these (there were 6650 samples collected in total). Features were extracted and simple classification such as instance-based learning and symbolic learning techniques were validated. The feature sets were then combined and the overall accuracy tested.

With 95 signs, some selected for their similarity to each other, the accuracy obtained was approximately 80 per cent. It is computationally simple and fast enough to fit on a small portable computer. This compares extremely favourably with the state of the art, with a lexicon almost two and a half times larger than any previous published results.

Potentials for future expansion to a large lexicon system and its expansion to handle inter-signer recognition are explored.

For more information, see the following Web page:

`http://www.cse.unsw.edu.au/~waleed/thesis.html`

Contents

List of Figures

List of Tables

In the Name of the One True God, the Beneficent, the Merciful

Acknowledgements

I first of all would like to thank Allah, the Creator, for giving me the ability to complete this thesis, and for guiding me while doing it.

My thanks go to my family, who have supported me, and especially my sister, Hediah, for putting up with my cooking, and for her wonderful spaghetti.

Also to the wonderful people in the Deaf community, who have put up with my bugging them, but still teaching and helping me a great deal. In particular, David Parker and Glenn Welldon at *Silent Messenger*, Lynne Rogers and Michelle Macguire for teaching me the little Auslan that I know, and Anne Horwood for putting me in touch with people. Especially, I would like to thank Andrew Young, who volunteered his time to provide me with some sample signs.

Academically, I would like to thank Adam Schembri for helping me with some of the technical aspects of Auslan, for helping me correct some errors I had made in the selection of signs and for providing me with sample signs. Also, Peter Vamplew for providing the handshape datasets and for offering advice. Similarly my thanks to all the people who helped with this research through the 'net – Chris Hand, Thad Starner, David Sturman, James Kramer, Dean Rubine, Diego Montefusco and Toni Emerson among them.

Fifthly, to the staff of the School of Computer Science and Engineering for trusting me with a thesis that is out of their usual domain of study and supporting me with the resources I needed to complete this thesis. In particular to Andrew Taylor, my supervisor, to Charles Willock for giving me ideas on how to look at the data and to Phil Preston for dealing with the equipment. Also to Simon Warfield, for his help with the implementation of the IBL algorithm; to Paul Wong for his suggestions and discussion, and to Ross Quinlan for his advice on using IBL and his suggestions for further work.

Finally, in every body of work, there are innumerable people who contribute, but are forgotten inadvertently, and for this I apologise. This is because of a shortcoming on my part and not on yours.

Chapter 1

Introduction, goals and objectives

What Are We Trying To Do Here?

1.1 Introduction

Traditionally, the first line of a thesis is supposed to be some deeply meaningful and insightful statement meant to pull the reader in.

Oh well ...

This investigation analyses the data from an instrumented glove – a glove that returns information on finger position, hand position and/or orientation – for use in recognition of the signs that are part of Australian Sign Language (Auslan). A system is developed for recognising these signs, which is termed GRASP (Glove-based Recognition of Auslan using Simple Processing)[1].

The results will show that despite the noise and accuracy constraints of the equipment used, reasonable accuracy rates were achieved. More importantly, the techniques developed here can be applied to improved hardware when (in accordance with an interface device version of Moore's Law[2]) it becomes available. While GRASP may not be

[1]So it's lame. But all the names having "glove" in them have been taken :-).

[2]Moore's Law states that the amount of computing power available for a given price doubles

viable for everyday use, it is possible that with improved instrumented gloves, future systems may be.

1.2 Goals

The goals of this investigation were:

- To investigate the possibility of recognition of a subset of the single discrete signs found in Auslan using simple techniques and cheap equipment.
- To investigate the feasibility of extension of GRASP into a real-time, continuous, large-lexicon Auslan recognition system.

1.3 Objectives

The objectives of the investigation, and the plan followed in its execution are summarised below:

- To review literature in the field and to summarise certain background material necessary for an informed investigation.
- To select the equipment and resources required for the investigation, while remaining within the financial and temporal constraints of an undergraduate thesis.
- To determine on the exact avenues to be explored, with a view to exploring an original method that appears feasible and assessing its efficacy.
- To construct the framework in which data would be gathered and tested and if necessary, write software for this purpose. This includes:
  - Creating software to interface the glove to the computer if such software did not exist.

once every two years. In fact, Moore's Law was made to look conservative when the improvements in instrumented glove technology that occurred while this investigation was being carried out are considered.

  - Creating software to read the data from the glove and record it for later use.
  - Determining what data was to be gathered, including the list of signs that were to be sampled.
  - Constructing a protocol for the recording of the training and testing examples, with a view to minimising random variations.
  - To create a tool for viewing gathered data to provide insigns into the data and to validate its correctness.
  - Creating the necessary conversion program to convert between the output format of the glove and the information required by learning algorithms.

- To examine the data carefully and to try to extract features of the data that would serve as good discriminants between signs when used with the learning algorithms.
- On the basis of the above, assess the feasibility of the system in terms of accuracy, real-time capability, cost and expandability.
- If time permitted, to consider inter-signer recognition (recognition of signs by people on whom GRASP was not trained), in addition to the intra-signer recognition already considered.
- To document the results of the investigation[3] in a manner comprehensible to people in the field and to anyone interested in the findings.

1.4 Potential Applications

The techniques developed here could form the basis of many potential applications.

These include:

1.4.1 Inverse mapping from Auslan to English for dictionary use

This is perhaps the application that would be easiest to build from this thesis. A user would put on an instrumented glove and make a sign that they wanted to know

[3] This document.

the meaning of. The software would then find the closest set of matching signs. The user could then examine the meaning of the word, in English, or perhaps it could be explained in terms of Auslan itself[4]. There are currently a variety of methods for indexing signs, but they are usually not as intuitive as making the sign itself[5].

This would be viable, since it uses discrete signs, and a high error rate is not a large problem, since several candidate signs can be returned. Furthermore, the algorithms discussed here can both be expanded to return multiple values[6].

The only problem with this is that it requires that GRASP be able to adapt to new signers – ie be able to successfully recognise signs by people other than those on whom it was trained.

1.4.2 Generalised Gesture Recognition

If effective ways of recognising signs are discovered and it is illustrated that these techniques are generalisable to gesture recognition, then this will add to the variety of tools that computer engineers and scientists have available to them to interpret gestures. Gesture recognition is a wider field than sign language recognition, since even hearing people use gestures; some professionally (e.g. dogmen at working sites, signalmen at airports and so on).

1.4.3 Another interface tool available to the Deaf community

In the hearing community we are seeing limited commercially viable voice recognition tools being developed. Although they are typically limited to a single user, and require that the user carefully pronounce the words, they are becoming more popular. There is no equivalent for members of the Deaf community. A system which recognised signs as a way of input to computers would give similar access to Deaf people.

[4] Plans are already underway to create a CD-ROM version of Trevor Johnston's Auslan Dictionary [Joh89]. It would not be difficult to record the definitions in Auslan itself as well as in English.

[5] One example of this is known as the Stokoe (*stoa-kee*) notation, which looks to the untrained eye as if it was lifted off the side of an Egyptian pyramid – essentially it appears to be a series of hieroglyphic symbols.

[6] By using a k-nearest neighbour approach with IBL1, rather than the 1-nearest neighbour, and with C4.5 by using "soft thresholds", where if a comparison being made on a particular attribute and the value of that attribute is close to the thing it is being compared against, we explore both alternatives.

1.4.4 Communication between Deaf and speaking people

Eventually, it might be possible to make a portable version of GRASP, taking input from the instrumented glove and outputting English. Originally, the system would only be capable of handling Signed English (a version of sign language where there is a one-to-one mapping from signs to English words, as compared to Auslan, which is really another language), but as natural language translation technology improves, it may be viable to go from Auslan to English. This would remove a great deal of the trouble members of the Deaf community have in interacting with non-Auslan speakers. Ideas for two way communication can be developed as Kramer [KL90, Kra91] has suggested, by having voice recognition software or a plain keyboard going the other way.

1.5 Overview of the report

We begin with an introduction to the field, covering some background information that is useful in understanding the issues. Other research in the area is then reviewed.

The hardware and software requirements are considered, and the choices made are then justified.

The software developed for this thesis is summarised and documented, and the techniques employed for data collection are explained.

The different techninques used in analysing the data and the techniques eventually employed are then explained, illustrated, tested and assessed for their accuracy.

These features are then put together and tested. Special attention paid to the potentials for expansion of the techniques to larger lexicon (i.e. greater number of signs) systems.

The conclusions we can draw from this work are documented, and a considerable number of avenues for future research are suggested.

I'll try to keep it as interesting as possible, and I hope you don't get bored.

Chapter 2

Previous work

What Background Material Would be Useful to Know, and Has Someone Done This Already?

2.1 Introduction

As computing power increased in the early 1980's, people began to consider attaching more complex peripheral devices to their computers. People began to understand the importance of the "man-machine" ("person-machine"?) interface. As Rheingold outlines [Rhe91], early researchers, from Ivan Sutherland to Richard Bolt at ArchMac[1] began to realise that the more natural the interface, the more useful the computer would become.

At the same time, various approaches were being made to improve techniques for computer-based pattern recognition, a task which was easy for humans, yet difficult for computers. There were a number of early failures, due to the huge computing power required to do such tasks. However, a number of approaches developed, such as neural networks (a version of the earlier "perceptrons"), Bayesian and other statistical methods, instance-based learning algorithms (such as IBL) and symbolic learning packages (such as ID3 and C4.5).

A spin-off of the great advances in interface design was the creation of "manual" input devices. There were of course the very conspicuous forms, which were to use

[1] As the Architecture Machine Group at MIT came to be known.

Sturman's words "devices for interaction via the hand and not 'whole-hand input'[2] devices". This included light-pens, styli, and mice. The first real "non-conspicuous" interaction device was the "Put-that-there" system, developed at ArchMac by Richard Bolt [Bol80]. This device used a combination of the newly commercialised Polhemus 6D tracking device and some simple voice recognition software. On the screen, the user saw objects. The user would then "point" at an object and say "Put that ..." move his/her finger to where he/she wanted the object to be, and say "there".

While this does not connect directly to the matter at hand[3], it laid the groundwork for future developments.

2.2 The field of study

The area of recognising gestures made by humans using computers is generally known as "gesture recognition". It is necessary to make a distinction between "posture recognition" and "gesture recognition". Posture recognition is the recognition of static positions of the body, while gesture recognition must include catering for dynamic changes in posture. Posture recognition is thus a subset of gesture recognition.

Note that there are no bounds on how the posture and gesture information may be obtained – it need not be hand-based at all, and in general can include facial gestures, body movement and so on. In general gesture recognition is understood to be the recognition of movements and postures that indicate a concept. For example, using a stylus (a computerised pen) for drawing objects would not be gesture recognition. However, if you were to use that stylus to "cross out" a drawing, indicating that you didn't want it, and the computer would recognise this as an indication of that fact, that would be considered gesture recognition. Recognition of sign language is a specific example of gesture recognition. So far, since sign languages tend to be manual in basis (although they are not exclusively so, and typically also include a component of facial gesture), most research in this area has focussed on measuring hand movement, and basing recognition on this, using one of a variety of approaches. Some work, however, has been done at recognising signs at a body level.

[2]This term comes from D. J. Sturman's PhD thesis [Stu92].

[3]Yes, that is an awful pun.

2.3 Physiology of the hand

To better understand sign language, it may be useful to have some background information about how the hand operates and how complex its movements can be. This requires an examination of the internal structures of the hand.

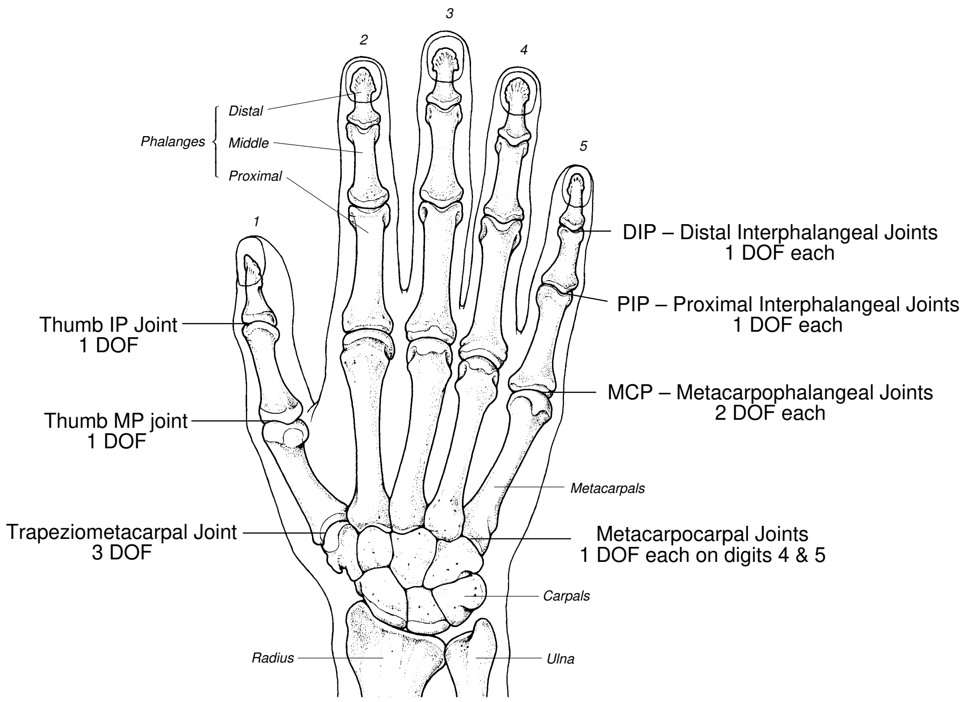


Figure 2.1: The joints in their hand together with their associated degrees of freedom.

Taken from [Stu92].

In figure 2.1 we can see that there are a total of 23 degrees of freedom available solely in the hand above the wrist, to which we usually add another six degrees of freedom, which cover its movement in 3 dimensions. A hand can move not only up-down (y-axis), left-right (x-axis) and forward-backward (z-axis), but also rotate about these axes. For examples, palms can face up or down (rotation about z-axis – known as *roll*), about the ball of the wrist – for example when indicating stop to being lame-wristed (rotation about the x-axis – known as *pitch*) and from side-to-side, as when "dealing cards" (rotation about the y-axis – known as *yaw*). A number of signs differ with respect to some of these factors only. For example, weight and doubt differ only in the yaw of the palms – weight has fingers pointing outwards and doubt has the blade of the hands parallel to the chest.

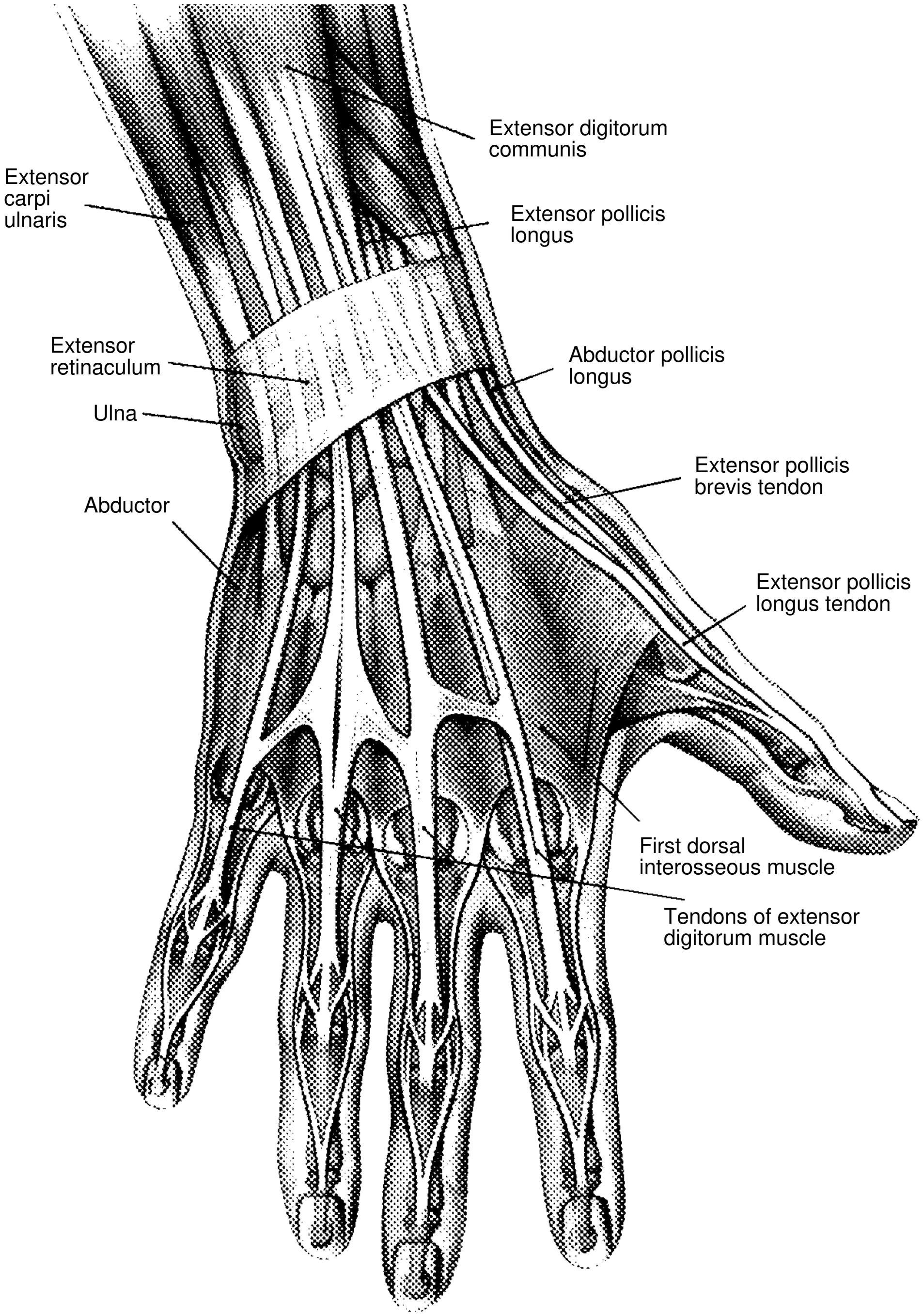


Figure 2.2: The tendons that are present in the hand. Again, note the high complexity of the muscles and tendons within the hand, leading to a wide range of possible motions.

Taken from [Stu92].

We can see the great number of muscles involved in motion of the hand in figure 2.2. Many of the muscles are actually in the forearm, and are connected via tendons to the hand itself, with the tendons transmitting the force to the hands. Thus any device trying to measure the effect of these muscles is going to be limited or expensive or both.

Two other important measures of finger movement are flexion (the "bend" of a finger along its axis) and abduction (the degree of bend between fingers). These are illustrated diagrammatically in figure 2.3. The thumb, in particular has a wide variety of possible movements – in addition to the flexion/extension and adduction/abduction of the other four fingers, it has much more complex movements with both radial and palmer abduction, and anteposition/retroposition.

2.4 What is Auslan?

This section is largely based on information in the Auslan Dictionary [Joh89], but some of the material is also from several other sources ([Aus95, Uni89]).

As stated previously, Auslan is the sign language used by the Australian Deaf and non-vocal communities under normal circumstances and without outside intervention.

Auslan is related to British Sign Language, although not identical to it. It is dissimilar to American Sign Language, although some signs have been taken from American Sign Language and incorporated into Auslan.

Sign languages, such as Auslan, in general have very different grammars to those of spoken languages. This is to be expected, since typically it takes twice as long to make a sign as to say a word, while it takes the same amount of time to make a sentence in Auslan as in English (typically) – the obvious and empirically correct conclusion being that Auslan uses about half the number of words that a sentence in English uses[4]. Sign languages in general thus tend to be far more concise than spoken languages.

Word order, for example is usually not important[5], and there is usually no form of

[4]This makes one think carefully about why English uses so many words. Other languages, such as Arabic and Chinese, use fewer words to convey a similar message. One possible cause is that with vocal languages, high levels of redundancy are used as a means of allowing error correction. The same level of redundancy may not be required in signing, or for that matter in other vocal languages.

[5]This is not always the case. If subject order is important, then it does matter. For example,

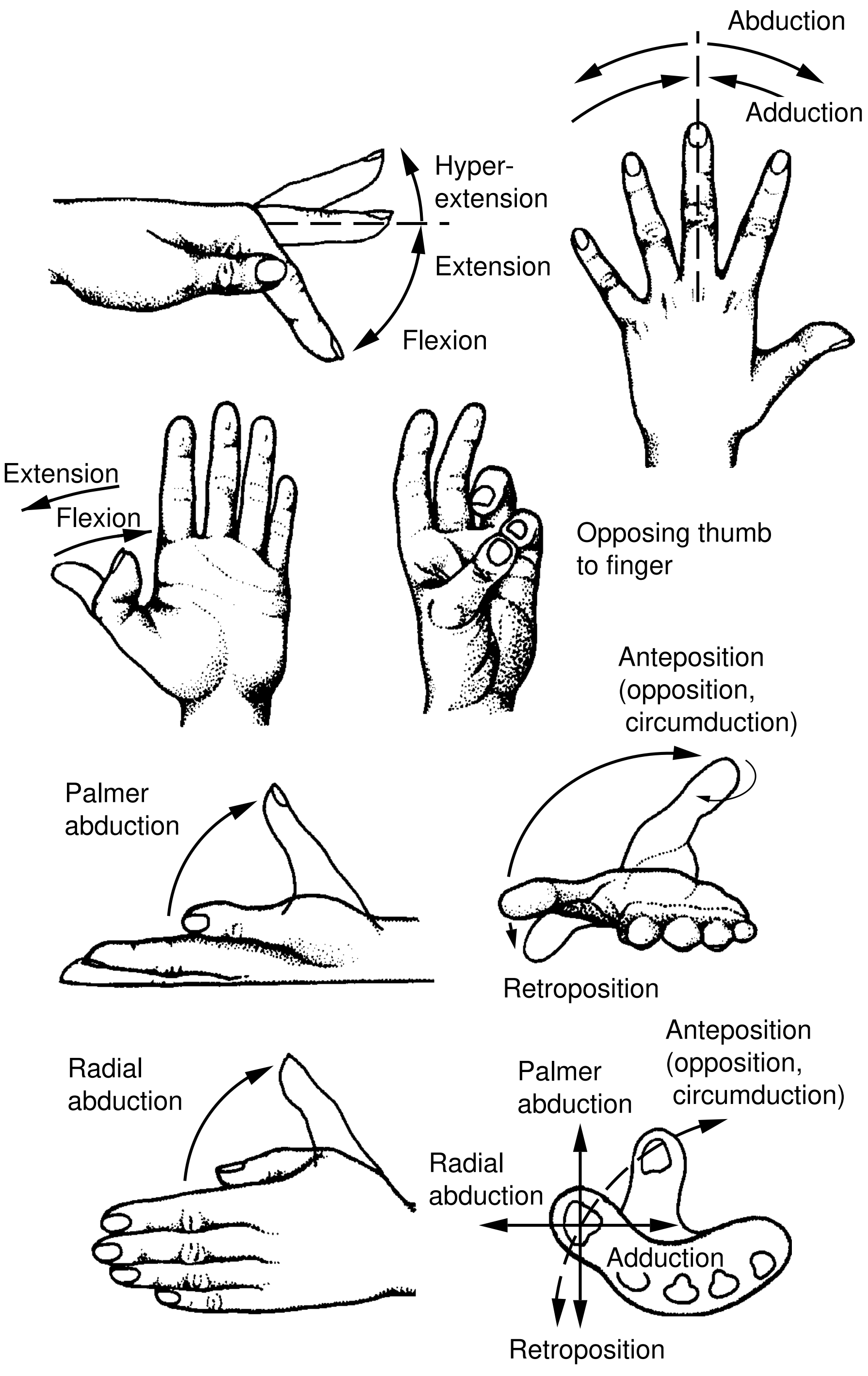


Figure 2.3: The wide number of possible motions that the hand can make.

Taken from [Stu92].

words like “to be”. Similarly, usually tense information is indicated adverbially, rather than as part of the sign itself – for example, rather than “earned” the sign would be “before earn”.

Auslan also has about 4000 signs in it. However, there are many ways these signs can be modified to alter the meaning slightly, and even more powerfully, signs can be combined to give new signs. Furthermore, there is the idea of a “classifier” – a symbol which describes the physical attributes of something, by miming them. At any point in time, an English word for which there is no exact Auslan equivalent can be finger-spelled (in fact a number of signs are evolutions of finger-spelling, such as the days of the week). All of these serve to enrich Auslan to an extent equivalent to or greater than that of spoken languages.

Finger-spelling in Auslan is accomplished through using a two-handed alphabet, taken from British Sign Language (BSL). Most other sign languages (such as American Sign Language (ASL) and Irish Sign Language) have one-handed alphabets. Finger-spelling is not only used in Auslan when there is no sign that covers the meaning of a word, but also for things with proper names[6], or sometimes even by people who cannot sign Auslan but still need to communicate.

In the following section, we will focus on individual signs and their structures, since that will also be the focus of the thesis.

2.4.1 Variations on Auslan

There are a number of similar sign systems and variations to Auslan that need to be explained:

“man shoot dog” and “dog shoot man” are equivalent in Auslan, since it is clear that the dog cannot shoot the man. On the other hand, “man shoot woman” and “woman shoot man” have very different meanings. Usually the most emphatic phrase is at the beginning of the sentence. So you can still use constructions such as “woman *pause* man shoot”, if you want to emphasise that it was the woman in particular who the man shot.

[6]Sometimes, however, objects with proper names have a sign attributable to an aspect of that object. For example, some signers have a sign name which is something about the way the look or act. The sign for Sydney, for example, is the hands making the shape of the Harbour Bridge.

Finger-spelling

Finger-spelling is the representation of each letter of the alphabet by a sign – so there are signs for A to Z. It is also interesting to note that many signs use handshapes based on finger-spelling, and that occasionally abbreviated finger-spellings become signs. For example, group is made by making a horizontal circle with flat hands, while family is done with a similar gesture, but with the F handshape (index and middle fingers straight, other fingers completely bent).

Signed English

This, again, is not Auslan. Signed English is a word-for-word translation from spoken English to signs, augmented with finger-spelling for the end of words – in other words, a direct mapping from the English language to signs. Obviously, there are shared signs between Signed English and Auslan. Signed English remains the most common language taught to Deaf children[7]. When watching people using Signed English, it is quite clearly a less concise than Auslan, and people must slow down their rate of speech to sign at the same time. As unusual as it seems, Signed English actually appears very clumsy compared to Auslan which looks quite amazing[8].

Pidgin Signed English

This lies between Auslan and Signed English, as the name suggests. Johnston [Joh89] points out that there is a continuum between Auslan and Signed English and that members of the Deaf community will attune where they sign on the curve depending on who they are speaking to. Between two Deaf people, for example, the conversation would lie very much to the right of the curve, in the domain of Auslan. Between Deaf and speaking persons, they would use something somewhere in the middle, whereas full-on Signed-English speakers would be on the left.

[7]The Deaf community feels that such an approach is not particularly effective and based on recent studies, they suggest that a "bilingual" approach should be taken, where children are first taught Auslan, and are then taught a second language such as Signed English, or Oral English, in a similar way to the treatment given in ESL (English as a second language) for non-native English speakers.

[8]As clichéd as the words *Poetry in Motion* are, they seem nowhere more apt than watching an energetic Auslan speaker in full swing. The speed and coordination required to do so are surprising. If you don't believe me, go to a Deaf Club meeting, or watch [Uni89].

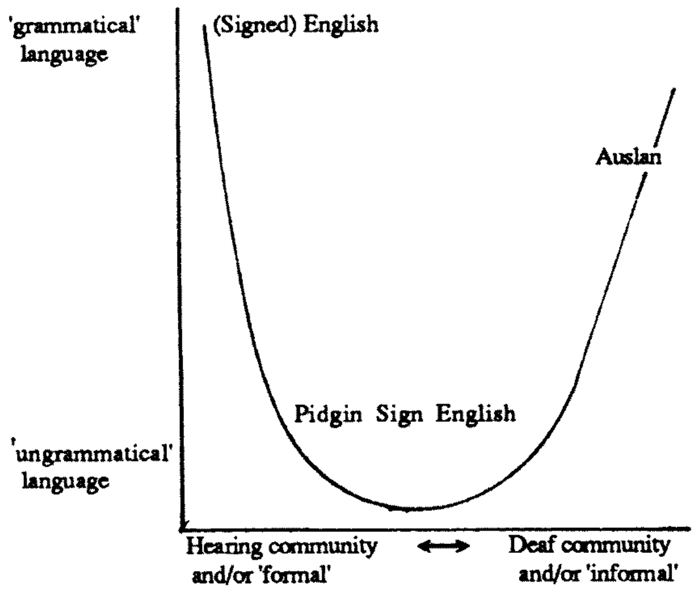


Figure 2.4: The continuum between Auslan and Signed English. PSE lies in the middle.

Taken from [Joh89].

Cued Speech

This can be thought of enhanced lip-reading – signs are made around the face to provide additional information about the phoneme being uttered. It must be pointed out that lip-reading is usually not considered a practical means of communication – typically, a person who is lip-reading is only recognising 30 per cent of the words being uttered.[9]. For example, b and p cannot be told apart – and thus "cued speech" indicates which phoneme is being uttered.

Typically, it is only used as a teaching aid and is not used widely in the Deaf community.

2.4.2 Structure of signs in Auslan

There are several components of a sign. By mixing and matching these, a wide variety of signs can be formed. These components are:

[9]One member of the Deaf community told me that most effective lip-readers are at least 50 years old – because that's how long it takes them to learn and get used to it.

Handshape Handshapes in Auslan can be classified into 31 different shapes (although there are 63 variations, these are typically considered minor).

Furthermore, there are one-handed signs (where only one hand is used), two-handed signs (where both hands are used, but each hand is executing a different handshape) and double-handed signs (where both hands have the same handshape). There is a tendency in two-handed signs for one hand to dominate (usually the right in right-handed signers) and also that double-handed signs tend to be symmetrical (that is, both hands also tend to move in the same, if mirrored, motion).

Location This is where the sign is started. There is also the concept of of "neutral space" – an area of space where it does not matter where a sign is made. This area extends in a half-ellipsoid in front of a person's chest (see figure 2.5). Outside the neutral space, location becomes significant, and there are a number of signs whose main difference is their location.

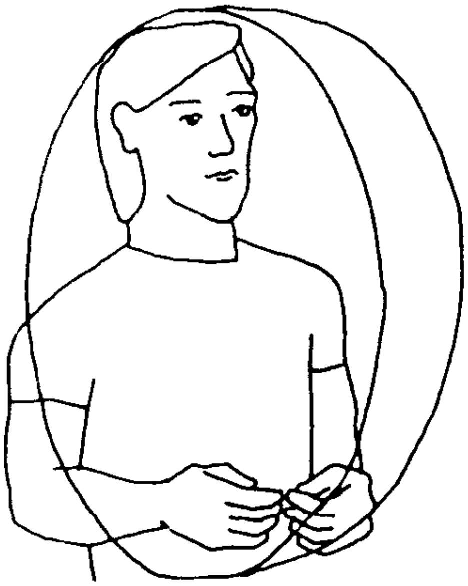


Figure 2.5: The neutral space for signs. Exact location is not important within this area.

Taken from [Joh89].

A secondary aspect of location is the way that in double-handed and two-handed signs the hands come in to contact.

Orientation This refers to the orientation of the hand. As previously discussed, this would be the three possible rotational movements of roll, pitch and yaw.

Movement There are a wide variety of movements possible with sign language, including arcs, straight lines and "wavy" patterns. They are not necessarily two-dimensional (consider the sign for crazy, which in Auslan is the universal gesture of pointing to the head and making small circles directed away from the head), and can include changes in orientation and sometimes in handshape. Typically, they can include multiple steps, such as the sign for science which is characterised by "mixing test-tubes".

Expression Facial expressions are an important part of signing. While it does not play a major role in individual signs, it becomes very important at the sentence level. For example, negation can be made by making the normal sign and shaking one's head at the same time, and questions are formed by raising the eyebrows or tilting the head (depending on the type of question).

We now move on to a discussion on the various ways of analysing signs.

2.5 The Device vs Vision debate

At the moment, there is a schism as to how to collect the data for gesture recognition in a general sense, and sign language recognition in particular. In general, the approaches are divided into one of two schools (although there also exist several hybrids):

Device-based measurement techniques: These techniques measure gestures using direct devices, such as gloves, styli, position-trackers, flexion sensors and so on. These are then transmitted to the computer, with the values of these then taken to indicate a variety of details about where the hand is. People using this approach include James Kramer ([KL89]), Peter Vamplew ([Vam]) and others.

Vision-based Approaches: These techniques utilise cameras and watch people while making signs. Typically, to simplify this activity, the individual involved wears a special glove with areas painted on it to indicate the positions of the fingers. Davis and Shah [DS93] for example, uses bright points on the edge of the fingers, while Dorner [Dor94] uses a much more complex system that colour-code individual knuckles on each finger in an intelligent way. Starner [Sta95] uses two simple gloves: one coloured yellow, and the other coloured orange. Others, such as Isibuchi et al [ITK92] and Krueger [Kru90] have managed to do so without special gloves using a 3D predictive model and a hand colour probability transformation, but at the cost of increasingly complex hardware.

Vision-based approaches have the advantage that the user is not encumbered with any complex devices (although they may still have to wear a glove, it is not as device-ridden or cumbersome as some of the device-based gloves). However, it has the significant disadvantage that an immense amount of computing power is typically required just to process the images to extract the hand position, before being able to analyse the data. While this would not be a major disadvantage for virtual reality applications (the mainstay of the hand-interface device industry) – since it is conceivable to have a virtual reality room (which some people have, in honour of *Star Trek: The Next Generation*, called "holodecks") – this would pose major difficulties for applications potentially requiring portability, such as sign language recognition. Also, while giving reasonable 2-D resolution, these systems typically do not handle the 3-D aspect of hand positioning well, unless two separate cameras are used, doubling the input data and complicating the already complex algorithms further.

On the other hand, the device-based approaches can suffer from the limitations of the devices used for measurement of hand movement. Different people have different hand-sizes, and calibration of these devices is a problem which has only been addressed recently. The motion detectors are subject to physical noise (for example, any metal near a Polhemus tracker tends to have rather strange effects on its behaviour), and some software is still required for filtering. On the other hand, they consume little computing power (in fact one of the first gloves, the Z-glove [ZL87] was attached to a Commodore 64 – hardly a rival to a Cray). This means that cost can be kept down. It also adds to their portability and some of the gloves (such as the CyberGlove [KL89]) have been road-tested, while attached to small palm-top computers.

2.6 “Whole Hand Input” Devices

Much of the discussion about instrumented gloves comes from the excellent review by Sturman and Zeltzer [SZ94].

The first “generation” of gloves were not intended for use for gesture recognition. This included the Sayre Glove, a glove which was really intended for control of as many devices as possible through the manipulation the hand.

This is the point where people began to consider the alternative of analysing video images, and see if this was a practical way of analysis.

The first widely recognised device for measuring hand positions was developed by Dr G Grimes at (surprise, surprise) AT & T Bell Labs [Gri83]. Patented in 1983, Grimes' “Digital Data Entry Glove”. It had finger flex sensors, tactile sensors at the finger-tips, orientation sensing and wrist-positioning sensors. The positions of the sensors themselves were changeable. It was intended for creating “alpha-numeric” characters by examining hand positions. It was intended as an alternative to keyboards, but it also proved to be effective as a tool for allowing non-vocal users to “finger-spell” words using such a system.

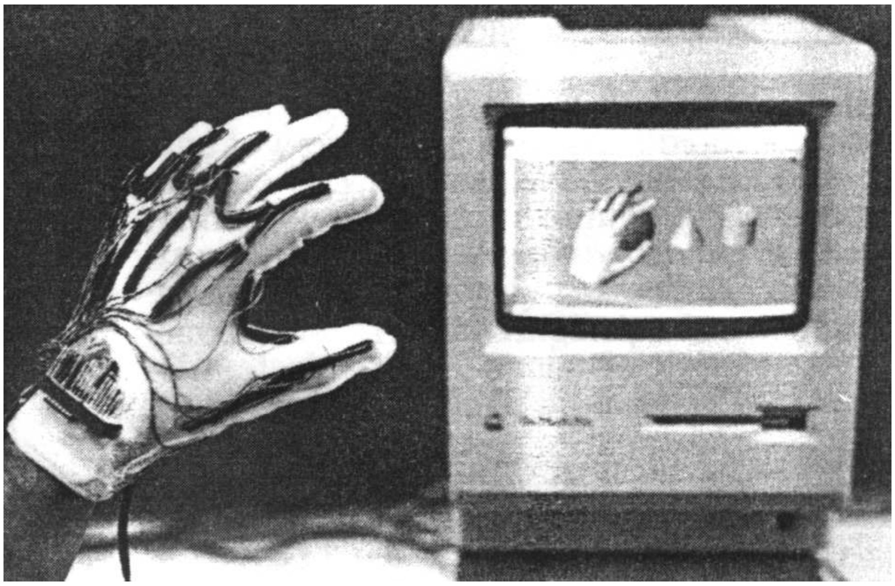


Figure 2.6: A picture of the internals of the VPL DataGlove, attached to a Macintosh.

Taken from [ZL87].

This was soon followed by a more "regular" device, which was later to become the VPL DataGlove [ZL87, ZL92], which is shown in figure 2.6. This device (which is, by the way, heavily protected by patent laws) was built by Thomas Zimmerman, who also patented the optical flex sensors used by the gloves. These sensors have fibre optic cables with a light at one end, and a photodiode at the other. Zimmerman had also built the Z-glove which he had attached to his Commodore 64, which was a simplified version. This device measured the angles of each of the first two knuckles of the fingers using the fibre optic devices, and was usually combined with a Polhemus tracking device. Some also had abduction measurements.

This was really the first commercially available glove. Even so, at in excess of US$9000 a piece, they were not quite within everyone's reach.

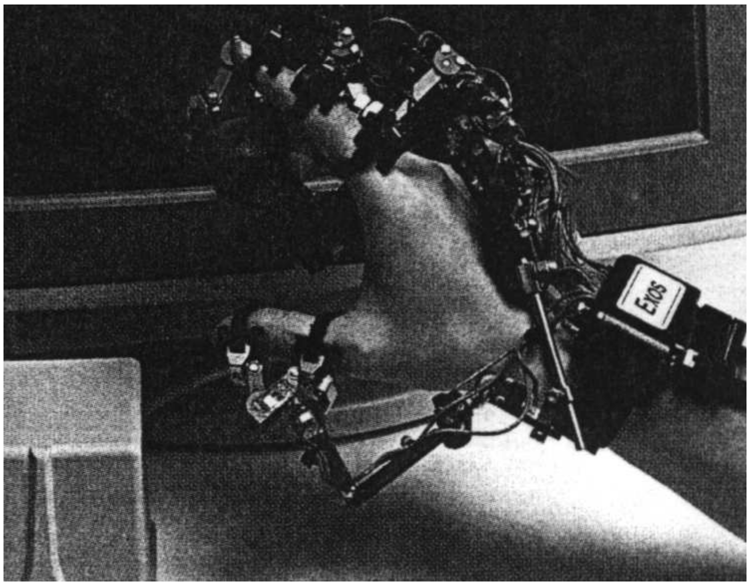


Figure 2.7: The Exos Dextrous Hand Master

Taken from [SZ94]

A completely new alternative was proposed at this stage, which was to become the Exos Dextrous Hand Master. This device makes no pretence of being a glove. Imagine an exoskeleton for your hand bedecked with sensors — you will have a pretty good image of the DHM (it is shown in figure 2.7). By all accounts, it takes a while to get on, but in terms of accuracy of information, it measures 20 degrees of freedom with 8 bits of accuracy, at up to 200 Hz, which is currently unrivalled. Usually a 6D tracker is attached, giving a total of 26 degrees of freedom. While ideal for tele-robotics (remote control of robots), it is overkill for some applications. It is also not cheap (around

US$15000).

Through a deal with VPL, Mattel/Nintendo began manufacturing PowerGloves. It has many design corners cut. These include only having sensors on the first four fingers, and a very poor tracking mechanism (4D - x, y, z, roll vs other gloves' 6D - x, y, z, yaw, pitch, roll). To quote Sturman and Zeltzer "Although the least accurate of the whole-hand input devices, the PowerGlove is also the cheapest by a factor of 100."[10] In addition, interface boxes exist that allows the easy connection to almost any machine through a standard RS-232 serial port (such as UIUC's PowerGlove Serial Interface, or the Menelli Box).

In parallel with this James Kramer at Stanford University was developing the CyberGlove – which is patented [Kra91]. This glove was specifically designed for use in his "Talking Glove" project, which is a system for communication between non-vocal people and speaking people. It comes in two models: one with 18 degrees of freedom and the other with 22 degrees of freedom and measures bend of the first two knuckles on each finger (three for the 22-DOF model), abduction between fingers and a number of additional measures around the thumb (since it has 5 degrees of freedom), by using strain gauges arranged in Wheatstone Bridge configuration. Together with a Polhemus tracker this would form the perfect system for this thesis. However, the cost is approximately US$6000 for the glove, not including the position-tracker.

As we speak, prices of these devices rapidly falling, as are the tracking devices themselves. In June 1995, Fifth Dimension Technologies released the 5th Glove '95[11], costing US$300, although it only had finger bend measures and no abduction or individual thumb measures. Also, in mid-September 1995 Abrams-Gentile Entertainment (AGE Inc) announced the release of the PC-PowerGlove, which was to have 8-bit information on each finger, 16-bit position information, 8-bit orientation information (roll, pitch, yaw) and also the possibility of a *pair* of gloves, which would allow much improved recognition, for the surprising price of US$120 each. The techniques developed in this thesis could be mapped to these new gloves in a matter of days.

In addition, high-accuracy magnetic trackers are also falling in price, with both Polhemus and Ascension trackers now available in the sub-US$1000 region.

So while currently the technology is not within the reach of many, this is changing

[10] While this was true at the time of writing of [SZ94], it is no longer the case, with several low-cost gloves now on the market.

[11] It seems that Microsoft has started a very depressing trend.

rapidly. While virtual reality may not be growing at the rate originally predicted, it is still growing at a sufficient rate to see the prices on these devices fall quickly.

2.7 Learning tools

Since one of the goals was to design an extensible system that was easy-to-use, this almost precludes the possibility of manually-programmed analysers. This implies that computer-based learning approaches to solving this problem might be viable. To do so, there are a number of learning methodologies that can be used to build a system capable of recognising gestures.

In general, the task of these learning algorithms is one of classification – given a set of inputs, commonly known as *attributes* these systems must decide the type or *class* of the object based on those attributes. The attributes need not in general be directly physical, and frequently some form of pre-processing is required to provide the attributes actually used by the learning algorithms.

2.7.1 Neural networks

Neural networks are intended to analyse data in a similar way to the way biological neural networks do.

To understand what a neural net is, it helps to understand what a single neuron does.

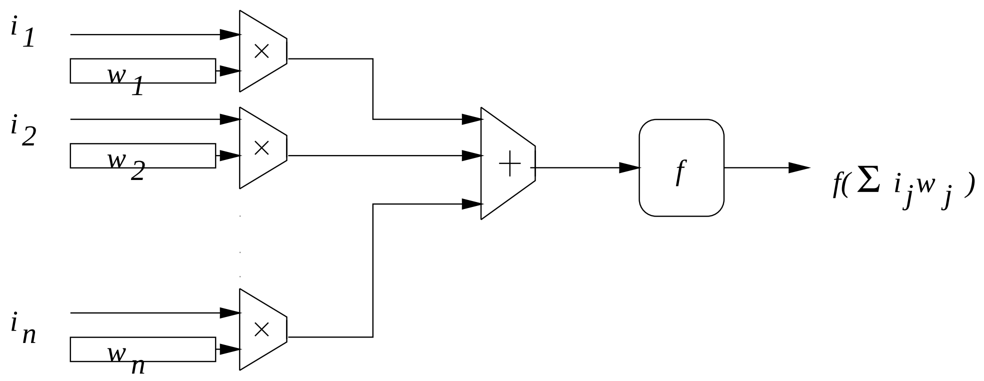


Figure 2.8: The internal structure of a neuron

A simple neuron is shown in figure 2.8. As can be seen, the neuron contains a set of inputs (i_j), a set of weights (w_j). The product of these input-weight pairs ($i_j w_j$) are then added together. An increasing monotonic function is then applied to the sum, such as the sigmoid function ($\frac{1}{1+e^{-x}}$). This output is then available. Typically the weights are considered to be between 0 and 1 and the the inputs and outputs similarly. However, the internal sum can be anywhere between 0 and the number of inputs.

This can be thought of in many ways. For example, this can be thought of as simply an information condenser – it takes a set of inputs and reduces it to a single output. The output can also be then can be interpreted in a number of ways. If instead of a sigmoid function a simple binary function is used, then the neuron can be thought of as a "decision-maker" – it takes a set of inputs and using its weights decides whether to output a "0" or a "1". The sigmoid function allows a "maybe" guess, somewhere between a "0" and a "1".

Another way at looking at the function of the neuron is that it partitions an n-dimensional space, where n is the number of inputs into the neuron. The weights describe a $n-1$ dimensional hyperplane through the origin, and depending on the values of the inputs, the point represented by the inputs is either "above" or "below" the plane[12].

A single neuron in isolation is interesting, but neurons are usually combined together to form neural nets, increasing their power by orders of magnitude, as shown in figure 2.9. Typically, these neurons are considered on a layer-by-layer basis. The first layer, known as the input layer, takes external input. The hidden layer then takes as input the outputs from the previous layer of input. There may be several hidden layers before the output layer, the layer that connects back to the real world, is reached. Each layer takes as input the previous layer's output[13].

The real power of a neural networks comes from the way that weights are set. Using supervised learning, the weights initially start out random and examples are fed through the neural net. The outputs are then compared to what the outputs should be, and using one of a variety of techniques (such as back-propagation), the weights of the neurons are adjusted. In this way, the neural net "learns" what the desired output

[12] The same argument we applied before is here. If you use a sigmoid function, you get additional information like "just above or just below".

[13] These sort of networks are commonly-known as "feed-forward". There are other types of neural nets that have feedback paths in them that feed the output of the previous layer back through the net again.

is and this is where it becomes useful.

There are a number of ways for using a neural net as a classifier. One is to have an output neuron for each possible class, thus each neuron will output a value "closer" to 1 when that particular class is indicated. A problem arises when more or less of the neurons claim that it is of that class. A second, less frequently used option, is to encode the classes in a binary manner. This alleviates the above problem, but may result in unusual behaviour, since the sequencing of the encoding affects the accuracy of the output function.

It will be noted that neurons are extremely "tweakable" – that is to say, there are a great number of variables that can be played with, such as the output encoding, the number of hidden layers and the number of neurons in each hidden layer, the interconnection of neurons (there can be for example feed-back neural networks), the neuron output function, the error function used for the learning algorithm and several others. What is worse, however, is that there is, so far, no structured way of choosing the correct configuration for a particular task – it is essentially a black art.

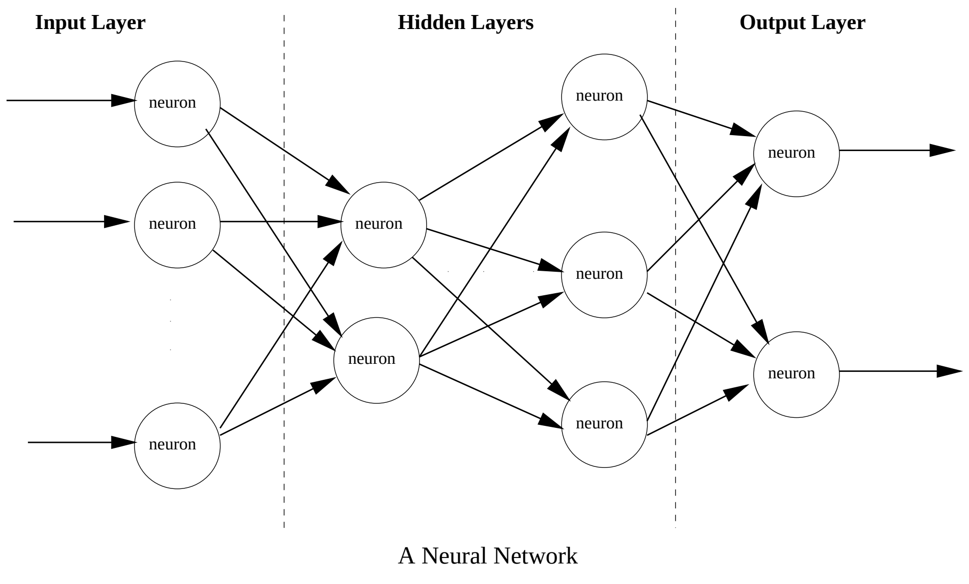


Figure 2.9: A neural network – a collection of neurons connected together.

2.7.2 Symbolic learning algorithms

Symbolic learning algorithms can be described as algorithms that determine a set of rules that form a relationship between the attributes and the classes. There are a variety of tools that work in this way, and they vary in the way that they construct these rules. For example, it might try to build rules in the form of nested if statements.

An example of such a tool is C4.5. C4.5 uses the concept of "information gain" to make a "tree" of decisions that lead it to the final outcome. The information gain can be described as the effective decrease in entropy (usually measured in terms of "bits") resulting from making a choice as to which attribute to use and at what level. For example, if I choose a specified attribute to discriminate at a given point, it will have some impact on how well the system can tell the classes apart. By considering which of the attributes is the best to use as a discriminant at a particular node in the tree, we can build up a tree of decisions that allows us to navigate from the root of the tree to a leaf node by continually examining attributes.

An example of such a tree is shown below[14]:

```
Thumb Bend <= 1 : Is Not a ``B'' (34.0/1.0)
Thumb Bend > 1 :
| Ring Finger Bend <= 2 : Is a ``B'' (7.0)
| Ring Finger Bend > 2 : Is Not a ``B'' (3.0)
```

The graphical representation of the tree is shown below:

As can be seen, by looking at attributes of the data about which we are trying to draw a conclusion, we can reach a leaf-node which will tell us what the output should be. This system is attuned to discrete outputs.

Essentially, however, and similarly to neural nets, symbolic learning algorithms do partition the space. This is shown in Figure 2.11. In general, there can be more than two partitions, and there can be many more attributes. In this situation in fact, there are eight attributes, but only two are interesting. As should be obvious, it is not usually sufficient to determine what a sign is, just by looking at the ring finger and thumb, it just happens to be in this case (for reasons of simplicity).

[14] This tree is actually one of the trees generated early on in the testing process. It determines whether a handshape is a "B" or not.

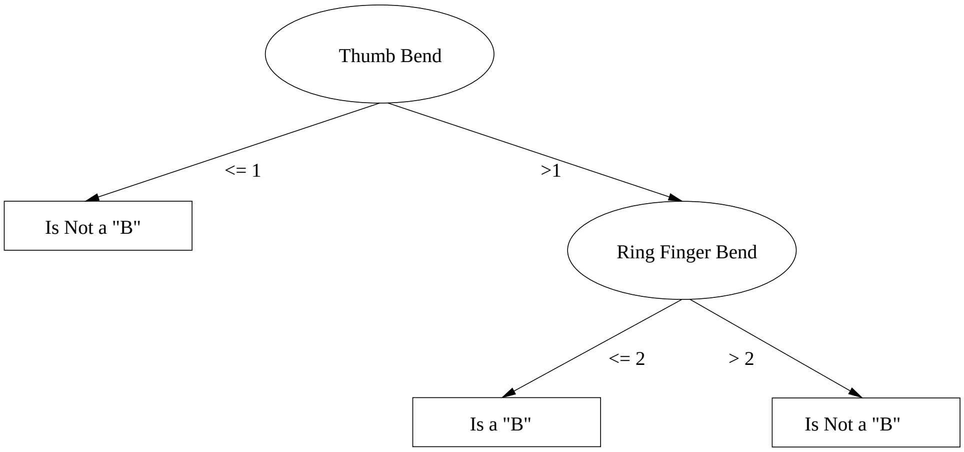


Figure 2.10: A graphical representation of the decision tree produced by C4.5

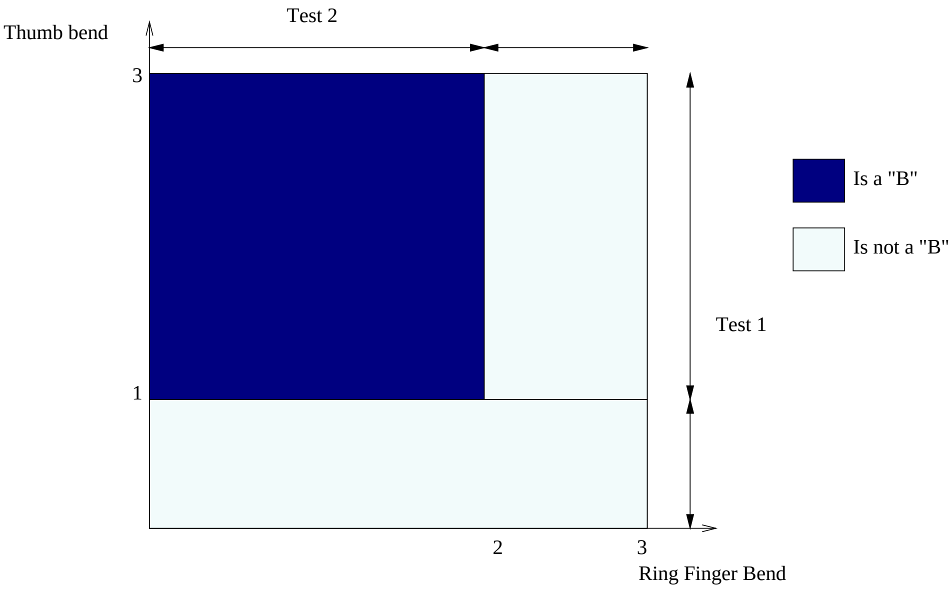


Figure 2.11: The decision tree discussed so far partition the space as shown.

Note, however, that some, such as C4.5, can only consider one attribute at a time – and can only segment the space into "orthogonal" pieces – that is it can only divide the space parallel to the axes. Imagine, for example, that a concept "boundary" (i.e. the values along which the real instances of the underlying data change from one classification to another - i.e. on one side of the boundary they are positive cases, and on the other they are negative) that was not horizontally or vertically aligned, but diagonal. Such a concept could only be approximated by a decision tree, as shown in figure 2.12.

Other learning algorithms can segment the space more arbitrarily. Neural nets and instance-based learning algorithms can do this "attribute space" segmentation in a much more flexible way, allowing them to cope more easily with non-orthogonal concepts. Of course, in theory, if the attributes were carefully selected for C4.5 the concept boundaries would be orthogonal to the attributes, but this violates one of the objectives of learning algorithms, since if we knew the concept boundary, we wouldn't be using C4.5 anyway.

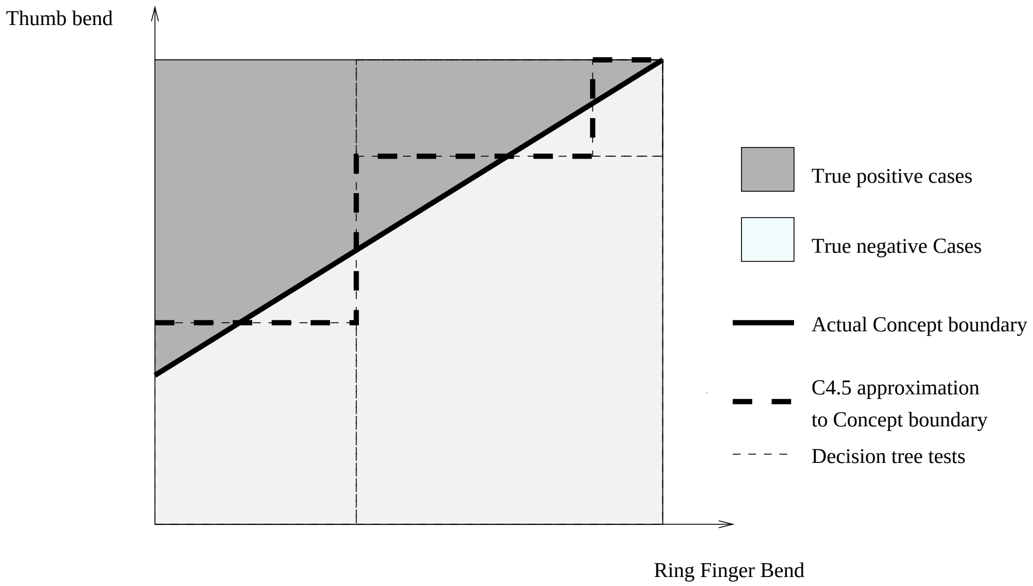


Figure 2.12: An example where the limitations of C4.5 come through. If the concept boundary is non-orthogonal, C4.5 has to make some approximations.

There are other similar systems that can provide variations, and not all symbolic learners have the limitations that C4.5 does. Some, for instance, can apply a function of the attributes at each node, rather than always having an absolute value. Another is to have non-binary trees, with it being possible to partition according to ranges of values of attributes. Another variation is that rather than having a discrete output, it is possible that different functions be applied in different partitioned areas of the space.

2.7.3 Instance-based Learning

Instance-based learning techniques work essentially by keeping typical attribute examples for each class.

Aha, Kibler and Albert [AKA90] define a set of instance-based learning algorithms that have three defining characteristics:

A similarity function: This tells the algorithm how close together two instance are. Although this sounds easy, there is a great deal of complexity in choosing the similarity function, especially in situations where some of the inputs are enumerated. For example, if you were trying to match people, and one attribute was hair colour, what does distance mean in the context of hair colour?

A "typical instance" selection function: This tells the algorithm which of the instances to keep as examples. How do you know which instances are "typical" and which are atypical?

A classification function: This function is the one that when given a new case, decides how it relates to the learned cases. For example, this function might be the instance to which it is closest in location.

IBL's are known by many other names, and there are many variations on the basic theme. There are three that Aha, Kibler and Albert propose in their paper:

IBL1: Store all example instances and simply find the closest instance – then the class of this instance is the class of the closest instance. The large number of instances that need to be stored, however, can require large amount of space.

IBL2: Similar to the above, but throw away instances in the training set that would have already have been correctly classified. This saves some space.

IBL3: Like IBL2, but makes some assumptions about the data and uses a statistical methods to "weed out" irrelevant or noisy instances.

In addition, the above can be augmented by the application of a k-nearest neighbour (k-nn) approach [CH67, Har68, Gat72]. Instead of looking at the single nearest point and classifying according to that, we look at a *collection* of the nearest points and use a "voting" mechanism to select between them[15].

They share many problems with neural networks, such as the "tweakability" of the system – due to having to set up the three functions discussed above, and finding an optimal set of them. They also suffer from the "extractability" problem, in that it is basically a very unstructured way of learning – it merely stores the typical values without processing the data in any way. No "concepts" are produced in a human readable format.

The other problem is that they sometimes require moderately large amount of storage, even with the optimisations of IBL2 and IBL3. This is not a problem in itself, but can impose problems for fast searching for the nearest point.

On the other hand, they are easy to test, conceptually simple, and are able to segment the space in complex ways.

2.7.4 Grammar-based techniques

Although not at the moment a mainstream learning technique, grammar-based learning algorithms may also be applied, although they are less general than other learning techniques.

A complete review of grammars is beyond the scope of this thesis, but a basic introduction will be given here.

A grammar consists of four things:

A set of terminals, where a terminal is the most primitive symbol in the grammar. In English, for example, words would be considered terminals.

A set of non-terminals, where non-terminals are symbols which are, as the name indicates, symbols which are not terminals. Typically, in English, this would be

[15] Of course, it's possible to get ties, and when this occurs, several methods exist for breaking the tie, such as reverting to 1-nn (ie just looking at the nearest neighbour), or taking the sum of the distances and finding the minimum.

things like sentences, noun phrases, adverbial phrases, subjects and so on.

A set of production rules, where a production rule is a transformation from one sequence of terminals and/or non-terminals to another sequence of terminals and non-terminals. The left hand side must have at least one non-terminal in it. An example of a production would be that if you have a noun_phrase (a non-terminal symbol) then it can be replaced with peartridge[16] (which is a terminal symbol). There can be more than one production for a given right-hand side.

A Start symbol, where a start symbol shows where all valid sequences of symbols we want to be able to produce can be derived from. For example, in English, the Start symbol might be sentence, from which we could derive some valid sentences, by following the production rules.

Given a stream of terminals, it is possible to take these terminals and build out of it a "parse tree" – a tree illustrating the steps taken to get from the start symbol to the sequence of terminals observed (this activity is of course known as parsing)[17]. This tree acts as a store of information, and by applying operations to this tree, meaning can be extracted from the sequence of apparently random terminal symbols.

Although this sounds complex, it is done on a daily basis by the compilers used in our computers. Although most people think of symbols as words, it might be possible to take a broader view and think of basic sub-gestures as the terminals (with some additional terminals for movement) of a language which could be the set of Auslan signs.

The problem with these systems at the moment is that these sort of grammars are typically hand-built. There is on-going research into automatic grammar learning programs, but these are still in their infancy.

2.7.5 Hidden Markov Models

A Hidden Markov Model has several components. It has a set of states, a set of output symbols, a set of transitions which have associated with them a probability and an

[16] A peartridge is a strange animal, known for being incredibly funny, almost as funny as frogs. It is also, by the way, also a noun, and hence a noun phrase.

[17] Of course, if you were using the grammar proposed in the previous section, you would end up with a peartridge in the parse tree, which, surprisingly, is a fact that is well known by "sobriety-challenged" people around the world, especially on the first day of Christmas.

output symbol, and a starting state. When a transition is taken, it produces an output symbol. The complicating factor is that the output symbol given is not necessarily unique to that transition, and thus it is difficult to determine which transition was the one actually taken – and this is why they are termed "hidden".

The way that HMM's are used is the opposite of producing output symbols. People who use HMM's extract meaning by answering the question: "Right. I have a set of output symbols. What was the sequence of states and transitions that resulted in those output symbols?" The output symbols might be something like words, and from them we might be trying to extract some meaning, such as the type of word (noun or verb).

Note that this is not deterministic, and since the model is hidden, there may in fact be more than one sequence of transitions that resulted in that sequence of symbols. Thus the most likely sequence of transitions is searched for, using an algorithm known as the Vitterbi algorithm.

To build an HMM that can actually be taught, the person typically decides on a set of states that he or she feels are important, and a set of all possible transitions (although it is possible to create a complete set of transitions and let the algorithms decide on the ones with transition probability of 0). The transitions are assigned random probabilities.

Then, using algorithms such as the Baum-Welch algorithm, it is possible to train the HMM by adjusting the weights of the transitions to better model the relationship of the actual training samples.

Although the HMM technique is different to neural nets, it has many of the same problems, including deciding what the states are. Again, the decision of what the states and transitions are is a black art.

2.8 Previous work in sign language recognition

In this section, we will discuss some of the more prominent research efforts made in the area of understanding sign language. This will include all forms of sign language and also finger-spelling, since most people in this field precede research into full-blown sign language recognition with more limited and easy to handle finger-spelling, and use similar methods for both approaches.

We will consider the two approaches – the image-based and the device-based approaches separately.

2.8.1 Image-based approaches to sign language

Charayaphan and Marble's research in to Image Processing for ASL

Charaphayan and Marble, in 1991-92 [CM92] examined the possibility of processing images in such a way as to understand ASL (American Sign Language). To do so, they used three criteria of movement:

- Where the hand stopped. They did this by measuring the standard deviation of the motion of the hands compared to previous positions and thus determined if the hand had come to a complete standstill.
- What "trajectory" – that is the course in space – the object followed. They did this by measuring the eccentricity between one "stop" and another. The eccentricity can be thought of as "how bent" the approach is – it can be defined as the ratio between the straight line connecting two stop points and the the farthest distance of the path from that straight line. See figure 2.13.

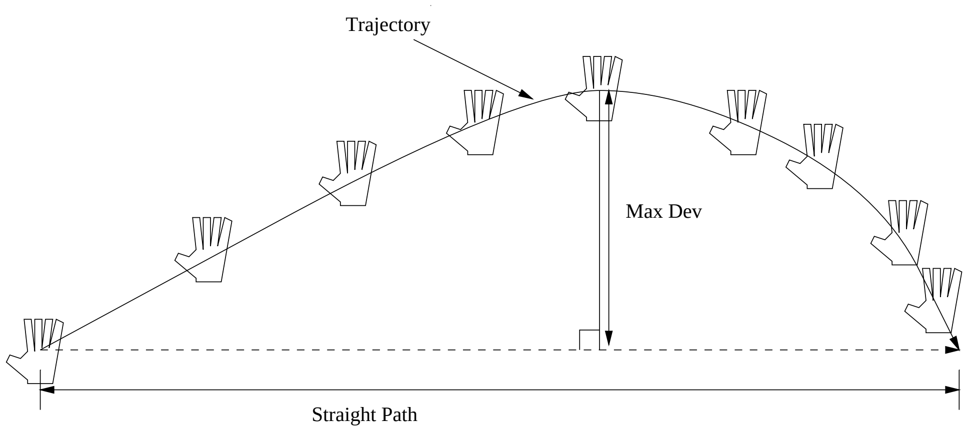


Figure 2.13: The eccentricity in the diagram is defined to be MaxDev/StraightPath. The value is positive if it is on our right as we follow the trajectory, and negative if it is on our left.

- What shape the hand was in when it stopped. When all else failed, they would look at the final handshape to determine what the sign was, using a Hough transform to compare the handshape to ones stored in a table.

They used no computer-based learning methods, but rather hand-built a system that has close similarities to an instance-based learning technique – that is they averaged the locations of their training set and used this as the information. To test the system, they captured each sign once and simulated what they believed was typical variation in the input. This system successfully classified 27 of the 31 ASL symbols by the second stage. Of the remaining four signs, it was shown that using the Hough transform, the correct sign could be decided upon most of the time.

Davis and Shah's work on Gesture Recognition using cameras

The research undertaken by Davis and Shah employed a simple black and white camera, and a glove which is black, except for white fingertips. A four-state finite state machine was used to recognise a simple set of seven gestures that would be used in interaction with computers – like left, right, up, down, rotate, grab, stop. The four states of the finite state machine were:

1. Keep hand still in start position.
2. Move hand position slowly to gesture position.
3. Keep hand in position for desired duration (ie how far left you want to go).
4. Move fingers back to start position.

As can be seen, this is a very simple system, and it works quite well. However, it has some significant shortcomings for use in full sign language – such as it does not monitor hand position, and requires explicit stop/start signals, and only operates at 4 Hz. Furthermore, the "finite state machine" approach is very limited and is really a "posture" recognition system rather than a "gesture" recognition system. The finite state machine has, in some of the experiments, gone prematurely into the wrong state, and in such situations, it is difficult to get it back into a correct state.

Starner's work with American Sign Language and Hidden Markov Models

A very new (February 1995) approach was suggested by Starner in his Master's Thesis [Sta95], and together with Pentland in another Tech Report [SP95].

First, they extended the HMM to be able to handle not just simple output symbols, but distributions of multiple variables, thus allowing them to use features extracted directly from the data, instead of having to do pre-process the data into a sequence of output symbols.

In terms of hardware, a colour camera was used and users wore a yellow glove on their right hand and an orange glove on their left. Five images are taken per second and fed into an SGI Indigo 2. A small selection of features are extracted from these images, such as the bounding ellipse and its eccentricity, the x and y positions for each hand and the axis of the bounding ellipse.

In this case, a raw correct rate was achieved of 91.3 per cent. By imposing a strict grammar on this, it was shown that accuracy rates in excess of 99 per cent were possible, with real-time performance. A selection of 40 signs were used and the simplifying assumption was made that signs have one grammatical class[18]. The signs were selected to allow a large set of coherent sentences to be constructed. Furthermore, the grammar was strictly "pronoun, verb, noun, adjective, pronoun", with pronouns and adjectives possibly empty. It is suggested that the bigram and trigram techniques could be used (a well-known method), whereby the preceding sign(s) are used to consider what the probability of the current sign is.

The system makes no attempt to consider the movements of the fingers, however. This is a limitation in a number of ways, since for large-lexicon system, finger position becomes increasingly important. Furthermore, there is no way that the system can handle finger-spelling.

Still, the application of the use of Hidden Markov Models was shown to be very promising and it may be effective to try to use Hidden Markov Models, even on glove-based sign recognition systems, since the HMM's are not directly related to the use of video at all; they are used on attributes extracted from the motion. This means that the HMM technique would be particularly well-suited to migrating to large-lexicon

[18] It is common to find words that can be verbs or nouns or adjectives in spoken languages. In sign languages this happens even more frequently.

Auslan recognition[19]. In fact, with very little modification, the features explored in this thesis can be used as the basis for the features used in the HMM.

Dorner and Hagen's work with a complete system

Dorner and Hagen are unique in that they have taken a holistic approach to the question of ASL interpretation [DH94]. Dorner's Master Thesis built a glove that would capture information about hand movement, while Hagen's Master Thesis involved building a deductive database that successfully translates from a standardised form of ASL into spoken English.

Dorner uses a cotton glove, with various areas of it painted different colours to enable tracking. Unlike most other models used by other people in this field, Dorner attempted to construct a 3+1D[20] model of the hand. Each joint is marked by three rings: A central band, indicating the finger it belongs to, and two bands of the same colour – either side of the central band – indicating which joint it is. To improve performance, a predictive model was emploted, limiting the possibilities of movement to those that are humanly possible. This approach appeared to work reasonably well; however the calculations involved became complex, stopping it for being real time in practical cases.

Hagen in the meantime, created a deductive database for ASL, adding the necessary features for purposes of indication which are above the lexical level – things like raising the eyebrows to indicate a question, and even allow 5 spaces for reference to objects (a technique often used by signers).

They then began to work on putting these two systems together, as described in [DH94]

Although not complete yet, they hope to connect these two systems together to form a complete ASL system. There is of course a great deal of filling to be put between these two slices of work.

[19] Most current large lexicon speech recognition systems, use HMMs as well as a number of other recognition techniques.

[20] Dorner uses this to indicate 3 dimensions in space plus one in time.

2.8.2 Device-based approaches to sign language

Glove-Talk and Glove-TalkII – Fels and Hinton's contribution

Fels and Hinton's work [FH93] and Fels' work [Fel94] take a completely different tack to solving the problem of allowing non-vocal people to communicate, by creating a connection between the signer and a speech synthesiser through an "adaptive interface" – an interface which changes over time to better match the behaviour of the user of the interface.

Fels employed a VPL DataGlove Mark II for this purpose, with a Polhemus tracker attached for position and orientation tracking. Glove-Talk was based on having a root word determined by handshape, with an ending depending on the direction of movement. Hand speed and displacement affected speed of speech and stresses respectively. For each of these, a separate neural network is employed, with an additional network (the "strobe network") used for sign separation. The "strobe" network was by far the most complex, since it used five pieces of information derived from 10 consecutive frames ($\Delta x, \Delta y, \Delta z$, velocity and acceleration) to determine the output, and it was the hardest to train, since it must be told when an action is beginning or ending, which is not an easy thing to do.

The outputs of the neural network are then connected to a speech synthesiser to form words (a DECTalk module in this case).

However, the system is very intensive because of the large neural networks employed (the "root handshape" network alone, for example, has 16 input nodes, 80 hidden nodes and 66 output nodes), and required an SGI 4D/240S to process the data in real time.

In Glove-TalkII, this system was refined and made practical for general-purpose use. A gesture-to-formant model was employed[21]. Instead of the five networks used above, three networks are used. One is the Vowel/Consonant decider, which chooses whether the current sound is a vowel or consonant, and the other two are the individual vowel selector and consonant selectors. In addition, a foot-pedal was used to control volume, and a keyboard had to be used for the "stop" sounds (like B, D, G, K, P), since it was found it was difficult to generate these by hand (since the motion would have been

[21] Formants are components of speech, identified by their unique spectral properties.

too fast). The system was taught to adapt dynamically to different users, based on their interpretation of an initial mapping. A CyberGlove was also used in place of the DataGlove, providing more information about hand movement (since the DataGlove lacks some of the sensors of the CyberGlove)[22].

Pausch and Davidson's CANDY system

CANDY (*C*ommunication *A*ssistance to *N*egate *D*isabilities in *Y*outh) is similar in concept to the GloveTalk system, although it is extremely simplified, using only two degrees of freedom to model the movement of the mouth and tongue. They used magnetic trackers placed on the bodies of the individuals involved. It is intended for use by people suffering from cerebral palsy and was designed with the intention of being made as flexible as possible. From the movement of the subject, they extract two variables: The tongue tip position and the tongue base position. By modelling the the tongue and mouth, it is possible to determine the sound such a position would generate.

Wexelblat's work on the AHIGS system

Wexelblat developed a generalised gesture recognition system as part of the work done by AHIG (Advanced Human Interface Group). In his Master's thesis [Wex93], he suggests a hierarchical approach to gesture recognition. In it, he used three Ascension Flock-of-Bird position trackers (one on each hand, and one on the head), together with a CyberGlove on each hand.

The outline of his system is shown in figure 2.14. We see that the very low level information feeds into the segmenters (which operate separately for each input, and inform the upper level of the system when there is a change in the behaviour of a particular measurement). These are then captured by proto-feature recognisers, and then feature recognisers, which when a particular set of "needs" are satisfied for the feature to be recognised, informs the path module, which looks at *all* the features. It then creates a frame. This output is then integrated temporally (seeing which things happened at the same time, and if so, encapsulate them). This output is then passed to a gestural interpreter.

[22] Several other mapping techniques were attempted in addition to the one presented, but the one presented was found to be the most effective.

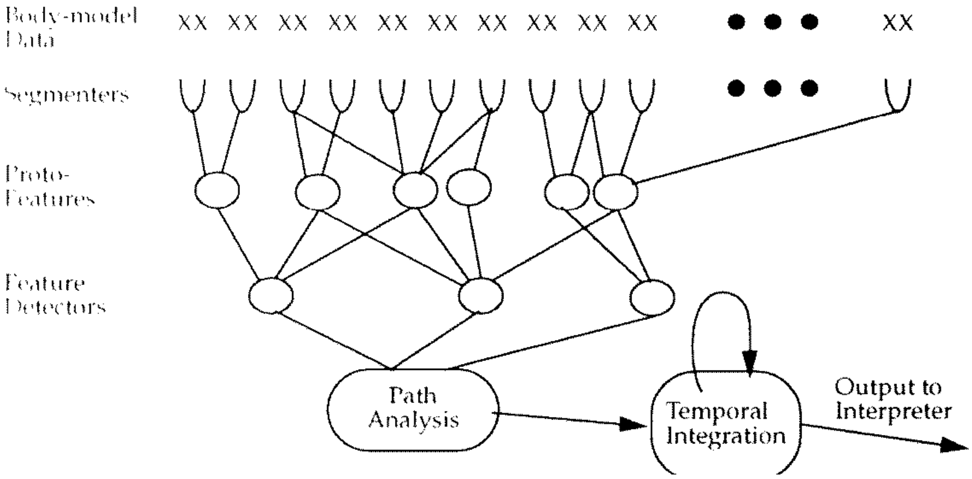


Figure 2.14: Wexelblat's system for use within AHIG.

Taken from [Wex93].

Although it has not yet been built, it is conceivable that a gestural interpreter which would be able to understand sign language could be designed. The problem, however, would be to design it to be as extensible as possible. It might be necessary to also add proto-features and features which are customised for use in sign language recognition.

Takahashi and Kishino's investigation

Takahashi and Kishino [TK91] investigated understanding the Japanese Kana manual alphabet (consisting of 46 signs) using a VPL DataGlove. They then analysed the data using "principal components". This method isolates the important factors using a statistical method, by finding the attributes that vary the least in a given set of instances, and then using a set of rules based on these instances.

Based on this, they built up a table that designated the positions of individual fingers and joints that would indicate a a particular handshape. By then matching the handshape against what is essentially this template, the handshape of the person is matched, and the class is decided on.

They found they could successfully interpret 30 of the 46 signs, while the remaining 16 could not be reliably identified, due to a variety of constraints, such as the fact that they were moving gestures and that sufficient distinction could not be made in situations where fingertips touched.

Recognition using recurrent neural nets

Murakami and Taguchi [MT91] investigated the use of recurrent neural nets for sign language recognition. A recurrent neural net is a neural net with a "feedback" loop in it. In this case, a copy of the hidden layer is taken and stored in the context layer, which is then used as input in the next cycle into the hidden layer. This is shown in figure 2.15.

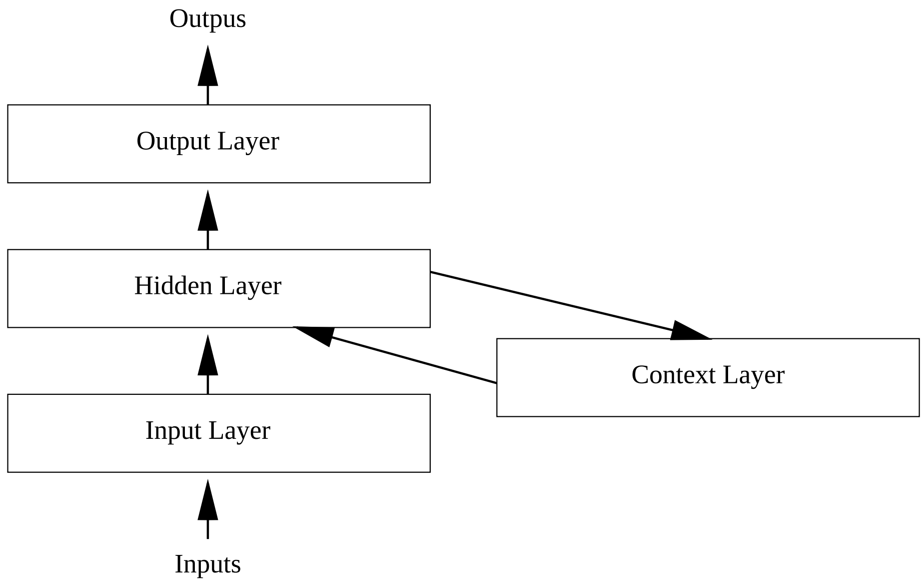


Figure 2.15: The structure of the recurrent neural network used by Murakami and Taguchi.

First,they trained the system on 42 handshapes in the Japanese finger alphabet, using a VPL Dataglove. This system was successfully built and obtained success rates of approximately 98 per cent.

Then, they managed to recognise continuous finger-spelling by only accepting a symbol as positive when a certain threshold was exceeded in the output layer of the neuron.

They then chose ten distinct signs (mostly in pairs – such as father and mother, brother and sister, memorise and forget, skilled and unskilled, like and hate – which in Japanese Sign Language are very distinct gestures).

Their neural network was very large: 93 in the input layer (consisting of the past three frames, each frame having finger position and absolute and relative position, as well as orientation information), with 150 in the hidden layer and the mirroring 150 in the context layer.

It recognised the beginning of signs by checking if it met any of a set of postures, and would stop when one of the output nodes consistently output a particular value.

It was quite successful, with accuracy of approximately 96 per cent. However, there was only a small number of signs (10) and it is not clear whether the technique would have generalised well (considering that the system already had in excess of 400 neurons already, it is highly likely that this will need to be increased, which will slow it down even further).

Kramer's Talking Glove Project

James Kramer and his supervisor, Larry Leifer, have been working on a method for communication between Deaf, Deaf-Blind and Non-vocal individuals. It is a complete system, which attempts to integrate a number of technologies together, in such a way that all parties can communicate. This is shown in figure 2.16.

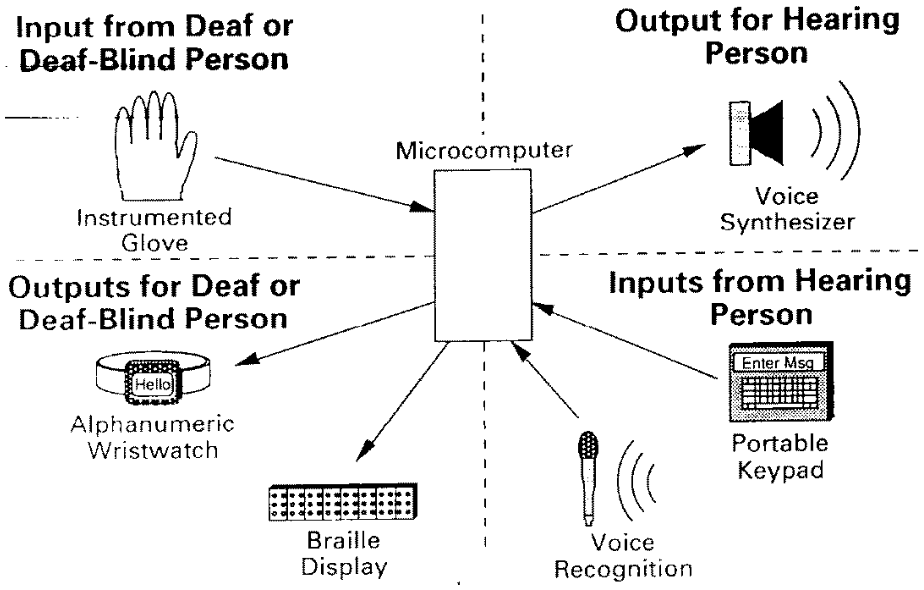


Figure 2.16: The complete Talking Glove system.

Taken from [KL90].

Of interest and relevance to us is the glove Kramer developed (the CyberGlove) and the technology used for recognising American finger-spelling.

At the moment, the system has acceptable finger-spelling performance, and James Kramer has set up a company, called Virtual Technologies, to market the system. In September 1995 VirTex released a commercial finger-spelling recognition package called GesturePlus. However, the system costs US$3500, and this price does not include the CyberGlove.

Initially, he used a prototyping algorithm. There were prototypes for each letter, which Kramer termed "beacons". These beacons which are points in the hand state vector space, are then surrounded by hyperspheres. When a hand enters the the hypersphere, a letter is produced. This hypersphere, which Kramer calls a "recognition ball", lies within another hypersphere, which Kramer terms the "hysteresis ball". To repeat a sign, the hand must move outside the hysteresis ball, before another handshape is recognised (this intuitively makes sense, since when finger-spelling and a sign is to be repeated, it is typically made distinct in a similar manner – ie the gesture is repeated twice). See figure 2.17. This shows the hand-state moving through a vector space, which in this case is simplified to three dimensions, but there are 16 dimensions in the actual model.

The above system has been road-tested and has been shown to be practical.

Currently, Kramer is working on moving the above model to a neural-network implementation, and also broadening the scope to more than finger-spelling – that is to full signs.

A Linguistic Approach to Recognising Gestures

Hand, Sexton and Mullan [HSM94] believe that perhaps a grammar-based approach to understanding gestures might be feasible. While they did not directly apply their techniques to sign language, as a personal opinion, I believe further investigation in this area might prove fruitful.

To do this, they define several postures that were the terminals of the language. These included things like HO – hand open, HF –hand fist, IF index finger outstretched. To these, they added several direction movements, like: MU, MD – move up and down; ML, MR – move left and right; and MT, MA – move towards and away.

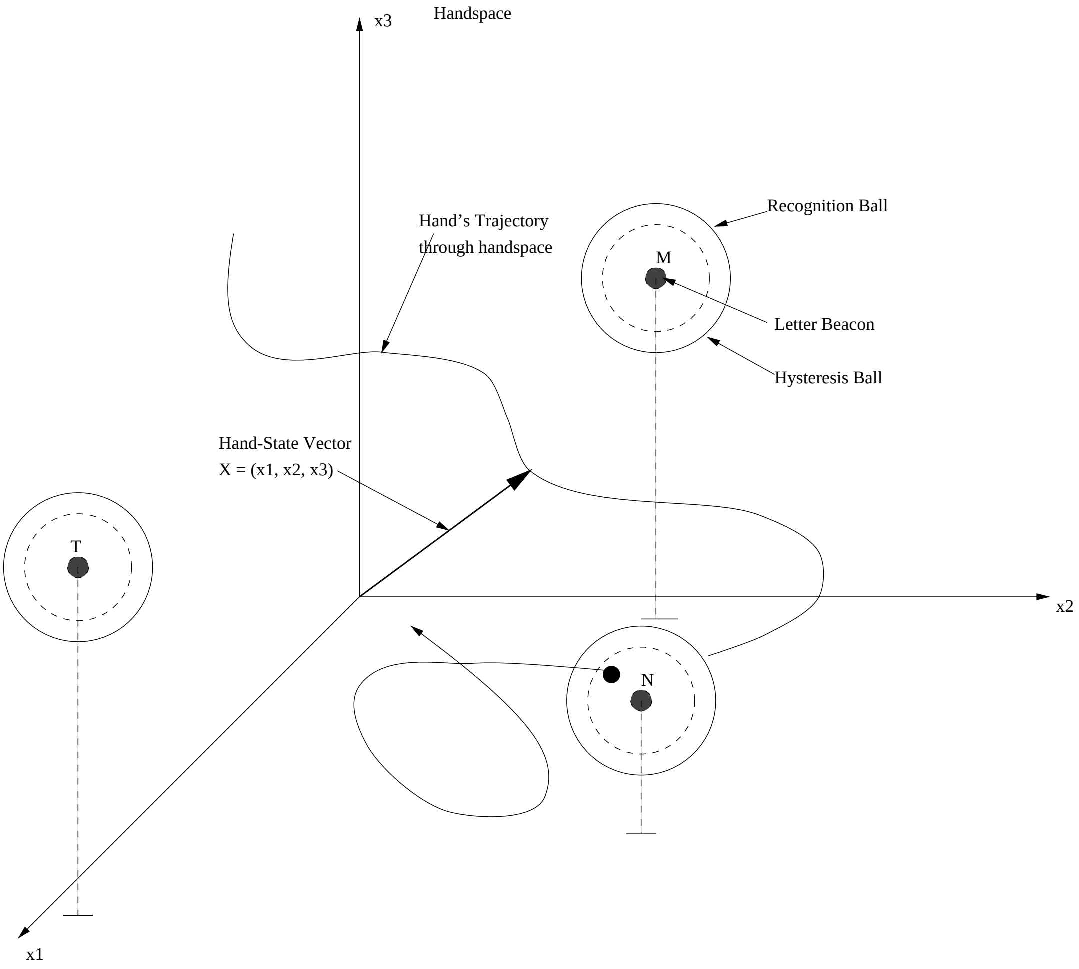


Figure 2.17: The approach adopted by Kramer to recognise finger spelling.

Based on a diagram from [KL89].

They then defined the production rules, a few of which are shown below:

```
<Gesture> :== <Pick> | <Point> | <Translate> | <Undo> | <Redo> | <Stop>
<Pick> :== <ChooseLocation> <Grab> <Drag> <Drop>
<ChooseLocation> :== TIF-Motion <ChooseLocation>
<Grab> :== TIFC
<Drag> :== TIFC-Motion <Drag>
<Drop> :== TIF
<Point> :== <MovePoint> <SelectPoint>
<MovePoint> :== IF-Motion <MovePoint>
<SelectPoint> :== TIF
<Translate> :== FTFO <Direction>
<Direction> :== ML | MR | MU | MD | MT | MA
<Undo> :== IMFO MA
<Redo> :== IMFO MT
<Stop> :== HF HO
```

This indicates for example that to stop, you would make a fist and then release it – or to drag an object, you would hold your thumb and index finger closed and continue to move it.

It is easy to see how such an approach could be adopted for sign language use, with the terminal symbols being handshapes, with additional terminal symbols for changes in orientation and position. Of course, there would be some complexity in the symbol generator, which would take the input from the glove and produce symbols. The second major drawback is that currently, automatic grammar construction is not sufficiently advanced to support this sort of approach, although several techniques exist (as discussed in [Cha93]).

Thus all new signs would have to be added and tested manually. Since the size of the grammar is not known in advance, it is possible that the grammar of Auslan might be "ambiguous" – that is it may not be possible to distinguish all signs, on the basis of structure alone.

Peter Vamplew's work with SLARTI

SLARTI is (as far as I know) the only other investigation of Auslan in particular. Peter Vamplew (University of Tasmania) has been adopting a neural-network approach to the problem of understanding Auslan. His thesis is still not complete. However, partial work illustrated ([Vam]) is extremely promising.

SLARTI (*S*ign *La*nguage Recogni*ti*on) is a system built around Kramer's CyberGlove [Kra91], augmented with a Polhemus tracking device, which gives the 6 additional degrees of freedom discussed so far. The focus is of course on using motion to analyse sign. At the moment, the system is effective at recognising the Auslan handshapes. A discussion of some of his results can be found in appendix A.

While still unfinished, it is important in that it focusses on Auslan. Auslan is also interesting because it is considered a dialect of British Sign Language (BSL) and thus any findings for Auslan may be generalisable to BSL as well, about which there is relatively little research so far.

2.8.3 Summary of previous research

There is some work that has been done in areas similar to sign language recognition, such as Fels' ([FH93, Fel94]) and Pausch's ([PW91]) work. Also, there is some of work in progress, such as Vamplew's ([Vam]) and Dorner & Hagen's ([DH94]). Some work has also been done on recognising handshapes ([ITK92, TK91, DS93]).

The work that has been published in the specific area of recognition of individual signs is still limited. Murakami and Taguchi considered ten signs using a recurrent neural net and obtained accuracy of 96 per cent ([MT91]). Charayaphan and Marble considered 31 ASL symbols, but only sampled each sign once and simulated the variation and consistently got 27 out of the 31 correct, with the remaining four sometimes correct using a Hough transform. Starner considered 40 signs used in brief sentences and ([Sta95, SP95]) obtained accuracies of 91.3 per cent on raw signs and 99.2 per cent by using a very strict grammar for sentences.

Chapter 3

Resources, Equipment and Exploration Areas

How much did you have to spend and how is your work different?

3.1 Introduction

We now consider the choices made with respect to the approach and resources used. First we consider the hardware and software requirements. We then summarise how this thesis is different from previous research.

3.2 Choices

There were many factors to consider here. The main limitations were cost and availability.

3.2.1 Choice of data collection technique

As mentioned previously, there are two schools of thought on the issue on how information about sign language should be gathered, whether via vision-based techniques or device-based techniques.

A device-based technique was selected for the following reasons:

- Cost. The cost of a camera is significant, but more importantly, graphics boards for capturing the information at a reasonable rate are also costly.
- Bandwidth and processing power. A very high-end glove might produce 40 kilobytes per second of data (200 packets arriving per second each packet consisting of 200 bytes). A low-end camera might produce 1 megabyte per second (320 by 200 image sent 5 times a second). Furthermore, each image must be analysed, relevant data extracted and so on. Storage of the data is also a problem. If four people make ten samples of each of 100 signs, then the signs from a glove would take 160 megabytes, the information from a camera would take 4 gigabytes.
- Complexity and space. Setting up a camera can be complex, and factors such as lighting and so on have to be accounted for. Although similar problems arise with a glove, they are not as dramatic in extent.
- Additional complexity of decoding image data. The thesis is only one year long, and there are limits to how much might be accomplished in one year. Significant effort would have to be exhausted simply extracting relevant data from the frames.

Given that a device-based approach was to be used, there were still some considerations that had to be made. Several options were considered.

- Building the glove from scratch, using as many "off the shelf" components as possible. This would offer the possibility to design a glove that suited the purposes of signing most ideally. After speaking with others, however, it was decided that this was a thesis in itself, and that it would be difficult to achieve in time, especially since some of the devices (the small strain gauges to be used for flex sensors and abduction sensors and the position tracking devices) were not locally available.
- Using one of the high-end pre-built gloves such as the VPL DataGlove, the VirTex CyberGlove, and the EXOS Dextrous Hand Master (the EXOS DHM less so because of the difficulty in setting it up on a person's hand). These

provide very accurate information about hand position. These would have been ideal, except for one slight point — they are expensive. The gloves in this range typically start at US$5000. Also, using gloves this expensive would violate one of the goals of the project which was to show that a system could be achieved at reasonable cost.

- Using a Mattel PowerGlove. The Mattel PowerGlove, originally designed for use with video games, surprisingly had its origin in VPL. It has, as previously discussed, many design corners cut.

The approach finally adopted to purchase two PowerGloves with PGSIs and then consider various augmentations to the glove to get it "up-to-scratch" if necessary. One glove would be kept as backup.

3.2.2 Choice of computer

The SGI Iris-4D/35 was selected for data acquisition, because it had good graphics features, which is useful for visualising the data. In addition, the SGI's have two serial ports, a convenient feature, since it was possible that a second input device could be used.

This was not felt to violate the "price" objective, since most of the software once it had been developed, could be easily ported to other computers, because the software developed was simple. In fact, if time remained, the possibility of porting the software to one of the new "palmtop" computers (such as the Hewlett-Packard HP200LX) was considered – just to prove the point. Obviously, however, the HP200LX is not exactly the ideal development environment.

3.2.3 Software choice

More interesting, however was the choice of the "pattern recognition" software. After some consideration, the decision reached was to examine two of the learning approaches concurrently – instance-based learning techniques and symbolic learning algorithms, with a strong emphasis on extracting features from the data using conventional data processing techniques, rather than trying to feed the raw data to these algorithms.

This approach was adopted for a number of reasons:

Originality of the Approach

As far as I know, the application of symbolic learning algorithm techniques and instance-based learning to sign language recognition has been limited.

This project would serve to examine the viability of symbolic learning algorithms and instance-based learning in the field of gesture recognition.

Discrete outputs

Symbolic learners and instance-based learners are tuned towards discrete outputs – and in this case, the discrete outputs are the signs.

Many other output methods are really function approximators, and while it is true that a discrete output can be modelled as a function, this can sometimes be counter-intuitive and "kludgey". Using neural networks, for example, two possible approaches would be to have an output layer of neurons, with the neurons producing a "binary digit" of high and low values representing the signs, or have a separate neuron for each possible output sign. As previously discussed, there are problems involved in using either of these.

Scalability

In many ways, this investigation is a "proof of concept" that shows it is viable to consider sign language recognition: we are using 95 signs out of the Auslan dictionary [Joh89] of 4000 or so. Classification time of decision trees is approximately logarithmic in the number of training instances (because of the tree structure employed for classification) and similarly, classification time for instance-based learning techniques can be made logarithmic in the number of training instances, using k-d trees ([WK91]). Learning times for simple instance-based learning techniques is linear, and it has been shown that under certain circumstances, so is C4.5 ([Squ94]). If we have more samples, we can simply add them in, without the major penalty of a quadratic increase in the time-cost of learning. Furthermore, both systems can operate incrementally (i.e.

more training instances can be added to the learner without significant time penalties), unlike neural networks, which much be retrained if new training instances need to be included.

"Tweakability"

All learning algorithms have certain aspects which may be labelled as "tweaks" – factors that significantly affect the behaviour of the algorithm for which there is no obvious or straightforward way to set. For example, in HMM's the set of states and transitions of the the HMM is difficult to arrive at. For neural nets, the number of neural nets in each layer, output encoding, neuron interconnection and error function need to be considered. In instance based learning techniques, deciding on the similarity function is a major difficulty if high performance is required. In symbolic learning there are also variable such as the pruning level, assessing the usefulness of particular attributes and so on.

However, instance-based learning and symbolic learning appear to be the least complex of these.

Furthermore, because of the speed of learning compared with other techniques, effective state-space searches (eg hill-climbing, simulated annealing or just good old exhaustive search) can be accomplished in a feasible time[1].

Speed

By speed, several things are meant. This means both classification and learning speed. Symbolic decision trees can obviously be evaluated very quickly, and it is possible to design a tree-builder which builds trees incrementally – i.e. as new instances come in, they can be added to the tree without having to destroy the old tree. IBLs are also incremental learners. Although naive IBLs can be quite slow, by using techniques like k-d trees quite good performance can be obtained.

[1]For example, if each execution of C4.5 takes 20 minutes, this means we could sample 72 possibilities a day. A simple neural net might take 3 hours to evaluate, which would mean 8 possibilities considered a day.

3.3 Original aspects of the investigation

This investigation is differentiated in four main ways from previous research:

Study of Auslan

It studies the application of sign language recognition techniques to Auslan. There are very few researchers working in this area, and most investigations have focussed on ASL (mainly because of the one-handed alphabet[2]). Systems developed from Auslan could also be easily converted for use in British Sign Language and vice versa, but it is harder to adapt American Sign Language to Auslan. So far to my knowledge only Peter Vamplew has worked on Auslan.

The use of Symbolic Learning Algorithms and Instance-based learning for Gesture Recognition

Symbolic learning algorithms, although conceptually simple, have been shown to be effective in some applications that are traditionally dominated by other learning methods, such as speaker recognition ([Squ94]). They have several advantages that would appear to make them a good candidate for applications in gesture recognition – such as advantages in speed of evaluation and computing power required, in terms of time and space.

Similarly, investigations of the use of instance-based learning techniques have been limited in scope, but are one of the most trivial forms of learners available.

Use of cheap equipment

A PowerGlove has extreme noise and accuracy limitations, as should be obvious by now. However, it is sometimes possible to use a small amount of information to extract enough identifying and discriminating attributes from data to uniquely identify it[3].

[2] And the greater research dollars available in the US, and the larger Deaf population to support it.

[3] Humans have become incredibly good at this, although we frequently don't appreciate it. We can for example determine a type of material simply by running our fingers over it, yet the sensation of

Also note that sign language is two-handed, but that we only have one glove. This is a significant problem, but its severity is alleviated by a number of factors:

- A very significant proportion of signs are one-handed (approximately 40 per cent).
- Of the signs that are two-handed or double-handed, a significant proportion are symmetrical or near-symmetrical.
- Of the remaining signs, the dominant hand[4] usually makes the most complex movements, with enough character to identify it uniquely.

This does not mean, however, that there are no signs solely distinguished by what the passive hand is doing; it is just that fortunately, this does not occur in a large number of cases.

This will certainly be a challenge, but if it is to be an effective communication tool, that can be used in the general domain, then a large proportion of the population could not afford to purchase US$5000 gloves. At US$500 or below, however, people might be able to afford it, even if they have to put up with a slightly higher error rate.

Relatively large lexicon

In this thesis, the goal lexicon is approximately 100 signs. Most previous research has focussed on small collections of signs. This introduces a range of issues that do not arise with small lexicon systems. As will be shown, issues of learning time and accuracy become dominant in large lexicons, and this is a further step in the direction of a full-size lexicon.

touch is not incredibly complex, and does not provide particularly accurate or clean data [Rhe91].

[4] The dominant hand may be either the left hand or the right hand, but is consistent for a given signer – just as in handwriting. However, the PowerGlove only comes in a right-handed model. This means that the system as it exists now is only suitable for right-handed signers. This will be fixed once the PC-PowerGlove arrives.

Chapter 4

Framework Development

What does GRASP actually look like?

4.1 Introduction

The complete system as envisaged is shown in Figure 4.1.

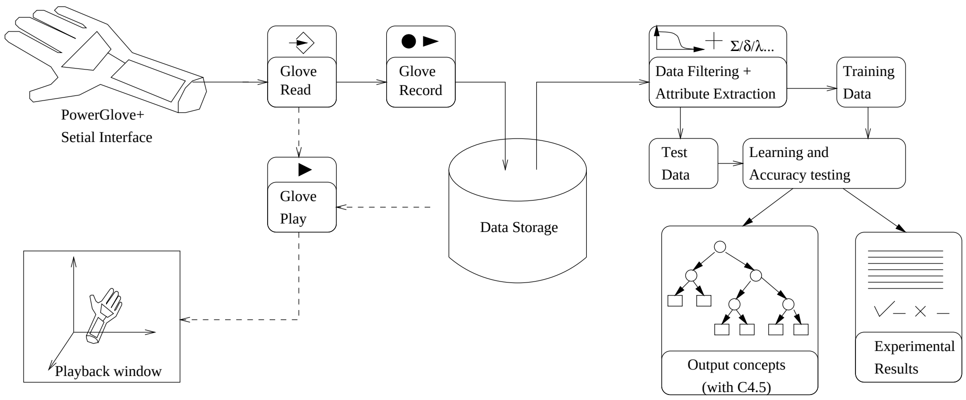


Figure 4.1: The complete GRASP system

The following section will discuss the design of each of these stages.

4.2 The Glove software family

After considering existing software, it was found that none had sufficient modularity for this task. Thus it became necessary to write our own driver and display software. To ensure "plug and play" at a software level, this was designed in a modular way.

The software is printed in appendix C. It is also available by ftp and can be obtained by contacting the author.

4.2.1 glovedata

Before any real work was begun on software applications, it was felt necessary that a library for the glove be developed. This library had a data structure glovedata which mirrored the packet of information coming from the glove, shown in table 4.1. However, there were also a number of fields that the PowerGlove did not provide but were included for flexibility and the possibility of replacing the PowerGlove with a better glove[1]. It had four functions:

printglove & fprintglove: When given a glovedata structure, these function print out the data structure (printglove to STDOUT and fprintglove to a file). The output is both human-readable and machine-readable.

readglove and freadglove: These functions complemented printglove and fprintglove by reading from STDIN (or a file in freadglove's case) and putting the information in a glovedata structure.

4.2.2 gloveread

In order to isolate the hardware level to a particular program, gloveread was written. This takes a continuous stream of bytes and converts them into glovedata objects before outputting them by calling printglove. It was implemented as a finite state machine that received individual bytes. This gives maximum flexibility and also also graceful recovery from bad packets.

[1] In the vague hope that by some miracle a $6500 CyberGlove would appear on the doorstep of the School of Computer Science and Engineering one day.

Packet Position (Byte)	**Name**	**Description**	**Legitimate values**
00	Header	Packet start	5F
01	Header	Packet start	5F
02	X	X dimension reading	0 - FF (-128 - 127)
03	Y	Y dimension reading	0 - FF (-128 - 127)
04	Z	Z dimension reading	0 - FF (-128 - 127)
05	Rotn	Rotation of wrist in 30 degree increments	0 - B (0 - 11)
06	Flex	Flex information byte	0 - FF
7:6		Thumb flex	0 - 3
5:4		Forefinger flex	0 - 3
3:2		Middle finger flex	0 - 3
1:0		Ring finger flex	0 - 3
07	Keys	Keys being pressed	0-FF
08	GSTAT1	General status 1	0 - 1
09	GSTAT2	Unused	0
10	RECVALS	Receiver values	0 - 3F
5		Left top received from left transmitter	0 - 1
4		Right bottom received from left transmitter	0 - 1
3		Right top receiver from left transmitter	0 - 1
2		Left top received from right transmitter	0 - 1
1		Right bottom received from right transmitter	0 - 1
0		Right top receiver from right transmitter	0 - 1

Table 4.1: Packets of information arriving from the glove.
Adapted from from [Stu94].

Typically this would be combined in the way shown below:

```
( stty raw -parity -echo ; gloveread ) < /dev/ttyd2
```

This sets up the second serial port on the SGI (/dev/ttyd2)[2] before passing the data on to gloveread. Gloveread then prints the results to STDOUT, in a comma-separated list that can be read by calling readglove on the stream, but is also human-readable.

4.2.3 gloverecord

The gloverecord program expects input that is readable using readglove, and implemented the sampling protocol discussed in section 4.3, by going through a list of words given in a file and asking the user to make that sign a given number of times. Each of these signs was then saved in a separate file.

Typical use might be something like:

```
( stty raw -parity -echo ; gloveread ) < /dev/ttyd2 | \
  gloverecord word-list save-dir 1
```

This would take the information from the serial port, which gloveread would convert to the readglove format, which would then be captured by gloverecord. The options to gloverecord are the list of signs to ask the user to provide, the directory in which the samples are to be saved and the number of times to ask the user to provide those signs. Files are saved in the directory using the name format `<sign-name><sample-number>.sign`.

4.2.4 gloveplay

Gloveplay expects a stream of lines readable using readglove. It then takes this information and creates a very simple 3D model of a hand, and models the movement of the fingers, using the SGI GL system.

[2] It was decided that it was best to separate the functions of setting the terminal and the actual glove-reading to allow the code to be maximally portable – even to platforms other than Unix. Provided the operating system the software is being ported to has some form of redirection possible, then gloveread will run on it.

Typical usage might be something like:

```
( stty raw -parity -echo ; gloveread ) < /dev/ttyd2 | gloveplay
```

which would produce a real-time moving hand or

```
cat yes3.sign 5 | gloveplay
```

which would "play back" an already existing sign, stored in the files `yes3.sign`. The "5" indicates a pause between consecutive frames in units of fiftieths of a second. This can be used for providing a "slow-motion replay" of a sign. In this case, playback occurs at 10 frames per second instead of the standard 25. An example of the display from gloveplay is shown in figure 4.2.

Together, these give a toolset that allows us to build applications for recording and visualising the data.

4.3 Selection of classes and data acquisition methodology

In this thesis it is obviously necessary to select a *subset* of Auslan to be considered for testing of GRASP, since it would take vast amounts of time to sample most or all of the signs in Auslan. Several issues then arise in the selection:

- How many signs do you choose?
- On what basis do you select the signs?
- How do you uniquely identify a sign?

The number of signs should at least be sufficient to incorporate as many of the common aspects of signs as is possible.

This includes things such as:

- The 31 handshapes.

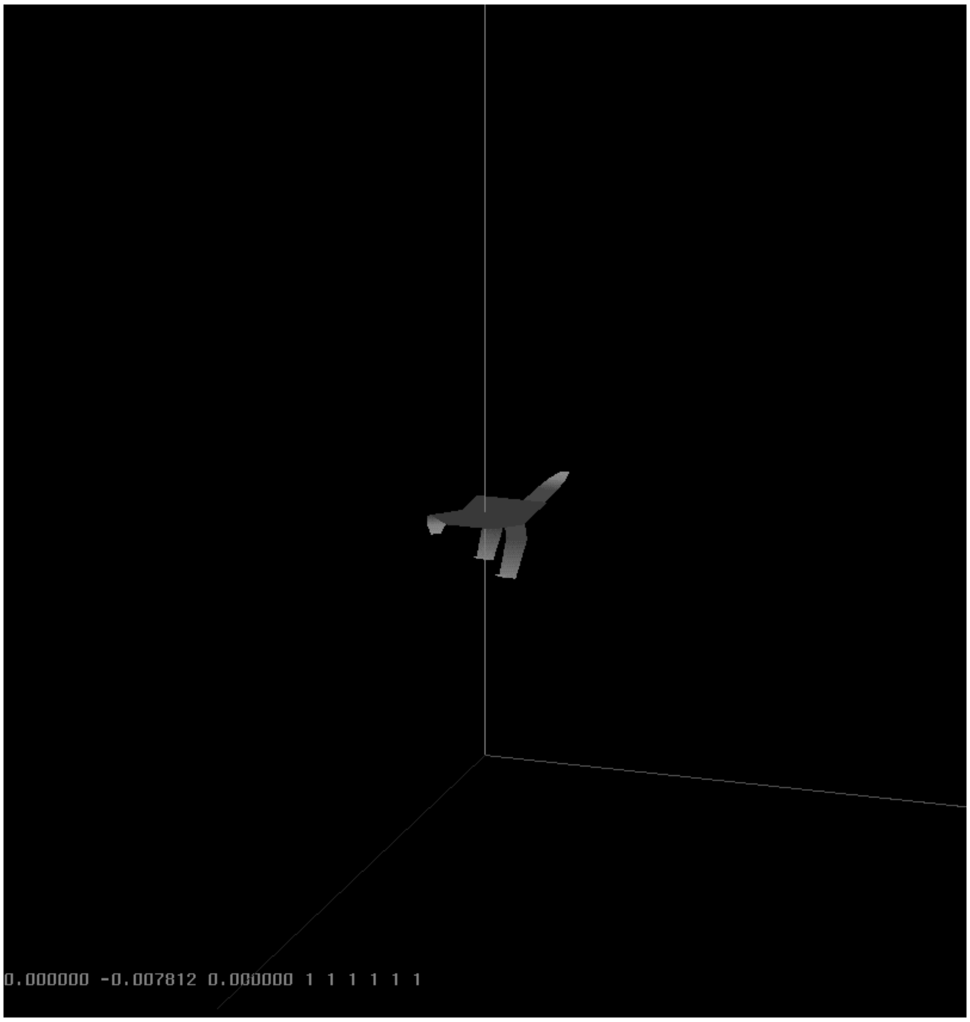


Figure 4.2: Screen dump of the gloveplay program.

- The various types of movements possible, such as shaking, rotatory, to and fro, pointing, slow and fast etc.
- The various areas of the body.
- The various sign types as far as hand involvement is concerned, such as one-handed, symmetrical double-handed (both hands do exactly the same thing), true double-handed (both hands have the same handshape but are not doing the same thing), two-handed with simple subordinate handshape (the hands doing different things, but the subordinate handshape being something like a flat palm or a fist) and true two-handed signs (each hand does completely independent things that are roughly equal in complexity).

At the same time, it is also important to focus attention on distinguishing signs that you would expect it to have difficulty differentiating. Thus, when signs were selected, one of the possible reasons for selection was its similarity to other signs. This can be seen, for example, in the selection of a large number of gestures around the right side of the face, such as when, think, know, forget, head, and in families of signs with similar gestures, such as shop, spend, buy, money, pen, draw, write and right, is-true, juice. This was in line with the goal of expanding the system to a full-lexicon system, and seeing if it would have trouble telling them apart. As will be shown, this did not appear to be a major obstacle.

While it is impossible to capture all aspects of sign language with anything less than, say, a thousand signs, almost all features can probably be encapsulated in one hundred or so signs.

The above aspect also cross in to the second area which is how we select which signs we are going to include in the project.

At a global level, we wish to meet the above criteria. But there are many sets that can fulfil the above criteria, so in addition, the following method was used for the selection of the signs to be included.

- A family of commonly used signs were included. To establish "common use" a number of methods were used. First, a brief dictionary of signs ([JRC88]) was consulted to see what one group of Deaf people believed were important signs. Second, the Auslan Dictionary [Joh89] was scanned, and the importance of each sign in the dictionary was personally assessed on the basis of the author's day-to-day experience.

- It was then seen how many of the signs covered interesting aspects. The Auslan Dictionary was rescanned for signs that were "interesting" as far as having aspects that were not found in the common signs. This included many shapes that made use of the rarer handshapes, as well as finding signs that were similar to see how well the system coped with them.

The complete list of signs can be found in Appendix B, and with the explanation of the basis for its selection. Some statistics about the signs are shown in table 4.2.

Item	**Number of signs**
Total number of signs	95
Hand usage	
- One-handed	59
- Double-handed	27
- Two-handed	9
Reasons for selection	
- Common sign	66
- Similarity to other signs	11
- Uncommon Handshape	9
- Sign Complexity	5
- Other	4

Table 4.2: Some important statistics on the signs used.

In table 4.2, there are a large proportion of signs that are one-handed. This is because most common signs are one-handed, even though one-handed signs only form approximately 40 per cent of the signs in Auslan. A sign was selected on the basis of "Sign complexity" because of some interesting aspect, like the large number of critical points or a changing handshape. An "Uncommon handshape" indicates the sign contained a handshape that was not found in the other signs selected. "Similarity" indicates that the sign was chosen because it was similar to some of the other signs selected. "Other" indicates that was chosen because the author felt like it, or some other obscure reason, like its relevance to the investigation.

Another problem was how to uniquely identify each sign. Since there is often more than one sign for a given word or a word does not translate well to a sign, it is not sufficient for a single word to be given. To solve this, a single name was given to each sign, but to uniquely identify it, the Auslan-dictionary reference number is given[3].

4.3.1 Data acquisition protocol

A major issue in this thesis was how to collect data in a way that minimises variations. As mentioned previously, in this thesis we only consider discrete signs. Thus we want the following aspects to be reasonably invariant:

- The location at which a sign is started, physically. This also allows us to calibrate our origin of the sign at a fixed location in space.
- The difference in the "calibration" between one sign and the next. In particular it was found that the fingers were subject to a slight hysteresis.

To solve this problem, it was decided that each sign would start at a fixed position around the body somewhere. A number of such locations were considered. These were narrowed down to the hand being placed on the shoulder and the hand being placed on an armrest to start with. It turned out to be very uncomfortable to touch the shoulder at the start of a sign; and in addition, the ultrasonic transmitters turned out to be less accurate when vertically aligned. Also, a fist was made at the start of each sign to minimise (although not completely remove) the effect of the calibration error.

To ensure calibration with each sign, gloverecord was programmed only to begin recording when the "CENTER" button the glove was pressed. Pressing the "CENTER" button on the glove not only sends a '00' as the key pressed, but also forces the PowerGlove to recognise this point as the new origin (ie the locations in space of all points after the last centre are relative to that centre). This also has the advantage of minimising the "limit effect" – which results because of the 8-bit limit on X, Y and Z position[4].

[3] As it turns out, even this is not sufficient, since frequently people will interpret the gloss differently. This turned out to be one of the reasons that it was difficult to get inter-signer recognition working as well as intra-signer recognition.

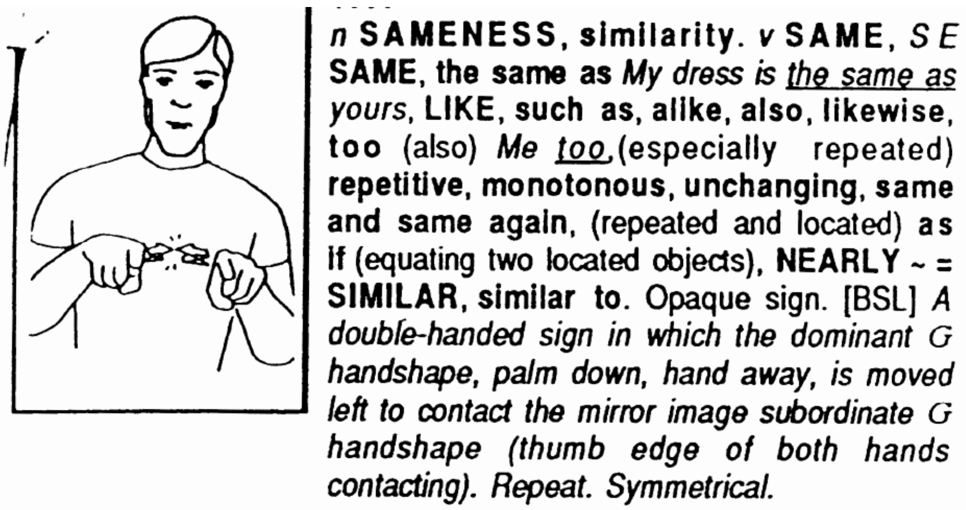
[4] Each "click" in the X dimension for example is about 0.5 cm. This only gives a total swing in each of the dimensions of about 120cm, which is not introduced by the hardware, but by the simple fact that you can only express 256 values with 8 bits. This is enough for a relative positioning system (since most signs do involve some bend at the elbow and people do not like to swipe each other out by swinging their arms around dramatically), but it is not enough for an absolute positioning system that is not centred relative to the body.

To indicate the end of a sign, a particular button ("A") is pressed on the PowerGlove. Although it might be possible to get the software to automatically stop recording when the PowerGlove re-entered the space near the origin, this would be dangerous, since it would be possible that erroneous data could result in the glove thinking it has returned to the origin, resulting in premature termination of recording. Similarly some signs could conceivably enter the space near the origin in the course of the sign itself.

The user was then asked if he wished to repeat the sign if he felt he did not "do it right", by pressing "B".

An example may help at this point.

Consider the sign for `same`. The gloss for `same` is shown in figure 4.3.



n **SAMENESS, similarity.** *v* **SAME,** *S E* **SAME, the same as** *My dress is the same as yours,* **LIKE, such as, alike, also, likewise, too** (also) *Me too,*(especially repeated) **repetitive, monotonous, unchanging, same and same again,** (repeated and located) **as if** (equating two located objects), **NEARLY ~ = SIMILAR, similar to.** Opaque sign. [BSL] *A double-handed sign in which the dominant G handshape, palm down, hand away, is moved left to contact the mirror image subordinate G handshape (thumb edge of both hands contacting). Repeat. Symmetrical.*

Figure 4.3: The Auslan Dictionary definition of same.

Taken from [Joh89].

Gloverecord would then, in the course of asking for a sequence of signs, ask the user to make the sign same:

```
Please sign the word: <<<same>>>.
Press Center to begin.
Press 'B' on the glove to redo the last sign.
Press 'A' on the glove to end, once the sign is complete.
```

The user begins by pressing the "Center" button of the glove, with his hand in a fist and his arm on the armrest, shown in figure 4.4.

Gloverecord starts capturing the information from the glove at this point. It saves all frames between this and the end in a separate file.

Figure 4.4: The user starting the sign

The user then continues to perform the sign (see figures 4.5, 4.6 and 4.7):

When the user has completed, he then tells Gloverecord to stop recording, by pressing "A" on the PowerGlove. If for any reason he/she is not satisfied, with the sign they just performed, they can repeat the sign by pressing "B".

Figure 4.5: Towards ...

Figure 4.6: Meeting fingers ...

Figure 4.7: Away ...

Figure 4.8: The user indicates the sign is complete.

Chapter 5

Feature Extraction and Selection

How do you turn a stream of noisy data into something a computer can chew on?

5.1 Introduction

We now have a stream of packets that are stored in files that originated as frames about glove state. There are approximately 25 such frames per second of signing. While it might be possible for this information to be fed directly into some learning algorithm (in fact, it was attempted by Murakami and Taguchi [MT91]), this is likely to be computationally expensive and does not take advantage of good-discriminating attributes that can be extracted simply from the data. Furthermore, a great deal of effort would need to be expended in extending existing learning algorithms to fulfil this obligations – since most common learning algorithms are not able to handle time-varying data in a sensible manner.

Thus in this investigation, the focus was on extracting pieces of information from the data, which we will term features or attributes[1] that are good for discriminating *between* signs.

[1] The words "feature" and "attribute" are used synonymously in this document. From my understanding of the issues, they are called features by the people trying to extract the features/attributes and attributes by the people who write the learning algorithms. Furthermore, when the word "feature" is used here, it is not in the same context as that used in Wexelblat's thesis, where a feature means an interesting change in a value of the raw data.

This, of course, is the central part of the investigation, and most of the effort was expended in trying to come up with, calculate, assess and validate features that can be extracted by simple processing.

5.2 Getting a clean signal

It was found that while the information coming from the fingers was reasonably clean because of the automatic calibration that occurred between each sign, but positional information (in terms of x, y and z) was often found to "glitch"; that is, to change to values that were not just randomly noisy, but physically impossible. There was of course much noise on the position (as would be expected with 8-bit resolution), but most of it was no more than 1 to 2 centimetres of displacement. Occasionally, however, the data returned would indicate that the hand had moved 40 or so centimetres in 1/25th of a second, which would hardly ever occur in the course of normal signing[2]. There are several causes for this:

- Random ultrasonic noise. Things like monitors and fluorescent light bulbs generate some ultrasonic frequency noises. Also, and this is probably the more problematic, reflections of the signals emitted by the glove itself[3] off rigid, non sound-absorbing surfaces will interfere with the positioning. This is remedied by the fact that most commonly the original signal will be the strongest.

  Steps were taken to reduce the effect of noise. For example, the environment was arranged so that the walls were at a 45° angle so that reflections were not likely to interfere. The walls were also covered with a thin material which reduces the intensity of the reflected signals. Tests with the fluorescent lights on and off showed that they had little effect on performance.

- Ultrasonic sensors work best in "line of sight". This means that if an object blocks the path between the glove and the L-bar, the glove becomes more susceptible to glitches, since it is possible in this case that a reflection is stronger

[2]Unless a signer is *really, really* annoyed and has entered into a very heated debate. In the same way that speaking people raise their voices when they're arguing, signs become more exaggerated and energetic as the arguments become more heated.

[3]It seems unusual, but the ultrasonic emitters are mounted on the glove itself, and not vice versa – i.e. a logical alternative would be to put the emitters at the L-bar and the receivers on the glove. As Chris Hand said in discussion of this apparent design flaw: "I don't know why they did it. It was a mistake anyway. If you go the other way round (as the Logitech flying mouse does), then you're sending ultrasonic pulses towards something soft and round (human being) rather than something hard and flat (monitor, desk) which reflects echoes of the pulses all over the place, causing glitches."

than the original signal. This can happen quite frequently, most commonly when the hand itself gets in the way. For example, consider waving hello. The hand blocks the space between the transmitter and the receiver. This effect is even more dramatic when the transmitter and the receiver are pointing in opposite directions, in which case the data coming in is almost random. This also happens frequently – consider the sign for I where you point towards yourself. There is very little you can do about this.

To remove these glitches a heuristic method was adopted. This is best expressed in a mathematical notation[4].

Each file analysed consists of a sequence of frames. In each frame there is information about x, y and z position, wrist rotation and finger bend.

Let x_i, y_i, z_i be the x-position, y-position and z-position stored in the ith frame of the sample.

Then we can define the change in direction from the last frame as:

$$\begin{aligned} \Delta x_i &= x_i - x_{i-1} \\ \Delta y_i &= y_i - y_{i-1} \\ \Delta z_i &= z_i - z_{i-1} \end{aligned}$$

The vector $(\Delta x_i, \Delta y_i \Delta z_i)$ is thus a pointer in the direction we are going in. Thus if we take the length of this vector and call it Δ_i , i.e.:

$$\Delta_i = \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$$

then this will give us a measure of speed. In effect this is the first (discrete) derivative with respect to time.

Similarly we define the second derivatives:

[4]Don't worry, nothing too complex.

$$
\begin{aligned}
\Delta^2 x_i &= \Delta x_i - \Delta x_{i-1} \\
\Delta^2 y_i &= \Delta y_i - \Delta y_{i-1} \\
\Delta^2 z_i &= \Delta z_i - \Delta z_{i-1} \\
\Delta_i^2 &= \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}
\end{aligned}
$$

Here again, the vector $(\Delta^2 x, \Delta^2 y, \Delta^2 z)$ is a vector that represent the *change* in direction that has occurred. Δ_i^2 is similarly the length or norm of that vector.

Thus an intuitive way to filter the glitches is:

1. See what the practical limits of movement are. This was done by creating special signs – such as fast movement in a straight line, making the hand move in small tight circles in a tight manner, and rapid to and fro movement. These files were "hand de-glitched" using the gloveplay program on the data files. They were then put through the analysis discussed above to find what the practical limits were.
2. When filtering a sign, check the Δ_i and Δ_i^2. If these exceed the practical limits, then average the previous point with the next point and set the value of this point to that average.

This filter is very empirical and the careful eye will observe some problems with it. For example, what happens if you get two glitch readings in a row? Clearly the above algorithm assumes that the next value after a glitch is a sensible one. It was found empirically that two glitches in a row rarely occurred. Furthermore, even if it did occur, the glitches would still be smoothed over, but to a lesser extent[5].

In fact, this can be thought of as nothing more than a specialised threshold filter.

Other filters were considered, such as Kalman filters, moving average filters and so on. However, these were not implemented for a number of reasons:

- Their exact meaning in a three-dimensional space is not always clear. For example, is the filter applied to the three dimensions separately, or is there some

[5]In fact it can be shown that in the limit with continuous glitching the filter behaves as $z_i = 0.5z_{i-1} + 0.5z_{i+1}$ (in Z transform notation) which is an unusual off-line linear filter.

way to map these methods more effectively to deal with all three variables at one time?

- Most of these filters are designed to smooth out random noise (which is usually assumed to be normally distributed). While there was certainly random noise to a certain extent, most of the feature extraction techniques would prove robust against such random noise. However, glitches were another story altogether, they are not distributed in the same way that the smoothing filters expect. The features used were found to be more sensitive to glitches than noise. For example, a glitch could easily throw off a bounding box, since the box would grow to accommodate the glitch point. Similarly if some distance or energy measurements were to be used, a glitch (if it was to happen) would represent a giant burst of energy relative to the rest of the motion.
- They were found to be computationally complex, while the filter proposed above is computationally simple. Furthermore, as shall be seen, a number of the synthesised features use the values calculated above anyway. If the filter was applied at the same time as the features were being extracted there would be no significant penalty.

This was implemented, as was all the feature selections in simple perl scripts.

No major sort of fitering of outlier data samples was taken, although some of the data that was incorrectly collected (such as one example where the user had taken a drink and forgotten to press "B" to stop data capture, resulting in a file with over 600 frames in it)[6]. This occurred four time in the course of sampling.

We will now go on to discuss and assess the features that we extract.

[6]The official reason for this is that it is only fair not to remove outlier points, and part of the testing of any system is to see how well it copes with noisy data. The non-official, pragmatic reason, is that 6 650 signs were collected and going through them, analysing them and deciding which ones were true outliers can take just a *little* time, a resource which an undergraduate thesis student has plenty of :-).

5.3 Feature extraction and validation

5.3.1 Validation methodology

To check the validity of a method, they were tested on the data collected.

In total five datasets were collected. These are named after the person who produced them. Each was collected on a different day, and for practical reasons, not the same number of samples were obtained from each signer.

The datasets are:

adam 8 samples per sign[7].

andrew 8 samples per sign.

john 18 samples per sign.

stephen 16 samples per sign.

waleed 20 samples per sign.

In total 6 650 signs were collected.

It was expected that there would be significant variations between the efficacy of the features on the various signers. This is because just as accents exist in spoken languages, each signer has his own gestural "accent". Some signers are conservative in their movements, some are energetic. Any system will prove to be better able to cope with some signers than with others, just as handwriting recognition systems are better able to work with some people's handwriting than with others.

Each of the features was tested with the learning algorithms already discussed.

In each of these cases, five-fold cross validation was used. Five-fold cross-validation means that the sample set is divided into fifths. One fifth is used as a test set and the

[7] This dataset was the first dataset taken. After the dataset was taken, the results appeared too consistent. After some thought, it was realised that this was because the words were appearing in the same order each time, and thus the "fatigue" effects were equivalent for a given signs. In subsequent samplings, the sign order was shuffled randomly to ensure that the fatigue effects were not causing unnatural consistency in the data.

learner is trained on the other four fifths[8]. This is repeated five times with a different fifth used for testing each time. The average error rate is then taken.

5.3.2 Distance, energy and time

Principle

One simple application of the data calculated above is to sum the Δ_i above. This would give us a measure of the total distance covered by the sign. This makes sense as a feature, since clearly some signs cover greater distances than others. For example the sign for all, a sign made by moving the palm face down in a large circle in front of the body, covers more distance than a sign like you, which is a short pointing gesture.

Thus we define the feature *distance* as:

$$distance = \sum_{i=1}^{n} \Delta_i$$

where n is the number of frames in the sample[9].

Similarly some signs turn out to be more energetic than others, even though the distance covered may be similar. For example, something like all and go have a similar distance covered. However, in go, it's a quick flick of the wrist, and a momentarily faster motion than all – thus it has more energy.

We cannot get an exact measure of the energy required to make a sign, because there are so many variables involved. Thus we make an approximation, based several assumptions:

- The packets come at equal time intervals. Thus the time between two consecutive frames is constant. This proved to be an empirically reasonable assumption.

[8]In fact there are several forms of cross-validation – for example there is n-fold cross validation, where the data is divided into n near-equally sized sets. The system is trained on $n-1$ of the sets and tested on the reamining set. The test is then repeated n times, each time with a different test set. At the extreme end there is leave-one-out cross validation. In this case, we train on all of the training set *except one* and then test the accuracy on the "left-out" sample. This is repeated with each sample in turn left out once. In practice, this is only computationally efficient with IBL1.

[9]Note that we ignore the first sample, since Δ_0 is clearly not defined, since x_{-1}, y_{-1}, z_{-1} are not defined.

- Most of the energy comes from the positional motion. This means that quick finger flicks, and wrist rotation are not included in the energy feature we calculate, although in some signs they can dominate, such as in maybe and flashlight.
- That the "effective mass" of the hand is some constant and the energy required to move the hand in all directions is approximately equal.

Using these assumptions, we can derive a measure of the energy. The textbook definition of energy is:

$$W = \int_0^T F ds$$

and a discrete approximation of this equation is given by:

$$W = \sum_{i=0}^{n} F \Delta s_i$$

Since $F = ma$ and m is a constant, and converting to the naming we have used then we can say:

$$W \propto \sum_{i=1}^{n} a \Delta_i$$

But $a = \frac{\Delta v}{\Delta t}$ and we have assumed that Δt is a constant. Also Δv is synonymous with Δ_i^2 then we have that[10]:

$$energy = \sum_{i=2}^{n} \Delta_i^2 \Delta_i$$

This *energy* features, while approximate, should still prove to be useful.

Finally, one feature that is obvious is the number of frames that were taken to make the sign. This is perhaps a very weak attribute, since it is dependant on the way a person started the sign, and observation of the samplers while they were doing a sign indicated wide intra-sign variation. However, it is sometimes a useful attribute for gross differentiation between signs.

[10]Note: We have ignored the first two samples, because, as previously discussed, Δ_0^2 and Δ_1^2 do not exist.

Results

The results are shown in table 5.1.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	90.6	90.6	90.5	92.1
andrew	96.5	96.4	96.4	94.1
john	94.0	94.3	94.3	92.1
stephen	93.3	93.3	92.6	93.3
waleed	91.4	91.4	92.0	92.3

Table 5.1: Results of learning algorithms with distance, energy and time features.

Discussion

The results indicate that these three factors are reasonably useful attributes, but not resoundingly so. They indicate that in about 7 per cent of cases, the sign can be differentiated on the basis of these three factors alone.

It must be remembered that on average, if a random guess was taken the error rate would be just under 99 per cent (since there are 95 signs – choosing one at random there is a probability of $\frac{1}{95}$ that you would be right).

Thus using the above attributes alone is seven times better than guessing.

At this stage, no statistically significant conclusion about the relative merits of the learners can be made.

5.3.3 Bounding boxes

Principle

Some signs are bigger than others and that they differ in their location around the body. So an obvious thing to think of as a feature is the bounding box of a sign.

This can be seen in figure 5.1. As can be seen, a total of six features are required to characterise the bounding box: $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$.

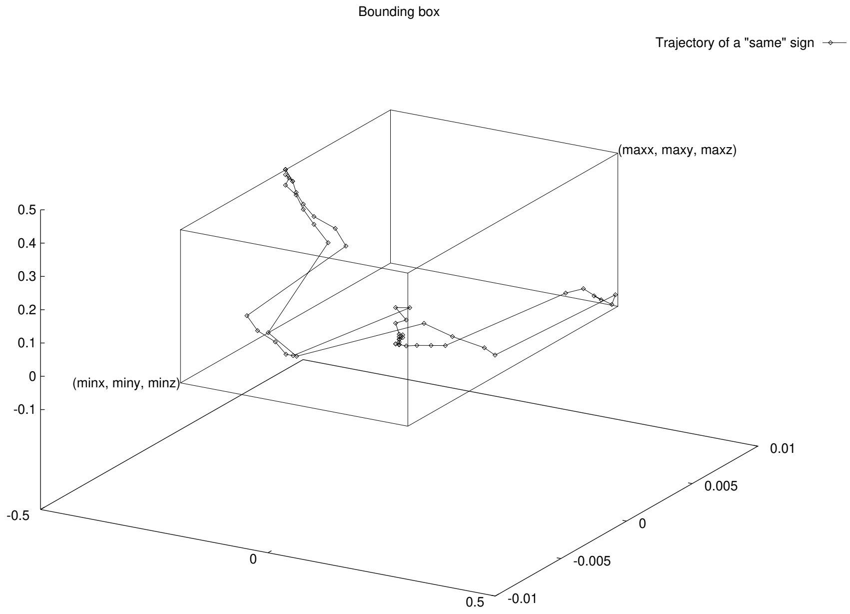


Figure 5.1: The bounding box for the sign same.

In the above, we consider the sign same. The whole sign fits in the box represented by $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$.

The calculations for a bounding box are relatively simple. Again, given a sequence (x_i, y_i, z_i), it is clear that the bounding box is:

$$\begin{aligned}
x_{min} &= \min(x_0, \ldots, x_n) \\
y_{min} &= \min(y_0, \ldots, y_n) \\
z_{min} &= \min(z_0, \ldots, z_n) \\
x_{max} &= \max(x_0, \ldots, x_n) \\
y_{max} &= \max(y_0, \ldots, y_n) \\
z_{max} &= \max(z_0, \ldots, z_n)
\end{aligned}$$

Results

The results are shown in Table 5.2.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	59.0	62.8	62.8	67.6
andrew	68.9	73.9	73.9	73.4
john	64.6	67.1	67.0	65.3
stephen	67.8	70.5	70.5	71.4
waleed	72.5	75.1	75.1	75.2

Table 5.2: Results of learning algorithms with bounding boxes

Discussion

The bounding box of a sign appears to be an effective feature. According to these results, 25 to 35 per cent accuracy can be obtained solely by considering the bounding box.

This is a little surprising, to say the least. This essentially says that by looking only at two points in space we can identify the signs in about three out of every ten cases.

At this point, no particular pattern is obvious about the relative quality of various learners, except that IBL2 and IBL3 do not appear to be doing very well. Also, the results for `adam` are quite good, but they should be taken carefully, since this is the data-set for which no fatigue adjustment technique was employed.

5.3.4 Positional histograms

Principle

Anyone acquainted with pattern recognition in any field will have come across histograms. Surprisingly however, their application to gesture recognition is, to the best of my knowledge, novel.

A histogram can be thought of as a discretised probability density function. Basically, you segment the range of possible values into subranges, and then count the number of instances of each subrange.

It is exactly this that we do with the signs and on a number of aspects of a sign. One obvious histogram to take is on the x, y and z positions of the hand.

You have to be a bit clever with how we do this, however, to get useful information. First, if signs are slightly more exaggerated, either in time or space, then we want it to remain reasonably invariant.

Thus we calculate the histograms in the following way: Let d be the number of divisions we wish to divide the ranges into, and let h_i with $0 \leq i < d$ be the "columns" of the histogram. Assume we are doing it for the x-position only. Then:

$$h_i = \sum_{j=0}^{n} \frac{1}{n} r_i(x_j), \ \forall i, 0 \leq i < d$$

where

$$r_i(x) = \begin{cases} 1, \text{ if } \frac{i(x_{max}-x_{min})}{d} \leq x < \frac{(i+1)(x_{max}-x_{min})}{d} \\ 0, \text{ otherwise} \end{cases}$$

Effectively, this "normalises" by two things:

- The length of the sequence. Since the sum term above includes a $\frac{1}{n}$ term, and every x_j must fall into exactly one h_i column, the net effect is that $\sum_{i=0}^{d-1} h_i = 1$.
- The bounding box of the sign. The column divisions are relative to the bounding box, and thus most of the h_i above will not be zero. This is desirable, since it essentially removes the issue of size of a sign, and low resolution on small signs, with lots of empty columns. The alternative would be to have absolute locations which would be nowhere near as closely correlated with the information in the sign itself.

To clarify what a histogram means, a 5-division histogram for the 'same' (that we saw in figure 5.1) sign shown in figure 5.2.

As you can see, there is a strong presence in the middle – this isn't a surprise, since this is where the sign starts. Also, it is weak to the right, since this seems to be a few points resulting from "swinging back" too far on the return. However, on the left hand side, there is another strong peak, which is caused by the slowing down of the motion that occurs in the sign[11].

[11]In this case the range is symmetrical about the x-axis, but this is just a coincidence in this case.

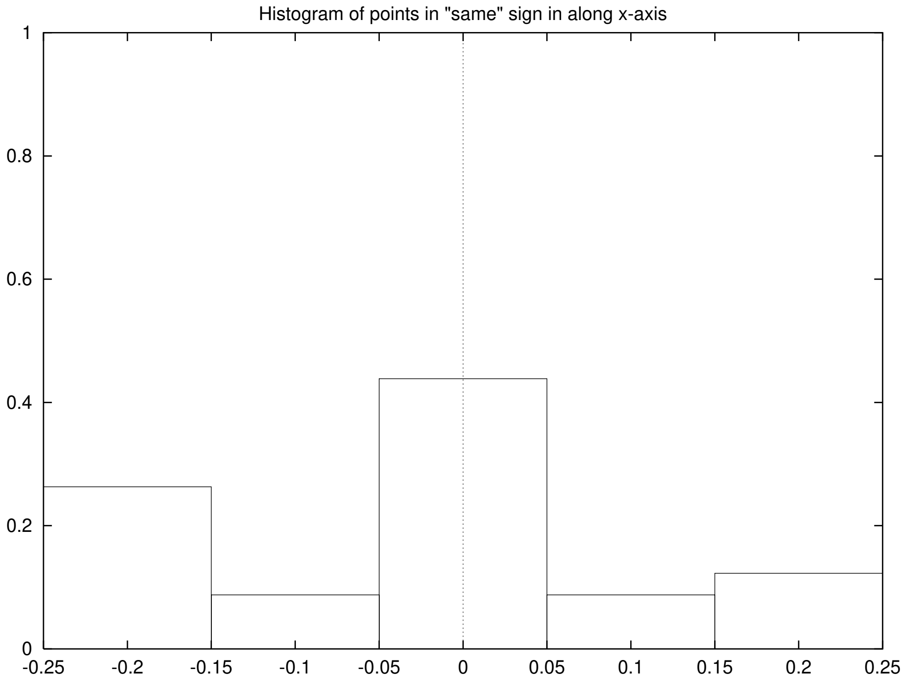


Figure 5.2: An example of a histogram – in this case of motion in the x-dimension.

In this set of features, we consider the histograms of the x-position, the y-position and the z-position. Histograms of other data will follow.

There is one question which cannot be answered by theory alone. This question is: What is the optimum number of divisions d that will result in the smallest error? The answer depends on a number of factors, such as the accuracy of the equipment, and the nature of the signs themselves. Too few divisions, and there will be insufficient information to distinguish signs; too many divisions and noise and variation in the data will cause crossover between adjacent columns in the histogram, resulting in erroneous classification.

Results

To investigate what the optimum number of divisions for the histogram were, it was decided to run a test with some of the sample data and using two learners, C4.5 and IBL1[12].

[12] It was thought there would be little difference in the behaviour of IBL1, IBL2 and IBL3.

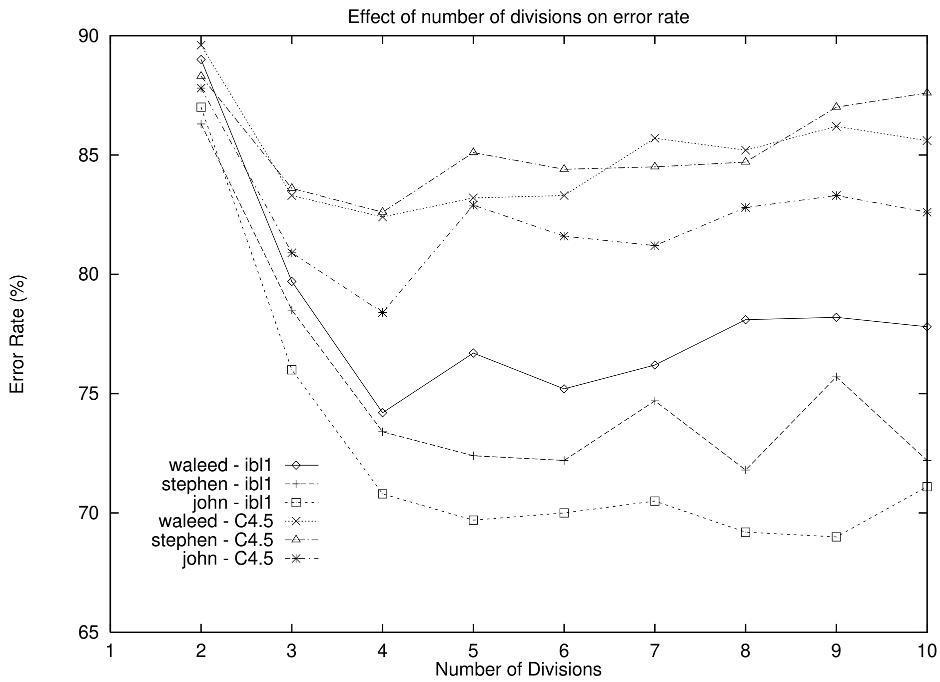


Figure 5.3: Graph of the relationship between error and the number of divisions used in making the histograms.

The results of the tests on accuracy using the 6-divisions agreed upon above are shown in table 5.3.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	75.1	76.7	76.7	87.7
andrew	78.1	79.7	79.7	88.9
john	70.0	76.9	76.9	81.6
stephen	72.2	77.9	78.0	84.4
waleed	75.2	78.4	78.4	83.3

Table 5.3: Results of learning algorithms on positional histograms.

Discussion

The results of the histogram size tests are somewhat surprising, and difficult to decipher. There seems to be a general downward pattern up to about 4, but it is not consistently decreasing by any means after 4 divisions. A reasonable "middle ground" appears to be around the 6-division histograms. While not the best, the behaviour is predictable up to 6 and is in all cases except one, no worse by greater than 2 per cent than the minimum. There is of course some noise in the data, and this choice of 6 is reasonably arbitrary.

Another pattern is arising. While with up to three divisions, C4.5 is able to keep up, IBL1 outperforms it significantly beyond this level, by margins of around ten per cent. What may be happening is that the IBLs are better than C4.5 at handling the high dimensionality of the feature space. Since C4.5 can only consider the value of one variable at each node in the decision tree, this forces it to segment the space in limited ways relative to the way instance-based learning can.

5.3.5 Rotation and finger position histograms

Principle

We can similarly apply histograms to the wrist roll and finger position. This is still an intuitively useful variable, since, for example, it tells us how much time the hand was at a particular roll level. Clearly signs such as give with the palms up, will have rotational histograms that significantly differ from those that have the palms down, such as take. Similarly, I, with a pointing gesture to oneself, will have a different finger histogram (hopefully) to mine which is a similar gesture to I except it is with a clenched fist.

They differ from the x, y and z position histograms in a number of ways, however:

- The finger and rotation information is not as high resolution as the x, y and z information (for example, x y and z position each have 256 possible values; rotation has 12 and fingers 4 each). This means that histograms can be built on each of the raw data values – rather than having to divide the range of values up.

- The idea of normalising by maximum and minimum values of the variables does not apply. For example, the rotation value indicates a particular angle between 0° and 360° (at 30° intervals). We want this in absolute terms, not relative terms. Similarly, because a value of 0 means a finger is fully extended and 3 means fully flexed, we want these pieces of information.

There are some special precautions we need to take, as well as some redundancy we can make use of.

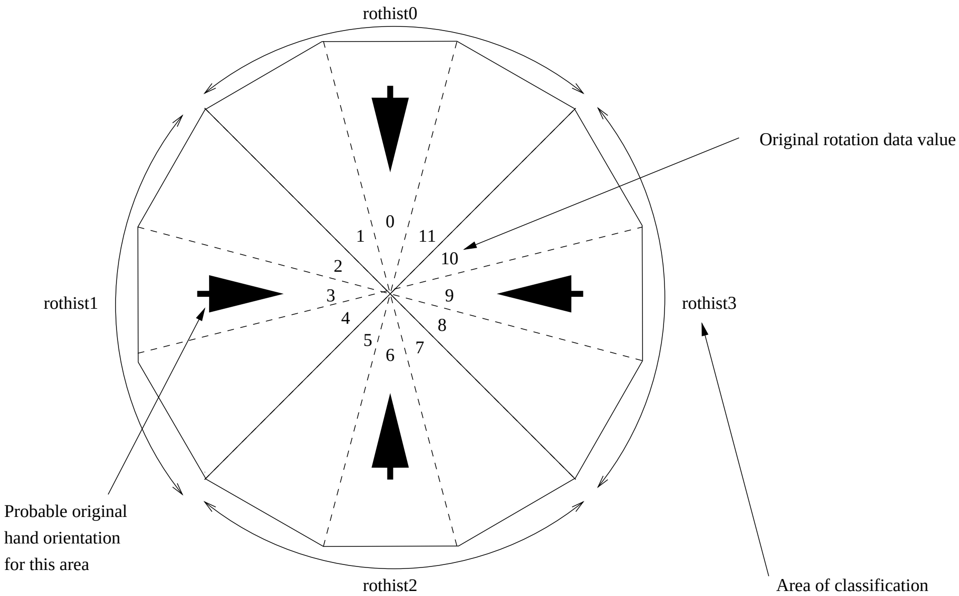


Figure 5.4: The possible rotation values coming from the glove and their encoding and classification into histograms.

In the rotation histogram, there are 12 possible values, as shown in figure 5.4. Although we could directly build histograms on this data, this would make the system extremely sensitive to noise, since practical observation of the glove indicates that the "roll" value is particularly sensitive to noise. Furthermore, it is unlikely that a sign will intentionally be made at an angle not parallel to one of the axes. Few, if any, signs, for example, require the hand to be inclined along a 45° angle.

Thus what we can do is segment the region into areas as shown in figure 5.4 – into 4 regions, centred on the positive and negative x and y axes. This gives us a much more noise-resistant classification. For values of 11, 0 and 1 we classify this as "palm down". Similarly values between 5 and 7 would indicate "palm up". Similarly for "palm left" and "palm right". Histograms are then built on the proportion of time spent with palm up, palm down, palm left and palm right.

For the fingers, we can make a similar suggestion (in fact Sturman [Stu92] did) – that 90 per cent of the time, the fingers are either fully opened or closed. While this method works in cases with high-accuracy gloves, in this case, there is no margin for error with only two bits of information available. Also, since there are only 4 possible values, it is practical to have a histogram for each finger position.

Results

The table of results obtained using these features is shown below in table 5.4.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	39.0	41.4	41.4	60.5
andrew	66.5	67.5	67.5	77.5
john	59.2	60.4	60.5	72.6
stephen	56.4	60.1	60.1	71.8
waleed	56.7	58.1	58.2	73.6

Table 5.4: Results of learning algorithms on rotation and finger histograms.

Discussion

Again, it is clear that these features are very useful, even by themselves and without x, y, and z positional histograms. This is good, since the two sources of information are distinct. It also indicates the potential sources of information finger position and wrist roll represent, and show that they can significantly help in deciding signs. In fact, these attributes are significantly better than the x, y and z histograms.

A consistent pattern is appearing here, however. As previously noted C4.5 is starting to “lag behind” the IBL’s.

5.3.6 Synthesised histograms

Principle

Histograms of synthesised attributes can be calculated as well as the raw data attributes. Two obvious ones will be to do the histograms on the Δ_i and Δ_i^2 features calculated above. This is basically a “poor man’s”[13] energy spectrum, since it gives

[13] A poor man in this situation is someone who does not have much CPU time or power available to him.

an estimate of the distribution of the acceleration and velocity of the sign.

To understand how this gives us an energy spectrum, consider a sign with smooth, slow, continuous motion. The distance covered will be uniform, so there will be a "peak" in the Δ_i with a low reading, and have a very low Δ_i^2 reading, since acceleration is constant. A smooth, fast sign will have a peak in the Δ_i at a higher reading level, but a similar Δ_i^2 histogram. Consider a movement that has shaking or waving in it; this will have a distinctive Δ_i^2 histogram, with peaks that will indicates periods of high acceleration. The Δ_i would show very little movement in its histogram, because it is about the same point.

Results

The table of results obtained using these features is shown below in table 5.5.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	95.3	95.3	95.3	94.6
andrew	98.6	98.6	98.6	95.6
john	96.3	96.6	96.6	95.4
stephen	95.5	95.6	95.6	96.4
waleed	96.8	97.0	97.0	94.7

Table 5.5: Results of learning algorithms on histograms of Δ_i and Δ_i^2 histograms.

Discussion

These results are not very promising. They show that these attributes are not to be useful (at least in isolation).

The failure could be for many reasons, such as the fact that they are normalised to a maximum. However, having it not normalised would be just as bad, since then our resolution would not be in a useful range. The other possible cause is that at a frame-to-frame level there is little difference in magnitude changes; and thus the aliasing can occur. Another probable cause is the random noise the glove is generating. It is possible that the major component of the distance covered and the energy generated by a sign are not from the underlying motion, but from noise generated by the glove.

5.3.7 Octant count

Principle

A three-dimensional space can be divided into octants, where each of x, y and z can be positive or negative. Now, if we take the $(\Delta x, \Delta y, \Delta z)$ we calculated above, we can easily classify each of the $(\Delta x, \Delta y, \Delta z)$ as belonging to one of the eight possible octants. By considering which of these octants dominate, we can get an effective measure of the general flow of a sign.

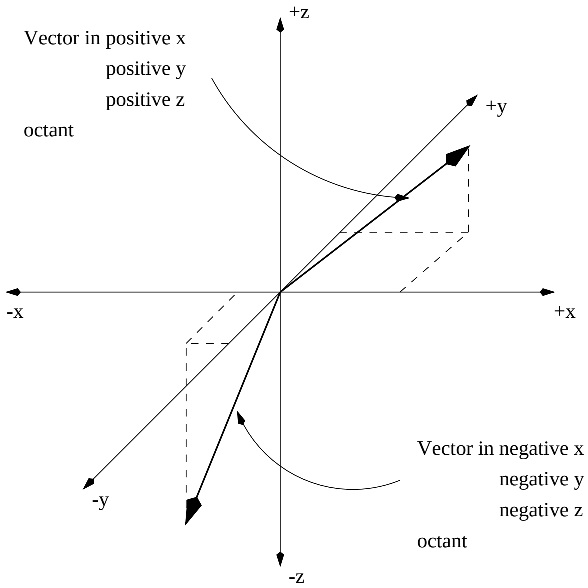


Figure 5.5: Two examples of vectors and the octants they fall into – one in octant $(+x, +y, +z)$ and the other in octant $(-x, -y, -z)$.

In figure 5.5 we can see two examples of such vectors.

In addition, we make sure that the length of the sign has no impact, by ensuring the sum of all "octant counts" is normalised to 1.

Results

The results of the above experiments are shown in table 5.6.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	87.3	87.1	87.1	91.0
andrew	92.7	93.1	93.2	94.4
john	86.6	86.7	86.7	89.2
stephen	87.9	88.7	88.7	91.8
waleed	89.1	90.1	90.1	92.0

Table 5.6: Results of learning algorithms on octant counts.

Conclusion

These features appear to be useful, but not incredibly so.

This measure was contrived to begin with, and its use was a result of mathematical simplicity rather than a strong empirical foundation. So it is not surprising to see it less successful than other attributes.

5.3.8 Simple time division

Principle

Another obvious way to find useful data in the samples might be to try a pattern-matching approach at a very low level; by simply dividing the sign into a fixed number of equally sized segments, find the average value of attributes such as the x, y and z positions, rotation value and finger positions over all the frames in that segment.

Mathematically, we can express this in the following way. For the moment, consider only the average value of the x position. Let s_i be the x position in the ith segment, and let d be the number of segments we want to divide the sign into. The the value of s_i is given by:

$$s_i = \sum_{j=l(i)}^{u(i)} \frac{x_j}{u(i) - l(i) + 1}, \ \forall i, 0 \leq i < d$$

where n is the number of frames and

$$l(i) = \lfloor i\frac{n}{d} \rfloor + 1$$
$$u(i) = \lfloor (i+1)\frac{n}{d} \rfloor$$

This segments a set of frames into d sets, and we take the average of x position over the range of values we accepted. Graphically, it is much easier to see what is happening.

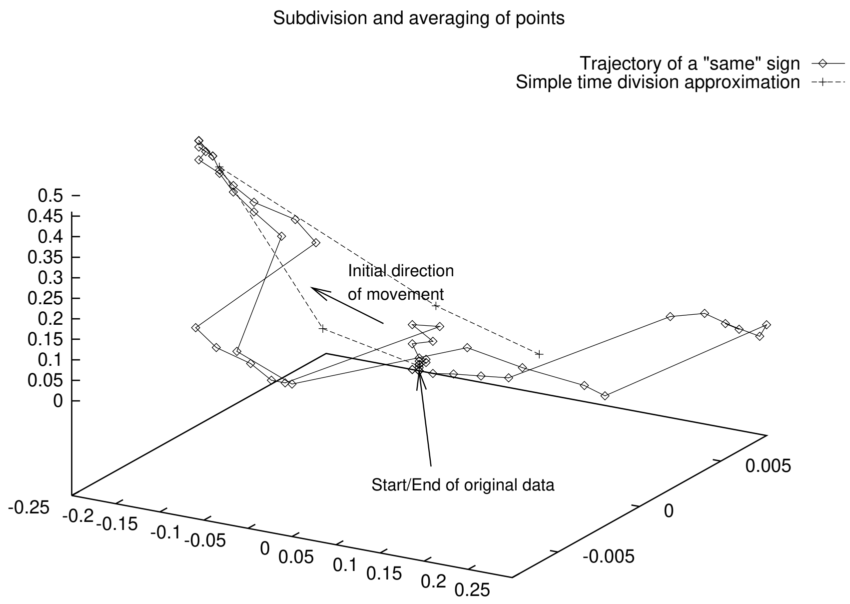


Figure 5.6: Original data and time-division approximation. Note the clarity of the time-division approximation – it also manages to reduce the effects of noise.

.

Effectively, what this does is reduce the data on the motion into a form more easily amenable to learning. It reduces the data to fixed size, and at the same time averages the data, resulting in smoothing.

Thus you end up with a rough approximation of the "shape" of the sign. In figure 5.6 we can see an example of this being applied to the x, y and z positions. In this case, we are using a 5-segment division. Also note how much "cleaner" the signal is – how much more characteristic of the sign it is, and that it is relatively free of noise. We can of course, do similar things for the rest of the variables.

Note also that because a fixed number of intervals are taken, it is relatively invariant to how long it takes to make a sign, provided that the relative timing is similar.

This also remedies one of the problems discovered with the histogram technique, when observing the type of errors made using histograms. Because the information in the histogram was not connected at all with time, signs that had the same "sub-gestures" but in a different order would have similar histograms. Most significantly, there are a number of signs that are the "reverse" of each other which were frequently confused. For example, want and don't want – want the flat palm pointing to the chest is moved down the chest, and in don't want it is moved up the chest. These two signs would have similar histograms. By using some information gathered in time, it becomes easy to distinguish these two.

Time division was applied to the x, y and z position as well as wrist roll and finger flexure.

The same issue arises as did with the histograms as far as optimisation is concerned: what is the most generally optimal size for the number of segments to make?

Again, this cannot be determined theoretically easily and is a balance of the factors of sufficient characterisation of the sign while balancing noise and resolution.

Results

We began by considering how the number of divisions affected the results. The results are shown in figure 5.7.

This shows some recurring patterns. Again, IBL1 is very significantly better than C4.5.

We notice that the minimum occurs around 5 or 6 segments. We will use five, simply because it reduces the feature count by 8 features, and appears to give similar error rates.

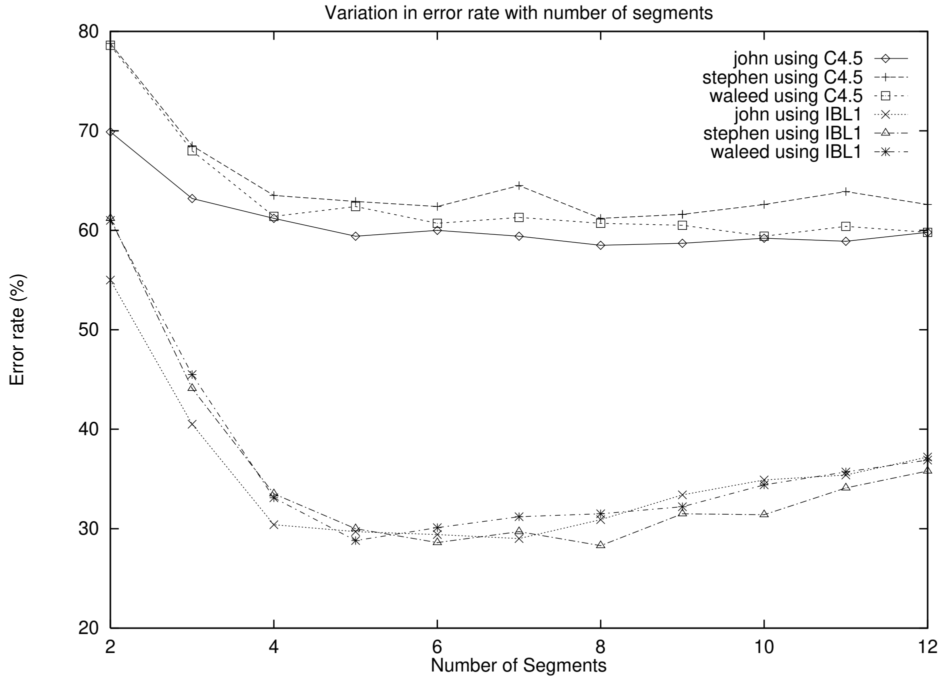


Figure 5.7: The variation in the error rate that occurs with segmentation.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
adam	20.7	25.1	25.2	58.6
andrew	52.7	55.7	55.6	78.2
john	29.7	34.3	34.3	59.4
stephen	30.0	34.2	34.3	62.9
waleed	28.8	33.8	33.9	62.4

Table 5.7: Results of learning algorithms on simple time division features.

Discussion

These results are very good. Basically, by using the time division technique, it is possible to obtain in excess of 70 per cent accuracy in cases with large numbers of training sets. This appears by far to be the most accurate method of assessing what the sign is.

5.3.9 Using synthesised features with time-division

Principle

We need not force ourselves to stick to the data coming directly off the glove to apply time division to. It might be for instance useful to look at the distance covered in each segment.

Thus we considered two such synthesised features with each segment made above. These were the distance covered in the segment and the energy expended in the segment.

Again, we can use equations similar to those already derived previously. The distance covered is the sum of the of the inter-frame distances. Similarly, the energy is the sum of the inter-frame energies within that time division.

For consistency, we used the same number of time divisions as we did for the simple time division techniques.

Results

The results of this are shown in table 5.8.

Discussion

Clearly, this set of attributes was not effective. Their accuracy is extremely low relative to some of the other attributes that we have seen.

Dataset	IBL1 Error (%)	IBL 2 Error(%)	IBL 3 Error(%)	C4.5 Error(%)
adam	94.0	94.0	94.0	95.7
andrew	97.1	97.5	97.5	96.4
john	94.6	94.8	94.8	93.6
stephen	94.3	94.6	94.6	94.9
waleed	96.8	93.2	93.2	93.8

Table 5.8: Results of learning algorithms synthesised time division features.

It seems to have been plagued by similar problems to those that resulted from taking histograms of distance and energy.

5.3.10 Other approaches considered but untested

There were of course, several other approaches that were considered. But these were eliminated for a number of reasons:

- A hierarchical approach to recognition. In this situation, the idea would have been to recognise the handshape using C4.5 or IBL, and then use the handshape decided upon as input into a "second layer" learner, again either C4.5 or IBL. However, it is unlikely this would have have worked for a number of reasons. As you can see in appendix A, the results of handshape recognition using the PowerGlove are not good at all. On average, the error rate for handshape recognition with the PowerGlove was 70 per cent. This would introduce significant noise into subsequent layers, which would probably have resulted in an increased error rate.
- A "spatiotemporal encoding" approach. In this sort of approach, the movement is broken into a series of component movements along a limited set of axes (eg up, down, left, right, forward, backward). These are output as strings and then compared at a string level. This was not used because of complexity and its vulnerability to noise.

We now consider ways to assemble these attributes to form a more accurate system.

Chapter 6

Synthesis

Ok - you have all the pieces ... what happens when you put them together?

6.1 Introduction

In chapter 5 a set of features were illustrated that appeared to, individually, be good discriminants between signs. None of them, however, was sufficiently accurate to use alone. In such a situation, a logical continuation is to combine such feature sets together.

Of course, we cannot expect that if we have one feature set which is 50 per cent accurate, and another which is also 50 per cent accurate, that when we combine them, we will have a 100 per cent accurate feature set; because there is an intersection in the input data of samples that would be correctly classified by either – in other words, there are "easy" samples that would be classified correctly by either feature set. In fact, for the above to occur, one set of features would have to get every instance that the other one got wrong right and vice versa. Also it is possible that adding a non-useful feature set to a useful feature set may result in a higher error rate than the useful feature set alone.

Another thing that needs to be investigated is how GRASP behaves as certain aspects change. For example, we may wish to consider some of the following:

- What the relationship is between the number of samples for each sign and the accuracy we obtain? This affects how long it takes to train GRASP on a new user.
- What the relationship is between the number of signs GRASP tries to learn and the accuracy it obtains? This would be an issue in expanding the lexicon of GRASP and other sign language recognition systems.
- How does the learning time scale and how well is it suited to incremental learning? This system would be used in an incremental environment, where it will be expected to learn new signs.
- Is it feasible for GRASP to understand the signs of people who it has not seen yet, or does it require separate training on each person? What is the disparity between its accuracy on seen and unseen signers ?

Finally, we consider ways of optimising the system to obtain higher accuracies.

In the previous sections, we saw that IBL1 was unlikely to have worst performance than IBL2 and IBL3, and that IBL2 and IBL3 were consistently somewhere between 2 and 7 per cent worse in accuracy than IBL1. We will thus focus on IBL1 and C4.5 from now on.

6.2 Putting everything together

6.2.1 All features

One of the obvious ways to test how well the system works is to simply combine all the feature sets discussed above into one test.

In total, 119 features were used. The same techniques are employed for assessment – 5-fold cross validation.

We can see, using the above, that altogether, GRASP can have an accuracy up to about 80 per cent. In all the large data sets, the error rate is approximately 80 per cent. The results of the andrew data-set indicate that perhaps that if an insufficient number of samples is provided, then the accuracy goes down rapidly. As previously

Dataset	IBL1 Error (%)	C4.5 Error(%)
adam	15.0	50.9
andrew	44.8	66.3
john	20.3	45.9
stephen	21.3	52.2
waleed	19.4	50.7

Table 6.1: Results of learning algorithms on all attributes combined

mentioned, the adam dataset should not be taken as representative, because when the data was sampled, the words were not randomly sorted. This of course meant that the fatigue factor was removed. This also perhaps indicates the relative sensitivity of GRASP to intra-signer consistency. From the above, it would appear that GRASP is sensitive to variations in the way a signer makes a sign.

The effects of the number of samples will be explored in section 6.3.1.

6.2.2 The most effective features

As a basis for comparison, what were judged to be the most effective attributes from the investigation in the previous chapter were used. It is possible that at high levels of accuracy, when the more effective attributes are working quite well, the less effective attributes not just being performance-neutral, but degrading performance, since they were creating "attribute noise".

These were thus removed. This meant that in particular the synthesised histograms and time features were excluded because of their poor performance.

This is a very simplified approach, and it may be that the elements feature set are not orthogonal – i.e. there is some overlap between the information they provide. If time permitted, a linear independence investigation could be undertaken to determine the relative accuracies of the attributes included here.

In fact, in general, we can optimise the feature set even more by modifying the similarity function of the IBL algorithm, or by creating addition weighted features for C4.5. Here we have chosen to include or not to include features. In general, you can in fact assign "weights" to attributes to give a better similarity function. Finding the optimal similarity function is not a trivial task, and searching algorithms have to be used.

The results of the tests on these smaller set of attributes are shown below:

Dataset	IBL1 Error (%)	C4.5 Error(%)
adam	12.6	51.4
andrew	41.5	66.1
john	19.4	45.4
stephen	18.4	52.8
waleed	17.0	48.5

Table 6.2: Results of learning algorithms on the most effective attributes combined

The above shows that in all of the IBL1 cases, the results are better than when the poorly discriminatory features are included, by somewhere between 1 and 5 per cent. The situation with C4.5 is more ambiguous, but at this point, C4.5 appears to be significantly worse than IBL1, by some 30 per cent. It does not appear to be a serious rival to IBL1.

This result suggests that further optimisations are possible and that using techniques such as forward selection, Gauss-Seidel hill-climbing, simulated annealing or any other searching techniques to obtain a more accurate weighting function, since some of the features are, as illustrated here, better discriminants than others and that we want to give the better discriminators a higher weighting.

6.3 Effects on accuracy

6.3.1 Number of samples per sign

Algorithmic learners usually get better with practice, that is, the more examples you give them, the better they get at discriminating between classes. Notice the "usually" – there are several phenomenon that can begin to occur if you give too much (or more particularly, too much *noisy*) data. One of the most common is over-fitting of the data. If too many samples are given, learners will develop very constrained concepts – too constrained – so that anything that does not match the concept fairly closely is not accepted as a member of that concept.

Various approaches exist for handling the problem that is adapted for each of the algorithms, for example with symbolic decision tree learners, the tree is "pruned" of nodes in the tree which do not perform well or do not generalise. In instance based learning, there are also ways of culling the instances, such as IBL3.

Still, it is logical that if we have more samples of a sign that the error rate will fall, because the more samples that any learning algorithm has, the better the concept description that it is able to develop.

To test the impact that changing the number of training instances has on the error rate, we tested each of the three large data sets with an increasing number of training instances for each sign – from 2 instances for each sign up to 14 with an increment of 2. The training examples used were selected at random. The remainder of the data-set was used for testing[1].

The results are shown in figure 6.1. As can be seen, there is a decrease in the error rate as the number of samples per sign increases. This effect, as we would expect, tapers off, since the first few signs learnt help a great deal, but subsequent signs only help a little more.

A logarithmic relationship was hypothesised. So the graph was re-plotted with a logarithmic axis (figure 6.2).

From the appearance of the above, there seems to be strong support for the hypothesis that a logarithmic relationship exists between the number of samples per sign and the error rate.

Also note that C4.5 does not appear to learn significantly faster than IBL. In figure 6.2, C4.5 does not appear to have a steeper gradient than IBL or vice versa. It indicates the number of samples per sign does not alter the relationship between the error rates of C4.5 and IBL even over a long period. It might have been, for example, that C4.5 would have been a better choice in the long term, because of the (negatively) steeper gradient. This hypothesis, however, is not supported by the data.

[1] This technique has certain statistically complicating factors. For example, it means that the error rate reported at low number of samples should be more reliable, except for the fact that the learner is probably very sensitive to which training examples were selected. At high number of examples, there is less sensitivity to selection of the training examples, but the testing examples are fewer and thus the reporting of error rate is very sensitive. Furthermore, we cannot apply cross-validation, since frequently the training set is smaller than the test set. Of course, inverse cross-validation might be possible, i.e. simply cycling through the possible training sets, rather than the test sets, but this makes comparison of test results unfair. In brief, however, expect the results to be a little noisy.

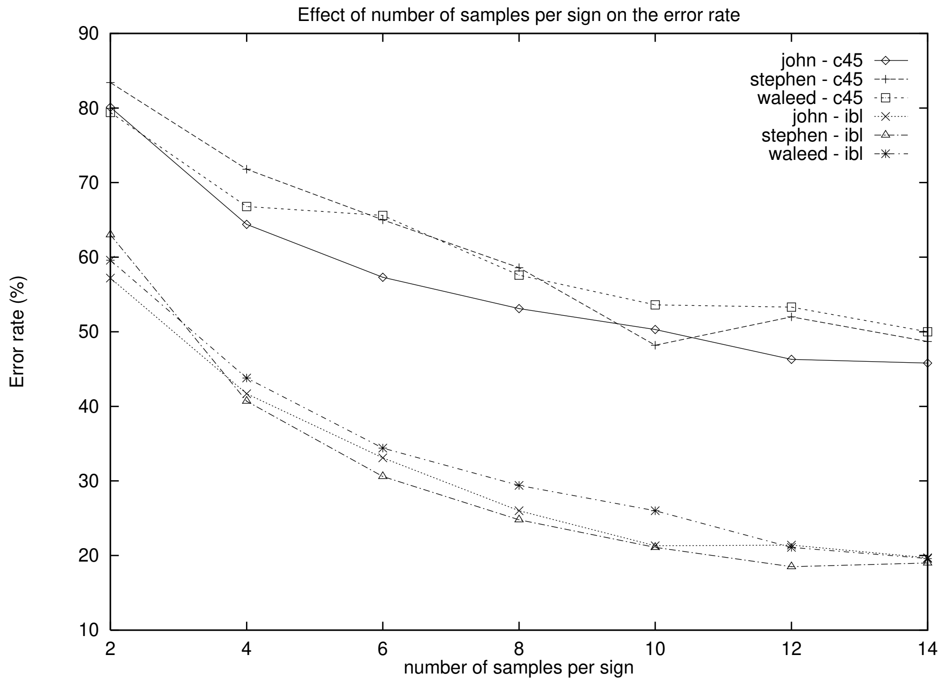


Figure 6.1: The effect that the number of samples has on the error rate.

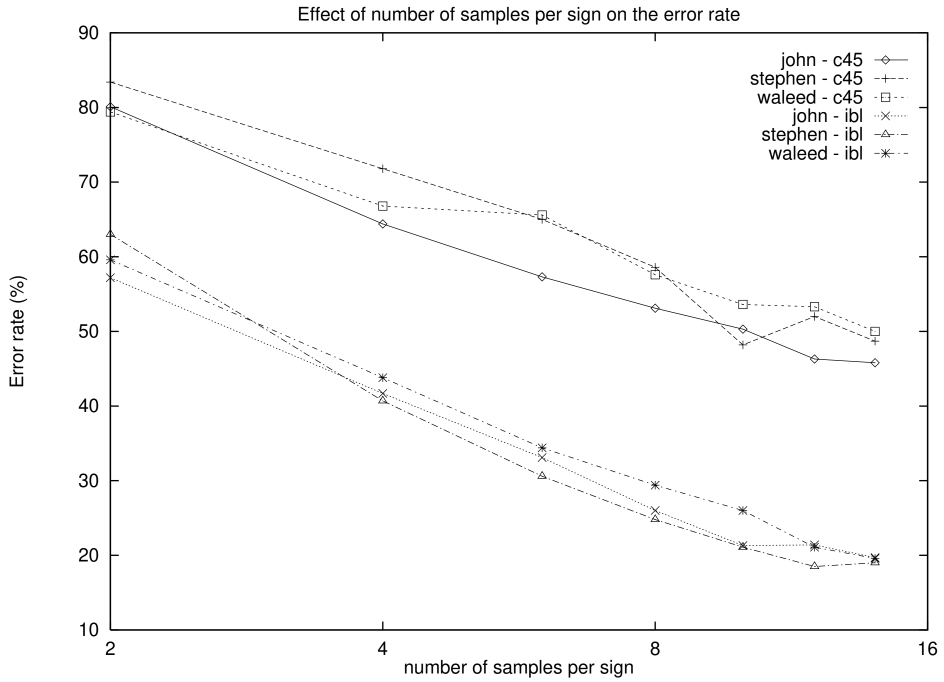


Figure 6.2: The effect of the number of samples on the error rate, this time with a logarithmic x-axis.

6.3.2 Number of signs

In order to assess the feasibility of extending the existing GRASP to a larger lexicon, it would be useful to know how the system "scales" with increasing numbers of signs. In section 6.2.2, it was shown that with 95 signs the error rate was 20 per cent. If we increase the number of signs to 200, will the error rate double to 40 per cent? If we went to 1000 signs, would we then get an error rate that was 100 per cent – would GRASP always be wrong?

It is clear that the error rate is likely to increase as the the number of signs increases. If we use the "random guess" comparison, then if you have 1000 signs, then you only have a 0.1 per cent chance of guessing the correct sign. A similar situation applies for learning algorithms. If there are more classes to distinguish between, then the algorithms have to be more capable of discriminating between signs. Essentially the feature space becomes filled with more instances, and thus developing concept boundaries that are tight enough to be accurate is more difficult.

We are still interested in the extent in the change of error rate, however. For example, one of the best possible results would be that we find that the error increases relative to the logarithm of the number of signs. This means, for example, that if we go from 100 signs to 200 signs, then the increase in the error is the same as going from 50 signs to 100 signs. In effect, this would mean large lexicon systems are feasible.

Thus tests were run on each of the three large datasets to observe their behaviour as the number of signs increased. 5-fold cross-validation was used, and we used the most effective attributes as discussed in section 6.2.2. The subsets of signs were selected at random, but were identical for each of the three datasets, to allow a fair comparison (since it is possible to choose some sets of signs that are easier to tell apart than others).

The results are shown in figure 6.3.

As we would expect, the error increases as the number of signs learnt does. However, there is a noticeable tapering off as the number of signs increases. The data is too noisy for IBL1 to really make any more substantial claims about its behaviour, although it appears that the error rate is increasing less than linearly.

Out of curiosity, the x-axis was logarithmically plotted, in the hope we we would see

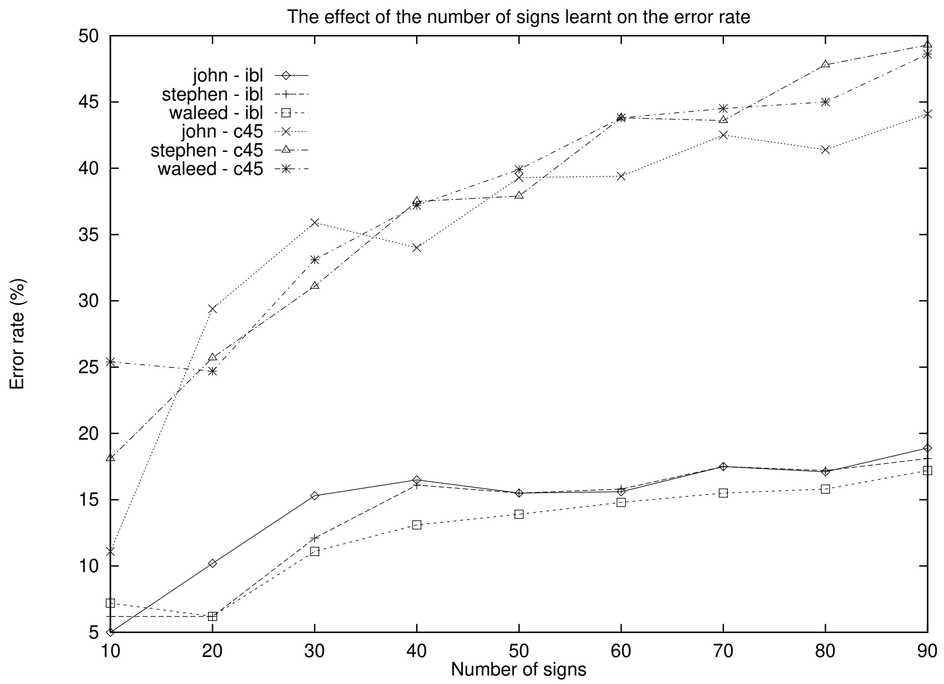


Figure 6.3: The effect of an increasing number of signs on the error rate.

a more interesting pattern in the data.

This is shown in figure 6.4.

It appears that there is a strong logarithmic relationship for C4.5, but the results are more ambiguous for IBL. In either case, however, it is clear from this graph that the rate of growth of the error rate is less than linear.

This bodes well for extension to larger systems. Because of this plateau effect, it is more likely that larger lexicons will not lead to proportionately larger error rates.

More significantly however, is that the C4.5 error rate seems to be increasing faster than that of the IBL datasets.

In conclusion, it appears that both systems behave better than linearly in the number of signs. The results for IBL1 are a little ambiguous and noisy, but for C4.5 it is plausible to suggest that the error rate increases as the logarithm of the number of signs. As mentioned above, although this does not guarantee performance with larger collections of signs, it does indicate that there is potential for the expansion of GRASP

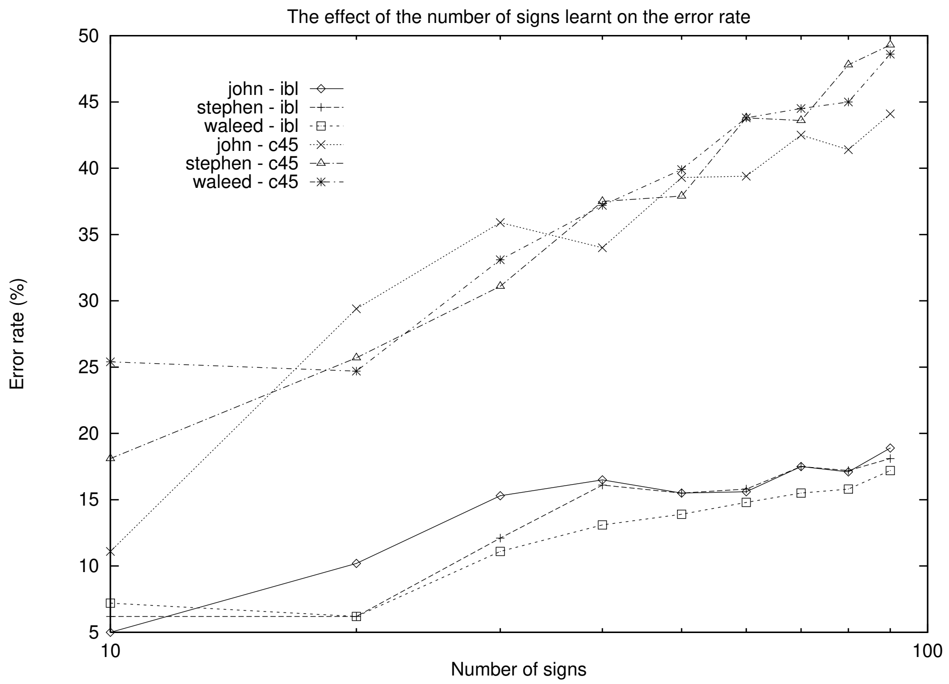


Figure 6.4: The effect of an increasing number of signs on the error rate using IBL1, this time with a logarithmic x-axis.

to a larger lexicon. Furthermore, it appears that IBL has an advantage over C4.5 in that the rate at which the error increases with increasing number of signs is lower. This suggests that IBL is more suitable for development of large lexicon systems.

6.3.3 Inter-signer recognition

So far, tests have been limited to investigating how adept GRASP is at recognising signs from the same signer. How well does it cope with signs it has seen from signers it has not seen?

Obviously, it might be useful to allow such learning if the effects on accuracy were not dramatic, since it would greatly reduce learning time. If the examples given to the learning algorithm were typical, this might allow GRASP to be trained on a general user base and then customised to suit each individual. This would reduce training time and thus allow GRASP to perform better "out of the box".

It could also result in better performance on more than one individual by providing more instances. As we saw in the previous section, the error rate is closely related to the the number of samples for each sign. Thus by using signs from other people, it might be possible to get extra samples "for free", thus reducing the error rate. For example, if we had 4 people with 20 samples each, this would result in 80 samples per sign.

On the other hand, there are a number of phenomena that could work against such a system. With 80 samples per sign, it is likely that noise can influence the data more significantly. There are a number of solutions to this, such as using the more noise-resistant version of IBL1, IBL3. Furthermore, and more dangerously, there may be a problem with concept boundaries being different for different people. Thus adding extra samples doesn't help, it confuses the learning algorithm by "corrupting" the feature space with noise.

So a test was performed. GRASP was trained on four of the data sets and then tested on the fifth. This was repeated on each of the four datasets. The results are shown in table 6.3. The name corresponds to the name of the person whose samples were used as the test dataset.

Dataset	IBL1 Error (%)	C4.5 Error(%)
adam	85.8	88.6
andrew	87.4	90.8
john	89.1	89.3
stephen	84.5	88.5
waleed	86.2	86.9

Table 6.3: Results of learning algorithms with inter-signer learning.

As we can see, the results are not so good. On average GRASP would only get a sign in every ten correct, if it is used by people on whom it is not trained.

There are a number of possible reasons for this. To investigate further, a "confusion list" was generated. This is a list of what signs were incorrectly classified, and what they were incorrectly classified as. A perusal of the data set illustrated an interesting occurrence – the errors were not always random, but frequently a consistent error was made – i.e. a given sign was constantly misinterpreted as another.

This suggests that the concepts for different people are different and that the problem is not with the learning algorithms themselves, but with the fact that while a particular sign for one person may represent a particular collection of attribute values, it may not be the same set of values for another person. In fact a particular set of attribute values for one signer may map to a *different* sign for another signer. This is shown in figure 6.5. Furthermore, this effect is probably more dramatic in our case, since in our validation of attributes, we selected attributes that were optimal for intra-signer recognition and *not* inter-signer recognition. There may in fact be a set of attributes that work well for inter-signer learning that we have not come up with.

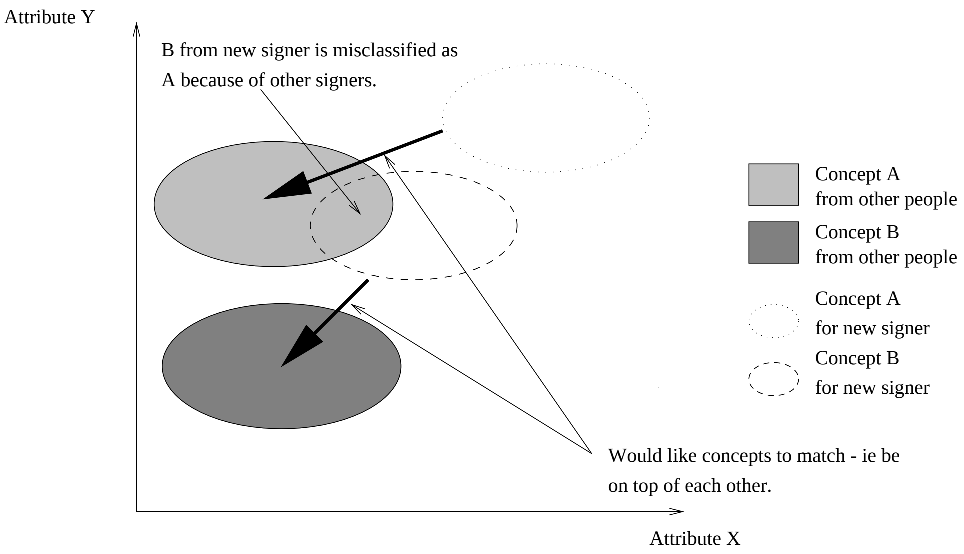


Figure 6.5: What can happen when GRASP is trained on one set of users and then tested on another.

In the above diagram, we see that based on training examples from other signer we can develop concepts A and B. Note that this is all in attribute space – and not in real-space, since if this happened with the concept boundary in real life, we would have difficulty adapting to unseen people's signing.

For a new signer, the concept boundaries in attribute space may not exactly match those that have been learnt, for two main reasons: aspects of that person's physical character that affect the way he/she signs.

Furthermore, "inter-signer concept overlap" can occur. In the diagram, there is a region where the concept learnt from the known signers overlaps with a *different* concept of the new signer – in this case concept B of the new signer is confused with concept A of the known signers.

From the fact we saw above that the errors it makes are consistent, it appears that we are indeed subject to the effects of inter-signer concept overlap.

To further investigate the following test was undertaken: GRASP was trained on all but one of the people. The dataset from that person was halved, with one half used for training and the other half for testing.

The results, with the corresponding values from the investigation from the number of samples is shown in table 6.4.

Dataset	IBL1 Error (%) with others	IBL 1 Error(%) without others	C4.5 Error (%) with others	C4.5 Error (%) without others
adam	28.2	24.2	72.1	58.7
andrew	48.2	45.5	78.4	71.3
john	24.0	23.7	65.0	51.6
stephen	28.7	24.8	71.3	58.6
waleed	31.8	26.0	69.7	53.6

Table 6.4: Results of learning algorithms with inter-signer learning, with partial training from that person.

The columns of the table show the error that occurs when tested on half of that person's dataset. In the first and third column, the datasets from other people are used for training in addition to the other half of the dataset from that person. In the second and fourth, only the other half of the dataset is used, without using other people's training sets.

From the above, we can see that to a limited and not too significant degree, such inter-signer concept confusion is indeed occurrring, since the error rate is higher when including other people's signs than when only using that person's signs. But the effect on the error rate is not too bad, ranging between 0.3 and 5 per cent. Thus it is feasible that we might start off the system trained on other people, and then with use, GRASP will increase its accuracy. Another approach that might be worth investigating would be the use of some selection criteria for the instances kept – such as keeping the twenty most recent versions of a sign; thus providing a balance between computation time required and accuracy.

What we really want, however, is for the new signer concepts to transform to the known signer concepts (as indicated by the arrows in figure 6.5, which show the direction we would like the transformation to move the new signers' concepts). Such a transformation of the attributes may exist and may work in making it easier to use the system for new signers. As mentioned above, the transformation would compensate for two factors: firstly, that each signer has different physical limitations such as reach, and secondly that each signer has an individual style.

Several attempts were made to address the problems. These included:

- Using calibration data from each of the signers but it was found to be of insufficient accuracy to be of practical use (for example, what use is calibrating finger position if you can only have four possible values?).
- Finding the mean and standard deviation of each of the attributes from a subset of the new signer and then normalising these to the set of known signers. While not pure unseen learning, it would have indicated that it might potentially be possible to sample some subset of the signs and then work out the "transformation" from that person to the known signers. However, this did not appear to work, because the mean and standard deviation from two datasets taken at different times from the same signer had similar distances to those that were found between signers.

It is believed that the stylistic aspects are the dominant cause of error. Even so, through analysis of the signs it might be possible to find a transformation derived by observing a small sample of the user's signs and then developing a suitable transformation. This is similar to modern speech recognition techniques, where a new user is asked to pronounce a small but significant subset of the lexicon of the system before being used, on which is developed a mapping to the standard set of instances. Of course, a complicating issue is that English can be segmented into a known collection of sub-sounds, while there is still some discussion over the feasibility of such an approach in sign language recognition.

However, trying to find optimal transformations is a thesis in itself and is certainly non-trivial.

An alternative and perhaps more difficult approach would be to try to extract attributes that are optimally suited to inter-signer recognition. The attributes we have selected are effective for intra-signer recognition; it may be that for inter-signer recognition a different set of attributes are required altogether. Each new set of attributes proposed would have to be validated in a way similar to that of chapter 5, but using inter-signer validation rather than intra-signer validation.

6.4 Time issues

Another issue is the time required to recognise the signs. However, performing tests on time is extremely subjective, and depends heavily on architecture used, other jobs running simultaneously, algorithm used, extent of optimisation and so on. This makes it difficult to form a complete comparison. Furthermore, do we judge time on the time required to learn the data or the time required to make a decision as to the class, assuming that the time required to make a decision is constant, which of course it isn't?

IBL1 and C4.5 tend to be at opposite end of the spectrum with respect to learning and testing time. IBL1 takes very little time to learn, since all it is effectively doing is reading a giant array of attribute values. C4.5 takes the attributes, considers the gain that a particular comparison results in, chooses the most effective decision and repeats the process until it gets to a leaf node in all cases – which can take a significant amount of time. On the other hand when testing occurs, C4.5 is trivial, simply traversing a tree, which can be done very quickly. It is unusual for a tree to have a depth greater than 20, so there is a nice upper bound. IBL, however, must find the instance that is closest to it. If a brute force algorithm is used, then it has to check every instance in the data set. Fortunately, as illustrated in [WK91], algorithms exist that allow this search in $O(\log n)$ manner, using a method known as k-d trees.

C4.5 is only one example of a symbolic decision tree learner. The implementation of the IBL algorithm used does not employ k-d tree. It uses an optimised search based on collecting all instances of the same class together and maintaining some information about the mean and range of those instances, generously provided by Simon Warfield ([Waron]). So the applicability of the results to other domains is limited.

The tests were performed on a Sun SuperSparc server running Solaris with two 60 MHz processors and 256 Mb of RAM. Only one processor was used for each algorithm, however, and there were a number of other users making use of the system simultaneously. This fact was adjusted for by using the “time” command to report the time used. The “CPU time” measure was used for all computations. This returns only the amount of CPU time used by the process alone for actual calculations, not I/O and so on. Even so, the figures are meant to give only a rough guide to the time taken.

As a measure of performance, the calculations are based on the average training and test time for each element of the test set – i.e. *(test time + training time)/(number of test cases)*.

First we considered the effects on time of increasing the number of samples, as was discussed in section 6.3.1. This shows how as we add signs the time taken increases.

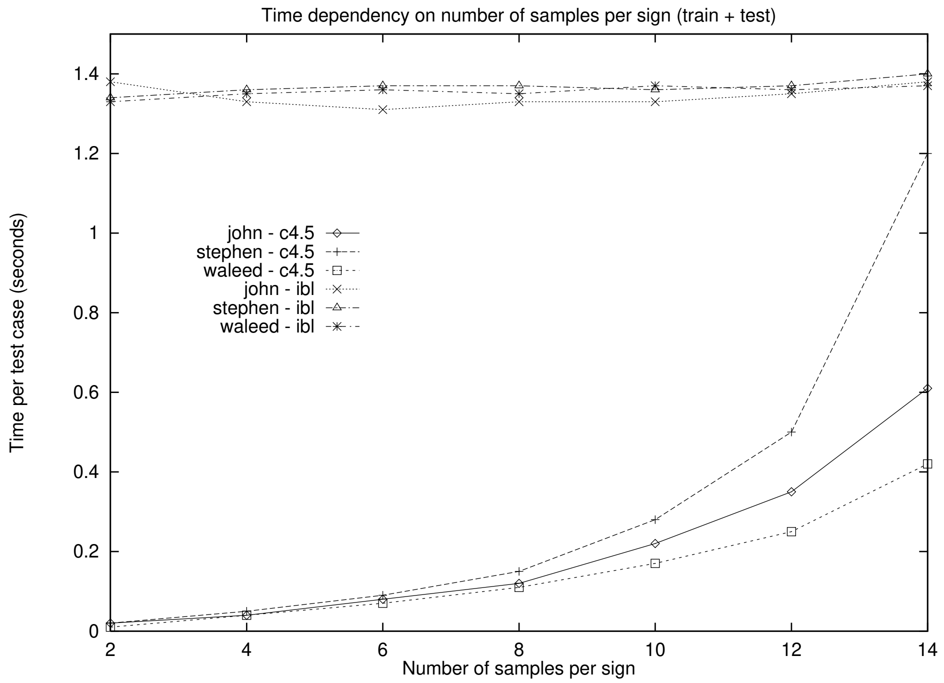


Figure 6.6: The relationship between the number of samples per sign and the error rate.

The effects appear unusual at first. It appears that the IBL algorithm is taking constant time under all circumstances. This is a result of the algorithm used. The optimisations used are that all instances from a given class are treated together – the mean and range for each feature is stored. Calculations are then performed on the means and ranges to eliminate bad class candidates automatically. This usually leaves a very small collection of classes that a test example can take, which are tested as usual. Since the number of classes is the same whether we have 2 samples for each class or 14, then this optimisation will take approximately the same time. As we will see when looking at the way it behave when given more signs, rather than more instances of that sign, it is still sensitive to the number of signs being taught.

Also note that while C4.5 is always faster in the cases shown than IBL, it is showing a tendency to grow faster than linearly. This is not a good thing to happen of course, since it means the more samples the more C4.5 would slow down, and in fact, it would slow down faster than the signs are being added (at least that's what the results here indicated). It also means that at some point (at a rough guess, less than one hundred samples) IBL would be faster than C4.5. If C4.5 were to be used for large lexicon systems, this would need to be dealt with, and some pre-processing to eliminate unnecessary instances (which is almost identical to what IBL3 does) would have to be employed. This is not usually an efficient task in itself and is not a good indication. It must be remembered that this is a very simplified view, and that we are looking at a combination of test and training times, and it is possible that training times are the dominant factor.

We then considered how the error rate changes with the number of signs learnt. This is shown in figure 6.7.

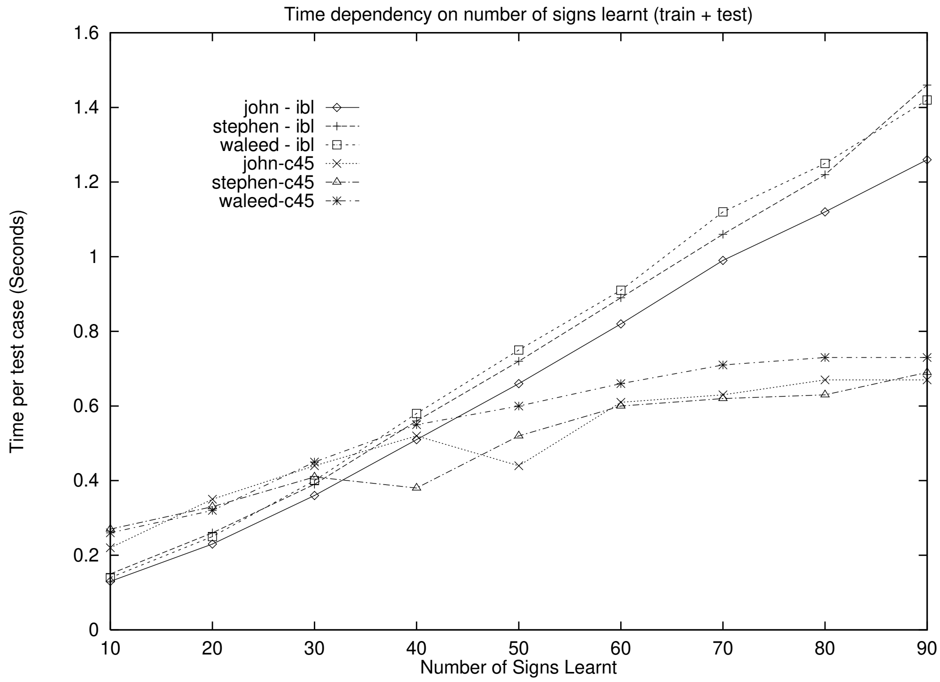


Figure 6.7: The time per test example and its relationship with the number of signs being learnt.

As we can see, both systems take similar time with low sign count, but C4.5 takes less time for larger number of signs than IBL. This should not, however be taken as definitive. In particular, it is possible to write IBL algorithms that runs in $O(\log n)$ time, rather than the algorithm here which was $O(n)$. However, none of the algorithms appear to be worse than linear – both having what appear to near-linear performance. Also, we can see that these systems, with their inefficient algorithms are near to real-time, even with 90 signs, with C4.5 taking 0.7 seconds and IBL taking around 1.4 seconds. It must be remembered that this includes both testing and training.

In conclusion, these results should not be taken as definitive and only approximate. However, it can be seen that C4.5 is probably no longer viable, because of the increase in time required to learn with increasing number of samples, and a greater number of samples leads to a smaller error rate (although only logarithmically). But C4.5 is only one example of a decision tree builder, and similarly the implementation of IBL used here is only one instance. The results as far as timing are concerned are thus not all that useful. Nonetheless, we have at least illustrated that both algorithms can at least be used with what appears to be a linear increase in time with increasing number of signs. Given that computing power grows exponentially, even if other algorithms do not exist that perform better, it is viable at some future point that a real-time system would be possible.

6.5 Optimisations

There are many possible ways to optimise the learning algorithm. These include techniques such as forward and backward selection ([JKP94]) as well as other space-searching techniques. However, the application of these is non trivial and usually takes considerable time.

The conclusion should not therefore be drawn that these results are even near-optimal.

Chapter 7

Summation

What can we learn from all of this, and what remains to be done?

7.1 Conclusion

In this investigation it has been shown that using extremely cheap hardware[1], computationally simple extraction of features, and similarly simple learning algorithms, we can recognise a large lexicon of individual Auslan signs (95 signs) at a useful accuracy rate (approximately 80 per cent).

This was established on a large data-set collected from 5 people with a total of 6650 instances. The set of signs used was specifically chosen to test how well GRASP performed at discriminating similar signs apart.

All of the algorithms presented here are simple enough to be implemented directly in hardware, or can run on a conventional micro-chip with a reasonably-sized EPROM and a medium amount of RAM. This is a good indication for future development. In fact, the current system with 95 signs – with all the data for one signer, feature extraction and learning algorithm – could easily fit in the memory of a palmtop computer (like the Hewlett-Packard 200LX). With further work on optimisation of search algorithms used in instance-based learning, the system could be made to work in real

[1] PowerGloves cost approximately US$50 each, and the interface device could be built for less than US$30.

time.

Features were found that were good discriminants between signs for a given (known) signer.

It also illustrates that instance-based learning does well at tasks involved in gesture recognition, while C4.5 does not seem so well-suited to such applications. There are a number of possible causes for this, such as that there are a small number of instances for each class relative to the number of classes; and that there are a large number of attributes that are inter-dependant, which due to the nature of decisions at each node, C4.5 is only capable of considering one of. Note that C4.5 is only one instance of a symbolic learner and that other symbolic learners may fare better (or worse).

Not only can the most simple method (IBL1) be used, but the more space-conservative and thus run-time conservative versions (IBL2 ands IBL3) of the algorithms can be used, at a slight, but typically constant, increase in the error rate[2].

It has also illustrated that both systems can adequately cope with addition of new signs and new samples. This is important, since this investigation is at best a stepping stone to future, more effective systems.

It also illustrates that it is possible to obtain reasonable results with very cheap hardware. The PowerGlove – which has been superseded – was inaccurate. Even so, it produces enough information to distinguish the signs. With a better glove (and even cooler, with a better *pair* of gloves), better results can be expected.

There are only a limited number of publications with which the quality of the results may be compared. These are the works of Charayaphan and Marble ([CM92]) and Starner ([Sta95]). Both of these used image-based techniques. Furthermore, Charayaphan and Marble used simulations to test their accuracy, rather than actual signs, which is of limited accuracy. A useful comparison can therefore be made to Starner's work. Doing so we would see the following:

- Slightly lower accuracy for GRASP at the same lexicon size (91.3 per cent for Starner's work, 85.0 per cent for GRASP). This can be attributed to a number of causes, such as sign selection and having access to information about one hand.

[2]Typical difference in performance was up to 5 per cent. At the same time, however, IBL3 significantly reduced the number of instances that had to be kept, sometimes by up to 70 per cent. This of course means less computing time spent comparing distances.

- In GRASP, no attempt is made to recognise continuous sign language, only individual signs, whereas Starner's system recognised sentences, albeit grammatically very constrained ones.
- The GRASP lexicon is almost two and a half times the size of Starner's lexicon (95 signs versus 40 signs). GRASP's accuracy is approximately 80 per cent even with the full size lexicon.
- Some of GRASP's signs were selected specifically to test discriminating ability, while Starner's were chose to provide as wide a range of coherent sentences as possible.
- GRASP uses a single (right-handed) PowerGlove, and does not require a camera[3] and image processing hardware.
- A larger amount of data collected (6 650 signs) from a range of people, rather than just one. Inter-signer recognition considered and assessed. Starner considered one signer only and collected 500 sentences, with approximately 4 words per sentence.

There are, as with any thesis, a number of serious short-comings as far as unexplored areas are concerned. These unexplored areas are discussed in the next section.

7.2 Suggestions and reasons for future research

The surface has barely been scratched in the area of sign language recognition in particular and gesture recognition in general.

There are also reasons why this research should be done, even if we put aside questions of equity of access to resources for Deaf people. The concepts are generalisable to other fields outside sign language, and there are serious implications for areas such as human-computer interaction that are more interesting to those who are financially inclined. Gesture is a natural part of communication, and it can be exploited to improve the quality of human-computer interaction.

Furthermore, if some of the issues suggested here are addressed, it is possible that the solutions could have spin-offs in other areas. In particular, techniques for segmentation

[3]You will recall that even though Starner's system makes use of a camera, it requires that the user wear a light yellow glove on one hand and a light orange glove on the other.

of continuous-time input and for transformation of new user information to existing information all have general applications in computer-based learning.

7.2.1 Short-term work

There are some tasks that could be done immediately, and closely on the heels of this thesis. These include:

- Updating the system to make use of the new gloves becoming available now. The main issues here would be re-optimising the division and segment numbers, which would have to be revised because of the increased accuracy; and coping with two hands instead of one and creating additional features to encapsulate the interrelationship between the two hands. In addition, work could be started on an effective two-handed finger-spelling system.
- Optimising the learning algorithm. The algorithms examined here are few. In addition, there are many parameters that were not played with, in particular the similarity function. We simply included or excluded feature sets, but in general it is possible not just to have a weight of "0" or "1" for each feature, which is effectively what we do by including or excluding features, but anywhere in between. Finding a global minimum in a 100-dimensional space is just slightly difficult, however.
- Development into a real-time system. Although this seems like a major development, it is not. We can easily write an instance-based learning algorithm into a program which has good real-time behaviour (using k-d trees we can actually search through n instances in $O(\log n)$ time), and we can even make it incremental. A preliminary attempt was made at this, and it appeared to work reasonably well.

7.2.2 Long-term work

We now consider major aspects of research that could carry on with the work of this thesis.

- Dealing with the segmentation issue. A major problem for computer-based learning is segmenting continuous-time data into its structural components. Sign language is no different. Breaking up continuous Auslan into its component signs is very difficult. It is expected that closely related to this would be the creation of a real-time continuous Auslan recogniser, which would be the next step.

- Investigating other approaches that make use of the features. There is no reason why some of the features used here could not be used as the input to a hidden Markov model. In particular, using the HMM would allow us to simultaneously address the problem of continuous Auslan. Similarly a probabilistic grammar approach might be just as effective.

- Combining the data with facial gesture information. We could also include information about what the face is doing as well, since this would provide a useful clue to the nature of the sign. Already, some work has been done on facial gesture recognition.

- Work on finding suitable transformations for new users of a system such as GRASP to a common space where the concept boundaries are signer independent; or developing more signer-independent features. This can be viewed as a highly complex issue in calibration.

- Use contextual information to aid in the recognition of signs by using techniques such as trigrams, bigrams, sign grammars and so on.

Research in this area is by no means limited to the above and there is much, much more scope for research. This area is interesting not just because of the challenges it poses to learning and to data analysis techniques, but also because of its potential application for creating a new channel of communication between humans and computers that can be far more intuitive than a keyboard.

Appendix A

Other tests performed

What about handshape recognition?

A.1 Introduction

As a test and comparison to see how well algorithms coped with handshape recognition, we took a data set generously donated by Peter Vamplew, and applied the learning algorithms discussed here.

The tests were performed on the basic handshapes used in Auslan[1]. Peter collected the information using a CyberGlove using only the finger information (i.e. not the information about wrist flexure). The features included were:

- Thumb's flexion towards the palm.
- Metacarpophalangeal ("first joint"[2]) flexion for each of the five fingers.

[1]There are actually 31 handshapes in Auslan, but 30 handshapes used in this test. The reason is that four handshape is identical to the second variant of the spread handshape. Their use in sign language is distinguished by the context in which the handshape occurs, much as in English spelling, where the "c" and "k" sounds can cause the same phoneme. We tell which is being used in spoken English using the context.

[2]Actually, although the MCP (who actually wants to read "metacarpophalangeal", let alone type it) is considered the first joint, because that's where our fingers start, there are *four* joints in each of our fingers – the trapeziometacarpal for the thumb and the metacarpocarpals for the other four digits. See figure 2.1 for a diagram.

- Proximal interphalangeal ("second joint") flexion for each finger.
- Inter-finger abductions (i.e. thumb-index, index-middle, middle-ring, ring-little).
- Rotation of the little finger towards the palm.

In total, there were 16 features. Peter collected a total of ten data sets from ten different people, each with five samples of the variant handshapes. The data sets were divided into two sets, the unseen test cases (3) and the seen cases (7). Of the seen cases, 4 of the samples were used for training and the fifth for testing. Thus there were two test sets - seen candidates, and unseen candidates.

A.2 Simple, straight attributes fed directly to the learning algorithm

The simplest test possible is to feed the handshape data directly to the learning algorithms with no pre-processing.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
seen	5.0	11.4	12.8	14.0
unseen	13.7	19.2	18.7	22.7

Table A.1: Results of learning algorithms on Vamplew's handshape data

As can be seen, the IBLs outperform the C4.5 learners, but IBL2 and 3 do not seem to do as well as we would expect (previously they were only a few per cent behind IBL1). However, the results on seen users are quite good (95 per cent correct). It is easy to see how if we could get the right handshape 95 per cent of the time, we could use this information to further increase the accuracy of the sign recognition, by adding an additional feature specifying handshape.

A.3 Creating additional attributes

As previously discussed, a sometimes effective way to decrease error rates is to develop attributes from the raw data that can be better discriminants than the raw data by itself. This is especially true of learners like C4.5, since if we select our attributes carefully so that the concept boundaries are parallel to the attributes we select, they will be able to segment the space more accurately. In this case, several attributes which were thought to be useful were added. These were:

- The difference between the bends of the MCPs. It was felt that in signing, the bend of a finger relative to the previous one might be significant, and provide more information than simply knowing the absolute bend of each finger.
- A measure of the closeness of the thumb to the little finger. This also was believed to be a useful piece of information, since some handshapes require the thumb and little finger to touch. The equation derived empirically and approximately for this was *little-finger-rotation*(little-finger-mcp+little-finger-pip)+4*thumb-palm-rotation*(thumb-mcp+thumb-pip).*
- Total finger bend. By summing the MCP and PIP we can get a measure of how far the finger is bent overall. It may be that this proves to be more invariant, particularly *between* signers than the raw values.
- The differences between total finger bend.

All up there were an additional 19 attributes suggested. They were tested together and their effect on the accuracy measured.

Dataset	IBL1 Error (%)	IBL2 Error(%)	IBL3 Error(%)	C4.5 Error(%)
seen	5.1	12.1	9.8	11.8
unseen	16.7	21.9	21.4	22.3

Table A.2: Results of learning algorithms on Vamplew's handshape data with additional attributes.

A.3.1 Discussion

The additional attributes appear to be useless. They seem to have had little effect on the performance, in some cases in fact leading to the error rate *increasing*. This means that perhaps we should ignore them and not use them at all, especially in the case of IBL1; they seemed to have had an improving effect on the results with C4.5.

A.4 Other investigations and notes

A.4.1 Neural network behaviour

Peter Vamplew reported that he got results significantly better than the above using neural nets. His results were on a fully-connected, back-propagating neural network with 16 input nodes, 40 hidden nodes and 30 output nodes. His results are shown below.

Dataset	Neural Net Error (%)
seen	3.0
unseen	15.0

Table A.3: Results of neural nets on Vamplew's handshape data with additional attributes.

Obviously, these results are pretty good, although they are closely comparable with the results from IBL1. The only drawback is the learning time is quite high and the complexity of the neural net likewise, with a total of 86 nodes, 40 with 16 inputs and 30 with 40 inputs.

A.4.2 How does a PowerGlove compare?

Out of curiosity, we thought it might be interesting to compare the behaviour of the CyberGlove to that of the PowerGlove. Unfortunately, however, there was insufficient data for a complete comparison; so the results from a single person using the PowerGlove are compared with the results of a single person using the CyberGlove.

As before, 5 instances total were collected for each of the variational handshapes, both with the CyberGlove and the PowerGlove. One out of the 5 was used for testing.

The results are shown in table A.4.

The results here show that the CyberGlove has less than half the error rate as the PowerGlove with C4.5 and one tenth or less when using IBL1. IBL performs worse when used with the PowerGlove, because it appears that C4.5 is capable of forming better concept boundaries with the small collection of attributes than IBL is. For the

Dataset	IBL1 Error (%)	C4.5 Error(%)
CyberGlove with no additional features (16 features)	6.5	29.5
CyberGlove with additional features (35 features)	1.6	27.9
PowerGlove with no additional features (4 features)	68.3	60.0
PowerGlove with additional features (9 features)	65.3	58.3

Table A.4: Results of comparison between PowerGlove and CyberGlove on handshape recognition

large number of attributes available with the CyberGlove, IBL performs much better.

The result with the CyberGlove with additional features using IBL1 seems a little alarming to say the least. It means we can recognise handshapes with an accuracy of 98.4 per cent, considerably better than before with the full test. This result seemed extremely suspect, and the test was repeated on all the other data-sets to verify that this was no coincidence, since before, a random one of the seven available subjects was chosen.

The results are shown in table A.5.

Dataset	IBL1 Error (%)
a	1.6
b	4.9
c	1.6
e	8.1
f	9.8
g	4.9
j	11.4
average	6.0

Table A.5: Results of CyberGlove data for each of the data-sets

It appears that the above was just a a "fluke", but it does illustrate some aspects of the variability. It must be realised that this result is not impossible, considering that the IBL algorithm is being used. What happened before is that we lumped *all* of the data-sets together, regardless or not of whether they were from that person or not. This thus introduces points into the data-set that did not originally belong to that person, but also to how someone else did the handshapes. Once we only include examples that belong only to that person, it now is only possible to match *that person's* version of the handshape and not someone else's *false* handshape.

Thus the results obtained above are only valid if the system is only trained on one person.

A.5 Conclusions

IBL behaves comparably to neural network approach and does better than it for unseen signers. The scale of the problem is bounded, however and the system does not have to be built in a scalar manner, since there will always be 31 handshapes. C4.5 is not competitive.

The difference in performance between the PowerGlove and CyberGlove is significant, as would be expected. This becomes an issue if GRASP was to be expanded to things such as finger-spelling.

Appendix B

Words selected for the project

B.1 Notes on the list

There are three types of sign: One-handed ("O"), two-handed ("T") and double-handed ("D"). For a discussion of the differences, see Chapter 2.

Word/ Sign	Auslan Dict Ref	Type of sign	Main Handshapes	Reason for Choice
alive	200	O	B	Common sign
all	158	O	B	Common sign
answer	2230	D	B	Common sign
boy	821	O	F	Common sign
building	512	D	O	Large number of critical points
buy	1731	D	A	Common sign
change(mind)	2885	O	V	Similarity (head area)
cold	1802	D	A	Common sign
come	2322	O	X	Common sign
computer(PC)	1065	D	G2	Relevance to project
cost	1234	O	5	Common sign
crazy	752	O	G1	Large number of critical points
danger	1219	D	5	Common sign
deaf	2086	O	6	Similarity (head area)
different	1072	D	G	Common sign

Word/ Sign	Auslan Dict Ref	Type of sign	Main Handshapes	Reason for Choice
draw	3454	O	12	12 Handshape & Similarity (write)
drink	2759	O	C	Common sign
exit	336	T	B	Common sign
eat	2632	O	O	Common sign
flashlight	2650	O	O, 5	Changing handsigns/similarity
forget	2613	O	O, 5	Changing handsigns
girl	785	O	G	Common sign
give	618	D	B	Common sign
glove	440	T	B	Relevance to project
God	848	O	G2	Large sweep of space.
go	243	O	B	Common sign
happy	387	T	B	Common sign
head	3	O	B	Common sign
hear	2747	O	C	Common sign
heat	1233	O	5	Common sign
hello	144	O	B	Common sign
hers	2132	O	6	Similarity (all, hello)
how	601	T	B	Common sign
hurry	1099	T	G	Common sign
hurt	2139	O	H	Common sign
innocent	1970	D	6	Similarity (responsible, voluntary)
I	839	O	G	Common sign
is(true)	2316	D	X	Common sign
joke	3424	O	3	3 handshape
juice	1347	O	5	Little movement & similarity (bad)
know	1942	O	6	Common sign
later	967	T	G	Common sign
lose	1507	D	5	Similarity (all, hello)
love	548	O	B	Common sign
make	1835	D	A	Common sign
man	1605	O	A	Common sign
maybe	3340	O	Y	Similarity (which) & Y handshape
mine	1617	O	A	Common sign

Word/ Sign	Auslan Dict Ref	Type of sign	Main Handshapes	Reason for Choice
money	1715	D	A	Common sign
more	330	D	B1	Common sign
name	2278	O	X	Similarity (head area)
need	3612	O	N	N handshape
not-my-problem	1530	D	5	Common sign
Ok	3043	O	F	Common sign
paper	1818	D	A	Common sign
pen	3381	O	11	Common sign
please	90	O	B	Common sign
polite	3589	D	K	K handshape
question	2429	D	X	Common sign
ready	3171	D	Gam	Gamma handshape
read	2983	T	V	Common sign
research	3005	D	V	True 2-handed sign
responsible	3848	O	R	R handshape
right	1990	T	6	Common sign
sad	94	O	B	Common sign
same	1066	D	G	Common sign
science	2723	T	O	True 2-handed sign
share	416	D	B	Double gesture
shop	1730	D	A	Common sign
soon	1641	O	A	Common sign
sorry	3608	O	Ani	Animal Handshape
spend	1714	O	A	Common sign
stubborn	3474	D	Rude	Rude Handshape
surprise	1266	O	5	Similarity (mine, I)
take	1318	O	5	Common sign
temper	3314	O	Y	Similarity (change-mind, name)
thank	91	O	B	Common sign
think	14	O	B	Common sign
tray	1127	D	G, A	Complex sign
us	845	O	G	Common sign
voluntary	3111	O	F	Similarity (responsible)

Word/ Sign	Auslan Dict Ref	Type of sign	Main Handshapes	Reason for Choice
wait	3212	D	Gam	Gamma Handshape
watch	2949	D	V	Common sign
what	876	O	G	Common sign
when	1232	O	5	Common sign
which	3345	O	Y	Common sign
who	861	O	G	Common sign
why	2497	O	L	Common sign
will	1597	O	A	Common sign
wrong	3226	O	I	Common sign
yes	1671	O	A	Common sign
you	889	O	G	Common sign

Table B.1: Signs included and the reason for their selection

Appendix C

Glove package source

C.1 glovedata.h and glovedata.c

C.1.1 glovedata.h

```
/*****************************************************************************
glovedata.h

The format about which information about the glove is stored.  Note
that while at the moment pitch and yaw measurements are not available,
they may be at some future time. Similarly for the little finger.

*****************************************************************************/

#include <stdio.h>

/* Buttons on the PowerGlove control pad */

/* 0 through to 9 map onto the 0 to 9 on the control pad. */

#define PGLOVE_PROG  (0x84)
#define PGLOVE_ENTER (0x80)
#define PGLOVE_CENTER (0x00)

/* Note: There is an ambiguity here -- pressing "Center" and "0" */
/* results in the same signal being sent -- so don't use "Center" and */
/* "0" buttons together. */

#define PGLOVE_UP    (0x0C)
#define PGLOVE_RIGHT (0x0D)
#define PGLOVE_DOWN  (0x0E)
#define PGLOVE_LEFT  (0x0F)
#define PGLOVE_SEL   (0x83)
#define PLOVE_START  (0x82)
#define PGLOVE_A     (0x0A)
#define PGLOVE_B     (0x0B)
/* No key pressed is indicated by FF */
#define PGLOVE_NOKEY (0xFF)

typedef struct {
  float x, y, z;
  /* Works out to be +/- 1 roughly where "1" = 1 metre approximately */
```

```
  float roll, pitch, yaw;
  /* Roll, pitch and yaw is expressed as a value between 0 and */
  /* 1. Negative values indicate the information is not valid or */
  /* available */
  float thumb, fore, index, ring, little;
  /* Finger positions. It is assumed these are between 0 (straight) */
  /* and 1 (flexed completely). Also assumed they are always available */
  unsigned char keycode, gstat1, gstat2, recvals;
  /* Various status bits about the glove. May be ignored if desired */
} glovedata;

/* Prints glove information in comma separated sequence in the same */
/* order above. Also, a file writing version. */

void printglove(glovedata* gd);
void fprintglove(FILE *outfile, glovedata* gd);

/* Complements printglove, and can read the output of printglove back */
/* into a glovedata struct. Also a file reading version. */
int readglove(glovedata* gd);
int freadglove(FILE *infile, glovedata *gd);
```

C.1.2 glovedata.c

```
/******************************************************************************
glovedata.c

The format about which information about the glove is stored.  Note
that while at the moment pitch and yaw measurements are not available,
they may be at some future time. Similarly for the little finger.

*****************************************************************************/


#include "glovedata.h"

void printglove(glovedata* gd){
  printf("%f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, 0x%X, 0x%X, 0x%X, 0x%X\n",
         gd->x, gd->y, gd->z, gd->roll, gd->pitch, gd->yaw, gd->thumb,
         gd->fore, gd->index, gd->ring, gd->little, gd->keycode,
         gd->gstat1, gd->gstat2, gd->recvals);
}

void fprintglove(FILE *outfile, glovedata* gd){
  fprintf(outfile, "%f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, 0x%X, 0x%X, 0x%X, 0x%X\n",
         gd->x, gd->y, gd->z, gd->roll, gd->pitch, gd->yaw, gd->thumb,
         gd->fore, gd->index, gd->ring, gd->little, gd->keycode,
         gd->gstat1, gd->gstat2, gd->recvals);
}


int readglove(glovedata* gd){
  int i, kc, gs1, gs2, rv;
  i= scanf("%f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, 0x%X, 0x%X, 0x%X, 0x%X\n",
              &(gd->x), &(gd->y),
              &(gd->z), &(gd->roll), &(gd->pitch), &(gd->yaw), &(gd->thumb),
              &(gd->fore), &(gd->index), &(gd->ring), &(gd->little),
              &kc, &gs1, &gs2, &rv);
  gd->keycode = (unsigned char) kc;
  gd->gstat1  = (unsigned char) gs1;
  gd->gstat2  = (unsigned char) gs2;
  gd->recvals = (unsigned char) rv;
  return i;
}
```

```
int freadglove(FILE *infile, glovedata* gd){
  int i, kc, gs1, gs2, rv;
  i= fscanf(infile, "%f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, 0x%X, 0x%X, 0x%X, 0x%X\n",
               &(gd->x), &(gd->y),
               &(gd->z), &(gd->roll), &(gd->pitch), &(gd->yaw), &(gd->thumb),
               &(gd->fore), &(gd->index), &(gd->ring), &(gd->little),
               &kc, &gs1, &gs2, &rv);
  gd->keycode = (unsigned char) kc;
  gd->gstat1  = (unsigned char) gs1;
  gd->gstat2  = (unsigned char) gs2;
  gd->recvals = (unsigned char) rv;
  return i;
}
```

C.2 gloveread.c

```
/******************************************************************************
 gloveread.c

A program that takes the output from any pipe providing input to the
program, expected in the format specified in the PGSI guide (AGE format) and
converts it to a human readable form, with all values between either
+/- 1 or 0-1, depending -- x, y, z are +/- 1 with all information
about finger flex being 0-1. This allows you to write tools that mean
that we can attach a new glove or tracking mechanisms to new devices.

*****************************************************************************/
#include <stdio.h>
#include "glovedata.h"


main(){
/* A finite state implementation is used. See text of thesis for */
/* fsm. */
  glovedata gd;
  unsigned char state = 0, c; /* State of the FSM */
  /* The above are status bits of various kinds. Doesn't make sense to */
  /* map them to doubles. */
  gd.yaw = -1;
  gd.pitch = -1;
  setbuf(stdout, NULL);
  setbuf(stdin, NULL);
  while((c = getchar()) != EOF){
/*    printf("State is %d, char is 0x%X\n", state, c); */
    switch(state){
/* States 0 and 1 check the header */
    case 0:
      if(c == 0x5F) state = 1;
      break;
    case 1:
      if(c == 0x5f) state = 2;
      else state = 0;
      break;
    case 2:
      gd.x = ((c > 127) ? c-255 : c)/128.0;
      state = 3;
      break;
    case 3:
      gd.y = ((c > 127) ? c-255 : c)/128.0;
      state = 4;
      break;
    case 4:
      gd.z = ((c > 127) ? c-255 : c)/128.0;
      state = 5;
      break;
    case 5:
      /* If rotation is out of bounds, then indicate using minus */
```

```
        /* value. */
        if(c > 0xB) gd.roll = -1;
        else gd.roll = (float) c / 12.0;
        state = 6;
        break;
      case 6:
        gd.thumb = ((c >> 6) & 3) / 4.0;
        gd.fore  = ((c >> 4) & 3) / 4.0;
        gd.index = ((c >> 2) & 3) / 4.0;
        gd.ring  = (c & 3) / 4.0;
        gd.little = gd.ring;
        /* A kludge for the powerglove -- assume */
        /* similarity because of the metacarpocarpal */
        /* joints and their relationship */
        state = 7;
        break;
      case 7:
        gd.keycode = c;
        state = 8;
        break;
      case 8:
        gd.gstat1 = c;
        state = 9;
        break;
      case 9:
        gd.gstat2 = c;
        state = 10;
        break;
      case 10:
        gd.recvals = c;
        printglove(&gd);
        state = 0;
        break;
      }
    }
  }
```

C.3 gloveplay.c

```
/*******************************************************************/
/* This program displays a simple hand icon based on raw powerglove*/
/* input.                                                          */
/*                                                                 */
/* This program was written after looking at Greg Newby's source   */
/* for the PGSI, flyinghand.c. However, it is very different to it */
/* in many ways -- such as the use of 3D.                          */
/*                                                                 */
/*******************************************************************/

#include <stdio.h>
#include <gl.h>
#include <math.h>
#include <string.h>
#include "glovedata.h"
#include <device.h>
#include <termios.h>
#include <fcntl.h>

#define MY_BLACK     0x0
#define MY_RED       0x0000ff
#define MY_WHITE     0xffffff
#define MY_GREEN     0x00ff00
#define MY_BLUE      0xff0000
#define MY_CYAN      0xffff00
#define MY_MAGENTA   0xff00ff
```

```
char msgstr[100];
int sleepval = 0;


/****************************************************************/
/* Main loop:  initialize everything and draw/move a hand image */
/****************************************************************/
main(int argc, char *argv[])
{
  glovedata globalgd;
  /* use argument for delay time, else use default */
  if (argc > 1) sleepval = atoi(argv[1]);

  /* Initialize graphics and powerglove */
  initialize(argc, argv);

  /* Main loop:  Read in data, draw the hand */
  while(!getbutton(QKEY))
    {
      query_glove(&globalgd);
      cpack(MY_BLACK); clear();
      drawsystem(&globalgd);
      writenote();
      swapbuffers();
      zclear();
      sginap(sleepval);
    }

  /* cleanup and go */
  gexit();
}


/****************************************************************/
/* Get glove and user ready; flush data from glove buffer       */
/****************************************************************/
initialize(int argc, char *argv[])
{
  /* Initialize graphics */
  setbuf(stdin, NULL);
  setbuf(stdout,NULL);
  keepaspect(1,1);
  prefposition(180, 1000, 60, 920);
  winopen((argc == 3 ? argv[2] : "PowerGlove Playback"));
/*  perspective(900, 1.0, 0.5, 5.0); */
  ortho(-2.0, 2.0, -2.0, 2.0, 0.5, 5.0);
  lookat(1, 1, 1, 0, 0, -2, 0);
  backface(FALSE);
  zbuffer(TRUE);
  zclear();
  doublebuffer();
  RGBmode();
  gconfig();
  cpack(MY_BLACK); clear();
}



/****************************************************************/
/* Get glove data                                               */
/****************************************************************/
query_glove(glovedata* gdata)
{
  int ret;
  char dying[]="Input is finished ... Dying !";
  /* disable buffering for stdout */


  ret = readglove(gdata);
  /* Call synthpy to synthesise pitch and yaw values if necessary */
  synthpy(gdata);
```

```
  if((ret == -1) || feof(stdin)){
    strcpy(msgstr,dying);
    writenote();
    sginap(200);
    exit(0);
  }
  return(0);
}

/****************************************************************/
/* Draw a simple 2D hand                                        */
/****************************************************************/


drawfinger(float knuck1, float knuck2, float knuck3){
#define TOBECALC 0
  float fingerpos[8][3] = {
    {0, 0, 0},
    {0.1, 0, 0},
    {0, TOBECALC, TOBECALC},
    {0.1, TOBECALC, TOBECALC},
    {0.01, TOBECALC, TOBECALC},
    {0.09, TOBECALC, TOBECALC},
    {0.03, TOBECALC, TOBECALC},
    {0.07, TOBECALC, TOBECALC},
  };
  int i;

/* Draws a finger of length 1 */
  fingerpos[2][1] = fingerpos[3][1] = 0.4 * cos(knuck1*M_PI_2);
  fingerpos[2][2] = fingerpos[3][2] = 0.4 * sin(knuck1*M_PI_2);
  fingerpos[4][1] = fingerpos[5][1] = 0.35 * cos((knuck1 +
                                                  knuck2)*M_PI_2) +
                                     fingerpos[2][1];
  fingerpos[4][2] = fingerpos[5][2] = 0.35 * sin((knuck1 +
                                                  knuck2)*M_PI_2) +
                                     fingerpos[2][2];
  fingerpos[6][1] = fingerpos[7][1] = 0.25 *
    cos((knuck1+knuck2+knuck3)*M_PI_2) + fingerpos[4][1];
  fingerpos[6][2] = fingerpos[7][2] = 0.25 * sin((knuck1 + knuck2 +
                                                  knuck3)*M_PI_2) +
                                      fingerpos[4][2];
  cpack(0x005050);
  bgnqstrip();
  v3f(fingerpos[0]);
  v3f(fingerpos[1]);
  v3f(fingerpos[2]);
  v3f(fingerpos[3]);
  cpack(0x009090);
  v3f(fingerpos[4]);
  v3f(fingerpos[5]);
  cpack(0x00A0A0);
  v3f(fingerpos[6]);
  v3f(fingerpos[7]);
  endqstrip();
}
  /* First point is origin */


drawhand(glovedata *gdata)
{
   float tcoord[3];
   int i, handsides = 7;
   float hand[7][3] = {
     {-0.1,0.2,0},
     {0.2,0.2,0},
     {0.2,-0.1,0},
     {0.1,-0.2,0},
     {-0.1,-0.2,0},
     {-0.2,-0.1,0},
     {-0.1,0,0}
   };
   pushmatrix();
```

```
   bgnpolygon();
   cpack(0x004040);
   for(i = 0; i < handsides; i++){
     v3f(hand[i]);
   }
   endpolygon();
   sprintf(msgstr,"%f %f %f %d %d %d %d %d %d", (gdata->x),
           (gdata->y), (gdata->z), (gdata->recvals) & 1,
           (gdata->recvals)>>1 & 1, (gdata->recvals)>>2 & 1,
           (gdata->recvals)>>3 & 1, (gdata->recvals)>>4 & 1,
           (gdata->recvals)>>5 & 1, (gdata->recvals)>>6 & 1);
   /* Thumb */
   pushmatrix();
   translate(-0.2,-0.1,0);
   rotate(300,'z');
   scale(1,0.2,0.2);
   drawfinger(gdata->thumb,gdata->thumb,gdata->thumb);
   popmatrix();
   /* Fore finger */
   pushmatrix();
   translate(-0.1,0.2,0);
   scale(0.9,0.4,0.4);
   drawfinger(gdata->fore, gdata->fore, gdata->fore);
   popmatrix();
   /* Index finger */
   pushmatrix();
   translate(0,0.2,0);
   scale(0.9,0.5,0.5);
   drawfinger(gdata->index, gdata->index, gdata->index);
   popmatrix();
   /* Ring finger */
   pushmatrix();
   translate(0.1,0.2,0);
   scale(0.9,0.4,0.4);
   drawfinger(gdata->ring, gdata->ring, gdata->ring);
   popmatrix();
   popmatrix();
}

synthpy(glovedata *gdata){

}

drawsystem(glovedata *gdata){

  float origin[3] = {0, -1, -2};
  float xaxis[3] = {100, -1, -2};
  float yaxis[3] = {0, 100, -2};
  float zaxis[3] = {0, -1, 100};
  /* Draw the axes */
  bgnline();
  cpack(MY_RED);
  v3f(origin);
  v3f(xaxis);
  v3f(origin);
  cpack(MY_GREEN);
  v3f(yaxis);
  v3f(origin);
  cpack(MY_BLUE);
  v3f(zaxis);
  endline();
  pushmatrix();
  translate((gdata->x),(gdata->y),(gdata->z)*9-2);
  scale(1,1,-1);
  rotate(900,'x');
  rotate(3600*(1-(gdata->roll)), 'y');
  drawhand(gdata);
  popmatrix();
}


/****************************************************************/
/* Place stationary text on the screen                         */
```

```
/***************************************************************/
writenote()
{
  pushmatrix();
  ortho2(0.0,1024.0,0.0,768.0);
  cmov2(10.0,30.0);
  cpack(MY_MAGENTA);
  charstr(msgstr);
  popmatrix();
}
```

C.4 gloverecord.c

```
/******************************************************************************
gloverecord.c

This file reads in a list of words, stores them in an array and
outputs them into a directory specified in the command line.

Usage: gloverecord word-list output-directory sample-count

*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include "glovedata.h"
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

#define MAXWORDS 300
#define MAXLENGTH 50

char wordlist[MAXWORDS][MAXLENGTH];
FILE *tty;
readwords(FILE *wordfile, int* wordcount){
  int i = 0;
  int retval = 1;
  while(i < MAXWORDS && retval > 0){
    retval=fscanf(wordfile, "%s", wordlist[i++]);
  }
  if(i >= MAXWORDS-1)
    fprintf(tty, "Warning: Maximum word count exceeded - modify source !\n");
  *wordcount = i - 1;
}

scanwords(int wordcount, int samples){
  char outfilename[100];
  int i = 0,j = 0;
  char backtracked = 0;
  glovedata gd;
  FILE *outfile;
  while(i < wordcount){
    if(!backtracked) j = 0;
    while(j < samples){
      backtracked = 0;
      fprintf(tty, "\nPlease sign the word: <<<%s>>>.\nPress Center to begin.\n",
              wordlist[i]);
      fprintf(tty, "Press 'B' on the glove to redo the last sign.\n");
      fprintf(tty, "Press 'A' on the glove to end, once the sign is complete.\n");
      sprintf(outfilename, "%s%d.sign", wordlist[i], j);
/*      fprintf(tty, "The file will be called: %s\n", outfilename); */
      readglove(&gd);
      while((gd.keycode != PGLOVE_B) && (gd.keycode != PGLOVE_CENTER)){
        readglove(&gd);
      }
      if(gd.keycode == PGLOVE_B){
        while(gd.keycode == PGLOVE_B){
```

```
          /* Wait for them to take their finger off the "B" button. */
          readglove(&gd);
        }
        backtracked = 1;
        /* B is for backtrack ... decrement j and i as necessary */
        j--;
        if(j < 0){
          if (i == 0){
            fprintf(tty, "Can't backtrack any more\n");
            i = 0;
            j = 0;
          }
          else{
            i--;
            j = samples-1;
          }
        }
        fprintf(tty, "Backtracking to %d: %s, sample %d\n", i+1, wordlist[i],j);
        break;
      }
      else if(gd.keycode == PGLOVE_CENTER){
        if(!(outfile = fopen(outfilename, "w"))){
          fprintf(tty, "Could not open file %s\n", outfilename);
          exit(1);
        }
        fprintf(tty, "Now recording data in file %s ...\n", outfilename);
        readglove(&gd);
        while(gd.keycode != PGLOVE_A){
          fprintglove(outfile, &gd);
/*        printglove(&gd); */
          readglove(&gd);
        }
        fclose(outfile);
      }
      if(!backtracked){
        j++;
      }
      else break;
      fprintf(tty, "Scanned word %d: %s, sample %d\n", i, wordlist[i],j);
    }
    if(!backtracked) i++;
  }
}



int main(int argc, char *argv[]){
  FILE *wordfile;
  int samples;
  struct stat dirinfo;
  int wordcount;
  /* To get optimal performance, we need this: */
  setbuf(stdin, NULL);
  setbuf(stdout, NULL);
  if(argc != 4){
    printf("Usage: %s word-list-file output-directory sample-count\n",
           argv[0]);
    return(1);
  }

  samples = atoi(argv[3]);
  if(!(tty = fopen("/dev/tty", "w+"))){
    printf("Could not open the console\n");
    return(1);
  }
  setbuf(tty,NULL);
  if(!(wordfile = fopen(argv[1], "r"))){
    printf("Could not open %s\n");
    return(1);
  }
  if(stat(argv[2], &dirinfo) < 0){
    /* Means directory does not exist */
    if(mkdir(argv[2], S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IWGRP |
```

```
          S_IXGRP) < 0){
      printf("Could not create directory %s \n", argv[2]);
      return(1);
    }
  }
  else{
    /* Means directory does exist */
    if(!S_ISDIR(dirinfo.st_mode)){
      printf("%s is not a directory. Bummer !!\n", argv[2]);
      return(1);
    }
  }
  if(chdir(argv[2]) < 0){
    printf("Huh ? I can't get into %s. Any idea why ?\n", argv[2]);
    return(1);
  }
  readwords(wordfile, &wordcount);
  scanwords(wordcount, samples);
}
```

Appendix D

A Note on the bibliography

Many of these references are available on the Internet. Where possible, the URL of all such documents is indicated, in the hope that this will make tracking down the information easier. Note that URL's tend to be capricious and change more frequently than say, the publisher of a book. In many ways, access to the Internet is necessary for research into this area, since it enables information to be transmitted quickly. There are also a number of other resources such as the gesture-l mailing list and the sci.virtual-worlds newsgroup which have been invaluable in this thesis. As you can see, some of the information in this thesis is as recent as July 1995.

Bibliography

[AKA90] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Draft submission to Machine Learning*, 1990.

[Aus95] Deafness Resources Australia. Sign language and the deaf community course notes, July 1995. Address available in: `http://www.cse.unsw.edu.au/ waleed/local_deaf.html` .

[Bol80] R. A. Bolt. Put that there: Voice and gesture at the graphics interface. In *Proceedings SIGGRAPH*. ACM Press, 1980.

[CH67] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, January 1967.

[Cha93] Eugene Charniak. *Statistical Language Learning*. MIT Press, 1993.

[CM92] C. Charayaphan and A. Marble. Image Processing system for interpreting motion in American Sign Language. *Journal of Biomedical Engineering*, 14:419–425, September 1992.

[Cov68] Thomas M. Cover. Estimation by the nearest neighbour rule. *IEEE Transactions on Information Theory*, IT-14(1):50–55, January 1968.

[DH94] Brigitte Dorner and Eli Hagen. Towards an American Sign Language Interface. *Artificial Intelligence Review*, 8(2–3):235–253, 1994.

[Dor94] Brigitte Dorner. Chasing the Colour Glove: Visual hand tracking. Master's thesis, Simon Fraser University, 1994. Available at: `ftp://fas.sfu.ca/pub/thesis/1994/BrigitteDornerMSc.ps` .

[DS93] James Davis and Mubarak Shah. Gesture recognition. Technical Report CS-TR-93-11, University of Central Florida, 1993.

[Fel94] S. Sidney Fels. *Glove-TalkII: Mapping Hand Gestures to Speech Using Neural Networks – An Approach to Building Adaptive Interfaces.* PhD thesis, Computer Science Department, University of Toronto, 1994.

[FH93] S. S. Fels and G. Hinton. GloveTalk: A neural network inteface between a DataGlove and a speech synthesiser. *IEEE Transactions on Neural Networks*, 4:2–8, 1993.

[Gat72] Geoffrey E. Gates. The reduced nearest neighbour rule. *IEEE Transactions on Information Theory*, pages 431–433, May 1972.

[Gri83] G. Grimes. Digital Data Entry Glove interface device. Patent 4,414,537, AT & T Bell Labs, November 1983.

[Hag94] Eli Hagen. A flexible American Sign Language interface to deductive databases. Master's thesis, Computer Science, Simon Fraser University, 1994. Available at `ftp://fas.sfu.cs/pub/theses/1994/EliHagenMSc.ps` .

[Har68] Peter E. Hart. The condensed nearest neighbour rule. *IEEE Transactions on Information Theory*, pages 516–517, May 1968.

[HB78] D. J. Hand and B. G. Batchelor. Experiments on the edited condensed nearest neighbour rule. *Information Sciences*, 14:171–180, 1978.

[HSM94] Chris Hand, Ian Sexton, and Michael Mullan. A linguistic approach to the recognition of hand gestures. In *Designing Future Interaction Conference*. Ergonomics Society/IEE, April 1994. Also available at `http://www.cms.dmu.ac.uk/People/cph/Publications/DFI94/gestures.html` .

[ITK92] Koichi Ishibuchi, Haruo Takemura, and Kumio Kishino. Real-time hand shape recognition using a pipe-line image processor. In *Proceedings of the IEEE International Workshop on Robot and Human communications*, pages 111–116, 1992.

[JKP94] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Machine Learning Conference*, pages 121–129, 1994.

[Joh89] Trevor Johnston. *Auslan Dictionary : a Dictionary of the Sign Language of the Australian Deaf Community.* Deafness Resources Australia Ltd, 1989.

[JRC88] Raymond C. Jeanes, Brian E. Reynolds, and Bernadette C. Coleman. *Basic Signs for Communication with the Deaf.* Deafness Resources Australia, 1988. Reprinted extracts from the *Dictionary of Australasian Signs* with permission from the Victorian School for Deaf Children.

[KL89] Jim Kramer and Larry Leifer. The Talking Glove: A speaking aid for non-vocal deaf and deaf-blind individuals. In *Proceedings of RESNA 12th Annual Conference*, pages 471–472, 1989.

[KL90] James Kramer and Larry J. Leifer. A "Talking Glove" for nonverbal deaf individuals. Technical Report CDR TR 1990 0312, Centre For Design Research – Stanford University, 1990.

[Kra91] Jim Kramer. Communication system for deaf, deaf-blind and non-vocal individuals using instrumented gloves. Patent 5,047,952, Virtual Technologies, 1991.

[Kru90] Myron W. Krueger. *Artificial Reality.* Addison-Wesley, Reading Mass., second edition, 1990.

[MT91] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *CHI '91 Conference Proceedings*, pages 237–242. Human Interface Laboratory, Fujitsu Laboratories, ACM, 1991.

[OTK91] T. Onishi, H. Takemura, and E. Kishino. A study of human gesture recognition for an interactive environment. In *7th Symposium on Human Interaction*, pages 691–696, 1991.

[PW90] Randy Pausch and R. D. Williams. Tailor: Creating custom user interfaces based on gesture. Technical Report TR-90-06, University of Virginia, 1990.

[PW91] R. Pausch and R. D. Williams. Giving CANDY to children: User-tailored gesture input driving an articular-based speech synthesiser. Technical Report TR-91-23, University of Virginia, 1991.

[Rhe91] Howard Rheingold. *Virtual Reality.* Touchstone Books, 1991.

[Rub90] Dean Rubine. *The Automatic Recognition of Gestures.* PhD thesis, Computer Science, Carnegie-Mellon University, 1990.

[SP95] Thad Starner and Alex Pentland. Visual recognition of American Sign Language using Hidden Markov Models. Technical Report TR-306, Media Lab, MIT, 1995. Available at:

ftp://whitechapel.media.mit.edu/pub/tech-reports/TR-306.ps.Z
.

[Squ94] Brett Squires. Automatic speaker recognition, an application for machine learning. Undergraduate thesis, University of New South Wales, 1994.

[Sta95] Thad Starner. Visual recognition of American Sign Language using Hidden Markov Models. Master's thesis, MIT Media Lab, July 1995. Available at: ftp://whitechapel.media.mit.edu/pub/tech-reports/TR-316.ps.Z .

[Stu92] D. J. Sturman. *Whole Hand Input*. PhD thesis, MIT, February 1992. Available at: ftp://media.mit.edu/pub/sturman/WholeHandInput/* .

[Stu94] Student Chapter of the Association for Computing Machinery. *Power Glove Serial Interface*. Student Chapter of the Association for Computing Machinery, UIUC, 2.0 edition, 1994.

[SZ94] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, January 1994.

[TK91] Tomoichi Takahashi and Fumio Kishino. Gesture coding based in experiments with a hand gesture interface device. *SIGCHI Bulletin*, 23(2):67–73, April 1991.

[Uni89] Griffith University. Signs of life. Video, 1989.

[Vam] Peter Vamplew. The SLARTI sign language recognition system: A progress report. Unpublished report.

[Waron] Simon Keith Warfield. *Segmentation of Magnetic Resonance Images of the Human Brain*. PhD thesis, University of New South Wales, In preparation.

[Wex93] Alan Daniel Wexelblat. A feature-based approach to continuous gesture analysis. Master's thesis, MIT, 1993. Available at: ftp://media.mit.edu/wex/MS-Thesis.ps.gz .

[WK91] Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufman, 1991.

[Zim91] Thomas G. Zimmerman. Optical flex sensor. Patent, VPL Inc, 1991.

[ZL87] Thomas G. Zimmerman and Jaron Lanier. Hand gesture interface device. In *CHI + GI Conference Proceedings*, pages 189–192, 1987.

[ZL92] Thomas G Zimmerman and Jaron Lanier. Computer data entry and manipulation apparatus method. Patent Application 5,026,930, VPL Research Inc, 1992.